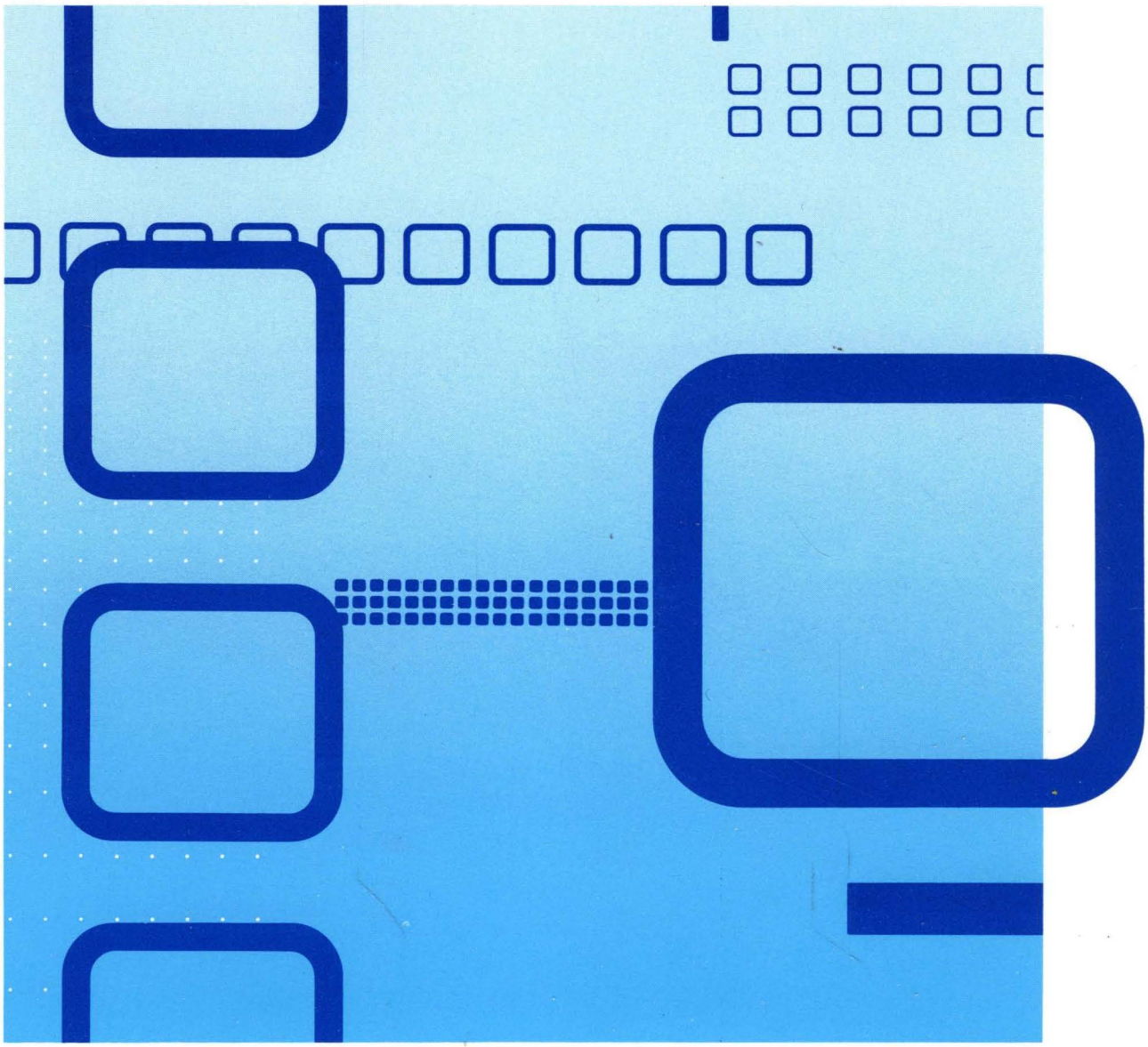
IBM

Token-Ring Network

SC30-3374-02
39F9354

Architecture Reference



Token-Ring Network

Architecture Reference

SC30-3374-02

This product is intended for use within a single establishment and within a single, homogeneous user population. For sensitive applications requiring isolation from each other, management may wish to provide isolated cabling or to encrypt the sensitive data before putting it on the network.

Third Edition (September 1989)

Changes are made periodically to the information herein; these changes will be incorporated in new editions of this publication. It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

The following statement does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Communication Systems Information Development, Department E02, PO Box 12195, Research Triangle Park, North Carolina, U.S.A. 27709. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

Note: This reference describes in detail the architecture of the IBM Token-Ring Network. Describing this architecture does not constitute a commitment by IBM to provide products that implement all aspects of the architecture. For information regarding the current implementation of this architecture in IBM products, refer to the IBM publications listed in *IBM Local Area Network Administrator's Guide*, GA27-3748, or contact your IBM marketing representative.

This reference does not describe specific equipment that connects to the IBM Token-Ring Network, or specific programs that implement this architecture.

To understand this reference, you need a background in the concepts of network design and implementation. You must be familiar with the terms and concepts of IBM Systems Network Architecture (SNA). It may be beneficial if you are also familiar with the networking standards established by:

- The European Computer Manufacturers Association (ECMA)
- The Institute of Electrical and Electronics Engineers, Inc. (IEEE)
- The International Standards Organization (ISO)
- The International Telegraph and Telephone Consultative Committee (CCITT).

This reference is divided into five parts:

- **Part 1** provides an overview of the IBM Token-Ring Network architecture, including basic functional layering concepts and a description of the format of the frames used in the IBM Token-Ring Network.
- **Part 2** describes in detail the Medium Access Control (MAC) sub-layer of the Data Link Control layer.
- **Part 3** describes in detail the Logical Link Control (LLC) sub-layer of the Data Link Control layer.
- **Part 4** describes in detail the LAN manager and the concepts of network management.
- **Part 5** contains the appendixes, list of abbreviations, glossary, and index.

Prerequisite Publication

The *IBM Token-Ring Network Introduction and Planning Guide*, GA27-3677, contains information needed before using this reference. It introduces the IBM Token-Ring Network and explains in detail the process of planning for an IBM Token-Ring Network.

Related Publications

The following publications contain information that may be helpful:

- *IBM Systems Network Architecture Technical Overview*, GC30-3073.
- *ECMA-89*
  Token-Ring Local Area Network Standard
- *ISO 8802/5*
  Token-Ring Local Area Network Standard
- *ISO 8802/2*
  Logical Link Control Standard for Local Area Networks

- *ISO 4335 (Revised)*
  Information processing systems, data communication, high-level data link control procedures, consolidation of elements of procedures
- *ISO 7809-1984*
  Information processing systems, data communication, high-level data link control procedures, consolidation of classes of procedures
- *CCITT — Recommendation X.25 (LAPB)*
  Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode on public data networks.
- *SNA Format and Protocol Reference Manual: Management Services*, SC30-3346
- *SNA Formats*, GA27-3136
- *IBM Local Area Network Technical Reference*, SC30-3383.

A more comprehensive list of other related publications can be found in *IBM Local Area Network Administrator's Guide*, GA27-3748.

New in This Edition

This edition contains new information concerning:

- Early token release
- Logical Link Control (LLC) State Tables
- Network Manager (IBM LAN Manager Version 2.0 protocols, frames, and flows)
- Largest frame size values
- Spanning tree protocol
- Functional addresses
- Alert transport frames
- IEEE and user-defined SAPs (Service Access Points)
- Wire fault regions.

Contents

Figures

Part 1. The IBM Token-Ring Network

Chapter 1. Introduction to the Architecture

Note: This reference describes in detail the architecture of the IBM Token-Ring Network. Describing this architecture does not constitute a commitment by IBM to provide products that implement all aspects of the architecture. For information regarding the current implementation of this architecture in IBM products, refer to "Related Publications" on page iii or contact your IBM marketing representative.

The IBM Token-Ring Network is a high-speed communication network that consists of *physical equipment* and *architecture*.

Physical equipment includes adapters, attaching devices, and interconnecting cable; these are described in the *IBM Token-Ring Network Introduction and Planning Guide*, GA27-3677.

The architecture, which is described in this reference, is:

> The description of the logical structure, formats, protocols, and operational sequences for transmitting information through, and controlling the configuration and operation of, the IBM Token-Ring Network.

Product designers and developers, system programmers, and others who need detailed information about the architecture of the IBM Token-Ring Network should therefore use this reference.

This architecture reference describes in detail the Data Link Control, Physical layers, and Management for the IBM Token-Ring Network. The architecture for the IBM Token-Ring Network can support a variety of higher-layer network protocols. As a Physical and Data Link Control layer for IBM Systems Network Architecture (SNA), the IBM Token-Ring Network supports communication between SNA nodes.

The Data Link Control Layer

Each node in an IBM Token-Ring Network contains a Data Link Control layer, called DLC.LAN. DLC.LAN consists of a manager function, called DLC.LAN.MGR; a Logical Link Control (LLC) sub-layer, which includes one or more link stations, an access channel control, and a user datagram service; and a Medium Access Control (MAC) sub-layer, which includes one or more medium access controls.

Notes:

1. A *link* is a logical connection between two link stations, providing data transfer between two nodes; a *node* is either of the end points of the link; and a *link station* is a protocol machine that manages the elements of procedure required for the exchange of data, and that schedules data transfer over the link.
2. In this reference, specific states and functions are written as a set of qualifiers separated by periods. For example, DLC.LAN.MGR is the manager function (MGR) for the local area network component (LAN) of the Data Link Control layer (DLC).
3. In qualifiers that are composed of more than one word (such as some parameters), the individual words are connected by underscores to indicate that they are all part of a single qualifier, rather than being separate qualifiers. Examples include timer_values and desired_functional_address_mask.

Figure 1-1 on page 1-3 illustrates the relationships among these architecture components.

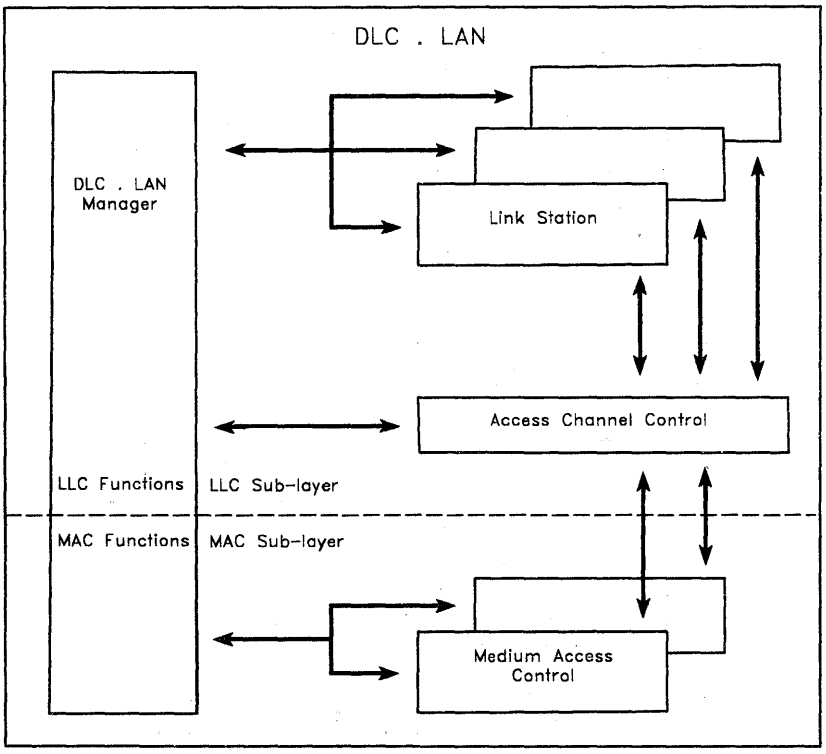


Figure 1-1. Component Structure for DLC.LAN

Note: The structural decompositions in this reference represent the meta-implementation, not the actual implementation, of the IBM Token-Ring Network. They are meant to aid and guide implementers; they are *not* meant to restrict implementers if the meta-implementation is not well-suited to a given environment.

DLC.LAN.MGR

DLC.LAN.MGR covers both the LLC and MAC sub-layers, and provides management functions for both. It supervises the operation of DLC.LAN and directs the flow of information through the MAC and LLC sub-layers. DLC.LAN.MGR controls link activation, the attachment of ring stations to the ring, and the removal of ring stations from the ring.

DLC.LAN.MGR also acts as the interface between DLC.LAN and the Physical Unit. In this capacity, DLC.LAN.MGR handles those records from the Physical Unit that require action on the part of DLC.LAN.MGR, and passes other records from the Physical Unit to appropriate link stations.

For more details, see Chapter 4, "Operation of DLC.LAN.MGR" on page 4-1.

The Logical Link Control (LLC) Sub-Layer

The Logical Link Control (LLC) sub-layer provides sequential, *connection-oriented* data transfer and non-sequential, *connectionless* data transfer (see page 9-1). For a detailed description of the LLC sub-layer, see "Part 3. The Logical Link Control (LLC) Sub-Layer." The sub-components of the LLC sub-layer and the functions they perform on behalf of DLC.LAN are described in the following sections.

Link Stations

The link stations provide sequential *connection-oriented* data transfer and error recovery for one or more links.

The Path Control layer (see page 4-1) passes basic transmission units (BTUs) to the link stations, which add appropriate control information. On each link, the local and remote link stations use the HDLC asynchronous balanced mode of operation to keep LPDUs in sequence and to detect and correct, by retransmission, LPDUs that are out of sequence. Link stations pass the BTU and the control field to the access channel control.

See Chapter 11, "Operation of Link Stations" on page 11-1.

Access Channel Control

The access channel control multiplexes message units flowing between the link stations and the MAC sub-layer, and between DLC.LAN.MGR and the MAC sub-layer.

The access channel control builds LLC protocol data units (LPDUs) from the information in its routing table for that link station, and transmits the LPDUs to the MAC sub-layer. It also routes the LPDUs it receives from the MAC sub-layer to the appropriate link station or to DLC.LAN.MGR.

See Chapter 10, "Operation of the Access Channel Control" on page 10-1.

User Datagram Service

The user datagram service, which is part of the DLC.LAN.MGR, provides connectionless data transfer where data is sent and received without any correlation to previous or subsequent data and without the need for the establishment of a data link connection.

A higher-layer protocol passes the data to be transferred and the source and destination addressing information to the user datagram service, which adds appropriate control information. The user datagram service passes the addressing information, control field, and data to the access channel control. The user datagram service does not provide acknowledgment of data, nor does it provide any flow-control or error-recovery procedures. See Chapter 9, "Connectionless Service" on page 9-1.

The Medium Access Control (MAC) Sub-Layer

The Medium Access Control (MAC) sub-layer controls the routing of information between the Physical layer and the Logical Link Control sub-layer. It provides:

- Address-recognition — for initiating the copying of a frame based on the destination address in the physical header. Each ring station must be able to recognize at least its own individual MAC address and an all-stations broadcast address. In addition, ring stations should be able to recognize one or more group addresses.
- Frame-copying — for copying a frame off the ring.
- Frame control recognition — for determining the frame format and the type.
- Delimiting of frames — for determining the start and end of a frame. This includes originating frames for transmission, and analyzing received frames.

- Frame status generation and verification — for providing and verifying additional information (the frame check sequence, bits in the ending delimiter and frame status field) in each frame to detect transmission errors.
- Priority management — for gaining access to the transmission medium based on priority and issuing the proper priority level tokens.
- Routing — for determining which function in the node (or above) should process the frame.
- Timing — to provide the timers required by the MAC management protocols.
- Token management — for gaining access to the physical transmission medium, including appropriate supervision protocols in case of errors.

For a detailed description of the MAC sub-layer, see "Part 2. The Medium Access Control (MAC) Sub-Layer."

The Physical Layer

Each node in an IBM Token-Ring Network contains a Physical layer. The Physical layer provides attachment to the transmission medium, and contains the cable and the circuit switches that are used to reconfigure the physical equipment.

The primary function of the Physical layer is to encode, transmit, recognize, and react to the following signals:

- Bits (B'0', B'1')
- Code violations
- Signal losses (recognition and reaction).

See Appendix B, "The Differential Manchester Code" on page B-1 for more information on these signals.

The Physical layer also manages the following functions:

- The master clock, which generates timing information
- The latency buffer, which compensates for minor differences in timing between two ring stations
- The phantom circuit, which provides voltage so that ring stations can attach to the ring.

See Appendix C, "Physical Interfaces" on page C-1 for details of the required physical interface to the IBM Token-Ring Network.

Chapter 2. MAC Frame Format

The basic transmission unit on the IBM Token-Ring Network is the *frame*. Frames are composed of a number of fields of 1 or more bytes, as shown below:

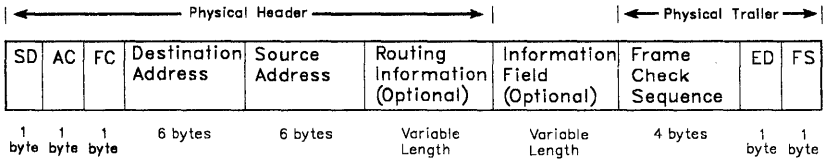


Figure 2-1. IBM Token-Ring Network Frame Format. SD is the starting delimiter; AC is the access control field; FC is the frame control field; ED is the ending delimiter; and FS is the frame status field.

Each field named above is described in detail in this chapter. In each field, and in the frame as a whole, the high-order byte (byte 0) is transmitted first, as is the high-order bit (bit 0) within each byte (that is, left to right in the figure above).

The physical header contains the starting delimiter, the access control and frame control fields, the destination and source addresses, and the optional routing information field.

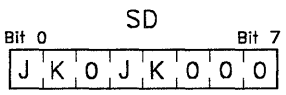
The physical trailer contains the frame check sequence, the ending delimiter, and the frame status field.

Code violation protection covers the access control and frame control fields, the destination and source addresses, the optional routing information field, the information field, and the frame check sequence.

The frame check sequence covers the frame control field, the destination and source addresses, the optional routing information field, the information field, and the frame check sequence itself.

Starting Delimiter

The starting delimiter for a frame is a single byte with the following format:



J = Code Violation
K = Code Violation

Figure 2-2. Starting Delimiter

All valid frames (and tokens) start with this byte, exactly as shown above. A frame or token that starts with any other byte or combination of bits is invalid.

J and K indicate *code violations*, which identify the byte as a delimiter. See Appendix B, "The Differential Manchester Code" on page B-1 for more about code violations.

Access Control Field

The access control field is a single byte with the following format:

P = Priority Bits
T = Token Bit
M = Monitor Bit
R = Reservation Bits

Figure 2-3. Access Control Field

Note: The term "priority" used in this reference always means "access priority."

Priority Bits

The priority bits indicate the priority of a token.

In a multiple-priority system, each ring station is assigned an allowed access priority, which indicates the maximum token priority the ring station can use to transmit data. A ring station can use a token at a priority less than or equal to the ring station's allowed access priority.

A ring station may assign different priority levels to data. The priority level assigned to data must be less than or equal to the ring station's allowed access priority, except for specialized management data. When a ring station with data to transmit detects a token with a priority less than or equal to the data's assigned priority, the ring station changes the token to a frame (by appending the data to be transmitted and the appropriate fields) and transmits the frame.

There are eight priority levels, from B'000' (the lowest) to B'111' (the highest). For example, B'110' is a higher priority level than B'011'. For a full discussion of the use of priority bits, see "Access Priority" on page 3-16, "Token Transmission Finite State Machine" on page 7-30, and "Frame Transmission Finite State Machine" on page 7-32.

Token Bit

In a token, this bit is set to B'0'; in a frame, it is set to B'1'.

Monitor Bit

The monitor bit is used to prevent a token whose priority is greater than B'000', or any frame, from continuously circling the ring. If an active monitor detects a frame or a priority token with the monitor bit set to B'1', it then purges the ring and issues a new token. The precise protocol for setting and interpreting this bit is discussed in "Monitoring Token and Frame Transmission" on page 3-22.

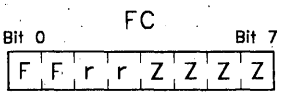
Reservation Bits

The reservation bits allow ring stations with high access priorities to request (in frames or tokens as they are repeated) that a token be issued at the needed priority. There are eight reservation levels, from B'000' (the lowest) to B'111' (the highest). For example, B'110' is a higher reservation level than B'011'.

The precise protocol for setting these reservation bits is described in "Access Priority" on page 3-16.

Frame Control Field

The frame control field defines the type of frame, as well as certain MAC and information frame functions. The frame control field is a single byte with the following format:



F = Frame Type Bits
r = Reserved Bits
Z = Control Bits

Figure 2-4. Frame Control Field

Frame Type Bits

The frame type bits indicate the type of frame, as follows:

B'00' = MAC frame
B'01' = LLC frame
B'10' = Undefined frame (reserved for future use)
B'11' = Undefined frame (reserved for future use).

Medium Access Control Frames

If the frame type bits indicate a MAC frame, then all ring stations that have a matching destination address (individual or group) copy the MAC frame, using normal or express buffers as indicated by the control bits. (Express buffers are described in "Express Buffer and Other Frame Control Field Values" on page 5-24.)

Logical Link Control Frames

If the frame type bits indicate an LLC frame, the control bits are reserved by IBM for future use. They are transmitted as zeros (X'0'); their value is ignored by receiving ring stations.

Undefined Format

The undefined format values (B'10', B'11') are reserved for frame types that may be defined in the future. However, although currently undefined, any future frame formats will adhere to the following conditions:

- The format will be delimited by the starting delimiter and access control field, and by the ending delimiter and frame status field, as defined in this chapter. (Additional fields may follow the frame status field.)
- The position of the access control and frame control fields will not change. The definition of the format of the access control field, and of the first 2 bits of the frame control field, will not change.
- The access control field and the ending delimiter will be separated by an integral number of bytes. This will be at least 1 byte (the frame control field); the maximum length is subject to the constraints of the T(any_token) timer (see "T(any_token)" on page A-2). A ring station that detects a non-integral number of bytes will set the error-detected bit in the ending delimiter of repeated frames to B'1' (see "Ending Delimiter" on page 2-13).
- All bits between the starting and ending delimiters will be either B'0' or B'1' (no code violations). A station that detects any other bit value will set the error-detected bit in the ending delimiter of repeated frames to B'1'.

Control Bits

For a MAC frame, the control bits indicate how the frame is to be buffered. This is described in detail in "MAC Frame Characteristics" on page 5-24.

For LLC and undefined-format frames, the control bits are reserved by IBM for future use. They are transmitted as X'0's; their value is ignored by receiving ring stations.

Reserved Bits

These bits are reserved by IBM for future use. They are transmitted as B'0's; their value is ignored by receiving ring stations.

Destination Address

The destination address identifies the ring stations that are to copy the frame. Destination addresses always consist of six 8-bit bytes:

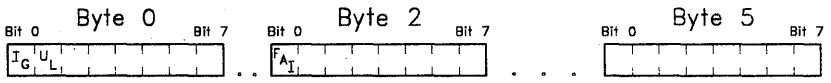


Figure 2-5. Destination Address

- Bit 0 of byte 0 (called the I/G bit) indicates whether the destination address is an individual address (B'0') or group address (B'1'). See "Individual and Group Addresses" on page 3-9 for more details.
- Bit 1 of byte 0 (called the U/L bit) indicates whether the address is universally administered (B'0') or locally administered (B'1'). See "Universal and Local Administration" on page 3-9 for more details.
- Bit 0 of byte 2 (called the functional address indicator) indicates whether a locally administered group address is a functional address (B'0') or a group address (B'1'). See "Functional Addresses" on page 3-10 for more details.

The above indicators are an integral part of each station's address, and must be considered during the address-recognition process.

Source Address

The source address identifies the station that originated the frame. Source addresses always consist of six 8-bit bytes:

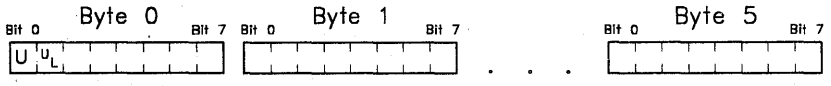


Figure 2-6. Source Address

- Source addresses are always individual addresses, so the individual/group address bit distinction of destination addresses is not needed. Instead, bit 0 of byte 0 of the source address (called the RII bit) is set to B'1' when there is a routing information field present in the frame, and to B'0' when no routing information field is present.

  Ring stations that do not use source routing always set the RII bit to B'0'. This enables such ring stations to coexist on the same IBM Token-Ring Network with ring stations that do use source routing. However, while a source-routing ring station can coexist with a non-source-routing ring station on *another* ring in the same IBM Token-Ring Network, the two stations cannot communicate. If a ring station that does not use source routing receives a frame with a routing information field, it will not interpret the frame correctly.

- As in destination addresses, bit 1 of byte 0 (called the U/L bit) indicates whether the address is universally administered (B'0') or locally administered (B'1'). See "Universal and Local Administration" on page 3-9 for more details.

Routing Information Field

Following the source address is the optional routing information field; this field is omitted if the frame is not going to leave the source ring. This field, when present, consists of a 2-byte routing control field and up to eight 2-byte route designators, as shown below:

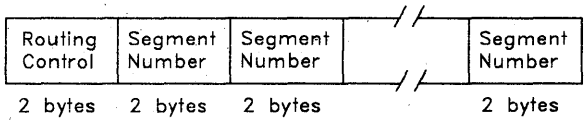
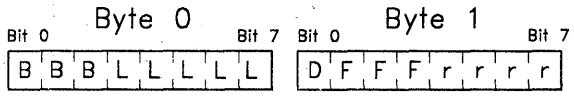


Figure 2-7. Routing Information Field

Routing Control Field

The format for the routing control field is shown below:



B = Broadcast Indicators
L = Length Bits
D = Direction Bit
F = Largest Frame Bits
r = Reserved Bits

Figure 2-8. Routing Control Field

Broadcast Indicators

The broadcast indicators indicate whether the frame is to be sent along a specified path, to all the segments in a network (potentially resulting in multiple copies on a given segment), or to all the segments such that only one copy of the frame appears on each segment in the network. For more about broadcasting, see "Broadcast Terminology" on page 3-2. For a description of how bridges respond to broadcast and non-broadcast frames, see "Route Designator Fields" on page 2-10.

- B'0XX' = Non-Broadcast: This indicates that the route designator field contains a specific route for the frame to travel through the network.
- B'10X' = All-routes broadcast: This indicates that the frame will be transmitted along every route in the network to the destination station. Frames transmitted as all-routes broadcast will result in as many copies at the destination station as there are different routes to the destination station.

  Note: An all-routes broadcast is independent of an all-stations broadcast, which is indicated by all ones in the DA field. An all-stations broadcast implies that every station on the segment will copy the frame, while an all-routes broadcast implies that every bridge in a network will copy and forward the frame to its adjoining segment (unless the next route designator already appears in the routing information field).
- B'11X' = Single-route broadcast: This indicates that only certain designated bridges will relay the frame from one segment to another with the result that the frame will appear exactly once on every segment in the network.

Note: X'' means the bit can be either a 0 or a 1. Its value does not affect the meaning of the indicator.

Length Bits

The 5 length bits indicate the length in bytes of the routing information field, enabling ring stations to parse the rest of the frame correctly. (A ring station *parses* a frame by separating it into its individual fields. When a station parses a frame, it also checks for errors in the formatting of the frame.)

For all-routes or single-route broadcast frames, the originating ring station initializes the length field to X'2', to represent the 2 routing control bytes. Bridges alter the routing information field in broadcast frames by adding route designators.

For non-broadcast frames, which are already carrying routing information, the length field indicates the length of the routing information field, and remains unchanged as the frame traverses the network.

Each bridge checks the length bits. If the length is an odd number of bytes, or if it is less than 2 bytes or greater than 18 bytes, the bridge does not forward the frame.

For all-routes broadcast frames, the length field indicates to a bridge where to append the route designator. The first bridge to forward the frame adds X'4' to the length value (2 bytes for the first route designator and 2 bytes for the next ring's route designator). After that, every bridge that forwards the frame adds X'2' to the length field (2 bytes for the next ring's route designator).

At any given time after crossing the first bridge, the formula {[(Length - 2)/2] - 1} indicates the number of bridges crossed.

Direction Bit

The direction bit enables the bridge to correctly interpret the route designators when it forwards the frame.

If the direction bit is set to B'0', the bridge interprets the routing information field from left to right; if it is set to B'1', it interprets the field from right to left. Using this bit allows the list of ring numbers and bridge numbers in the routing information field to appear in the same order for frames traveling in either direction along the route.

For all-routes broadcast frames, the originating ring station sets the direction bit to B'0'. Bridges do not need the direction bit in broadcast frames, but receivers could uniformly complement the received bit when they obtain routing information from frames with routing information fields.

For off-ring non-broadcast frames, the originating ring station sets the direction bit to B'0' in all frames transmitted to the target, while the target sets the direction bit to B'1' in all non-broadcast frames to the originating ring station.

Largest Frame Bits

These bits specify the largest-size information field (frame excluding headers, see Figure 2-1 on page 2-1) that can be transmitted between two communicating stations on a specific route.

A station that originates a broadcast frame sets the largest frame bits to B'111', the largest possible frame that can travel any path. Bridges that relay a broadcast frame examine the largest frame bits. If the designated size of the largest frame is greater than the capability of that part of the route, the bridge reduces the largest frame encoding to indicate the maximum information field.

The largest field value returned in the responses to the broadcast indicates the largest possible frame each specified route can handle.

The largest frame code points have the following values:

- 000 - As many as 516 bytes in the information field. 516 represents the smallest maximum frame size that a medium access control must support under ISO 8802/2 LLC and ISO connectionless-mode network service (ISO 8473).
- 001 - As many as 1500 bytes in the information field. 1500 represents the largest frame size that ISO 8802/3-standard local area networks can support.
- 010 - As many as 2052 bytes in the information field. 2052 represents a frame size that is useful for transferring a (typical) screen-full of data; that is, this frame size will support the transfer of data for an 80 X 24 screen plus control characters.
- 011 - As many as 4472 bytes in the information field. 4472 represents the largest frame size that can be transmitted using the Fiber Distributed Data Interface (FDDI) Draft Proposed American National Standard. It is also the largest frame size possible for ISO 8802/5-standard stations.
- 100 - As many as 8144 bytes in the information field. 8144 represents the largest frame size that ISO 8802/4-standard local area networks can support.
- 101 - As many as 11407 bytes in the information field.
- 110 - As many as 17800 bytes in the information field. 17800 represents the maximum frame size that a medium access control supports for ISO 8802/5 - standard stations.
- 111 - Used in all-routes broadcast frames

Note: Source-routing end stations on media with a maximum frame size should not send frames in which the headers, routing information fields, and information fields exceed that maximum frame size.

Reserved Bits

These bits are reserved by IBM for future use. They are transmitted as B'0's; their value is ignored by receiving ring stations.

Route Designator Fields

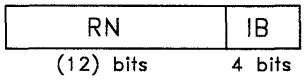
Each ring in a given multiple-ring network is assigned a unique ring number; each bridge is assigned a bridge number, which may or may not be unique. Together, the ring and bridge number form a route designator. When an all-routes broadcast frame is transmitted, each bridge that forwards the frame to another ring adds its bridge number and that ring's number to the frame's routing information field.

When a bridge receives a frame to forward to a ring, the bridge compares the route designators already present in the routing information field with its attached ring numbers and bridge number.

- If there is a target ring number match in an all-route or single-route *broadcast* frame, the bridge discards the frame because it has already circled the target ring.
- If there is *not* a target ring number match in an all-route or single-route *broadcast* frame, the bridge adds its route designator to the frame's routing information field and forwards it.
- If there is a ring number, bridge number, and ring number combination match in a *non-broadcast* frame, the bridge forwards the frame to the indicated ring.
- If there is *not* a ring number, bridge number, and ring number combination match in a *non-broadcast* frame, the bridge discards the frame.

When the frame reaches its destination, the sequence of route designators describes the path from the source ring to the destination ring.

The 2 bytes of the route designator are divided into the ring number portion (12 bits) and the individual bridge number portion (4 bits), as shown below. The individual bridge portion allows parallel bridges to exist, and to share traffic between the same two rings.



RN = Ring Number Portion
IB = Individual Bridge Portion

Figure 2-9. Route Designator Field

Ring Number Portion

Bridges that are attached to different rings have different values for the ring number portion of the route designator; bridges that are attached to the same ring have the same value.

Individual Bridge Portion

Bridges that are attached to the same ring can have the same value for the individual bridge portion of the route designator. However, parallel bridges (those that are attached to the same *two* rings) must have different values.

Because the end of a route is a ring and not a bridge, the individual bridge portion of the last route designator in the routing information field is not defined (that is, it is all B'0's).

Information Field

The variable-length information field contains an integral number of bytes of data or IBM Token-Ring Network management information. For more details, see Chapter 5, "MAC Frames" on page 5-1 and Chapter 8, "LLC Frames" on page 8-1.

Maximum Frame Size

On the IBM Token-Ring Network, a ring station can hold a token for 10 milliseconds (the duration of the T(any_token) timer), which limits the maximum frame size that a station can transmit.

Frame Check Sequence

The frame check sequence is a 4-byte cyclic redundancy check (CRC) covering the frame control field, the destination and source addresses, the optional routing information field, the information field, and the frame check sequence itself.

The ring station begins accumulating the frame check sequence with the first bit of the frame control field, and continues until the end of the frame check sequence. The type of frame used determines the position of the CRC within the frame, and therefore the protection provided by the CRC.

The frame check sequence is generated using the following standard generator polynomial:

$G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + X^0$.

The frame check sequence is the one's complement of the sum (modulo 2) of the following:

1. The remainder of $X^{**}k(X^{31} + X^{30} + X^{29} + \ldots + X^2 + X + 1)$ divided (modulo 2) by G(X), where k is the number of bits in the frame control field, destination and source addresses, optional routing information field, and information field.
2. The remainder after multiplication by X^{32} and then division (modulo 2) by G(X) of the content (treated as a polynomial) of the frame control field, destination and source addresses, optional routing information field, and information field.

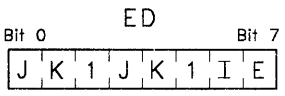
The frame check sequence is transmitted commencing with the coefficient of the highest term. As a typical implementation, at the transmitter, the initial remainder of the division is preset to all B'1's and is then modified by division of the frame control field, destination and source addresses, optional routing information field, and information field by the generator polynomial G(X). The one's complement of this remainder is transmitted, most significant bit first, as the frame check sequence.

At the receiver, the initial remainder is preset to all B'1's. If there are no transmission errors, dividing the number of serial incoming bits of the frame control field, destination and source addresses, optional routing information field, and information field by G(X) results in a unique non-zero remainder. The unique remainder value is the polynomial:

$X^{31} + X^{30} + X^{26} + X^{25} + X^{24} + X^{18} + X^{15} + X^{14} + X^{12} + X^{11} + X^{10} + X^8 + X^6 + X^5 + X^4 + X^3 + X + 1$.

Ending Delimiter

The ending delimiter is a single byte with the following format:



J = Code Violation
K = Code Violation
I = Intermediate Frame Bit
E = Error-Detected Bit

Figure 2-10. Ending Delimiter

The first 6 bits of the ending delimiter must be as shown (J K 1 J K 1) or the delimiter is invalid. J and K indicate *code violations*, which identify the byte as a delimiter. See Appendix B, "The Differential Manchester Code" on page B-1 for more about code violations.

Intermediate Frame Bit

The intermediate frame bit is set to B'1' to indicate the first frame of a multiple-frame transmission using a single token, or any intermediate frame of a multiple-frame transmission using a single token. It is set to B'0' for a token, for a single-frame transmission, or for the last frame of a multiple-frame transmission using a single token.

When copying a frame for forwarding to an off-ring destination, a bridge does not propagate the value of this bit.

Error-Detected Bit

A ring station that originates a token, frame, or abort sequence sets the error-detected bit to B'0'. Other ring stations repeat the token or frame with this bit set to B'0', unless a ring station detects one of the following:

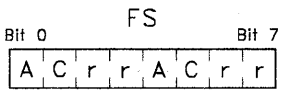
- A frame that contains a code violation between the starting and ending delimiters
- A frame that contains a non-integral number of bytes
- A frame that contains a cyclic redundancy check error.

A ring station that detects such an error checks the setting of the error-detected bit. If it is set to B'0', the ring station is the first to detect the error, and it changes the setting to B'1' and increments the appropriate counter. If it is already set to B'1' (another ring station detected the error and already set the error-detected bit), the ring station repeats the frame or token with the error-detected bit set to B'1'.

When copying a frame for forwarding to an off-ring destination, a bridge does not propagate the value of this bit; it is set to B'0'.

Frame Status Field

The frame status field is a single byte with the following format:



A = Address-Recognized Bits
C = Frame-Copied Bits
r = Reserved Bits

Figure 2-11. Frame Status Field

Address-Recognized Bits and Frame-Copied Bits

The address-recognized (A) and frame-copied (C) bits are used in neighbor notification (see page 3-20), in the duplicate address test (see page 5-11), and in assured delivery (see page 5-26).

A ring station that is originating a frame sets these bits to B'0'. If another ring station recognizes the destination address as its own address or as an applicable group address, or a bridge recognizes a frame to copy due to an RI field match, it sets the A bits to B'1'. If the receiving ring station copies the frame into its receive buffer, it also sets the C bits to B'1'. This allows the originating ring station to determine whether:

- The designated receiving ring station is non-existent or inactive
- The designated receiving ring station exists but did not copy the frame
- The frame was copied.

The A and C bits occur twice in the frame status field because this field is not protected by the frame check sequence. To minimize the possibility of error, the ring station considers the A and C bits valid only when both A bits are equal and both C bits are equal. In addition, only the following values of the A and C bits are considered valid:

- AC=B'00': No ring station recognized the destination address and the frame was not copied. Or if a routing information field is present, no bridge on the ring recognized the need to forward the frame on its adjoining ring.
- AC=B'11': A ring station recognized the destination address and copied the frame. Or in a frame with an routing information field, a bridge recognized a condition to result in forwarding the frame to its adjoining ring and copied the frame for forwarding.
- AC=B'10': A ring station recognized the destination address but did not copy the frame. Or a bridge, in a frame with a routing information field, recognized a condition to copy the frame for forwarding but was unable to copy the frame. The receiving ring station logs each occurrence of this bit combination.

The combination AC=B'01' is considered invalid, because this combination indicates that no ring station recognized the address but the frame was copied.

Note: If a destination ring station detects that the A bits have already been set in an on-ring frame, and the destination address is not a group address, the ring station assumes that a duplicate address problem exists. The ring

station increments the frame-copied error counter of the Non-Isolating Error Counts subvector for MAC frames (see page 5-20).

If the routing information field indicates an off-ring destination, a bridge forwards the frame *on the source ring* with the A bits set to B'1', and with the C bits set to B'1' if the bridge copies the frame for forwarding. However, the bridge forwards the frame *to the target ring* without setting the A and C bits (that is, with A=C=B'0').

Reserved Bits

These bits are reserved by IBM for future use. They are transmitted as B'0's; their value is ignored by receiving stations.

Chapter 3. Token-Ring Concepts

This chapter describes in detail the terms, concepts, and procedures that are basic to the operation of the IBM Token-Ring Network.

The Ring

In the IBM Token-Ring Network, a *ring* consists of ring stations and the transmission medium to which they are attached. A *ring station* is the combination of functions that allows a device to attach to the ring and to use the access protocols. A sample ring configuration is shown below:

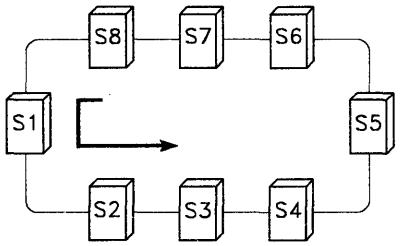


Figure 3-1. Sample Ring Configuration. S1 through S8 are ring stations.

A ring station transfers data to the ring, where the data travels sequentially from ring station to ring station, along the path indicated by the arrow in the figure above. Each ring station repeats the data, checking it for errors, and copying it if appropriate. When the data returns to the originating ring station, the station removes it from the ring.

Each ring station can serve one or more attached devices (such as terminals and printers), allowing them to communicate with other attached devices on the ring.

A minimum 24-bit delay is required on the ring, to allow a 24-bit token to circle the ring successfully (see "Tokens, Frames, and Abort Delimiters" on page 3-12). The total delay on the ring consists of the sum of the ring station delays plus the propagation delay introduced by the transmission medium.

The IBM Token-Ring Network requires addressing functions so that communication between any two ring stations can be uniquely identified. Addressing is independent of the underlying physical configuration.

The IBM Token-Ring Network also requires data checking functions to preserve the integrity of its access control and ring management transmissions.

The above requirements overlap the customary Data Link Control functions of addressing, framing, and error detection.

Multiple-Ring Connections

This architecture supports multiple-ring connections with bridges, as shown below:

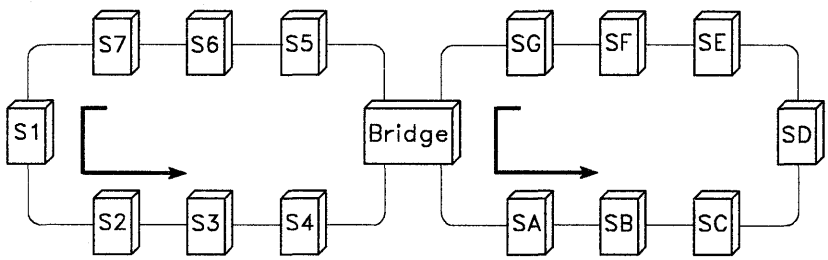


Figure 3-2. Multiple-Ring Connections. S1 through S7 and SA through SG are ring stations.

Data can be transmitted from a ring station on one ring to a ring station on any other ring.

The bridge, which also acts as a ring station on each ring to which it is attached, copies frames destined for other rings, and transmits frames from other rings destined for the local ring, or for rings beyond it. The routing information field or destination address of a frame determines whether the bridge copies the frame (see "Routing Information Field" on page 2-6).

Source Routing

Source routing is the way the IBM Token-Ring Network routes frames through a multiple-ring local area network. A *route* is the path that a frame travels through a network from an originating station to a destination station. Source routing does not require centralized routing tables; in source routing, each frame carries information about the route it is to follow. This routing information is acquired through a search process that originates at the source station. Routing information is acquired using the TEST or XID command LPDU. For information on these commands, see "Test (TEST) Command" on page 8-12 and "Exchange Identification (XID) Command" on page 8-12.

Broadcast Terminology

In the IBM Token-Ring Network, to *broadcast* a frame is to send it to more than one ring station, or to more than one ring. A frame can be broadcast when a specific destination address is known but the location and route to the destination are not known.

All-routes broadcast is sending a frame to all interconnected rings of a local area network. All-routes broadcast is indicated by a frame with the broadcast indicator of the routing information control field set to B'10X' (see "Routing Information Field" on page 2-6). The addressing of particular ring stations on those rings is determined by the destination address field.

Single-route broadcast is the sending of a frame to all interconnected rings of a local area network so that only one copy of a frame appears on each ring. This is different from the all-routes broadcast, where multiple copies of a frame may end up on a ring if there are multiple routes to a ring in a network. Single-route broad-

cast is indicated by a frame with the broadcast indicators of the routing information control field set to B'11X'.

All-routes and single-route broadcasts are independent of all-stations broadcast, which is sending a frame to all ring stations on a ring or rings if accompanied by an all-routes or single-route RI field.

On-Ring Determination

The originating station sends a TEST or XID command LPDU on the local ring with the address of the destination in the destination address field and to the null SAP address. The destination station responds with a TEST or XID response LPDU. If the originating ring station does not receive a response LPDU, the destination is not on the local ring.

Off-Ring Determination

A station can dynamically discover the routing information when the station needs it by several ways, two of which are described below.

All-Routes Broadcast Route Determination

The originating station sends a TEST or XID command LPDU to all rings. This frame passes through the interconnected rings, searching for the destination address and accumulating routing information as it passes through bridges on the way.

As the LPDU fans out through the multiple-ring network, copies are created, all of which continue to search for the destination address. If more than one route to the destination address exists, then more than one LPDU will reach the destination station. As the destination station receives each LPDU, it returns the acquired routing information to the originating station in a TEST or XID response LPDU, which follows the original route in reverse.

If more than one route to the destination address was found, all are returned to the originating station, which chooses a preferred route. The destination station learns the preferred route when it receives the first non-broadcast frame from the originating station. The destination station then uses the preferred route, followed in the opposite direction, for subsequent transmissions to the originating station. In that way all subsequent transmissions follow the same route. (See "Link Activation Races" on page 11-14.)

Single-Route Broadcast Route Determination

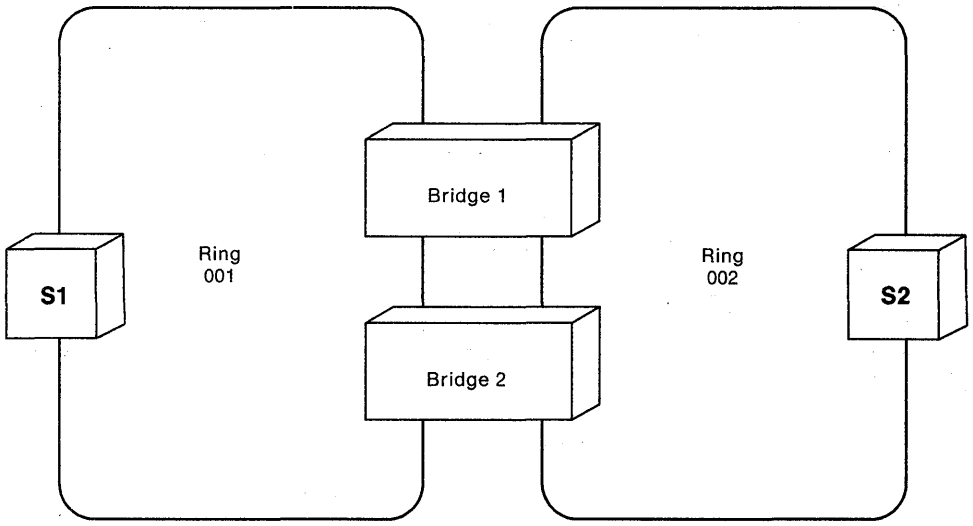
Another way a station can determine the route to a destination station is by sending an XID or TEST command LPDU through the network so that exactly one copy appears on each ring. This technique uses the single-route broadcast feature defined by source routing. The destination station, upon receipt of the XID or TEST command LPDU, sends an XID or TEST response LPDU to the originating station using the all-routes broadcast feature of source routing. As a result, multiple copies of the response may be received by the originating station. The originating station chooses the routing information from one of the received responses. With this technique, the routing information is collected in the response.

Description

A single-route broadcast path through a network provides a spanning tree configuration on the network topology. It guarantees that only one copy of a limited broadcast frame traverses each network segment. Bridges in the spanning tree are configured for forwarding single-route broadcast frames. Bridges can be configured for single-route broadcast in two ways, manually or automatically. The automatic configuration is accomplished by using the spanning tree algorithm where bridges communicate with each other to establish and dynamically maintain a single-route broadcast path through the network.

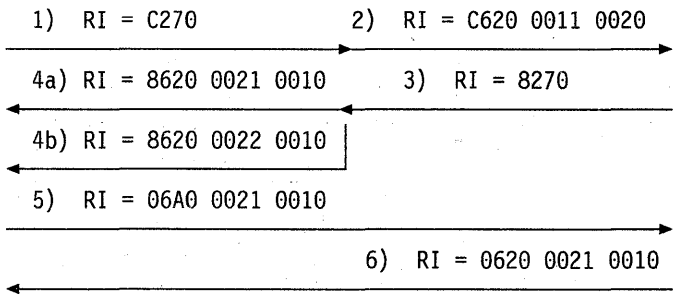
Single-route broadcast route determination is preferred over all-routes broadcast route determination. A frame that is sent all-routes broadcast will be duplicated as it traverses parallel bridges. Since the TEST or XID frame being sent by the originating station may be sent to a group or functional address, it would be best if only one copy of this frame were delivered to each ring through the use of single-route broadcast. That way other stations with the same group or functional address will only be interrupted once. When the destination station responds using an all-routes broadcast frame, only the originating station will receive the multiple copies of this frame allowing the originating station to choose a preferred route.

EXAMPLE: The following figure and example text describe single-route broadcast route determination:



Bridge 1 has single-route broadcast active.
Bridge 2 does not have single-route broadcast active.

Figure 3-3. Single-Route Broadcast Route Determination



1) S1 sends a single-route broadcast UI frame to S2.

2) Bridge 1 forwards the frame to ring 2 and adds RI data.
Bridge 2 does not forward the single route broadcast frame.

3) S2 responds to S1 with an all-routes broadcast frame.

4a) Bridge 1 forwards the frame and adds RI data.

4b) Bridge 2 forwards the frame and adds RI data.

5) S1 selects a path from one of the two frames and sends a routed frame by flipping the direction bit.

6) S2 responds by flipping the direction bit to reverse the path selected by S1.

Route Building by Bridges

LAN bridges are characterized by the following variables:

- Ring numbers. Each ring in a multi-ring network is assigned a unique ring number. The bridge is characterized by the numbers of the two rings it connects.
- Bridge number. Each bridge between any pair of rings is assigned a unique bridge number.
- Largest frame size. The largest frame size is either a function of the largest frame size supported on either of the rings it connects or of the bridge itself.
- Hop count limit. This variable specifies the number of rings an all-route broadcast frame can traverse before it is discarded by the bridge.
- Single-route broadcast indicator. This variable specifies whether the bridge is configured to forward single-route broadcast frames.

Each frame in a multiple-ring network contains a destination address, a source address, and possibly additional routing information. The routing information consists of an ordered list of ring and bridge numbers through which the frame is to pass to reach the destination address. Bridges add the routing information to the routing information field of all-routes and single-route broadcast frames they forward. Bridges also make frame-forwarding decisions based on the contents of the routing information field of non-broadcast frames. When an all-routes or single-route broadcast frame is forwarded from its originating ring to the next ring, the first bridge adds the number of the ring from which the frame was forwarded, its bridge number, and the number of the next ring. The first bridge also indicates (in the frame) the largest frame size that can be forwarded through the bridge. Subsequent bridges that copy the all-route or single-route broadcast frame add their bridge number and the number of the next ring on which the frame is transmitted. They also examine the largest frame size indication in the received frame and if the largest frame size supported by the bridge is smaller than that indicated in the frame, the bridge indicates its largest frame size in the frame before forwarding it.

To prevent the establishment of routing information that would cause a frame to continuously circulate around a loop of interconnected rings, bridges only forward frames that have not traversed the next ring. Bridges examine the routing information in received frames. If the ring number of the next ring is already present in the routing information of a frame, the bridge does not forward the frame, since it has already traversed that ring. Bridges also examine the number of rings already traversed by all-route broadcast frames they forward. If the number of rings indi-

cated in the routing information field of a received all-route broadcast frame exceeds the hop count limit for the bridge, the bridge does not forward the frame.

When a frame reaches its destination, the routing information in the frame indicates the path taken by the frame through the network. This routing information can be used for subsequent communication between the stations, eliminating the requirement to broadcast frames and conserving network bandwidth. In this way, source routing provides a mechanism to dynamically determine a route from one station to another in a multi-ring network.

Spanning Tree Protocol

Bridges can be configured to determine automatically if they should forward single-route broadcast frames. Bridges configured for automatic determination of the single-route broadcast path use the spanning tree algorithm to communicate with other bridges in the network. Bridges using the spanning tree algorithm automatically adjust for changes in topology caused by bridges entering or leaving the network. Bridges that are running the spanning tree algorithm use the following frame to communicate with each other.

Hello BPDU Frame Format

The HELLO BPDU is sent to and from the bridge spanning tree protocol SAP as an LLC type 1 or U1 frame:

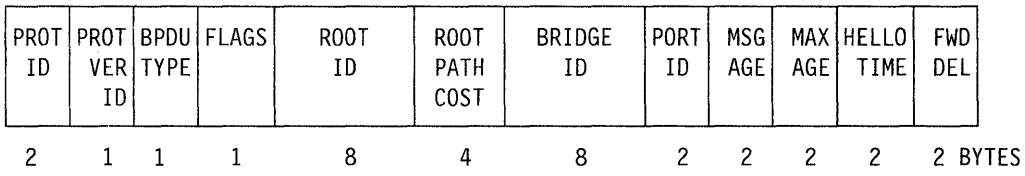


Figure 3-4. HELLO BPDU Frame Format

- Protocol Identifier

  This field is 2 bytes long and takes the value X'00 00', which identifies the Spanning Tree Algorithm and Protocol.

- Protocol Version Number (version 0 of the standard)

  This field is 2 bytes long and is used to identify the protocol version number that is being used.

- BPDU Type

  This field is one byte long and denotes a Configuration BPDU.

- Flags

  This field is one byte long and takes the value of '0000 0000'.

- Root Identifier

  This field is 8 bytes long and is the bridge identifier of the bridge believed to be the root.

- Root Path Cost

  This field is 4 bytes long and is the sum of the designated cost and the root path cost from a previously received HELLO BPDU on the root port (the port closest to the root). The default for the designated cost for each bridge is 10. This value should be adjustable.

- Bridge Identifier

  This field is 8 bytes long and has the form X'UUUU MMMM MMMM MMMM'.

  Where:

  - UUUU is set by the user
  - MMMM MMMM MMMM represents MAC address of the port with the lowest port identifier.

- Port Identifier

  This field is 2 bytes long and is represented by X'RRRB'.

  Where:

  - RRR represents the ring number to which the port is attached
  - B represents the number of the bridge.

- Message Age

  This field is 2 bytes long and is set to X'0000' by the root bridge. Bridges that receive a HELLO BPDU on their root port store the message age. Every second this timer (counter) is incremented. It must be incremented at least once in a bridge. When the bridge sends its HELLO BPDU, the stored message age is used for this field.

- Max Age

  This field is 2 bytes long and has the value of the parameter MAX_AGE(BRIDGE). Its default is 6 seconds.

- Hello Time

  This field is 2 bytes long and has the value of the parameter HELLO(BRIDGE). Its default is 2 seconds.

- Forward Delay

  This field is 2 bytes long and has the value of the parameter FORWARD_DELAY(BRIDGE). Its default is 4 seconds.

In a network using the spanning tree protocols, each bridge assumes one of three roles:

1. The root bridge
   - The root bridge has single-route broadcast set to active in both directions.
   - There is at any one time only one root bridge in the network.
   - The root bridge is the active bridge with the lowest bridge ID in the network.
   - The responsibility of the root bridge is to send a HELLO BPDU (containing its bridge ID, a path cost of zero, and timing information) every two seconds on both LAN segments to which it is connected.

2. A designated bridge
    - A designated bridge has single-route broadcast active in both directions.
    - A designated bridge is either not parallel to any other bridge, or is the only bridge of two or more parallel bridges that has single-route broadcast active.
    - The responsibility of a designated bridge is to recognize and receive Hello BPDUs from the root bridge, update the path cost and timing information in each message, and forward the Hello BPDUs to its other LAN segment.
3. A stand-by bridge
    - A stand-by bridge has single-route broadcast set to inactive in both directions; it cannot forward single-route broadcast frames.
    - The responsibility of a stand-by bridge is to monitor, but not update and forward, the Hello BPDUs. As bridges enter and leave the network, a stand-by bridge may need to assume the role of designated or root bridge and begin forwarding single-route broadcast frames. The Hello BPDUs will indicate when this is necessary.
    - A stand-by bridge is directly parallel to a designated or root bridge, or is at the end of a path that is parallel to a designated bridge. A stand-by bridge either has a path cost that is greater than the designated bridge or, if the path cost is equal, it has a lower priority (higher bridge ID) than the designated bridge.

Bridges use the bridge ID, path cost and timing information in the Hello BPDUs to do the following:

- Determine which role a newly active bridge should assume
- Determine whether a bridge is a parallel bridge or in a parallel path
- Determine which one of two or more parallel bridges should have single-route broadcast active
- Detect when the root bridge or a designated bridge has left the network
- Reassign the bridge roles as necessary when bridges enter and leave the network.

SPANNING TREE OPERATION

The bridge with the highest priority (lowest bridge ID) is chosen as the root bridge. Each bridge then selects the port closest (lowest path cost) to the root bridge as its root port. The root port receives BPDUs. The bridge that provides a LAN with the lowest path cost to the root bridge is selected as the designated bridge for that LAN. A port in a bridge connected to the LAN for which that bridge is designated is selected as a designated port. The designated port transmits BPDUs. Any bridge that is parallel to a root or designated bridge will not have a designated port. The non-root port is selected as a blocking port and the bridge becomes a stand-by bridge.

After a waiting period to ensure that no temporary loops are formed while the topology stabilizes, the root and designated bridge place their root and designated ports in forwarding state. These ports can then forward single-route broadcast frames. Stand-by bridges put their ports in blocking state where they do not forward single-route broadcast frames. Stand-by bridges will forward all-routes broadcast and non-broadcast frames.

TOPOLOGY CHANGES

If a bridge in the spanning tree topology fails or is removed from the network, it will no longer transmit BPDUs. The bridges that receive these BPDUs will time-out and will attempt to reconfigure so that the spanning tree bypasses the failed bridge. If the root bridge fails, the bridge with the highest priority of the remaining bridges will become the new root bridge and the other bridge with the lowest path cost for a LAN to this root bridge will become the designated bridge for that LAN. If a designated bridge fails, then a parallel bridge that was stand-by (with a port in blocking mode) with the lowest path cost will take its place and become the new designated bridge. If two bridges have the same path cost, the bridge with the highest priority (lowest bridge ID) will become the designated bridge.

Addresses

The IBM Token-Ring Network associates a ring station or group of ring stations with a unique MAC sub-layer address. This enables any ring station to attach to the IBM Token-Ring Network.

Individual and Group Addresses

An *individual address* identifies a particular ring station on the IBM Token-Ring Network.

A *group address* identifies a group of destination ring stations on the IBM Token-Ring Network.

Universal and Local Administration

Universal administration means that all individual addresses are assigned, administered, and guaranteed to be unique across all local area networks by the IEEE. This method eliminates customer involvement in the administration of individual addresses, which eliminates the need for site address administrators and address administration programs.

Local administration means that all individual addresses are administered by an authority other than the IEEE. Locally administered addresses must still be unique in the IBM Token-Ring Network in which they occur.

Null Address

X'0000 0000 0000' is the special individual destination address that is considered a *null address*. A frame with a null destination address is not addressed to any ring station; it can be sent, but not received. When the frame returns to its originating ring station, the station strips the frame of its data and issues a new token.

All-Stations Broadcast Addresses

X'FFFF FFFF FFFF' and X'C000 FFFF FFFF' are special *all-stations broadcast addresses*, meaning the frame is addressed to all ring stations on a given ring or interconnected rings. All ring stations *must* be able to recognize at least X'C000 FFFF FFFF' as an all-stations broadcast address. Whether the frame leaves the source ring is determined by the routing information in the frame (see "Routing Information Field" on page 2-6).

Functional Addresses

IBM Token-Ring Network architecture provides bit-specific *functional addresses* for widely used functions, such as the configuration report server. Ring stations use functional address "masks" to identify these functions.

For example, if function G is assigned a functional address of X'C000 0008 0000', and function M is assigned a functional address of X'C000 0000 0040', then ring station Y, whose node contains functions G and M, would have a mask of X'C000 0008 0040'.

All functional addresses are locally administered group addresses (bits 0 and 1 of byte 0 of the destination address are set to B'11'). Bit 0 of byte 2 of the destination address (the functional address indicator) is set to B'0' for functional addresses. The remaining 7 bits in byte 2, and the 8 bits each in bytes 3, 4, and 5, allow up to 31 functional addresses to be defined (because the addresses are bit-specific).

The functional addresses listed below have been defined; all other functional addresses are reserved.

Function Name	Functional Address	Identifying Bit
Active monitor	X'C00000000001'	Byte 5, bit 7
Ring Parameter Server	X'C00000000002'	Byte 5, bit 6
Ring Error Monitor	X'C00000000008'	Byte 5, bit 4
Configuration Report Server	X'C00000000010'	Byte 5, bit 3
NETBIOS	X'C00000000080'	Byte 5, bit 0
Bridge	X'C00000000100'	Byte 4, bit 7
LAN Manager	X'C00000002000'	Byte 4, bit 2
User-defined	X'C00000080000' through X'C00040000000'	Byte 3, bits 0-4; Byte 2, bits 1-7

Figure 3-5. Defined Functional Addresses

Numbering Practices and Order of Transmission

In the IBM Token-Ring Network, a *bit* is a signal that represents a binary value (B'0' or B'1'); a *byte* is made up of 8 bits; a *field* is made up of 1 or more bits. Bytes and bits are numbered in decreasing order of magnitude. That is, byte 0 has a higher order than byte 1, and is transmitted first. Similarly, in an 8-bit byte, bit 0 (the high-order bit) has the value 2^7, and bit 7 (the low-order bit) has the value 2^0; bit 0 is transmitted first.

The figures in this reference, such as the example below, show bits *as they are transmitted on the line, from left to right.*

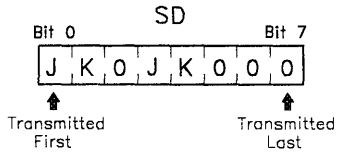


Figure 3-6. Order of Bit Transmission. This is the starting delimiter (SD), which is discussed in detail in Chapter 2, "MAC Frame Format" on page 2-1.

Tokens, Frames, and Abort Delimiters

The IBM Token-Ring Network uses tokens and frames to transmit data on the ring.

A *token* is a control signal that is passed from ring station to ring station between transfers of data. A *frame* is the unit of data transmission on the IBM Token-Ring Network, and includes delimiters, control characters, information, and checking characters. The figure below shows the format of the token:

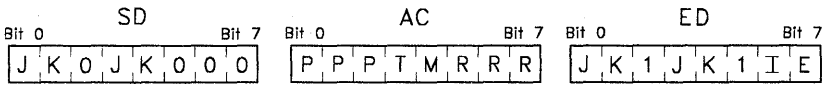


Figure 3-7. Token. In this figure, SD and ED are the starting and ending delimiters; AC is the access control field (see Chapter 2, "MAC Frame Format" on page 2-1).

A ring station that is waiting to transmit data captures a token, changes it to the start of a frame, appends destination and source information and data to the frame, and passes it on. A destination station along the way copies the data and passes the frame on. As the frame returns to its origin (having circled the ring), the originating ring station removes it from the ring and releases a new token for another ring station to capture and carry on the process.

This is called *single-token* protocol, because only one token can circulate on the ring at any time. For more about frames, see Chapter 2, "MAC Frame Format" on page 2-1, Chapter 5, "MAC Frames" on page 5-1, and Chapter 8, "LLC Frames" on page 8-1.

An originating ring station can abort a frame that it is transmitting at any time, by transmitting an *abort delimiter:*

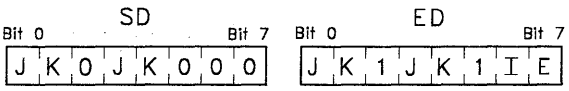


Figure 3-8. Abort Delimiter. In this figure, SD and ED are the starting and ending delimiters (see Chapter 2, "MAC Frame Format" on page 2-1).

A ring station transmits an abort delimiter and takes the specified action when it detects one of the following conditions:

- A transient error internal to the ring station. A transient error is one from which the ring station recovers without having to remove itself from the ring. After transmitting the abort delimiter, the ring station enters Normal Repeat mode (see page 3-15).
- A hard (permanent) error internal to the ring station. After transmitting the abort delimiter, the ring station removes itself from the ring.
- A token in which the third byte is not an ending delimiter (see "Normal Transmit Mode" on page 3-14). After transmitting the abort delimiter, the ring station enters Normal Repeat mode (see "Normal Repeat Mode" on page 3-15).

MAC Sub-Layer Operating Modes

This section describes the normal operating modes, or methods of operation, for the MAC sub-layer. Precise protocol specifications for frame transmission are given in Chapter 7, "Finite State Machines" on page 7-1.

Transmit-Pending Mode

```
Do while station has data to transmit.
 If a starting delimiter is received then
  If there are no code violations in the priority bits then
   If the priority is usable for the frame waiting transmission then
    If the token bit is set to indicate a token (B'0') then
     Call NORMAL_TRANSMIT (see Figure 3-10).
    Else
     Set the priority reservation if greater than the reservation
      in the passing frame.
```

Figure 3-9. TRANSMIT_PENDING Procedure

A ring station in Transmit-Pending mode (a subset of Normal Repeat mode) waits for a token. The ring station assumes that it has received a token when it receives a starting delimiter followed by an access control field with no code violations in the priority bits, and with the token bit set to B'0'. The ring station examines the token's priority bits:

- If the token is not of a usable priority, the ring station sets the reservation bits, if the current reservation in the passing token is less than the priority request of the station (see "Access Priority" on page 3-16 and "Token Transmission Finite State Machine" on page 7-30). It then waits for another token.
- If the token is of a usable priority, the ring station enters Normal Transmit mode (see page 3-14).

Using the single token protocol described above, the ring is idle for frames shorter than the ring length until the transmitting station receives its header, allowing the station to release a token. When a station transmits frames shorter than the length of the ring, the station must wait for the frame to circle around the ring before releasing a token, so that information carried in the header can be used to determine the priority of the token.

To allow greater use of the ring, the architecture allows an option to release the token early, called *early token release*. With early token release, a transmitting station releases the token after transmitting the ending delimiter. If the header of the transmitted frame had not been received from around the ring, the station releases a token with the priority and reservation of the token used for transmission.

If the station does receive the header of the frame from around the ring before releasing the token, the token is released according to the priority and reservation in the transmitted frame, and the stored priority levels in the station (see "Priority Bits" on page 2-2).

Normal Transmit Mode

```
Set the token bit to B'1' to indicate a frame.
Transmit the access control field and frame control field
 for the frame being transmitted.
If the ending delimiter is received then
   Transmit the addressing information (source address, destination
    address, and routing information [if necessary])
    (For information on these fields, see "Source Address" on page 2-5,
    "Destination Address" on page 2-5, and
    "Routing Information Field" on page 2-6.)
   Do while (there is data to transmit).
    Transmit the data.
   Transmit the calculated frame check sequence
    (see "Frame Check Sequence" on page 2-12).
   Transmit the ending sequence
    (ending delimiter and frame status field).
   Start the physical-trailer timer.
   If (physical header has not been received) then
    If early token release option selected then
     Release the token at the priority and reservation of
     the captured token.
      Call WAIT (Figure 3-11).
    If physical-trailer timer expired then Exit.
     If (the early token release option is not selected) or
        ((the early token release option is selected ) and
        (the token has not been released)) then
   Transmit a token at the appropriate priority
   Do while (frame ending sequence not received
    (ending delimiter and frame status field)) and (physical-trailer
    timer has not expired)).
    Transmit idles.
   If physical-trailer timer expired then Exit.
   Cancel the physical-trailer timer.
Else (* ending delimiter not received *)
   Transmit an abort delimiter.
Exit.
```

Figure 3-10. NORMAL_TRANSMIT Procedure

```
Do while (physical header, excluding the RI field, has not been received
with SA equal to station address) and (physical-trailer timer
has not expired).
 Transmit idles.
```

Figure 3-11. WAIT Procedure

In Normal Transmit mode, the ring station sets the token bit to B'1' (indicating a frame), transmits the rest of the access control field, and transmits the frame control field. The ring station then checks for the presence of the token's ending delimiter.

If the ring station cannot find the ending delimiter, it has incorrectly identified a frame as a token; it transmits an abort delimiter and returns to Transmit-Pending mode.

If the ring station finds the ending delimiter, it transmits the rest of the frame: the destination and source addresses; the optional routing information field; the information field, containing the data; the frame check sequence; the ending delimiter; and the frame status field. (See Chapter 2, "MAC Frame Format" on page 2-1 for more about these fields.)

The ring station then starts its T(physical_trailer) timer (see page A-10). If the ring station does not receive its physical header (see page 2-1) by the time it transmits the frame status field, it transmits idles (B'0's) and waits for the physical header to return. If the early token release option has been selected, the station releases the token that it captured for the transmission. If T(physical_trailer) expires and the physical header has not returned, the ring station returns immediately to Normal Repeat mode, without originating a token.

When the physical header returns (excluding the RI field), the ring station compares the returned source address with the source address it transmitted (which is the ring station's individual address). If the addresses are identical and either the early token release option is selected and the token has not been released, or the early token release option is not selected, the ring station transmits a token of the appropriate priority (see "Access Priority" on page 3-16), followed by idles. Otherwise, the ring station continues to wait for its transmitted physical header. The ring station transmits idles until it receives its transmitted physical header and has completely removed its frame from the ring, at which time it returns to Normal Repeat mode.

If the ring station never receives the ending delimiter and frame status field, it transmits idles until T(physical_trailer) expires, at which time it returns to Normal Repeat mode without releasing a token.

Normal Repeat Mode

A ring station that is in Normal Repeat mode checks the data in the tokens and frames it receives, and sets the error-detected, address-recognized, and frame-copied bits, as appropriate, as it repeats the token or frame.

Access Priority

Note: Access priority is not the same as message priority within a node. The term "priority" used in this section means "access priority."

The priority of a token or frame is indicated in the first 3 bits (the priority bits) of the access control field (see "Access Control Field" on page 2-2). Any reservation for a different priority is indicated in the last 3 bits (the reservation bits) of the same field. A ring station uses the reservation bits to request that a token originated on the ring be at the requested priority. The algorithms for making reservations and for transmitting priority frames are described in "Frame Transmission Finite State Machine" on page 7-32.

Not all ring stations will use access priority by transmitting priority frames, but all must be able to originate priority tokens. The allowable token and frame priorities are listed in Figure 3-12.

Priority Bits	Priority
B'000'	Normal user priority, MAC frames that need no token and response-type MAC frames
B'001'	Normal user priority
B'010'	Normal user priority
B'011'	Normal user priority and MAC frames that need tokens
B'100'	Bridge
B'101'	Reserved by IBM for future use
B'110'	Reserved by IBM for future use
B'111'	Specialized station management

Figure 3-12. Allowable Token and Frame Priorities

A ring station can transmit a frame at a given priority using any available token with a priority less than or equal to that of the frame. If an appropriate token is not available, the ring station may reserve a token of the required priority in a passing token or frame as follows:

- If another ring station has reserved an equal or higher priority, the ring station cannot make a reservation in the frame or token.
- If the reservation bits have not been set, or if they have been set to a lower priority than that required by the ring station, it sets the reservation bits to its required priority.

When a ring station removes one of its frames from the ring and finds a non-zero value in the reservation bits, it must originate a non-zero priority token. The way the ring station determines the priority of the token is described in "Token Transmission Finite State Machine" on page 7-30, based on the priority used by the ring station for the recently transmitted frame, the reservation received in the returning frame, and any stored priority.

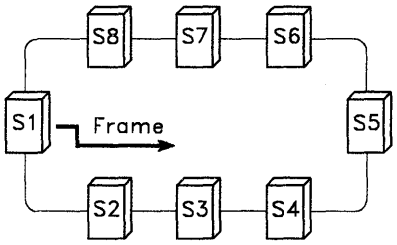
To prevent a ring station from continually transmitting priority frames (thereby keeping non-priority ring stations from transmitting), the IBM Token-Ring Network provides "fairness" within each priority. While a priority can be preempted at any time by a request for a higher priority, once the highest priority has been satisfied the priority reverts to a lower priority, and eventually to the normal priority.

That is, a ring station that originates a token of increased priority must eventually replace it with a token of the original priority. In this way the priority of originated tokens eventually returns to the normal priority. A ring station that is waiting to replace a high-priority token with a token of the original priority is in *priority-hold* state, because it is holding the original priority for eventual transmission.

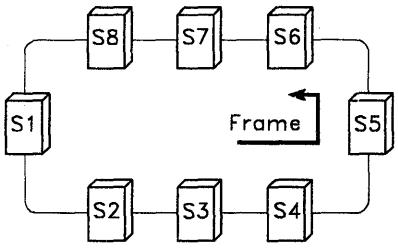
A ring station in priority-hold state must keep track of all priorities involved. Because of the manner in which reservations are made, a ring station must be able to hold up to four possible priority transitions. See "Finite State Machines for Priority Operation" on page 7-29.

For the ring station to have enough time to recognize and change the priority of a token, it introduces an 8-bit delay between its receiving and its transmitting components when it enters priority-hold state. When the ring station leaves priority-hold state, it removes this delay.

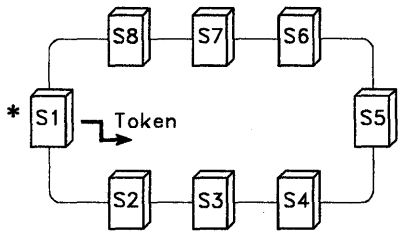
The following sequence shows how fairness within a priority is maintained:



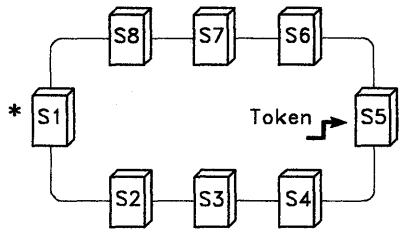
S1 transmits its frame at normal priority, using the received token.



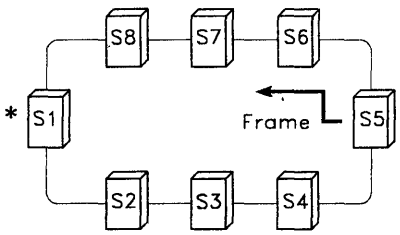
S5 reserves a higher priority in the passing frame.



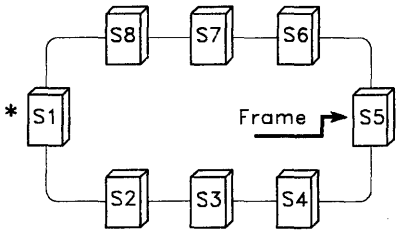
S1 removes its frame, originates a token of the priority that S5 reserved, and enters priority-hold state (denoted by *).



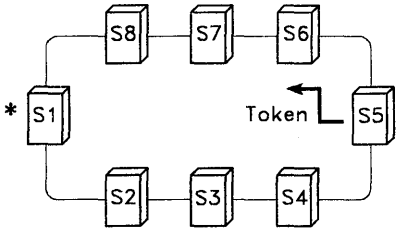
S2, S3, and S4 have no priority traffic, and the token makes its way to S5.



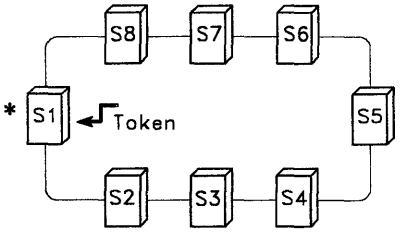
S5 transmits its priority frame.



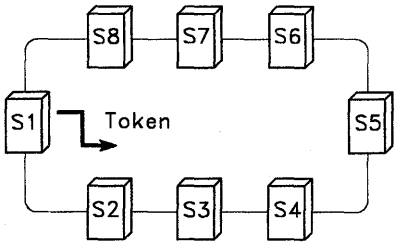
The frame returns to S5.



S5, having completed its transmission, issues a token of the priority it just used (the higher priority). S1, still in priority-hold, is awaiting a token of that priority (the priority that S5 requested, and that S1 originated).



S1 receives S5's token and recognizes the priority that it originated.



S1 leaves priority-hold state (assuming there was no new priority reservation) and originates a token at normal priority. If S2 had a normal priority frame queued, it would now be able to transmit the frame.

Duties of the Active Monitor

One ring station on each ring — called the *active monitor* — provides token-monitoring and other functions. Any operating ring station can be assigned the active monitor responsibility. Other ring stations act as *standby monitors*, prepared to take over if the active monitor fails (see "Duties of the Standby Monitor" on page 3-23). The active monitor function is part of the architecture of every ring station (see "Monitor Functions Finite State Machine" on page 7-25).

The active monitor resolves the following conditions on the ring:

- Lost tokens
- Frames and priority tokens that circle the ring more than once
- Other active monitors on the ring
- "Short" ring (a ring with such a low bit delay that it cannot hold a token)
- Clocking.

Several features, described in the following sections, help the active monitor perform its functions.

Maintaining the Master Clock

The active monitor maintains the ring's master clock, which controls timing and ensures that all other clocks on the ring are synchronized.

Ensuring Proper Ring Delay

To ensure that a token can be completely transmitted before returning to the originating ring station (overlapping), the active monitor introduces a 24-bit delay (the length of a token) into the ring. This guarantees proper delay on the ring.

Initiating Neighbor Notification

The active monitor periodically broadcasts the Active Monitor Present MAC frame to all ring stations on its ring, allowing each to acquire the address of its nearest active upstream neighbor (NAUN). (MAC frames are described in Chapter 5, "MAC Frames" on page 5-1.) The NAUN address is used during error isolation to determine if there is a failing component in a given ring station's fault domain; see Chapter 15, "Ring Error Monitor." For more information, see "Neighbor Notification Process" on page 3-26.

Monitoring Neighbor Notification

At any time during the neighbor notification cycle, an exception event can occur that could modify the progression of neighbor notification around the ring. For more information, see "Neighbor Notification Process" on page 3-26. These exceptions are described below.

- The T(neighbor_notification) timer has a comparatively long time-out (see page A-8). Each time it expires, the active monitor transmits an Active Monitor Present MAC frame. If the previous neighbor-notification process is not complete, the active monitor also transmits a Report Neighbor Notification Incomplete MAC frame to the ring error monitor (see "Report Neighbor Notification Incomplete MAC Frame, X'27'" on page 5-12 and Chapter 15, "Ring Error Monitor").

- The active monitor uses its T(receive_notification) timer (see page A-11) to ensure that an Active Monitor Present MAC frame circles the entire ring periodically. If this timer expires, the Active Monitor Present MAC frame did not circle the ring (possibly because a ring station is transmitting continuously, which is called *streaming*), and the active monitor initiates token-claiming.
- If the active monitor receives a Standby Monitor Present MAC frame after neighbor notification has been successfully completed, the frame is ignored.
- If the active monitor copies an Active Monitor Present MAC frame that contains a source address other than its own, an active monitor is already on the ring, and the active monitor that copied the frame deactivates its active monitor functions.
- A hard error can cause the suspension of the token protocol, and thus terminate the neighbor-notification process. The neighbor-notification process is initiated again by the active monitor after recovery from the hard error.
- If the ring is busy transmitting data (especially priority traffic), neighbor notification can be delayed or interrupted. The active monitor transmits a Report Neighbor Notification Incomplete MAC frame to the ring error monitor.

Monitoring Token and Frame Transmission

The monitor bit in the access control field is set to B'0' in every transmitted token and frame. When the active monitor repeats a frame or a non-zero priority token, it sets the monitor bit to B'1'. If the bit has already been set to B'1', the active monitor assumes that the token or frame has already circled the ring once (the ring station that originated the token or frame did not remove it). The active monitor purges the ring (see "Purging the Ring") and originates a new token.

Detecting Lost Tokens and Frames

The T(any_token) timer (see page A-2) has a relatively short time-out that exceeds the time required for the longest possible frame to circle the ring. The active monitor restarts this timer each time it repeats a starting delimiter. If T(any_token) expires, the active monitor assumes that the token or frame was lost on the ring. The active monitor purges the ring (see "Purging the Ring") and originates a new token.

Purging the Ring

The active monitor broadcasts the Ring Purge MAC frame to all ring stations on its ring before originating a new token. Receipt of the returned frame indicates to the active monitor that a frame can circle the ring without incident. The active monitor then restarts the token protocol.

The Ring Purge MAC frame resets the ring stations to Normal Repeat mode and cancels or restarts appropriate timers. When the Ring Purge MAC frame that indicates the end of token-claiming (see page 3-23) interrupts the ring stations, each station that is in Claim Token Repeat mode (that is, each station that is repeating received Claim Token MAC frames) takes the following actions:

- Cancels T(claim_token)
- Starts T(good_token) and T(receive_notification).

The active monitor ignores the value of the monitor bit in returning Ring Purge MAC frames. However, it does copy the reservation bits of the last returning Ring Purge MAC frame into the new token it originates.

An active monitor that is interrupted by a Ring Purge MAC frame that it did not originate assumes itself to be a duplicate monitor and becomes a standby monitor (by removing the 24-bit delay and master clock).

Duties of the Standby Monitor

Standby monitors detect failures in the active monitor and disruptions on the ring. Each standby monitor uses two token-protocol timers, T(good_token) and T(receive_notification).

The standby monitor restarts its T(good_token) timer (see page A-7) each time it repeats a priority zero token or a priority token greater than zero followed by a frame. This timer, with a duration longer than that of the active monitor's T(any_token), ensures that the active monitor's token management functions are active. If T(good_token) expires, the standby monitor initiates token-claiming (see page 3-23).

The standby monitor restarts its T(receive_notification) timer (see page A-11) each time it copies an Active Monitor Present MAC frame. This timer has a duration longer than that of the active monitor's T(neighbor_notification).

Expiration of T(receive_notification) indicates that the standby monitor has not received an Active Monitor Present MAC frame in an extended period. The standby monitor assumes that an active monitor is not present on the ring, or that the active monitor has malfunctioned and initiates token-claiming.

The Token-Claiming Process

Token-claiming is how ring stations elect an active monitor (if the current active monitor fails). Token-claiming starts whenever one of the following conditions occurs:

- The active monitor detects a loss of signal, detects expiration of its T(receive_notification), or cannot receive enough of its own Ring Purge MAC frames.
- A standby monitor detects a loss of signal, or detects expiration of its T(good_token) or T(receive_notification). If a standby monitor initiates token-claiming, the active monitor cannot participate; it must enter Claim Token Repeat mode, repeating received Claim Token MAC frames. This prevents a failing active monitor from continually reestablishing itself.
- A ring station attaches to the ring and does not detect an active monitor on the ring.

The ring station that detected the condition inserts its master clock and 24-bit delay. It enters Claim Token Transmit mode by broadcasting Claim Token MAC frames, paced at a specified interval (see "T(transmit_pacing)" on page A-15), to all ring stations on the ring. It broadcasts those frames without waiting for a token and follows them with idles (B'0's). The ring station then starts its T(claim_token), which specifies how long a ring station can remain in Claim Token Transmit or Claim Token Repeat mode. If T(claim_token) expires and the station is attached, the ring station enters Beacon Transmit mode. Otherwise, if T(claim_token) expires and the station is not attached, the station terminates the attachment process and removes itself from the ring. See "T(claim_token)" on page A-5.

Note: Every ring station contains an option that indicates whether it will participate in token-claiming. The default is not to participate. However, even a non-participating ring station must participate if it is the ring station that detects the condition for initiating token-claiming.

The other ring stations copy the Claim Token MAC frame. Each non-participating ring station then enters Claim Token Repeat mode and starts its T(claim_token).

Each participating ring station not already in the Claim Token process compares its individual address with the Claim Token MAC frame's source address:

- If the source address is greater than the ring station's individual address, the ring station enters Claim Token Repeat mode and starts its T(claim_token).
- If the source address is less than the ring station's individual address, the ring station transmits its own Claim Token MAC frame and starts its T(claim_token) and T(transmit_pacing) timers. When its T(transmit_pacing) timer expires, the ring station transmits another Claim Token MAC frame.

Each participating ring station already in Claim Token Transmit compares its individual address with the Claim Token MAC frame's source address:

- If the source address is greater than the ring station's individual address, the ring station transmits the received Claim Token MAC frame, removes its master clock and 24-bit delay, enters Claim Token Repeat mode, and restarts its T(claim_token).
- If the source address is less than the ring station's individual address, the ring station continues to transmit its own Claim Token MAC frame and restarts its T(claim_token) timer.
- If the source address is the same as the ring station's individual address, it continues broadcasting until it has transmitted three Claim Token MAC frames. When the station has received three of its own Claim Token MAC fames, the ring is viable and the ring station has won token-claiming. The ring station then cancels T(claim_token), purges the ring (which cancels all other T(claim_token) timers), starts its active monitor timers, originates a new token, and queues a Report New Active Monitor MAC frame to the configuration report server (see Chapter 16, "Configuration Report Server").

Below is an example of token-claiming. S1 through S8 are ring stations. The numbers in the figure indicate the events that occur.

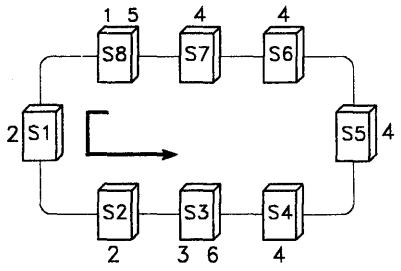


Figure 3-13. Example of Token-Claiming

1. S8's T(good_token) expires and S8 inserts master clock and 24-bit delay, begins broadcasting Claim Token MAC frames at a specified time interval (see "T(transmit_pacing)" on page A-15), and starts its T(claim_token).
2. S1 and S2 are not participants; each copies S8's Claim Token MAC frame, repeats S8's Claim Token MAC frame, enters Claim Token Repeat mode, and starts its T(claim_token).
3. S3 is a participant, with a higher address than S8. S3 inserts master clock and 24-bit delay, begins broadcasting its own Claim Token MAC frames, pacing them at a specified time interval (see "T(transmit_pacing)" on page A-15), enters Claim Token Transmit mode, and starts its T(claim_token).
4. S4, S5, S6, and S7 are not participants; each copies S3's Claim Token MAC frame, repeats S3's Claim Token MAC frame, enters Claim Token Repeat mode, and starts its T(claim_token).
5. S8 copies S3's Claim Token MAC frame, sees that the address of the received frame is higher than its own, stops transmitting its Claim Token MAC frame, removes master clock and 24-bit delay, retransmits S3's Claim Token MAC frame, enters Claim Token Repeat mode, and restarts its T(claim_token).
6. After S3 has transmitted three Claim Token MAC frames, S3 copies at least three of its own Claim Token MAC frames, indicating that S3 has won token-claiming. S3 cancels its T(claim_token), transmits Ring Purge MAC frames to reset the other ring stations, starts its active monitor functions, originates a new token, and queues a Report New Active Monitor MAC frame to the configuration report server.

Neighbor Notification Process

Neighbor notification begins when the active monitor broadcasts an Active Monitor Present MAC frame to all ring stations on its ring. At this time the active monitor also resets the "neighbor notification complete" flag to B'0', indicating that neighbor notification is in progress.

The first ring station that receives the Active Monitor Present MAC frame copies it and sets the address-recognized (A) and frame-copied (C) bits to B'1'. (These bits are described in "Frame Status Field" on page 2-14.) It then saves the source address field from the copied frame as its NAUN address (the address of the active monitor) and starts its notification-response timer. When its notification-response timer expires, it transmits a Standby Monitor Present MAC frame with the A and C bits set to B'0' to all ring stations on its ring.

If the frame is not copied, the neighbor-notification process is terminated. It will be started again by the active monitor when its neighbor-notification timer expires.

The next ring station downstream disregards the Active Monitor Present MAC frame (except to reset its T(receive_notification) timer), because the frame's A and C bits have already been set to B'1'. The ring station then copies its NAUN from the Standby Monitor Present MAC frame that was transmitted by its upstream neighbor, sets the A and C bits in that frame to B'1', and starts its notification-response timer. When its notification-response timer expires, it transmits its own Standby Monitor Present MAC frame with the A and C bits set to B'0'.

The next ring station downstream disregards the Active Monitor Present MAC frame (except to reset its T(receive_notification) timer), and the first ring station's Standby Monitor Present MAC frame, copies the second ring station's Standby Monitor Present MAC frame, sets that frame's A and C bits to B'1', and transmits its own Standby Monitor Present MAC frame with the A and C bits set to B'0'.

In this way neighbor notification proceeds around the ring, with other ring stations transmitting their Standby Monitor Present MAC frames, until the active monitor copies the last Standby Monitor Present MAC frame, in which the A and C bits are set to B'0'. The active monitor then sets the "neighbor notification complete" flag to B'1', indicating that neighbor notification has been successfully completed. Neighbor notification thus enables a ring station to learn its NAUN address, and to provide its address to its downstream neighbor.

If the active monitor copies its Active Monitor Present MAC frame and the A and C bits are still set to B'0', it assumes that it is the only station on the ring and sets the "neighbor notification complete" flag to B'1'.

Attaching to the Ring

The following sections describe the phases a ring station follows to attach to and function on a ring; for more details see "Station Attachment Finite State Machine" on page 7-21. It is assumed that each ring station has a pre-assigned address, which is unique across the IBM Token-Ring Network. Addresses are discussed in "Addresses" on page 3-9.

At the completion of these phases, the station activates its standby monitor functions if not elected active monitor in phase 1, and activates its wire fault error process.

Note: A ring station does not activate its standby monitor functions until it has completed the attachment process.

Phase 0: Lobe Test

Lobe testing is part of the attachment process, done before the ring station has attached to the ring. In this reference, a *lobe* is the section of cable that attaches a device to an access unit; the *access unit* allows the devices to access the ring from a central point (for example, in a wiring closet).

A lobe test consists of sending a series of Lobe Test MAC frames on the ring station's lobe only, to make sure there is not a fault in the lobe. If the frames traverse the lobe without fault, the ring station tests the receive logic by transmitting Duplicate Address Test MAC frames. If they are received correctly, the station attaches to the ring. Otherwise, the ring station terminates attachment with an error.

Following the power-on diagnostics and lobe test, a device containing a ring station is physically connected to the ring, and moves to Phase 1.

Phase 1: Monitor Check

The ring station starts the T(attach) timer (see page A-3).

- If the ring station receives an Active Monitor Present, Standby Monitor Present, or Ring Purge MAC frame before this timer expires, it assumes an active monitor is present on the ring and proceeds to Phase 2.
- If the ring station does not receive one of the above frames before the timer expires, either it is the first ring station on the ring, no active monitor is present on the ring, or the inserting station has broken the ring. The ring station initiates token-claiming (see page 3-23). (If this is the first ring station on the ring, it becomes the active monitor.)

During this phase, the ring station responds to inputs according to the "Station Attachment Finite State Machine" on page 7-21. In later phases, the ring station also responds to input conditions appropriate to its status as the active monitor, or not.

Phase 2: Duplicate Address Check

The ring station checks for the presence on its ring of another ring station with the same address using the Duplicate Address Test MAC frame (see page 5-11). If a duplicate address is found, the ring station removes itself from the ring.

Phase 3: Participation in Neighbor Notification

During this phase, the ring station participates in neighbor notification. This allows the station to learn its nearest active upstream neighbor's (NAUN) address and identify itself to its nearest active downstream neighbor (see page 3-20).

Before completing neighbor notification, if the ring station detects beaconing or the need for beaconing, it removes itself from the ring with an indication of beaconing. (See "Hard-Error Detection and Reporting" on page 3-30.) After neighbor notification is complete, the detection of beaconing or of the need for beaconing causes the ring station to begin beaconing *as if it had completed attaching to the ring*. Attachment is then continued after the ring recovers.

Phase 4: Request Initialization

The ring station requests changed operational parameters from a ring parameter server if one is present. If one is not present, default values are used.

A Request Initialization MAC frame sent to the ring parameter server includes registration information for the LAN manager. This information includes:

- The individual address of the ring station's NAUN
- The product instance ID of the attached product
- The ring station's microcode level.

If any of this information is incorrect or a threat to ring integrity, or if too many stations are already on the ring, the ring parameter server can notify the LAN manager. The LAN manager can request the configuration report server to issue a Remove Ring Station MAC frame, forcing the ring station to remove itself from the ring.

The ring parameter server responds with a Initialize Ring Station MAC frame. The parameters that can be set are Physical Location, Soft Error Report Timer Value, and Ring Number.

The ring parameter server serves as a focus to guarantee that ring stations on the ring have the same Soft Error Report Timer Values and Ring Numbers.

Soft-Error Detection and Reporting

Soft errors are intermittent faults that temporarily disrupt normal operation of the IBM Token-Ring Network; soft errors are normally tolerated by error recovery procedures. Soft errors are indicated by architectural inconsistencies (such as cyclic redundancy checks or time-outs) in received or repeated frames, and by a ring station's inability to process received frames. If soft errors result in degraded ring performance, the LAN manager can reconfigure the ring to bypass the faulty node.

Each ring station maintains a set of counters to measure the frequency of occurrence of the most critical soft errors (see "Isolating Error Counts, X'2D'" on page 5-19 and "Non-Isolating Error Counts, X'2E'" on page 5-20). When T(soft_error_report) expires, the ring station transmits a Report Soft Error MAC frame to the ring error monitor (see "Report Soft Error MAC Frame, X'29'" on page 5-13). After successfully transmitting the Report Soft Error MAC frame, the ring station resets these counters.

The Report Soft Error MAC frame reports the number of errors detected since the last report was made. This report identifies the transmitting ring station's NAUN and includes all the errors and their values, whether or not the appropriate error counter was incremented. This report may also contain a physical location ID for the ring station, if this information is available to the ring station.

Hard-Error Detection and Reporting

Hard errors are permanent faults, usually in equipment, that cause the ring to stop operating within the normal IBM Token-Ring Network architecture protocols. A ring station downstream from the hard fault recognizes a hard error at the receiver side of its attachment. The ring must be reconfigured to bypass the error. This restores the ring to an operational state. Repairs may be required to restore full operation to the ring.

When a ring station detects a failure of token-claiming following a hard error, it transmits Beacon MAC frames, pacing them at a specified time interval (see "T(transmit_pacing)" on page A-15), with an all-stations address to its ring only, until its input signal is restored, or until it removes itself from the ring. This is called *beaconing*. All other stations that receive the Beacon MAC frame enter Beacon Repeat mode. (See "MAC Recovery Finite State Machine" on page 7-13.)

The Beacon MAC frame identifies the beaconing ring station's NAUN and the type of error detected. It may also contain a physical location ID for the ring station, assuming that this information is available to the ring station.

When the beaconing ring station's NAUN has copied eight of these Beacon MAC frames, the NAUN removes itself from the ring and tests itself and its lobe, using the Lobe Media Test and Duplicate Address Test MAC frames (see page 5-11). If the test is successful, the NAUN reattaches to the ring without going through the normal attachment process. If the test fails, the NAUN remains unattached and notifies its DLC.LAN.MGR.

If the ring does not recover after a specified period (see "T(beacon_transmit)" on page A-4), the beaconing station assumes its NAUN has completed its self-test and removes itself from the ring and tests itself and its lobe, using the Lobe Media Test and Duplicate Address Test MAC frames. If the test is successful, the beaconing ring station reattaches to the ring without going through the attachment process. If the test fails, the beaconing ring station remains unattached and notifies its DLC.LAN.MGR.

If the ring does not recover after both the NAUN and the beaconing ring station have tested, the error cannot be repaired using automatic recovery; manual intervention is required. Network management is notified of the beaconing condition (see Chapter 15, "Ring Error Monitor").

Note: A ring station removes and tests itself only once during a ring recovery sequence.

Chapter 4. Operation of DLC.LAN.MGR

The following figure illustrates the component structure of the IBM Token-Ring Network in an SNA node:

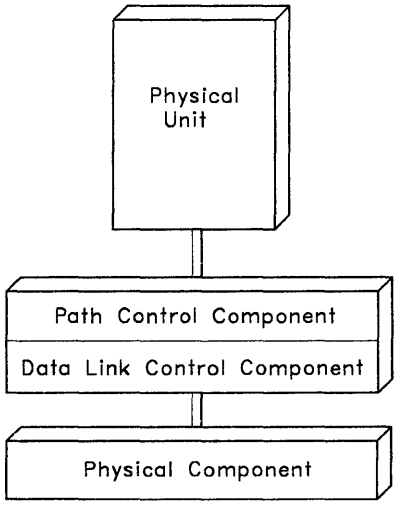


Figure 4-1. Component Structure for the IBM Token-Ring Network in an SNA Node

A Physical Unit manages and monitors the resources of links and link stations. Each SNA node contains a Physical Unit.

A Path Control component manages the sharing of these link resources and routes basic information units through the IBM Token-Ring Network. Each SNA node contains a Path Control component.

The Data Link Control and Physical components are described in "The Data Link Control Layer" on page 1-2 and "The Physical Layer" on page 1-5. Each SNA node and each node in an IBM Token-Ring Network contains a Data Link Control component and a physical component. This chapter describes in detail the operation of the manager function for the Data Link Control component, called DLC.LAN.MGR.

DLC.LAN.MGR activates and deactivates ring stations and link stations on command from the Physical Unit. DLC.LAN.MGR also manages information transfer between DLC.LAN and the Physical Unit. When DLC.LAN.MGR receives a record from the Physical Unit, it either acts on the record (if the record is for DLC.LAN.MGR) or passes it on to appropriate link stations. MAC frames intended for functions other than DLC.LAN.MGR (indicated in the Major Vector ID of the MAC frame) are routed by the DLC.LAN.MGR to the appropriate Physical Unit transaction programs.

In this chapter, a *record* is a collection of related data, treated as a unit; a *port* is the same as a ring station.

Physical Unit-to- DLC.LAN.MGR Records

An internal record interface is maintained between DLC.LAN.MGR and the Physical Unit. The records exchanged on this interface from the Physical Unit to DLC.LAN.MGR are described in this section. The records exchanged from DLC.LAN.MGR to the Physical Unit are described in the next section.

Note: The structural decompositions in this reference represent the meta-implementation of the IBM Token-Ring Network. They are meant to aid and guide implementers; they are not meant to restrict implementers if the meta-implementation is not well-suited to a given environment.

ACTIVATE_PORT

This record is used to map a single physical port to a specific Data Link Control component. ACTIVATE_PORT carries the following data:

- Port identifier
  A required field that accommodates a local identifier. This field identifies the specific ring station being activated.
- Any port parameters
  An optional field that contains a field identifier and length and parameter data. It also contains information used by DLC.LAN.MGR to initialize the port, including the following parameters:
  - individual_MAC_address (not required if ring station address is universally administered)
  - group_MAC_address (optional)
  - product_instance_ID (see page 5-21)
  - desired_functional_address_mask
- Any MAC parameters
  An optional field that contains a field identifier and length and parameter data. It contains information needed by DLC.LAN.MGR to create and initialize a MAC component tailored to the port.
- Any access channel control parameters
  An optional field that contains a field identifier and length and parameter data. It contains information needed by DLC.LAN.MGR to create and initialize the access channel control.
- Any DLC.LAN.MGR parameters
  An optional field that contains a field identifier and length and parameter data. It contains information needed by DLC.LAN.MGR to properly supervise DLC.LAN and manage information transfer from DLC.LAN to the Physical Unit.

ASSIGN_ALS_ADDRESS

This record carries the adjacent link station address assigned by the Physical Unit to the adjacent link station with which a connection has been established. An adjacent link station is a link station that is directly connected to a given node. The Physical Unit uses the same correlator it received in the ALS_CONNECTED_IN record from the DLC.LAN.MGR. The record contains correlator and adjacent link station address fields.

CONNECT_IN_ALS

This record informs DLC.LAN.MGR that a specified number of connection requests from adjacent link stations may be accepted through the access channel. It contains the following fields:

- Number of connection requests that may be accepted
- Link station parameters.

The link station parameter field is optional and contains a field identifier and length and parameter data. It also contains information needed by DLC.LAN.MGR to create and initialize the link station, including timer values, maximum send and receive window values (see "System Parameters" on page 11-3), and number of retries on error.

CONNECT_OUT_ALS

This record requests DLC.LAN.MGR to establish a connection to an adjacent link station. It contains the following fields:

- Adjacent link station identifier
- Signaling information
- Link station parameters.

The signaling information field is optional and contains a field identifier and length and signaling data. Signaling data includes the local ring station address, the remote ring station address and, optionally, the local and remote service access point (SAP) addresses to be used. In this reference, a service access point is the logical point at which a Path Control component acquires the services of the Data Link Control layer (see page 4-1). The default address for both local and remote SAPs is X'04'.

The link station parameter field is optional and contains a field identifier and length and parameter data. It also contains information needed by DLC.LAN.MGR to create and initialize the link station, including timer values, maximum send and receive window counts, and number of retries on error.

CONTACT_ALS

This record instructs DLC.LAN.MGR to send a SET_ABME record (see page 4-7) to a specified link station. It contains the following fields:

- Adjacent link station identifier
- Mode/role parameters
- Path control connection parameters.

The mode/role parameter field contains information regarding the operation of the local link station derived during the XID negotiation. Examples are operational mode, which specifies ABME, and station role, which is either primary or secondary (this determines which link station should send the SABME command LPDU).

The path control connection parameter field contains information regarding the connection to path control for the identified adjacent link station.

DISCONNECT_IN_ALS

This record instructs DLC.LAN.MGR not to accept any more connection requests through the access channel control.

DEACTIVATE_PORT

This record instructs the DLC.LAN.MGR to deactivate the port. Any currently active links will also be deactivated.

DEFINE_ALS

This record defines to DLC.LAN.MGR the adjacent link station that has just been connected. It contains an adjacent link station identifier and adjacent link station parameters.

The adjacent link station parameter field contains a field identifier and length and parameter data. It also contains information needed by DLC.LAN.MGR to tailor the link station, including timer values, maximum send and receive window counts, counter thresholds, and an optional access priority.

DISCONNECT_ALS

This record instructs DLC.LAN.MGR to terminate link-level contact with the specified adjacent link station, and to deactivate the connection to the adjacent link station. It contains an adjacent link station identifier field.

SEND_XID

This record instructs DLC.LAN.MGR to exchange identification with the adjacent link station. It contains an adjacent link station identifier and an XID information field. The XID information field contains a field identifier and length and information field data.

DLC.LAN.MGR-to- Physical Unit Records

This section describes the records exchanged on the internal record interface from DLC.LAN.MGR to the Physical Unit.

ALS_CONNECTED_OUT

This record informs the Physical Unit that the requested connection to an adjacent link station has been established. It contains an adjacent link station identifier field.

ALS_CONNECTED_IN

This record informs the Physical Unit that a connection to an adjacent link station has been established. It contains a correlator field, used by DLC.LAN.MGR to correlate the adjacent link station address sent in reply by the Physical Unit (in ASSIGN_ALS_ADDRESS) with the connected adjacent link station.

ALS_CONTACTED

This record informs the Physical Unit that an SABME/UA exchange with an adjacent link station has occurred. It contains an adjacent link station identifier field.

ALS_DEFINED

This record returns the link station process identifier to the Physical Unit so that the link between a path control component and the link station can be established. It contains an adjacent link station identifier and a link station process identifier.

ALS_DISCONNECTED

This record informs the Physical Unit that link-level contact with an adjacent link station has been terminated, and that the connection to the adjacent link station has been deactivated. It contains an adjacent link station identifier field.

INOPERATIVE

This record informs the Physical Unit that a failure has occurred, and identifies the affected resources. It contains the following fields:

- Port identifier of affected ports
- Adjacent link station identifiers of adjacent link stations to which connectivity has been lost
- Reason code for the failure.

The port identifier is present only when a port failure has occurred. Reasons for port failures include internal errors, receiver exceptions, wire faults, beaconing, and removed stations. Reasons for link failures include the N2 retry count (see page 11-4) reaching its limit, and receipt of a DM response, FRMR response, or DISC command LPDU.

PORT_ACTIVATED

This record informs the Physical Unit that a specific port has been activated as part of the access channel control. It contains a port identifier and an individual_MAC_address parameter (if provided by the ring station).

PORT_DEACTIVATED

This record informs the Physical Unit that a specific port has been deactivated and is no longer a part of the access channel control. It contains a port identifier field.

RCV_XID

This record indicates to the Physical Unit that an XID LPDU has been received from an adjacent link station. It contains an adjacent link station identifier and an XID information field received from the adjacent link station. The XID information field contains a field identifier and length and information field data.

RCV_SET_MODE

This record informs the Physical Unit that an SABME command LPDU has been received from the adjacent link station. It contains an adjacent link station identifier.

DLC.LAN.MGR-to- Link Station Records

This section describes the records that DLC.LAN.MGR sends to link stations to establish and terminate links, and to activate and deactivate data transfer.

- ACTIVATE_LS(ALS_identifier)
  Creates the link station and initializes it to asynchronous disconnected mode.
- DEACTIVATE_LS(ALS_identifier)
  Terminates the link station and removes any resource allocation associated with it.
- SEND_XID(ALS_identifier, xid_info_field)
  Causes the link station to send an XID command LPDU, with the P bit set to B'1', at the next opportunity to send a poll, or to send an XID response LPDU, with the F bit set to the value of the P bit in the previously received XID command LPDU.
- TEST_LINK(ALS_identifier, test_info_field)
  Causes the link station to send a TEST command LPDU, with the P bit set to B'1' and with test_info_field in the information field.
- SET_ABME(ALS_identifier)
  Causes the link station to send the SABME command LPDU, with the P bit set to B'1' to respond to a received SABME command LPDU with the UA response LPDU.
- SET_ADM(ALS_identifier)
  Causes the link station to send a DISC command LPDU, with the P bit set to B'1', or a DM response LPDU to a received DISC command LPDU as appropriate.
- SET_PARAMETERS(ALS_identifier, link_station_parameters)
  Used by DLC.LAN.MGR to pass link station parameters to the link station. (See "DEFINE_ALS" on page 4-4.)
- ATTACH_PC(ALS_identifier, path_ctl_conn_parameters)
  Tells the link station the internal reference (if different from the ALS identifier) for its associated path control element, so that the link station can route received BTUs to the proper destination.

Link Station-to- DLC.LAN.MGR Records

This section describes the responses that are sent from link stations to DLC.LAN.MGR.

- XID_RECEIVED(ALS_identifier, xid_info_field)
  Signals the receipt of an XID command or response LPDU.
- SABME_RECEIVED(ALS_identifier)
  Indicates that the link station has received an SABME command LPDU from the adjacent link station.
- CONTACTED(ALS_identifier)
  Indicates that the link station has successfully established a link with the remote link station. CONTACTED is sent to DLC.LAN.MGR when one of the following occurs:
  - The link station has sent SABME and received UA.
  - The link station has sent UA in response to an SABME *and* has subsequently received either an I- or S-format (RR or RNR) LPDU from the adjacent link station.
- LINK_TESTED(ALS_identifier, test_info_field)
  Contains the response to a previously issued TEST_LINK record.
- DISC_RECEIVED(ALS_identifier):
  Indicates that the link station has received a DISC command LPDU from the adjacent link station.
- FRMR_RECEIVED(ALS_identifier, frmr_info_field)
  Signals that a frame reject error condition has been established by the remote link station. The information field indicates the cause of the exception condition.
- DISCONNECTED(ALS_identifier)
  Indicates that the link station has successfully terminated the link with the remote link station. DISCONNECTED is sent to DLC.LAN.MGR after the link station has sent a DISC command LPDU and received a UA or DM response LPDU from the remote link station.
- INOPERATIVE(ALS_identifier, reason_code, link_station_status)
  Signals that the link station considers the link inoperative. Reason_code gives the specific cause. Link_station_status indicates whether the local link station was in a busy or REJ condition; whether the remote station was in a busy condition; the values of the send, receive, and last_received_Nr state variables; and the number of times T1 expired (see page 11-3).

DLC.LAN.MGR-to- Access Channel Control Records

Communication with the access channel control consists of records to attach and detach link stations (that is, to add or delete link stations from the access channel control's routing table), and to modify routing information for link stations already known to the access channel control.

This section describes the records sent by DLC.LAN.MGR to the access channel.

Note: The primitives shown apply only to connection-oriented service. Primitives for connectionless service must be provided by the implementer.

- ATTACH_LS(ALS_identifier, remote_MAC_address, local_MAC_address, remote_SAP_value, local_SAP_value, routing_information, svc_class)
  Requests the access channel control to add the specified link station to its routing table.

  The ALS identifier is used as the internal reference between the link station and other sub-components.

  The local and remote MAC addresses and SAP addresses provide the access channel control with the information needed to map a link station to a port.

  The routing_information parameter contains the routing information.

  The service class indicates the priority of the connection (if the underlying MAC supports different levels).

- DETACH_LS(ALS_identifier)
  Requests the access channel control to delete the specified link station from its routing table.

- MODIFY_LS(ALS_identifier, routing_information)
  Used to pass new routing information to the access channel control for a link station already contained in the access channel control's routing table. This primitive can only be issued during ADM.

Access Channel Control-to- DLC.LAN.MGR Records

This section describes the responses sent from the access channel control to DLC.LAN.MGR.

Note: The primitives shown apply only to connection-oriented service. Primitives for connectionless service must be provided by the implementer.

- LS_ATTACHED(ALS_identifier)
  Indicates that the specified link station has been added to the access channel control's routing table.
- LS_DETACHED(ALS_identifier)
  Indicates that the specified link station has been deleted from the access channel control's routing table.
- LS_MODIFIED(ALS_identifier, new_routing_information, old_routing_information)
  Notifies DLC.LAN.MGR that new routing information has replaced the previous routing information used for the specified link station. Both the new and old routing information fields are given to DLC.LAN.MGR.

MAC-to- DLC.LAN.MGR Services

This section describes the services provided at the boundary between DLC.LAN.MGR and the MAC sub-layer. DLC.LAN.MGR uses this boundary to monitor and control the operations of the MAC sub-layer.

The primitives listed below enable DLC.LAN.MGR to request service from the MAC sub-layer. (In this reference, a *primitive* is an abstract, implementation-independent interaction between a user of a service and the provider of the service.)

- REQUEST_MAC_CONFIGURE
- CONFIRM_MAC_CONFIGURE
- REQUEST_MAC_CONTROL
- CONFIRM_MAC_CONTROL
- MAC_STATUS_INDICATION
- SEND_MAC_DATA
- RECEIVE_MAC_DATA
- CONFIRM_MAC_DATA
- REQUEST_MAC_VALUES
- REPORT_MAC_VALUES.

Each service names the particular primitive and the information passed between the MAC sub-layer and DLC.LAN.MGR.

REQUEST_MAC_ CONFIGURE

DLC.LAN.MGR uses this primitive to change the operational parameters of the MAC sub-layer.

Semantics of the Service Primitive

```
REQUEST_MAC_CONFIGURE  (
  individual_MAC_address,
 *group_MAC_addresses,
 *functional_address_mask,
  product_instance_ID,
  participant_option,
  pass_all_MAC_frames,
  indicate_for_rcv_beacon_frames,
  indicate_for_rcv_attention_frames,
  )
```

Note: For this primitive, those parameters marked with an asterisk (*) can be set at any time. All others must be set before the MAC has been activated and the ring station attaches to the ring.

The individual_MAC_address parameter is the byte string used by the MAC sub-layer as its individual address. Its characteristics are:

- Bit 0 of byte 0 is set to B'0' (individual address).
- It is *not* X'0000 0000 0000'.

This parameter can be set only before the MAC is activated.

The group_MAC_addresses parameter is the set of byte strings used by the MAC sub-layer as its group MAC addresses (not functional addresses). Its characteristics are:

- Bit 0 of byte 0 is set to B'1' (group address).
- Bit 1 of byte 0 is set to B'1' (local administration).
- Bit 0 of byte 2 is set to B'1' (group address).
- It is *not* X'C000 FFFF FFFF' or X'FFFF FFFF FFFF'.

The default is not to recognize any group address.

The functional_address_mask parameter, which can be set at any time, is the byte string used by the MAC sub-layer to determine which destination functional addresses it may copy. Only the MAC sub-layer can enable and disable the active monitor functional address. Ring stations can request any functional addresses at initialization except those listed below:

- Active monitor (may not be requested across this protocol boundary)
- Ring parameter server.

The product_instance_ID parameter is the byte string used by the MAC sub-layer to report the identity of the product within which it resides. It has a maximum length of 18 bytes. In order for the product_instance to be valid, it must be supplied during initialization. This parameter can be set only before the MAC is activated. (For the format of the Product Instance ID see page 5-21.)

The participant_option parameter is to designate whether the ring station will participate in token-claiming. If the ring station chooses not to participate, then it will enter token-claiming only if it detects an error in the active monitor on the ring. The default is not to participate. This parameter can be set anytime.

The pass_all_MAC_frames parameter is set by DLC.LAN.MGR to indicate to the MAC layer to pass all MAC frames not recognized by it or which do not fit into one buffer to DLC.LAN.MGR for further routing. This parameter can be set anytime.

The indicate_for_rcv_beacon_frames parameter is the value the MAC sub-layer uses to decide whether to generate RECEIVE_MAC_DATA primitives for received Beacon MAC frames with source address or beacon type change. The default is not to receive Beacon MAC frames. This parameter can be set anytime.

The indicate_for_rcv_attention_frames parameter is the value the MAC sub-layer uses to decide whether to generate RECEIVE_MAC_DATA primitives for received attention MAC frames. This indication occurs only if the control bits are not equal to those of the last attention frame received. An attention MAC frame is a frame in which the Control Bits (ZZZZ) in the frame control field have a value greater than X'1'. The default is not to receive attention MAC frames. This parameter can be set anytime.

Note: All parameters of this primitive are optional. If a parameter is omitted, the MAC sub-layer uses the most recently provided value for this parameter or, if no value has been previously provided, the default value for the parameter is used. The default value for the individual_MAC_address parameter is not defined here.

When Generated

This primitive is generated by DLC.LAN.MGR, whenever DLC.LAN.MGR requires that the MAC sub-layer be reconfigured. This primitive can be issued at any time; in most cases the MAC sub-layer need not be activated (see "MAC Activation and Deactivation" on page 5-27).

Effect On Receipt

Receipt of this primitive causes the MAC sub-layer to reset its protocol and establish the values of its addresses and other initialization parameters. The individual_MAC_address, product_instance_ID, participant_option, pass_all_MAC_frames, indicate_for_rcv_beacon_frames, and indicate_for_rcv_attention_frames parameters cannot be changed if the MAC is activated. Upon completion of this primitive, the MAC sub-layer generates a CONFIRM_MAC_CONFIGURE.

CONFIRM_MAC_ CONFIGURE

This primitive is used by the MAC sub-layer to inform DLC.LAN.MGR that the REQUEST_MAC_CONFIGURE primitive is complete.

Semantics of the Service Primitive

```
CONFIRM_MAC_CONFIGURE  (
  status
  )
```

The status parameter indicates the success or failure of the REQUEST_MAC_CONFIGURE. If success is indicated, this parameter has the value NORMAL_COMPLETION.

If failure is indicated, this parameter has one of the following values:

- NO_INDIVIDUAL_MAC_ADDRESS — the individual_MAC_address parameter was not specified, and one was required.
- INVALID_INDIVIDUAL_MAC_ADDRESS — either the individual_MAC_address parameter was all B'0's, or it did not begin with a B'0'.
- INVALID_GROUP_MAC_ADDRESS — the group_MAC_address parameter was all B'1's, or it did not begin with a B'1', or it was X'C000 FFFF FFFF' (an invalid group MAC address).
- INVALID_PRODUCT_INSTANCE_ID — the product_instance_id parameter was an invalid length.
- INVALID_FUNCTIONAL_ADDRESS_REQUESTED.

When Generated

This primitive is generated by the MAC sub-layer upon completion of a REQUEST_MAC_CONFIGURE.

Effect On Receipt

The effect of this primitive on DLC.LAN.MGR is unspecified.

REQUEST_MAC_CONTROL

DLC.LAN.MGR uses this primitive to cause the MAC sub-layer to attach itself to and remove itself from the ring.

Semantics of the Service Primitive

```
REQUEST_MAC_CONTROL  (
  control_action
  )
```

The control_action parameter is one of the following:

- MASTER_RESET — causes the ring station to unconditionally remove itself from the ring.
- INSERT — causes the ring station to attach to the ring. Failure to attach is indicated by the CONFIRM_MAC_CONTROL primitive.

When Generated

This primitive is generated by DLC.LAN.MGR whenever it requires the MAC sub-layer to take one of the above control actions.

Effect On Receipt

Receipt of this primitive causes the MAC sub-layer to take the action specified.

CONFIRM_MAC_CONTROL

The MAC sub-layer uses this primitive to inform DLC.LAN.MGR of the results of a REQUEST_MAC_CONTROL.

Semantics of the Service Primitive

```
CONFIRM_MAC_CONTROL  (
  status
  )
```

The status parameter indicates the success or failure of the REQUEST_MAC_CONTROL. If success is indicated, this parameter has the value NORMAL_COMPLETION.

If failure is indicated, this parameter has one of the following values:

- MAC_NOT_ACTIVATED — MASTER RESET was attempted, but the MAC sub-layer was not active.
- INTERNAL ERROR — an internal error occurred during execution of the REQUEST_MAC_CONTROL.
- REMOVE_RECEIVED
- TRANSMISSION_MEDIUM_ERROR (phase, type) — a cable error occurred during MAC activation. Phase defines when the transmission medium failure occurred:
  - DISCONNECTED
  - MONITOR_CHECK
  - DUPLICATE_ADDRESS_CHECK
  - NEIGHBOR_NOTIFICATION
  - REQUEST_PARAMETERS

  Type defines the error:
  - FUNCTION_FAILURE
  - RECEIVER_EXCEPTION (signal loss)
  - FREQUENCY_ERROR
  - WIRE_FAULT

- TIME-OUT_ERROR (T(attach) expires during ADDRESS_VERIFICATION, NEIGHBOR_NOTIFICATION, or REQUEST_PARAMETERS phases)
- MONITOR_PURGE_FAILURE
- BEACONING_OCCURRING
- DUPLICATE_INDIVIDUAL_MAC_ADDRESS
- PARAMETER_REQUEST (time-out on receiving Set Parameters, when one was expected).

When Generated

This primitive is generated by MAC upon completion of a REQUEST_MAC_CONTROL.

Effect On Receipt

The effect of this primitive on DLC.LAN.MGR is unspecified.

MAC_STATUS_ INDICATION

The MAC sub-layer uses this primitive to inform DLC.LAN.MGR of errors and significant status changes. This primitive can occur only after attachment to the ring.

Semantics of the Service Primitive

```
MAC_STATUS_INDICATION  (
  status_report
  )
```

The status_report parameter is one of the following:

- FRAME_GOOD — a good frame was repeated or copied.
- FRAME_BAD — the Error-Detected (E) bit was set on in a bad frame.
- CLAIM_TOKEN — token-claiming transmit state has been entered.
- ACTIVE_MONITOR — the ring station has assumed the active monitor role.
- STANDBY_MONITOR — the standby monitor role has been assumed.
- DUPLICATE_ADDRESS_ENCOUNTERED — a duplicate individual MAC address has been encountered.
- BEACONING_CHANGED.
- BEACONING_STOPPED.
- SET_PARAMETERS_MAC_FRAME_RECEIVED.
- INOPERATIVE
  - INTERNAL_ERROR
  - RECEIVER_EXCEPTION — no received signal transitions on the medium
  - WIRE_FAULT
  - FREQUENCY_ERROR
  - BEACONING_OCCURRING
    - BEACON_FRAME_RECEIVED, not Self_beaconing
    - BEACON_FRAME_RECEIVED, Self_beaconing
    - SELF_BEACONING, not Beacon_frame_received
  - REMOVE_RECEIVED (removal_code) — a remove_ring_station MAC frame was received
  - AUTO_REMOVE_FAULT — a ring station that caused the ring to enter beaconing removed itself from the ring, tested, detected an error, and remained off the ring
  - SINGLE_STATION_INDICATOR — the ring station detects that it is the only ring station on the ring.

When Generated

This primitive is generated by the MAC sub-layer when one of the reported conditions occurs.

Effect On Receipt

The effect of this primitive on DLC.LAN.MGR is unspecified.

SEND_MAC_DATA

This primitive defines the transfer of data from a local DLC.LAN.MGR to the local MAC sub-layer.

Semantics of the Service Primitive

```
SEND_MAC_DATA  (
  frame_control_field,
  destination_address,
  routing_information,
  MSDU,
  requested_service_class
  )
```

The frame_control_field parameter specifies the value for the frame's frame control field. This parameter has the following characteristics:

- Bits 0 and 1 of byte 0 are each set to B'0'. (indicating a MAC frame)
- Bits 4 through 7 have a value between B'0000' and B'0110' their value is a valid combination with the MAC command value.

The destination_address parameter may specify either an individual or a group MAC address. It contains sufficient information to create the destination address field that is appended to the frame by the local MAC sub-layer, as well as any lower-level address information. The Remove Ring Station and Response MAC frames cannot have an all-stations broadcast destination address. (See "MAC Frames That Cannot be Sent to All Stations" on page 5-25.)

The routing_information parameter specifies the frame routing information, if the frame is targeted to an off-ring destination.

The MSDU parameter specifies the MAC service data unit to be transmitted by the MAC sub-layer. There is sufficient information associated with MSDU for the MAC sub-layer to determine the length of the data unit. The source class value within MSDU is examined by the MAC sub-layer to ensure that the source class does not equal X'0', and that it is not restricted by the current source_class_vector.

The requested_service_class parameter specifies the priority desired for the data unit transfer.

Note: Certain MAC frames generated by the MAC sub-layer itself (for example, Active Monitor Present) may be transmitted with a priority that exceeds the maximum transmit priority allowed for frames passed to the MAC sub-layer by the LLC sub-layer or DLC.LAN.MGR.

When Generated

This primitive is generated by DLC.LAN.MGR whenever data must be transferred to one or more peer DLC.LAN.MGRs.

Effect on Receipt

Receipt of this primitive causes the MAC sub-layer to append all MAC-specific fields, including the destination and source addresses and any fields that are unique to the particular medium access method, and to pass the properly formed frame to the lower layers of protocol for transfer to the peer DLC.LAN.MGRs.

RECEIVE_MAC_DATA

This primitive defines the transfer of data from the MAC sub-layer to DLC.LAN.MGR.

Semantics of the Service Primitive

```
RECEIVE_MAC_DATA  (
  frame_control_field,
  destination_address,
  source_address,
  routing_information,
  MSDU,
  reception_status
  )
```

The frame_control_field parameter is the frame control field received.

The destination_address parameter may be either an individual or a group MAC address, as specified by the destination address field of the received frame.

The source_address parameter must be an individual address as specified by the source address field of the received frame.

The routing_information parameter specifies the routing information contained in the received frame.

The MSDU parameter specifies the MAC service data unit as received by the local MAC sub-layer.

The reception_status parameter indicates the success or failure of the received frame. It consists of the following elements:

- frame_status: FR_GOOD
- E_value: B'0', B'1'.
- A_&_C_value: B'00', B'01', B'10', B'11'.

When Generated

The RECEIVE_MAC_DATA primitive is generated by the MAC sub-layer to one or more DLC.LAN.MGRs to indicate the arrival of a MAC frame. The MAC information field is reported only if it is validly formed and received without error, and if the destination address designates the local MAC sub-layer.

Effect on Receipt

The effect of this primitive on DLC.LAN.MGR depends upon the validity and content of the frame.

Additional Comments

If the local MAC sub-layer is designated by the destination_address parameter of an SEND_MAC_DATA primitive, the RECEIVE_MAC_DATA primitive is also invoked by the MAC sub-layer to the local DLC.LAN.MGR. This duplex characteristic of the MAC sub-layer may be due to unique function capabilities within the MAC sub-layer, or to duplex characteristics of the lower layers. For example, frames transmitted to the broadcast address invoke RECEIVE_MAC_DATA primitives at all ring stations in the IBM Token-Ring Network, including the ring station that originated the request.

CONFIRM_MAC_DATA

This primitive responds to SEND_MAC_DATA, signaling success or failure.

Semantics of the Service Primitive

```
CONFIRM_MAC_DATA  (
  transmission_status,
  provided_service_class
  )
```

The transmission_status parameter passes status information back to the local requesting DLC.LAN.MGR. It indicates the success or failure of the associated SEND_MAC_DATA. If success is indicated, this parameter has the value PDU_ACCEPTED.

The failure reason codes indicate an error that causes the MAC sub-layer to fail to queue the frame for transmission. If failure is indicated, this parameter has one of the following values:

- MAC_NOT_ACTIVATED
- INVALID_FRAME_CONTROL_FIELD_VALUE (bit values not allowed)
- INVALID_DESTINATION_ADDRESS
  - Improperly formed
- INVALID_ROUTING_INFORMATION
  - Less than 2 bytes
  - Greater than 18 bytes
  - Improperly formed (odd length)
- INVALID_MSDU
  - Greater than the maximum allowed length
  - Not authorized to transmit using this source class
  - Invalid source/destination class combination
- INVALID_REQUESTED_SERVICE_CLASS
  - Priority value not allowed.

The provided_service_class parameter specifies the priority provided for the data unit transfer.

When Generated

This primitive is generated by the MAC sub-layer in response to a SEND_MAC_DATA from the local DLC.LAN.MGR.

Effect on Receipt

The effect of this primitive on the DLC.LAN.MGR is unspecified.

Additional Comments

It is assumed that sufficient information is available to DLC.LAN.MGR to associate the response with the appropriate request. For example, the association may be implied by the order of the responses, because the MAC sub-layer requires that the requests be serviced in a first-in, first-out manner. This does not preclude MAC from buffering more than one data unit for transmission.

REQUEST_MAC_ VALUES

DLC.LAN.MGR uses this primitive to request the values of the current MAC sub-layer operational information.

Semantics of the Service Primitive

```
REQUEST_MAC_VALUES  (
  individual_MAC_address
  )
```

The individual_MAC_address parameter specifies which local MAC sub-layer is to transfer its operational information to DLC.LAN.MGR in a REPORT_MAC_VALUES primitive.

When Generated

This primitive is generated when desired by DLC.LAN.MGR.

Effect on Receipt

The REPORT_MAC_VALUES primitive is generated.

REPORT_MAC_ VALUES

This primitive defines the transfer of operational information from the local MAC sub-layer to DLC.LAN.MGR.

Semantics of the Service Primitive

```
REPORT_MAC_VALUES  (
  operational_values
  )
```

The operational_values parameter contains the current values used by the MAC sub-layer, including:

- Individual_MAC_address
- Group_MAC_address
- Source_class_vector
- Functional_address_mask
- NAUN

When Generated

This primitive is generated by the MAC sub-layer when it receives a REQUEST_MAC_VALUES primitive.

Effect on Receipt

The effect of this primitive on DLC.LAN.MGR is unspecified.

Part 2. The Medium Access Control (MAC) Sub-Layer

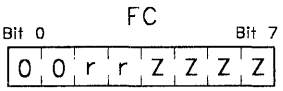
Chapter 5. MAC Frames

MAC frames are originated, received, and acted upon by ring stations. These frames control the operation of the IBM Token-Ring Network and any ring station operations that affect the ring.

MAC Frame Format

The overall MAC frame format is the same as that given in Chapter 2, "MAC Frame Format" on page 2-1.

However, all MAC frames have the same frame control field format, with bits 0 and 1 set to B'0', as indicated below:



r = Reserved Bits
Z = Control Bits

Figure 5-1. MAC Frame Control Field

The reserved bits are reserved by IBM for future use. They are transmitted as B'0's; their value is ignored by receiving ring stations.

The control bits for a MAC frame are described in detail in "Express Buffer and Other Frame Control Field Values" on page 5-24.

The format of the MAC frame information field, also called a major vector, is shown below:

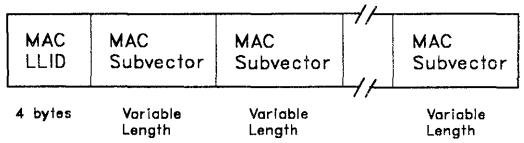


Figure 5-2. MAC Information Field

The major vector consists of a MAC length and ID (LLID) and 0, 1, or more MAC subvectors.

The format of the MAC LLID is shown below:

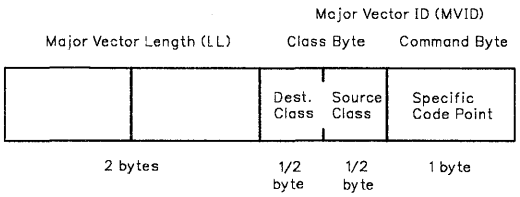


Figure 5-3. MAC LLID Format

Major Vector Length

The Major Vector Length (LL) is a 2-byte field that gives the length, in bytes, of this specific MAC major vector, including the Major Vector Length field itself.

Major Vector ID

The Major Vector ID (MVID) is a 2 byte field that identifies the function that this major vector is to perform. The Major Vector ID is divided into two sub-fields, Function Class and Command. The Command sub-field is a single byte that uniquely identifies the type of MAC frame.

The Function Class sub-field is further divided into two 4-bit fields that define the destination and source function classes for the MAC frame.

The destination and source fields provide two functions:

1. They provide a way to route a received MAC frame to the desired handling function. For example, if the destination class field specifies Ring Station, then the ring station parses the frame, instead of passing it to DLC.LAN.MGR. If the destination class field is DLC.LAN.MGR, then the ring station immediately passes the frame to DLC.LAN.MGR, without parsing it.

   If the destination class is not Ring Station, then a router located in DLC.LAN.MGR must route the frame to the proper function class manager for that function class. For example, if the destination class is configuration report server, then the router must pass the frame to the configuration report server for parsing.

   Some MAC frames are destined for the Ring Station function class, but are not actually handled by ring stations. In these cases, the ring station receives the MAC frame, sees that it is destined for the ring station, parses the frame but sees that the remainder of the major vector identifier is unrecognized, and passes the frame to the DLC.LAN.MGR router. The router must handle these frames also. Whether an unknown MAC frame is transferred to the DLC.LAN.MGR router depends on the destination function class or on the optional setting of pass_all_MAC_frames (see page 4-12).

2. They provide a way to filter MAC frames that are built and *sent* by an attached product, using the source class field. They also provide a way to filter MAC frames received from the ring.

Each ring station contains a 2-byte mask specifying which of the 16-source function classes the attached product may use. For example, if the mask says that the attached product may not send configuration report server-class MAC frames, but the attached product builds and tries to send such a frame (such as Remove Ring Station), then the ring station will reject the frame without sending it.

Destination and source classes are listed below:

Class Value	Function Class
X'0'	Ring station
X'1'	DLC.LAN.MGR
X'4'	Configuration Report Server
X'5'	Ring Parameter Server
X'6'	Ring Error Monitor

Figure 5-4. Destination and Source Function Classes

Ring stations and DLC.LAN.MGR are covered in Chapter 3, "Token-Ring Concepts" on page 3-1; the following sections describe the other function classes.

Configuration Report Server

The configuration report server is a network management function that resides on every ring in a multiple-ring environment for which stations are to be managed. It serves four purposes for the IBM Token-Ring Network:

- It collects configuration information from the ring (Report NAUN Change and Report New Monitor MAC frames) and reports this information to the LAN manager. This assures that the LAN manager always has complete and accurate ring configuration information for all rings.
- It can request status information from stations on its local ring as requested by the LAN manager.
- It can set the values of operational parameters for stations on its local ring as directed by the LAN manager.
- It can change the configuration of its local ring by requesting a station to remove itself from the ring as directed by the LAN manager.

Ring Parameter Server

The ring parameter server is a network management function that resides on every ring in which the operational parameters are centrally managed. It serves two purposes for the IBM Token-Ring Network:

- It is the target for all Request Initialization MAC frames that are sent by ring stations during attachment to the ring. This allows a ring station to send the frame to a known address (that is, the ring parameter server functional address) on its own ring only, without having to broadcast on all the other rings. The ring parameter server sends this registration information to LAN manager.
- It makes the following parameters readily available to all ring stations on the ring (using the Initialize Ring Station MAC frame):
  - Ring Number
  - Ring Station Soft Error Report Timer Value
  - Physical Location.

  This guarantees that the Ring Number and Ring Station Soft Error Report Timer Value are the same for all ring stations on the ring.

Ring Error Monitor

The Ring Error Monitor provides three functions:

- It collects error reports from stations on the attached ring. These include the "Report Neighbor Notification Incomplete MAC Frame, X'27'" on page 5-12, "Report Active Monitor Error MAC Frame, X'28'" on page 5-11, and "Report Soft Error MAC Frame, X'29'" on page 5-13.
- It further analyzes the soft error reports and when thresholds are exceeded, reports the fault domain and the error condition to the LAN manager.
- It forwards the other reports received from stations on the ring to the LAN manager.

This server is present on rings for which errors are to be monitored or analyzed. Its functional address is the destination address for error reports generated by ring stations.

Subvector Format

A subvector has the format shown below:

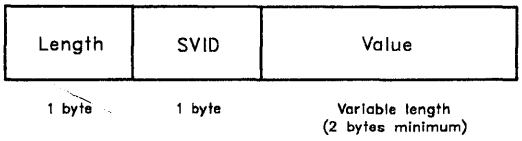
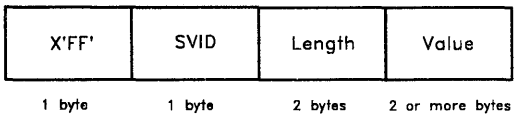


Figure 5-5. MAC Subvector

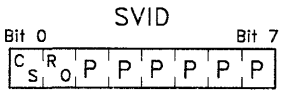
Subvector Length

This is a 1-byte field indicating the length, in bytes, of the MAC subvector. A length of X'FF' means that the vector is longer than 254 bytes and the format of the entire subvector becomes:



Subvector Type (SVID)

The subvector type code of the MAC subvector (SVID) is a single byte with the following format:



C/S = Common/Specific Indicator Bit
R/O = Required/Optional Indicator Bit
P = Code Point Bit

Figure 5-6. SVID Format

- Bit 0 of the SVID is the common/specific indicator. It is used to distinguish between types of subvectors.
  - The MAC subvectors with code points from X'00' through X'7F' (see "List of Subvectors" on page 5-17) are used so that certain *common* (to many MAC major vectors) strings of data can be formatted and labeled in a standard way. This standardization facilitates sharing of data between local area network applications, and helps make the data application independent. Additionally, the design effort for formatting and documenting the vectors need not be repeated for every future instance of their use. The type codes for these vectors are unique across local area network applications and are controlled by IBM Token-Ring Network architecture to ensure this uniqueness.
  - The MAC subvectors with code points from X'80' through X'F8' are for *specific* definition within a particular MAC major vector by command versus reply types. They need be unique only within the context of the specific command major vector and reply major vector definitions.

    For example, the MAC subvector X'90' can have an entirely different definition in every different MAC major vector, and can differ depending on whether the major vector is a command or reply. The MAC subvector X'40' has only one definition across all MAC major vectors and applications.
- Bit 1 of the SVID is the Required/Optional Indicator Bit (R/O). This bit is set by the sending ring station to inform the target ring station that the receiver must handle this subvector. If this bit is set to B'1' in the subvector, and the target ring station does not recognize this subvector, the frame is rejected and a negative response is sent. If this bit is set to B'0' in the subvector, and the target ring station does not recognize the subvector, the subvector is ignored and the frame is accepted.
- Bits 2-7 of the SVID are the subvector Code Point Bits (P). Note that a code point of X'FF' means that an expanded SVID is being used and is contained in the next 2 bytes.

Subvector Value

This field contains information to process the subvector.

MAC subvectors themselves may contain other MAC subvectors. They may also contain other types of vectors and optional fields that are unique only to the particular MAC subvector to which they belong.

MAC Frame Specification Table

Table 5-1 on page 5-8 lists all IBM Token-Ring Network Medium Access Control frames, in numerical order of their major vector values.

Abbreviations used in Table 5-1 on page 5-8

O A subvector that is *optional* in a given frame. The receiver assumes a default value if this subvector is not present.

R A subvector that is *required* in a given frame. If this subvector is not present, the ring station rejects the major vector and sends a response stating the reason for rejection. See the validity checking rules in "MAC Frame Response" on page 5-27 for the handling of MAC frames with missing required subvectors.

NAUN Nearest active upstream neighbor.

Table 5-1 (Page 1 of 2). MAC Frame Specification Table

MVID Cmd Byte	MAC Frame	Frame Control Field	Destination Class	Source Class	Destination Address	Subvectors
X'00'	Response	X'00'	Source class of received frame	Ring station	Source address of received frame	X'09' [O] Correlator (from Request) X'20' [R] Response Code
X'02'	Beacon	X'02'	Ring station	Ring station	All stations this ring	X'01' [R] Beacon Type X'02' [R] NAUN X'0B' [O] Physical Location
X'03'	Claim Token	X'03'	Ring station	Ring station	All stations this ring	X'02' [R] NAUN X'0B' [O] Physical Location
X'04'	Ring Purge	X'04'	Ring station	Ring station	All stations this ring	X'02' [R] NAUN X'0B' [O] Physical Location
X'05'	Active Monitor Present	X'05'	Ring station	Ring station	All stations this ring	X'02' [R] NAUN X'0B' [O] Physical Location
X'06'	Standby Monitor Present	X'06'	Ring station	Ring station	All stations this ring	X'02' [R] NAUN X'0B' [O] Physical Location
X'07'	Duplicate Address Test	X'01'	Ring station	Ring station	Station's own individual address	None
X'08'	Lobe Test	X'00'	Ring station	Ring station	Null address	X'26' [R] Wrap Data
X'09'	Transmit Forward	X'00'	Ring station	Config. Report Server	Target address	X'27' [R] Frame Forward
X'0B'	Remove Ring Station	X'01'	Ring station	Config. Report Server	Target address	None
X'0C'	Change Parameters RESPONSE REQUIRED	X'00'	Ring station	Config. Report Server	Target address	X'03' [O] Local Ring Number X'04' [O] Assign Physical Location X'05' [O] Soft Error Report Timer Value X'06' [O] Enabled Function Classes X'07' [O] Allowed Access Priority X'09' [O] Correlator (for Response)
X'0D'	Initialize Ring Station RESPONSE REQUIRED	X'00'	Ring station	Ring Parameter Server	Target address	X'03' [O] Local Ring Number X'04' [O] Assign Physical Location X'05' [O] Soft Error Report Timer Value X'09' [O] Correlator (for Response)
X'0E'	Request Ring Station Address	X'00'	Ring station	Config. Report Server	Target address	X'09' [O] Correlator (for Response)
X'0F'	Request Ring Station State	X'00'	Ring station	Config. Report Server	Target address	X'09' [O] Correlator (for Response)
X'10'	Request Ring Station Attachments	X'00'	Ring station	Config. Report Server	Target address	X'09' [O] Correlator (for Response)
X'20'	Request Initialization	X'00'	Ring parameter server	Ring station	Ring parameter server functional address, this ring	X'02' [R] NAUN X'22' [R] Product Instance ID X'23' [R] Ring Station Microcode Level

Table 5-1 (Page 2 of 2). MAC Frame Specification Table

MVID Cmd Byte	MAC Frame	Frame Control Field	Destination Class	Source Class	Destination Address	Subvectors
X'22'	Report Ring Station Address	X'00'	Config. Report Server	Ring station	Sender of request	X'02' [R] NAUN X'09' [R] Correlator (from Request) X'0B' [R] Physical Location X'2B' [R] Group Address X'2C' [R] Functional Address(es)
X'23'	Report Ring Station State	X'00'	Config. Report Server	Ring station	Sender of request	X'09' [R] Correlator (from Request) X'23' [R] Ring Station Microcode Level X'28' [R] Station ID X'29' [R] Ring Station Status Vector
X'24'	Report Ring Station Attachments	X'00'	Config. Report Server	Ring station	Sender of request	X'06' [R] Enabled Function Classes X'07' [R] Allowed Access Priority X'09' [R] Correlator (from Request) X'22' [R] Product Instance ID X'2C' [R] Functional Address(es)
X'25'	Report New Active Monitor	X'00'	Config. Report Server	Ring station	Config. Report Server functional address, this ring	X'02' [R] NAUN X'0B' [R] Physical Location X'22' [R] Product Instance ID
X'26'	Report NAUN Change	X'00'	Config. Report Server	Ring station	Config. Report Server functional address, this ring	X'02' [R] NAUN X'0B' [R] Physical Location
X'27'	Report Neighbor Notification Incomplete	X'00'	Ring error monitor	Ring station	Ring error monitor functional address, this ring	X'0A' [R] Address of Last Neighbor Notification
X'28'	Report Active Monitor Error	X'00'	Ring error monitor	Ring station	Ring error monitor functional address, this ring	X'02' [R] NAUN X'0B' [R] Physical Location X'30' [R] Error Code
X'29'	Report Soft Error	X'00'	Ring error monitor	Ring station	Ring error monitor functional address, this ring	X'02' [R] NAUN X'0B' [R] Physical Location X'2D' [R] Isolating Error Counts X'2E' [R] Non-Isolating Error Counts
X'2A'	Report Transmit Forward	X'00'	Config. Report Server	Ring station	Config. Report Server functional address, this ring	X'2A' [R] Transmit Status Code

List of Major Vector MAC Frames

The 25 MAC frames with unique major vector IDs are described below in alphabetic order. These MAC frames are listed in the numeric order of their Major Vector ID values in Table 5-1 on page 5-8.

Active Monitor Present MAC Frame, X'05'

This frame is sent by the active monitor to all other ring stations on the same ring (X'C000 FFFF FFFF') at the completion of the ring purge process or at the expiration of the active monitor's T(neighbor_notification) timer. This frame is used to:

- Inform standby monitors of the active monitor's presence
- Indicate that the ring is functioning correctly
- Initiate neighbor notification.

Because all standby monitors depend on the timely arrival of this frame to indicate the active monitor's presence, this frame must be transmitted at the highest priority (access control field set to X'F0', indicating an access priority of 7). Standby monitors must also be able to recognize that the frame has been transmitted, so they will not initiate token-claiming inappropriately.

The interval between transmissions of the Active Monitor Present MAC frame is controlled by T(neighbor_notification) in the active monitor. (See Appendix A, "Token-Protocol Timers" on page A-1 for more detail about timers and their relation to the neighbor notification process.) When this timer expires, the active monitor queues an Active Monitor Present MAC frame.

The active monitor uses its T (receive_notification) timer to ensure that a Active Monitor Present frame cycles the entire ring periodically.

The information field of this MAC frame contains:

- X'0012 0005' (the Active Monitor Present major vector)
- X'0802 xxxx xxxx xxxx' (the NAUN subvector, with length=8, type=2, and the active monitor's NAUN's individual address)
- X'060B xxxx xxxx ' (the Physical Location subvector, with length=6, type=B, and the assigned physical location of the active monitor).

Beacon MAC Frame, X'02'

If a ring station detects the expiration of T(claim_token) during the token-claiming process, then the ring station transmits this frame to all other ring stations on the same ring.

Change Parameters MAC Frame, X'0C'

This frame is sent by the configuration report server to a station to set any appropriate ring operational values.

Claim Token MAC Frame, X'03'

This frame is transmitted by a ring station to all other ring stations on the same ring during the token-claiming process (see "The Token-Claiming Process" on page 3-23).

Duplicate Address Test MAC Frame, X'07'

This frame is sent by a ring station addressed to itself to assure that:

- During attachment no other ring station on the same ring has the same individual address
- During self-test the station can receive frames.

Initialize Ring Station MAC Frame, X'0D'

This frame is sent by the ring parameter server to respond to a ring station's Request Initialization MAC frame, and to set any appropriate ring operational values.

Lobe Test MAC Frame, X'08'

This frame is sent by a ring station to test the bit error rate in its loop-back path:

- Before the ring stations physical attachment to the ring
- During the beacon self-test process.

The destination address of this frame is the null address (X'0000 0000 0000').

Remove Ring Station MAC Frame, X'0B'

This frame is sent by the configuration report server to instruct a ring station to remove itself from the ring.

Report Active Monitor Error MAC Frame, X'28'

This frame is sent to the ring error monitor functional address by the active monitor when it receives a Ring Purge or an Active Monitor Present MAC frame that it did not transmit, or when it receives a Claim Token MAC frame. It is also sent by a ring station in Claim Token Transmit mode if it receives a Claim Token MAC frame with its individual source address but a different NAUN. When Report Active Monitor Error is sent as a result of receiving a Claim Token MAC frame, its transmission takes place after token-claiming is resolved.

Report NAUN Change MAC Frame, X'26'

This frame is sent to the configuration report server functional address (X'C000 0000 0010') by any ring station (including the active monitor) that detects a change in the address of its nearest active upstream neighbor (NAUN) during neighbor notification. This address change means that either a ring station has attached to the ring, or a ring station has removed itself from the ring. The LAN manager can use this information to maintain an accurate configuration table.

To ensure that the configuration report server has reliable information, ring stations use assured delivery (see page 5-26) to deliver this frame.

The information field of this MAC frame contains:

- X'0012 4026' (the Report NAUN Change major vector)
- X'0802 xxxx xxxx xxxx' (the NAUN address subvector, with length=8, type=2, and the NAUN's individual address)
- X'060B xxxx xxxx' (the Physical Location subvector, with length=6, type=B, and the assigned physical location of the ring station).

Report Neighbor Notification Incomplete MAC Frame, X'27'

This frame is sent by the active monitor to the ring error monitor functional address (X'C000 0000 0008') when its T(neighbor_notification) expires before the completion of the previous neighbor-notification process.

When T(neighbor_notification) in the active monitor expires, the active monitor checks the "neighbor notification complete" flag to determine if the preceding neighbor notification cycle has been successfully completed. If the flag is not set, the active monitor queues a Report Neighbor Notification Incomplete MAC frame to the ring error monitor functional address.

The information field of this MAC frame contains:

- X'000C 6027' (the Report Neighbor Notification Incomplete major vector)
- X'080D xxxx xxxx xxxx' (the source address from the last copied Active Monitor Present or Standby Monitor Present MAC frame).

Note: The Active Monitor Present MAC frame is transmitted before any Report Neighbor Notification Incomplete MAC frame.

Report New Active Monitor MAC Frame, X'25'

This frame is sent by a ring station to the configuration report server functional address to report that it has become the active monitor.

Report Ring Station Address MAC Frame, X'22'

This frame is sent by a ring station to the configuration report server to respond to the latter's Request Ring Station Address frame.

Report Ring Station Attachments MAC Frame, X'24'

This frame is sent by a ring station to the configuration report server to respond to the latter's Request Ring Station Attachments frame.

Report Ring Station State MAC Frame, X'23'

This frame is sent by a ring station to the configuration report server to respond to the latter's Request Ring Station State frame.

Report Soft Error MAC Frame, X'29'

This frame is sent by a ring station to the ring error monitor functional address when a soft error is detected and T(soft_error_report) has expired. This enables the ring error monitor to maintain accurate error statistics.

To ensure that the ring error monitor has reliable error information, ring stations use assured delivery (see page 5-26) to deliver this frame.

Report Transmit Forward MAC Frame, X'2A'

This frame is sent by a ring station to the configuration report server functional address when it receives and forwards a Transmit Forward MAC frame.

Request Initialization MAC Frame, X'20'

This frame is sent by a ring station to the ring parameter server (functional address X'C000 0000 0002'). This is to inform the latter that the ring station has attached to the ring and will accept modified parameters from either the ring parameter server or the configuration report server.

Request Ring Station Address MAC Frame, X'0E'

This frame is sent by the configuration report server to a ring station to request the ring station's address information.

Request Ring Station Attachments MAC Frame, X'10'

This frame is sent by the configuration report server to a ring station to request information regarding the product in which the ring station resides.

Request Ring Station State MAC Frame, X'0F'

This frame is sent by the configuration report server to a ring station to request the state of the ring station's adapter.

Response MAC Frame, X'00'

This frame is sent from one ring station to another to acknowledge receipt of response MAC frames (see Table 5-1 on page 5-8), or to report a syntax error in, a MAC frame received with a ring station destination class (class 0).

- If the destination address field of the frame that contains the error is an all-station address, then the first ring station that sets the A and C bits on reports the error.
- If the DA field of the frame that contains the error is not a an all-station address, then any ring station that detects the error sends a report.

Ring Purge MAC Frame, X'04'

This frame is sent by the active monitor to all other ring stations on the same ring if one of the following conditions occurs:

- The token-claiming process is completed.
- A token error is detected. A token error is either a lost frame or token, a circulating frame, or a circulating priority token.

Standby Monitor Present MAC Frame, X'06'

This frame is sent by a standby monitor to all ring stations on the same ring (X'C000 FFFF FFFF') when it receives an Active Monitor Present or Standby Monitor Present MAC frame with A=C=0.

Standby monitors use T(receive_notification) to measure the time since they last recognized an Active Monitor Present MAC frame. The ring station waits to copy an Active Monitor Present or Standby Monitor Present MAC frame in which A=C=0. When this occurs, the ring station assumes that the frame is from its NAUN, and it uses the source address from the frame as its new NAUN address. The ring station starts its T(notification_response) and when it expires, queues a Standby Monitor Present frame to be used by the next downstream ring station.

The information field of this MAC frame contains:

- X'0012 0006' (the Standby Monitor Present major vector)
- X'0802 xxxx xxxx xxxx' (the NAUN subvector, with length=8, type=2, and the NAUN's individual address)
- X'060B xxxx xxxx' (the Physical Location subvector, with length=6, type=B, and the assigned physical location of the standby monitor.)

Transmit Forward MAC Frame, X'09'

This frame is sent by the LAN manager to a ring station to cause the ring station to construct a frame for transmission from the data contained in the information field.

MAC Frame Subvectors Specification Table

Table 5-2 lists all the MAC frame subvectors in numeric order by type value. The table shows a list of valid values or an explanation of the value field. The length shown includes the subvector type, length, and value fields.

Table 5-2 (Page 1 of 2). MAC Frame Subvectors Specification Table

Type	Length	Subvector Name	Value Field'
X'01'	X'04'	Beacon Type	Gives the reason for beaconing: X'0001' — Recovery mode set X'0002' — Signal loss error X'0003' — Streaming signal not Claim Token MAC frame X'0004' — Streaming signal, Claim Token MAC frame
X'02'	X'08'	NAUN	Indicates the individual address of the sending ring station's nearest active upstream neighbor (NAUN). If the value is all zeros, the NAUN is unknown.
X'03'	X'04'	Local Ring Number	Indicates the local ring number of the sending ring station.
X'04'	X'06'	Assign Physical Location	Indicates the physical location of the target ring station. This field must be manually entered and updated.
X'05'	X'04'	Soft Error Report Timer Value	Indicates the time-out value (in units of 10 milliseconds) for the ring station's T(soft_error_report) timer. soft-error report timer (see "Soft Error Report Timer Value, X'05'" on page 5-23).
X'06'	X'04'	Enabled Function Classes	Indicates the functional classes that the node is enabled to transmit (see "Enabled Function Classes, X'06'" on page 5-17).
X'07'	X'04'	Allowed Access Priority	The low-order 2 bits indicate the maximum token priority with which the attached node is authorized to transmit. The high-order 14 bits are ignored. X'0000' — Unauthorized Mode Not X'0000' — Authorized Mode
X'09'	X'04'	Correlator	Used by the sending ring station to relate requests and responses.
X'0A'	X'08'	Address of Last Neighbor Notification	Indicates the source address of the last Active Monitor Present or Standby Monitor Present MAC frames received by the active monitor before the notification cycle fails.
X'0B'	X'06'	Physical Location	It reports the assigned physical location of the sending ring station. This value is available only if manually assigned.
X'20'	X'06'	Response Code	Indicates whether or not a frame was received and handled correctly. The first 2 bytes specify the response code. The second 2 bytes contain source class, destination class, and major vector ID of the triggering MAC frame. For a list of response codes, see "Response Code, X'20'" on page 5-22.
X'21'	X'04'	Reserved	Used in the Request Initialization and Report Ring Station Address MAC frames. The value of this subvector is X'0000'.
X'22'	X'14'	Product Instance ID	Provides sufficient data to identify the hardware product instance of the attached product. For a list of values, see "Product Instance ID, X'22'" on page 5-21.
X'23'	X'0C'	Ring Station Microcode Level	Contains the sending ring station's microcode level specified in EBCDIC. Bytes 0-3 — Feature Code Bytes 4-9 — EC Level
X'26'	variable	Wrap Data	Used in the Lobe Test MAC frame.
X'27'	variable	Frame Forward	Consists of the access control and frame control fields, the destination and source addresses, and the routing information (optional) and information fields of the frame to be forwarded. The value field, if present, must be a Transmit Forward MAC frame.
X'28'	X'06'	Station Identifier	Uniquely identifies the station. It is used in the Report Ring Station State MAC frame.
X'29'	X'08'	Ring Station Status Subvector	Indicates the current state of the sending ring station's microcode. Its contents are implementation dependent.
X'2A'	X'04'	Transmit Status Code	Indicates the strip status of a transmitted frame. It is used in the Report Transmit Forward MAC frame.
X'2B'	X'06'	Group Address	Indicates the low-order 4 bytes of the group address recognized by this ring station.

Table 5-2 (Page 2 of 2). MAC Frame Subvectors Specification Table

Type	Length	Subvector Name	Value Field
X'2C'	X'06'	Functional Address	Indicates the functional addresses recognized by this ring station.
X'2D'	X'08'	Isolating Error Counts	Indicates the number of each type of error detected since the last soft error report (see "Isolating Error Counts, X'2D'" on page 5-19).
X'2E'	X'08'	Non-Isolating Error Counts	Indicates the number of each type of soft error detected since the last soft error report (see "Non-Isolating Error Counts, X'2E'" on page 5-20).
X'30'	X'04'	Error Code	Used in the Report Active Monitor MAC frame. It has the following code points: X'0001' — Monitor error X'0002' — Duplicate monitor X'0003' — Duplicate address

List of Subvectors

The 23 subvectors are described in the following sections, along with their respective subvector values.

Address of Last Neighbor Notification, X'0A'

This subvector has a value field 6 bytes long and is used in the Report Neighbor Notification Incomplete MAC frame. It indicates the source address of the last Active Monitor Present or Standby Monitor Present MAC frame received by the active monitor before the notification cycle fails.

Allowed Access Priority, X'07'

This subvector has a value field 2 bytes long and is used in the Change Parameters and the Report Ring Station Attachments MAC frames. The higher-order 14 bits are ignored, while the lower-order 2 bits indicate the maximum token priority with which the attached node is authorized to transmit.

Assign Physical Location, X'04'

This subvector has a value field 4 bytes long and is used in both Change Parameter and Initialize Ring Station MAC frames. It assigns the physical location of the target ring station. The Physical Location field must be manually entered and updated.

Beacon Type, X'01'

This subvector has a value field 2 bytes long and is used in the Beacon MAC frame. It gives the reason for the beaconing:

- X'0001' — Recovery mode set
- X'0002' — Signal loss error
- X'0003' — Streaming signal not Claim Token MAC frame
- X'0004' — Streaming signal, Claim Token MAC frame or intermediate detection of hard error.

Correlator, X'09'

This subvector has a value field 2 bytes long and is used by the sending ring station to relate requests and responses. It is used in the following MAC frames:

- Response
- Change Parameters
- Initialize Ring Station
- Request Ring Station Address, State, and Attachments
- Report Ring Station Address, State, and Attachments.

This subvector can be used in any major vector.

Enabled Function Classes, X'06'

This subvector has a value field 2 bytes long and is used in the Change Parameters and Report Ring Station Attachments MAC frames. It indicates the functional classes that the node is enabled to transmit. The valid range is X'0000' to X'FFFF' with each bit (0 to 15) corresponding to a function class. Bit values of B'1' indicate the function class is enabled. Ring station (class 0) and ring parameter server (class 5) cannot be enabled; DLC.LAN.MGR (class 1) and configuration report server (class 4) cannot be disabled.

Error Code, X'30'

This subvector has a value field 2 bytes long and is used in the Report Active Monitor Error MAC frame. It has the following code points:

- X'0001' — Monitor error, used when the active monitor receives a Claim Token MAC frame, indicating that another ring station detected a ring protocol error.
- X'0002' — Duplicate monitor, used when the active monitor receives a Ring Purge or Active Monitor Present MAC frame that it did not transmit, indicating the presence of another active monitor.
- X'0003' — Duplicate address, used when a ring station in Claim Token Transmit mode receives a Claim Token MAC frame in which the source address equals the ring station's individual address but the NAUN is different from the ring station's NAUN. This indicates that another ring station on the ring has the same individual address.

Frame Forward, X'27'

The length of this subvector is a product implementation choice. It is used in the Transmit Forward MAC frame. This subvector consists of the access control and frame control fields, the destination and source addresses, and the routing information (optional) and information fields of the frame to be forwarded. The value field of this subvector, if present, must contain a Transmit Forward MAC frame.

Functional Address, X'2C'

This subvector has a value field 4 bytes long and is used in the Report Ring Station Address and Report Ring Station Attachments MAC frames. It indicates the functional addresses recognized by this ring station (see "Functional Addresses" on page 3-10).

Group Address, X'2B'

This subvector has a value field 4 bytes long and is used in the Report Ring Station Address MAC frame. It indicates the lower-order 4 bytes of the group address recognized by this ring station.

Isolating Error Counts, X'2D'

This subvector has a value field 6 bytes long and is used in the Report Soft Error MAC frame. It indicates the number of each type of error detected since the last soft error report.

- Byte 0 — line error

  This counter is incremented when a frame or token is repeated by the ring station, the error-detected bit of the ending delimiter is set to B'0', and the ring station detects:

  - A code violation between the starting and ending delimiters of the frame or token
  - A Frame Check Sequence error (only in a frame).

  The first ring station that detects a line error increments its Line-Error counter and sets the error-detected bit to B'1' in the ending delimiter of the frame; this prevents other ring stations from logging the error, and isolates the source of the line error to the proper fault domain.

- Byte 1 — internal error

  This counter is incremented when a ring station recognizes a recoverable internal error. This can be used for detecting a ring station in marginal operating condition.

- Byte 2 — burst error

  This counter is incremented when a ring station detects the absence of transitions for 5 half-bit times (burst-five error). Note that the first ring station to detect a burst-four condition (4 half-bit times without transitions) transmits idles on the fifth half-bit time.

- Byte 3 — A/C error

  A ring station receives an Active Monitor Present or Standby Monitor Present MAC frame with AC=B'00', and then receives a Standby Monitor Present MAC frame with AC=B'00' without first receiving an Active Monitor Present MAC frame.

- Byte 4 — abort delimiter transmitted

  This counter is incremented when a ring station transmits an abort delimiter (see "Tokens, Frames, and Abort Delimiters" on page 3-12).

- Byte 5 — reserved.

Local Ring Number, X'03'

This subvector has a value field 2 bytes long and is used in the Initialize Ring Station MAC frame. It indicates the local ring number of the sending ring station.

NAUN, X'02'

This subvector has a value field 6 bytes long and is used in the following MAC frames:

- Beacon
- Claim Token
- Report Active Monitor Error
- Report NAUN Change
- Report New Active Monitor
- Report Ring Station Address

- Report Soft Error
- Active Monitor Present
- Standby Monitor Present
- Ring Purge
- Request Initialization.

It indicates the individual address of the sending ring station's nearest active upstream neighbor (NAUN).

If the value is all zeros, the ring station's NAUN is unknown (for example, ring station entering token-claiming because it is the first station on the ring).

Non-Isolating Error Counts, X'2E'

This subvector has a value field 6 bytes long and is used in the Report Soft Error MAC frame. It indicates the number of each type of error detected since the last soft error report.

- Byte 0 — lost frame error

  This counter is incremented when a ring station is transmitting and its T(physical_trailer) timer expires.

  This counts how often frames transmitted by a particular ring station fail to return to it (and thus the active monitor must issue a new token).

- Byte 1 — receiver congestion

  This counter is incremented when a ring station is receiving/repeating a frame and recognizes a frame addressed to it, but has no buffer space available for the frame.

- Byte 2 — frame-copied error

  This counter is incremented when a ring station detects an on-ring frame with its individual address, but with A=C=B'1', indicating a possible duplicate address.

- Byte 3 — frequency error

  This counter is incremented when a ring station detects a frequency error.

- Byte 4 — token error

  This counter is incremented when an active monitor recognizes an error condition resulting in the need to transmit a token. This occurs when the active monitor function recognizes any of the following:

  - The monitor bit of a priority token has been set to B'1' (circulating priority token)
  - A frame has its monitor bit set to B'1' (circulating frame)
  - T(any_token) expires (lost frame or token).

- Byte 5 — reserved.

Physical Location, X'0B'

This subvector has a value field 4 bytes long; it reports the assigned physical location of the sending ring station. It is used in the following MAC frames:

- Beacon
- Claim Token
- Ring Purge
- Active Monitor Present
- Standby Monitor Present
- Report Ring Station Address
- Report New Active Monitor
- Report NAUN Change
- Report Active Monitor Error
- Report Soft Error.

This value is available only if manually assigned.

Product Instance ID, X'22'

This subvector is a maximum of 18 bytes long. It is used in the following MAC frames:

- Request Initialization
- Report Ring Station Attachments
- Report New Active Monitor.

The contents of this subvector are passed to the ring station by the attached product when the ring station is initialized. The contents must be supplied at initialization in order to be meaningful. It provides sufficient data to identify uniquely the hardware product instance of the attached product.

- Byte 0
  - bits 0-3 Reserved
  - bits 4-7 Product classification:
    X'1' IBM hardware
    X'3' IBM or non-IBM hardware
    X'4' IBM software
    X'9' Non-IBM hardware
    X'C' Non-IBM software
    X'E' IBM or non-IBM software.
- Byte 1 — Format type:
  - X'10': Product instance is identified by a serial number (that is, IBM plant of manufacture and sequence number) unique by machine type.
  - X'11': Product instance is identified by a serial number unique by machine type and model number.
  - X'12': Product instance is identified by a serial number unique by machine type (as in Format X'10' above). This format provides the model number not to identify a product instance uniquely, but for additional information only.
- Byte 2-5 — Machine type: four numeric EBCDIC characters.
- Byte 6-8 — Machine model number: three upper-case alphanumeric EBCDIC characters for format types X'11' and X'12'; these bytes are reserved by IBM for future use in format type X'10'.

- Byte 9-10 — Serial number modifier — IBM plant of manufacture: two numeric EBCDIC characters.
- Byte 11-17 — Sequence number: seven upper-case alphanumeric EBCDIC characters, right-justified, with EBCDIC zeros (X'F0') fill on the left.

Reserved, X'21'

This subvector has a value field 2 bytes long and is used in the following MAC frames:

Request Initialization
Report Ring Station Address.

The value of this subvector is X'0000'.

Response Code, X'20'

This subvector has a value field 4 bytes long and is used in the Response MAC frame. It consists of a 2-byte response code followed by another 2 bytes containing the source class, destination class, and the MVID in the received MAC frame that caused the ring station to send the Response MAC frame.

Following are the code points:

- X'0001' — Positive acknowledgment. The MAC frame was accepted by the ring station.
- X'8001' — Missing major vector. MVID is missing from the MAC frame.
- X'8002' — Major vector length error. Major vector length does not agree with the length of the frame or a subvector was found that did not fit within the major vector.
- X'8003' — Unrecognized MVID. The major vector ID is not recognized by the ring station.
- X'8004' — Inappropriate source class. The source class is not valid for the MVID.
- X'8005' — Subvector length error. Subvector length is less than 2, does not agree with the actual length, or exceeds the maximum allowed length.
- X'8006' — Transmit Forward invalid. The subvector in Transmit Forward MAC frame is not the Frame Forward subvector, or a parsing error was detected in a Transmit Forward MAC frame.
- X'8007' — Missing required subvector. A subvector required to process the MAC frame is not in the MAC frame.
- X'8008' — Unrecognized required subvector. A subvector that is marked required is not known by the ring station.
- X'8009' — MAC frame exceeds the receiving buffer size. MAC frame exceeds the expected maximum length.
- X'800A' — Function requested was disabled: for example, a specific MAC frame is received while the function is disabled.

Ring Station Microcode Level, X'23'

This subvector has a value field 10 bytes long and is used in the Request Initialization and Report Ring Station State MAC frames. It contains the sending ring station's microcode level, specified in EBCDIC.

Ring Station Status Subvector, X'29'

This subvector has a value field 6 bytes long and is used in the Report Ring Station State MAC frame. The format of this subvector is implementation dependent.

Soft Error Report Timer Value, X'05'

This subvector has a value field 2 bytes long and is used in the Change Parameters and Initialize Ring Station MAC frames. It indicates the time-out value (in units of 10 milliseconds) for the ring station's T(soft_error_report) timer, which is described in Appendix A, "Token-Protocol Timers" on page A-1.

The value of this subvector indicates the value of the timer in 10-millisecond increments. The default value of 2 seconds is established if:

- The ring parameter server is not present.
- This subvector is missing from the Change Parameters or Initialize Ring Station MAC frames.

Some example values are: X'000D' – maximum time delay (65,537 x 10 milliseconds), X'0001' – minimum time delay (1 x 10 milliseconds), and X'FFFF' – (65,536 x 10 milliseconds).

Station Identifier, X'28'

This subvector has a value field 6 bytes long and is used in the Report Ring Station State MAC frame. It uniquely identifies the station.

Transmit Status Code, X'2A'

This subvector is a 2-byte field indicating the strip status of a transmitted frame. It is used in the Report Transmit Forward MAC frame.

Wrap Data, X'26'

This subvector, with a length of 1500 bytes, is used in the Lobe Test MAC frame.

MAC Frame Characteristics

The following sections describe some special characteristics of MAC frames, including the use of express buffering, frames that can and cannot leave the source ring, frames that should not be sent to all stations, and the use of assured delivery.

Express Buffer and Other Frame Control Field Values

When a ring station receives a MAC frame, it examines the frame control field. If the frame cannot be copied, the ring station takes no action. If the frame can be copied, the ring station takes the action indicated below:

Any MAC frame that is marked *express buffer* must be processed by the MAC layer only. A frame intended for any other use causes the received MAC frame to be discarded with no action taken by the MAC layer.

Any MAC frame transmitted as express buffer must not also use the Assured Delivery transmit function since these two functions are mutually exclusive.

- If the control bits equal X'0', then the MAC frame is normal-buffered.

  The following MAC frames are normal-buffered:

  - Lobe Test
  - Report Active Monitor Error
  - Report NAUN Change
  - Report New Active Monitor
  - Report Neighbor Notification Incomplete
  - Report Ring Station Address
  - Report Ring Station State
  - Report Ring Station Attachments
  - Report Soft Error
  - Report Transmit Forward
  - Request Initialization
  - Request Ring Station Address
  - Request Ring Station State
  - Request Ring Station Attachments
  - Response
  - Change Parameters
  - Initialize Ring Station
  - Transmit Forward.

- If the control bits are greater than or equal to X'1', then the MAC frame is express-buffered.

  When a ring station's normal receive buffers are full, the ring station cannot copy subsequent frames unless some action is taken. Since certain MAC frames must be delivered immediately, these MAC frames require express buffering.

  A ring station's express buffers are used to copy express-buffered MAC frames when the ring station's normal receive buffers are full. When the express buffer is being used, non-express buffered frames are recognized but not copied. Express buffering therefore improves a ring station's chance to copy those MAC frames immediately.

The MAC frames that can be express-buffered are:

- Beacon
- Claim Token
- Ring Purge
- Active Monitor Present
- Standby Monitor Present
- Duplicate Address Test
- Remove Ring Station.

The ring station's routing function, which determines whether a frame is to be processed by the ring station or forwarded to higher layers, processes frames in the order they arrive at the ring station, regardless of whether they were copied by the normal receive buffers or by the express buffers.

MAC Frames That Should Not Leave the Source Ring

MAC frames that should not have a routing information field should not leave the originating ring station's ring. These MAC frames are:

- Active Monitor Present
- Beacon
- Claim Token
- Duplicate Address Test
- Lobe Test
- Remove Ring Station
- Report Active Monitor Error
- Report NAUN Change
- Report New Active Monitor
- Report Neighbor Notification Incomplete
- Report Soft Error
- Report Transmit Forward
- Report Ring Station Address
- Report Ring Station State
- Report Ring Station Attachments
- Request Ring Station Address
- Request Ring Station State
- Request Ring Station Attachments
- Request Initialization
- Response
- Ring Purge
- Change Parameters
- Initialize Ring Station
- Standby Monitor Present.

MAC Frames That Cannot be Sent to All Stations

The use of the all-stations address (either X'FFFF FFFF FFFF' or X'C000 FFFF FFFF') in any frame causes all ring stations on the intended ring or rings to receive the frame. (Whether the frame leaves the transmitter's ring depends on the contents of the routing information field, not on the contents of the address field.) The nature of certain MAC frames is such that sending them to all ring stations could seriously disrupt ring operation or degrade ring performance. The following MAC frames are of this nature, and cannot be sent with an all-stations address:

- Remove Ring Station
- Response.

Assured Delivery

The LLC sub-layer can use sequence numbers, acknowledgments, time-outs, and retransmissions to assure reliable, sequential delivery of LLC frames. However, the MAC sub-layer does not use these procedures. To assure the delivery of a MAC frame (on-ring only), the originating ring station can retransmit the frame if it appears that the target of the frame either failed to copy it or did not recognize it because of an error.

The originating ring station can determine that delivery has failed by examining the address-recognized (A) and frame-copied (C) bits of the transmitted frame's frame status field, and by the error-detected (E) bit of its ending delimiter. The ring station retransmits the frame if either of the following conditions is met:

- Both A bits are set to B'1' and both C bits are set to B'0', indicating that the target ring station recognized its address but did not copy the frame. This could occur if there is congestion at the target ring station.
- Both A and both C bits are set to B'0', and the E bit is set to B'1', indicating that an error was detected on the ring and the target ring station did not recognize its address or copy the frame.
- Transmit status indicates that the frame was not transmitted due to a corrupted token.

The ring station continues to retransmit the frame until either a threshold is reached, or until both A and both C bits are set to B'1' and the E bit is set to B'0'.

The originating ring station uses assured delivery with the Report NAUN Change and Report Soft Error MAC frames.

Note: Express delivery and assured delivery are mutually exclusive.

The A and C bits can only be used to determine that delivery has failed for frames that stay on the original ring. The A and C bits will be set if the frame was copied and forwarded to another ring by a bridge. In this case the originating station cannot determine if the destination station actually copied the frame by examining the A and C bits.

MAC Frame Response

A ring station parsing a MAC frame checks for a variety of errors. These errors are identified in "Response Code, X'20'" on page 5-22.

MAC Activation and Deactivation

The MAC sub-layer is activated when the DLC.LAN.MGR successfully executes the REQUEST_MAC_CONTROL primitive with the INSERT control_action. (In this reference, a *primitive* is an abstract, implementation-independent interaction between a user of a service and the provider of the service.) The MAC sub-layer uses whatever values it has at the time for its operational values. If no REQUEST_MAC_CONFIGURE has been previously executed, and the individual_MAC_address does not have a default hardware value, then the REQUEST_MAC_CONTROL will fail.

To specify group_MAC_addresses and other ring station characteristics, the normal activation sequence is:

```
—► REQUEST_MAC_CONFIGURE
◄— CONFIRM_MAC_CONFIGURE
—► REQUEST_MAC_CONTROL
◄— CONFIRM_MAC_CONTROL
```

DLC.LAN.MGR can deactivate the MAC sub-layer using a REQUEST_MAC_CONTROL with the MASTER_RESET control_action. The normal deactivation sequence is:

```
—► REQUEST_MAC_CONTROL
◄— CONFIRM_MAC_CONTROL
```

Figure 5-7 on page 5-28 illustrates the MAC activation procedure.

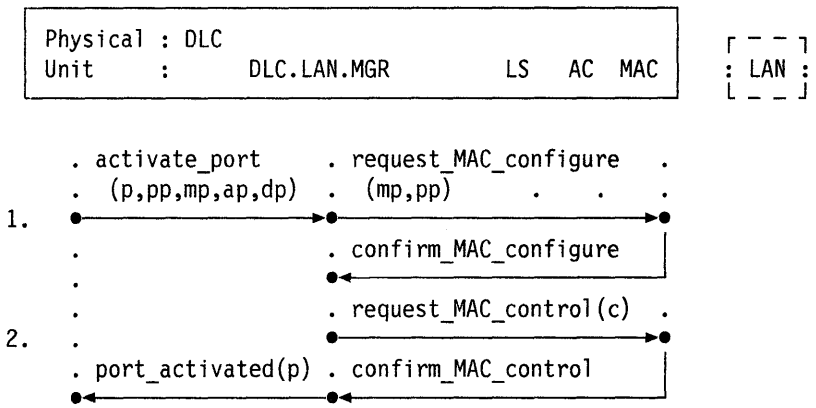


Figure 5-7. SNA Port Activation Flow for DLC.LAN Component

The following legend explains the symbols in Figure 5-7:

ap = access channel parameters
c = control action (for example, insert)
dp = DLC.LAN.MGR parameters for port activation
mp = MAC parameters
p = port address
pp = port parameters

The following notes are keyed to Figure 5-7.

1. Receiving an activate_port record from the Physical Unit causes DLC.LAN.MGR to initialize the medium access control component and to activate the physical port.

   DLC.LAN.MGR initializes the medium access control by sending a request_MAC_configure record containing parameters (group MAC addresses to recognize, a functional address mask, a participant option, and so on) to the medium access control. When initialization is complete, the MAC sub-layer informs DLC.LAN.MGR by sending a confirm_MAC_configure record.

2. Following initialization of the medium access control, DLC.LAN.MGR activates the port by sending a request_MAC_control record to the medium access control. This record requests that the ring station be attached to the ring. DLC.LAN.MGR is informed of the success or failure of the procedure by a confirm_MAC_control record.

   After activating the port, DLC.LAN.MGR passes the port activation results to the Physical Unit in a port_activated record.

Chapter 6. MAC-to-LLC Services

This chapter specifies the services provided by the MAC sub-layer to the LLC sub-layer, via the access channel. These services allow the local LLC sub-layer to exchange LLC data units with peer LLC sub-layers.

The following primitives enable the access channel to request service from the MAC sub-layer:

- SEND_AC_DATA
- RECEIVE_AC_DATA
- CONFIRM_AC_DATA.

Each service names the particular primitive and the information passed between the access channel and MAC sub-layer.

SEND_AC_DATA

This primitive defines the transfer of a MAC service data unit from the local LLC sub-layer to one or more peer LLC sub-layers.

Semantics of the Service Primitive

```
SEND_AC_DATA  (
  frame_control_field,
  destination_address,
  routing_information,
  MSDU,
  requested_service_class
  )
```

The frame_control_field parameter specifies the value for the frame control field (see "Frame Control Field" on page 2-3). This parameter has the following characteristics:

- Bit 0 is set to B'0' and bit 1 is set to B'1'.
- Bits 2 through 7 are each set to B'0'.

The destination_address parameter specifies either an individual or a group MAC address. It contains sufficient information to create the destination address field that is appended to the frame by the local MAC sub-layer.

The routing_information parameter specifies the frame routing information.

The MSDU parameter specifies the MAC service data unit (MSDU) to be transmitted by the MAC sub-layer. This parameter contains sufficient information for the MAC sub-layer to determine the length of the data unit.

The requested_service_class parameter specifies the priority desired for the data unit transfer. This priority has a value between B'000' and B'011'.

When Generated

This primitive is generated by the access channel whenever data must be transferred to a peer access channel. This can be in response to a request from higher layers of protocol, or from data generated internally to the LLC sub-layer.

Effect on Receipt

Receipt of this primitive causes the MAC sub-layer to append all MAC-specific fields, and to pass the properly formed frame to the lower layers of protocol for transfer to one or more MAC sub-layers.

RECEIVE_AC_ DATA

This primitive defines the transfer of data from the MAC sub-layer to one or more LLC sub-layers.

Semantics of the Service Primitive

```
RECEIVE_AC_DATA  (
  frame_control_field,
  destination_address,
  source_address,
  routing_information,
  MSDU,
  reception_status
  )
```

The frame_control_field parameter is the frame control field received.

The destination_address parameter may be either an individual or a group MAC address, as specified by the destination address of the received frame.

The source_address parameter must be an individual address, as specified by the source address of the received frame.

The routing_information parameter specifies the routing information contained in the received frame.

The MSDU parameter specifies the MAC service data unit, as received by the local MAC sub-layer.

The reception_status parameter indicates the success or failure of the received frame. It consists of the following elements:

- frame_status: FR_GOOD
- E_value: B'0', B'1'
- A_&_C_value: B'00', B'01', B'10', B'11'.

When Generated

The RECEIVE_AC_DATA primitive is generated by the MAC sub-layer to one or more LLC sub-layers to indicate the arrival of a frame at the local MAC sub-layer. The MAC information field is reported only if either the frame is validly formed and received without error, and their destination address designates the local MAC sub-layer or their source address designates the local MAC sub-layer if the ring station was so initialized (see the primitive "REQUEST_MAC_ CONFIGURE" on page 4-11).

Effect on Receipt

The effect of this primitive on the access channel depends upon the validity and content of the frame.

Additional Comments

If the local MAC sub-layer is designated by the destination_address parameter of a SEND_AC_DATA primitive, the RECEIVE_AC_DATA primitive is also invoked by the MAC sub-layer to the local LLC sub-layer. This duplex characteristic of the MAC sub-layer may be due to unique function capabilities within the MAC sub-layer or to duplex characteristics of the lower layers. For example, all frames transmitted to the broadcast address invoke RECEIVE_AC_DATA primitives at all ring stations in the network including the ring station that generated the SEND_AC_DATA.

CONFIRM_AC_ DATA

This primitive responds to SEND_AC_DATA, signaling success or failure.

Semantics of the Service Primitive

```
CONFIRM_AC_DATA  (
  transmission_status,
  provided_service_class
  )
```

The transmission_status parameter passes status information back to the local requesting access channel. It indicates the success or failure of the associated SEND_AC_DATA. If success is indicated, this parameter has the following value:

- PDU_ACCEPTED

The failure reason codes indicate an error that causes the MAC sub-layer to fail to queue the frame for transmission. If failure is indicated, this parameter has one of the following values:

- MAC_NOT_ACTIVATED
- INVALID_FRAME_CONTROL_FIELD_VALUE
  - Bit values not allowed
- INVALID_DESTINATION_ADDRESS
  - Less than the minimum allowed length
  - Greater than the maximum allowed length
- INVALID_ROUTING_INFORMATION
  - Less than the minimum allowed length
  - Greater than the maximum allowed length
  - Improperly formed
- INVALID_MSDU
  - Greater than the maximum allowed length
- INVALID_REQUESTED_SERVICE_CLASS
  - Priority value not allowed.

The provided_service_class parameter specifies the service class that was provided for the data unit transfer, if the transmission_status parameter indicates success.

When Generated

This primitive is generated by the MAC sub-layer in response to a SEND_AC_DATA primitive from the local access channel.

Effect on Receipt

The effect of this primitive on the access channel is unspecified.

Additional Comments

It is assumed that sufficient information is available to the LLC sub-layer to associate the response with the appropriate request. For example, the association may be implied by the order of the responses, because the MAC sub-layer requires that the requests be serviced in a first-in, first-out manner. This does not preclude MAC from buffering more than one data unit for transmission.

Chapter 7. Finite State Machines

This section contains a number of finite state machines (FSMs) that describe the operation of the IBM Token-Ring Network in an explicit manner. These FSMs illustrate the relationships between the normal token protocols (see "MAC Sub-Layer Operating Modes" on page 3-13) and the MAC recovery protocols used by the IBM Token-Ring Network to recover from error conditions. The diagram below illustrates these relationships:

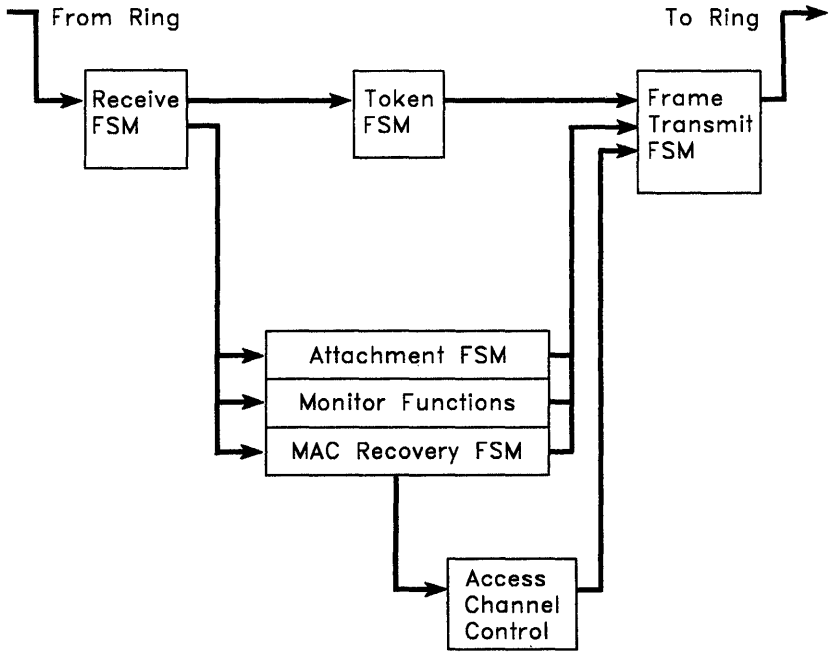


Figure 7-1. Finite State Machine Relationships

The Receiving FSM examines line input for frames to be received or stripped, errors in passing frames, delimiters, or non-standard bit patterns. It is responsible for setting the error-detected bit in the ending delimiter of passing frames, and for routing frames to the Station Attachment, Monitor Functions, and MAC Recovery FSMs. These FSMs handle received MAC and LLC frames, and can originate frames in response, or for ring recovery.

The Token Transmission FSM issues tokens; the Frame Transmission FSM uses them, if they originated in another ring station, to transmit pending frames.

Guide to Reading Finite State Machines

This guide is to help you understand the Routing Logic and Output code in the sections that follow.

The first section of each FSM (Function, Input, and Descriptions of States) describes the function being defined by the FSM, the inputs required, and the states within the machine.

The second section of the FSM (the INPUTS table) shows the states and the inputs that cause state transitions to occur.

A ring station acts according to the inputs received. "Routing Logic for the Finite State Machine" on page 7-3 describes the FSMs to be called, and the order in which to call them, given a particular input. After processing the input, a called FSM remembers the state it was in and returns control to "Routing Logic for the Finite State Machine."

"FSM_MONITOR_FUNCTIONS" refers to the Monitor Functions finite-state machine. "If the state of FSM_MONITOR_FUNCTIONS = ACTIVE then" means that if the station is currently in the ACTIVE state (see the particular FSM for different possible states) in the Monitor Functions finite-state machine, then do what the following statements specify. Variables, FSM names, and states are in all capital letters and words are separated by "_".

When a machine is in a particular state (the columns), the possible inputs (down the left edge of the FSM) determine the potential transitions. The possible actions performed (the rows) are as follows:

- '-' means there is no state transition; no action occurs.
- '-(A)' means there is no state transition; output action A occurs.
- 'n(A)' means there is a transition to state n; output action A occurs.
- '/' defines an invalid condition.

The last section of the FSM (the OUTPUT CODE table) defines the output codes referenced in the INPUTS table.

Routing Logic for the Finite State Machine

Function This logic describes what action to take for a given input, and in what order the particular inputs should be called. The input is processed as described by this machine and then control is returned to this machine.

Input The input consists of signals, recognized delimiters, and frames from the ring. Timer expirations, counters, and flags are inputs internal to the ring station. Descriptions of the flags and counters follow:

ACTIVE_MONITOR
This flag indicates whether the ring station is functioning as the active monitor for the ring. If the ring station is the active monitor, this flag is set to B'1'.

ATTACHED
This flag indicates whether the ring station is attached to the ring. After attaching to the ring (see "Station Attachment Finite State Machine" on page 7-21), this flag is set to B'1'.

CTFRAMES
This counter, used during MAC recovery, is decremented when a ring station is in Claim Token Transmit mode and receives its Claim Token MAC frame. When this counter reaches B'0', the token-claiming process is complete, and this ring station is the new active monitor.

DFC
The Duplicate Address Test Frame Counter (DFC) is used during attachment to the ring; it is initialized to DFC=n≥B'1'. When a Duplicate Address Test frame is stripped with AC≠B'00', DFC is decremented and another Duplicate Address Test frame is queued. If DFC reaches B'0', the address is assumed to be a duplicate.

MB
This flag, used during MAC recovery, indicates whether this ring station should be setting the monitor bit in received Beacon MAC frames. The ring station sets this flag to B'1' (enabling the setting of these bits) after receiving a specified number (M_BIT count) of Beacon MAC frames from its NAUN.

M_BIT
This counter, used during MAC recovery, is decremented when a ring station receives a Beacon MAC frame from its NAUN. When this counter reaches B'0', the ring station activates the function to set the monitor bit in repeated Beacon MAC frames. If a Beacon MAC frame is received that this ring station's NAUN did not transmit, this counter is set to its threshold value.

NADN
Nearest active downstream neighbor.

NDFC
The Non-Duplicate Address Frame Counter (NDFC) is used during attachment to the ring; it is initialized to NDFC=m≥B'1'. When a Duplicate Address Test frame is stripped with AC=B'00', NDFC is decremented and another Duplicate Address Test frame is queued. If NDFC reaches B'0', the address is assumed to be unique.

NEIGHBOR_NOTIFICATION
This flag is set to indicate that an Active Monitor Present or a Standby Monitor Present MAC frame was received with AC=B'00'.

NOPSC
The No Parameter Server Counter (NOPSC) is used during attachment to the ring; it is initialized to NOPSC = p≥1. When a Request Parameters frame is stripped with AC = B'00', NOPSC is decremented and another Request Parameters frame is queued. If NOPSC reaches B'0', no ring parameter server is present, the default values are used, and the ring station participates in normal ring operation.

RCVBEACON
This counter is used during MAC recovery. A ring station decrements this counter when it receives a Beacon MAC frame from its nearest active downstream neighbor (NADN). When this counter reaches B'0', the ring station removes itself from the ring and tests the functioning of its adapter.

REMOVE_TRANS
This flag is used to prevent a station transmitting beacon from removing twice during a recovery sequence.

RING_RECOVERED
This flag is set by the MAC Recovery FSM to indicate that token-claiming has successfully completed, and the ring station is a standby monitor.

RRTRYC
The Response Retry Counter (RRTRYC) is initialized to RRTRYC = q≥ B'1'. If T(response) expires, RRTRYC is decremented and another Request Parameters frame is queued.

SELF_TEST_COMPLETE
This flag indicates that the ring station has completed the self-test. It does not indicate the success or failure of the test.

SELF_TEST_SUCCESSFUL
This flag indicates the result of the self-test. It is set to B'0' if the test was unsuccessful, and to B'1' if successful.

SOFT_ERROR
This flag (when set to 1) indicates that a soft error has been detected and the soft-error timer is running. In this case, the error is just logged.

SOFT_ERROR_COUNTERS
This indicator refers to the soft error counters described in "Isolating Error Counts, X'2D'" on page 5-19 and "Non-Isolating Error Counts, X'2E'" on page 5-20.

STATION_WON_CLAIM_TOKEN
This flag indicates the station has won token-claiming and has been elected the active monitor.

Referenced Finite State Machines and Data Structures

MAC_FRAME_I-FIELD_PARSING

"MAC Frame Response" on page 5-27

FSM_MONITOR_FUNCTIONS

"Monitor Functions Finite State Machine" on page 7-25

FSM_MAC_RECOVERY

"MAC Recovery Finite State Machine" on page 7-13

FSM_STATION_ATTACHMENT

"Station Attachment Finite State Machine" on page 7-21

FSM_RECEIVING

"Receiving Finite State Machine" on page 7-19

FSM_TOKEN_TRANSMISSION

"Token Transmission Finite State Machine" on page 7-30

FSM_FRAME_TRANSMISSION

"Frame Transmission Finite State Machine" on page 7-32.

Select Appropriate FSM Based on Input

The following routing logic is used to select the appropriate FSM, based on the indicated input condition.

```
When ACTIVATE_MAC (DLC.LAN.MGR)
  Call FSM_STATION_ATTACHMENT
  ("Station Attachment Finite State Machine" on page 7-21).


When detect Signal loss
  If the state of FSM_STATION_ATTACHMENT ≠ (DISC|RESET|INIT) then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    If the state of FSM_MONITOR_FUNCTIONS = (ACTIVE|STANDBY)
    then
      Call FSM_MONITOR_FUNCTIONS
      ("Monitor Functions Finite State Machine" on page 7-25).
    If the state of FSM_STATION_ATTACHMENT = INIT then
      Call FSM_STATION_ATTACHMENT
      ("Station Attachment Finite State Machine" on page 7-21).
    Call FSM_MAC_RECOVERY
    ("MAC Recovery Finite State Machine" on page 7-13).


When (soft error detected) and (not SOFT_ERROR)
  Start T(soft_error).
  Set SOFT_ERROR.


When (recognize a starting delimiter)
  Call FSM_RECEIVING
  ("Receiving Finite State Machine" on page 7-19).
  If the state of FSM_MONITOR_FUNCTIONS = ACTIVE then
    Restart T(any_token).


When (DA = STA) | (SA = STA) | (recognize an ending delimiter) |
(detect a CRC error) | (recognize an abort delimiter) |
(recognize a burst 4 error) | (recognize a burst 5 error) |
(recognize a code violation between the starting and ending delimiters)
  Call FSM_RECEIVING
  ("Receiving Finite State Machine" on page 7-19).


When repeating a frame or priority token
and the state of FSM_MONITOR_FUNCTIONS = ACTIVE
  Call FSM_MONITOR_FUNCTIONS
  ("Monitor Functions Finite State Machine" on page 7-25).


When token
  If FSM_MONITOR_FUNCTIONS = STANDBY
    Restart T(good_token)
  Call FSM_TOKEN_TRANSMISSION
  ("Token Transmission Finite State Machine" on page 7-30).
  Call FSM_FRAME_TRANSMISSION
  ("Frame Transmission Finite State Machine" on page 7-32).
```

```
When stripped frame
  Call FSM_TOKEN_TRANSMISSION
  ("Token Transmission Finite State Machine" on page 7-30).
  Call FSM_FRAME_TRANSMISSION
  ("Frame Transmission Finite State Machine" on page 7-32).
 .If Duplicate Address Test MAC frame then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
    If (NDFC = B'0') | (DFC = B'0') then
      Call FSM_STATION_ATTACHMENT
      ("Station Attachment Finite State Machine" on page 7-21).
    If (ACTIVE_MONITOR)
    | (NEIGHBOR_NOTIFICATION flag set to B'1') then
      Call FSM_STATION_ATTACHMENT
      ("Station Attachment Finite State Machine" on page 7-21).
  Else
    If Request Initialization MAC frame then
      Call FSM_STATION_ATTACHMENT
      ("Station Attachment Finite State Machine" on page 7-21).
      If NOPSC = B'0' then
        Call FSM_STATION_ATTACHMENT
        ("Station Attachment Finite State Machine" on page 7-21).
        Call FSM_MONITOR_FUNCTIONS
        ("Monitor Functions Finite State Machine" on page 7-25).


When an LLC frame
  Call FSM_TOKEN_TRANSMISSION
  ("Token Transmission Finite State Machine" on page 7-30).
  Call FSM_FRAME_TRANSMISSION
  ("Frame Transmission Finite State Machine" on page 7-32).
  Route the frame to the access channel.


When received MAC frame
  Call FSM_TOKEN_TRANSMISSION
  ("Token Transmission Finite State Machine" on page 7-30).
  Call FSM_FRAME_TRANSMISSION
  ("Frame Transmission Finite State Machine" on page 7-32).


When SELF_TEST_COMPLETE
  If the state of FSM_STATION_ATTACHMENT = TEST then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    If not SELF_TEST_SUCCESSFUL then
      Remain bypassed.
      Notify DLC.LAN.MGR of self test failure.
```

Select Appropriate FSM Based on Type of MAC Frame

The following routing logic is used to select the appropriate FSMs, in the correct order, based on the indicated MAC frame.

```
When Beacon MAC frame
  If not ATTACHED then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    If the state of FSM_MONITOR_FUNCTIONS <>  (RESET) then
      Call FSM_MONITOR_FUNCTIONS
      ("Monitor Functions Finite State Machine" on page 7-25).
    Call FSM_MAC_RECOVERY
    ("MAC Recovery Finite State Machine" on page 7-13).
    If (Beacon is from this ring station's NADN) and (Beacon is not type 1) then
      Decrement RCVBEACON counter.
      If RCVBEACON counter = B'0' then
        Call FSM_MAC_RECOVERY
        ("MAC Recovery Finite State Machine" on page 7-13).
    Else
      Set RCVBEACON to its threshold value.
      If (Beacon is from this ring station's NAUN) and
      (Beacon is not type 1) then
        Decrement M_BIT.
        If M_BIT <> 0 then
          Reset MB to 0.
        Else
          Set M_BIT to threshold.
          Set MB to 1.
          Set REMOVE_TRANS to 1.
      Else
        Reset MB to B'0'.
        Set M_BIT counter to its threshold value.


When Claim Token MAC frame
  If the state of FSM_MONITOR_FUNCTIONS = (ACTIVE|STANDBY)
  then
    Call FSM_MONITOR_FUNCTIONS
    ("Monitor Functions Finite State Machine" on page 7-25).
  If the state of FSM_MONITOR_FUNCTIONS ≠ PURGE then
    Call FSM_MAC_RECOVERY
    ("MAC Recovery Finite State Machine" on page 7-13).
  If CTFRAMES = B'0' (*indicating token-claiming is complete*) then
    Call FSM_MAC_RECOVERY
    ("MAC Recovery Finite State Machine" on page 7-13).
    Call FSM_MONITOR_FUNCTIONS
    ("Monitor Functions Finite State Machine" on page 7-25).
    If the state of FSM_STATION_ATTACHMENT = CLTK then
      Set STATION_WON_CLAIM_TOKEN to 1
      Call FSM_STATION_ATTACHMENT
      ("Station Attachment Finite State Machine" on page 7-21).
```

```
When Active Monitor Present MAC frame
  If the state of FSM_STATION_ATTACHMENT ≠ (RESET|INIT) then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    If the state of FSM_MONITOR_FUNCTIONS = ACTIVE then
      Call FSM_MONITOR_FUNCTIONS
      ("Monitor Functions Finite State Machine" on page 7-25).
    Else
      Reset SINGLE_STATION flag
      Restart T(receive_notification).
    If AC=B'00' then
      Get the NAUN from the frame.
      If not ACTIVE_MONITOR then
        Start T(notification_response).
      If the NAUN differs from the stored NAUN then
        Queue a Report NAUN Change MAC frame.
        Store NAUN.


When Standby Monitor Present MAC frame
  If the state of FSM_STATION_ATTACHMENT ≠ RESET then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    If the state of FSM_MONITOR_FUNCTIONS = ACTIVE then
      Call FSM_MONITOR_FUNCTIONS
      ("Monitor Functions Finite State Machine" on page 7-25).
    If AC=B'00' then
      Reset SINGLE_STATION flag
      Get the NAUN from the frame.
      If not ACTIVE_MONITOR then
        Start T(notification_response).
      If the NAUN varies from the stored NAUN then
        Queue Report NAUN Change MAC frame.
        Store NAUN.


When Ring Purge MAC frame
  If the state of FSM_STATION_ATTACHMENT = MCK then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    If the state of FSM_MONITOR_FUNCTIONS = STANDBY then
      Cancel T(notification_response).
    If the state of FSM_MAC_RECOVERY = (CTX|CTR) then
      Call FSM_MAC_RECOVERY
      ("MAC Recovery Finite State Machine" on page 7-13).
    If RING_RECOVERED flag is set then
      If the state of FSM_STATION_ATTACHMENT = CLTK then
        Call FSM_STATION_ATTACHMENT
        ("Station Attachment Finite State Machine" on page 7-21).
      Else
        Call FSM_MONITOR_FUNCTIONS
        ("Monitor Functions Finite State Machine" on page 7-25).
    If FSM_MONITOR_FUNCTIONS = PURGE then
      Call FSM_MONITOR_FUNCTIONS
      ("Monitor Functions Finite State Machine" on page 7-25).
```

```
When (Change Parameters) | (Initialize Ring Station) MAC frame
  If (FSM_STATION_ATTACHMENT = INIT) and (SA = Specific Station Address)
  then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    Set the appropriate parameters in the ring station.


When Remove Ring Station MAC frame
  Remove ring station from the ring.
  Notify DLC.LAN.MGR ring station was removed.


When Request Ring Station Address MAC frame
  Queue a Report Ring Station Address MAC frame
  using the routing information field of the request frame.


When Request Ring Station State MAC frame
  Queue a Report Ring Station State MAC frame
  using the routing information field of the request frame.


When Request Ring Station Attachments MAC frame
  Queue a Report Ring Station Attachments MAC frame
  using the routing information field of the request frame.
```

Select Appropriate FSM Based on Expired Timer

The following routing logic is used to select the appropriate FSM, based on the indicated expired timer.

```
When T(physical_trailer)
  Call FSM_RECEIVING
  ("Receiving Finite State Machine" on page 7-19).


When T(good_token) | T(receive_notification)
  Call FSM_MONITOR_FUNCTIONS
  ("Monitor Functions Finite State Machine" on page 7-25).
  Call FSM_MAC_RECOVERY
  ("MAC Recovery Finite State Machine" on page 7-13).


When T(claim_token)
  If the state of FSM_STATION_ATTACHMENT ≠ (INIT|RESET) then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
  Else
    Call FSM_MAC_RECOVERY
    ("MAC Recovery Finite State Machine" on page 7-13).


When T(ring_purge)
  Call FSM_MONITOR_FUNCTIONS
  ("Monitor Functions Finite State Machine" on page 7-25).
  If ATTACHED then
    Call FSM_MAC_RECOVERY
    ("MAC Recovery Finite State Machine" on page 7-13).
  Else
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).


When T(escape) | T(beacon_transmit)
  Call FSM_MAC_RECOVERY
  ("MAC Recovery Finite State Machine" on page 7-13).


When T(attach)
  Call FSM_STATION_ATTACHMENT
  ("Station Attachment Finite State Machine" on page 7-21).
  If the state of FSM_STATION_ATTACHMENT = CLTK then
    Call FSM_MAC_RECOVERY
    ("MAC Recovery Finite State Machine" on page 7-13).


When T(response) expires
  Call FSM_STATION_ATTACHMENT
  ("Station Attachment Finite State Machine" on page 7-21).
  If RRTRYC = B'0' then
    Call FSM_STATION_ATTACHMENT
    ("Station Attachment Finite State Machine" on page 7-21).
```

```
When T(neighbor_notification) | T(any_token) expires
  Call FSM_MONITOR_FUNCTIONS
  ("Monitor Functions Finite State Machine" on page 7-25).


When T(transmit_pacing) expires
  Call FSM_MAC_RECOVERY
  ("MAC Recovery Finite State Machine" on page 7-13).


When T(notification_response) expires
  Queue a Standby Monitor Present MAC frame.


When T(soft_error) expires
  Queue Soft Error Report MAC frame.
  Reset SOFT_ERROR.
  Reset SOFT_ERROR_COUNTERS.
```

MAC Recovery Finite State Machine

Function

This FSM describes the actions of the MAC Recovery protocol. This FSM is active only under extraordinary conditions, such as certain timer expirations and the reception of recognized Beacon or Claim Token MAC frames. These frames are described in Chapter 5, "MAC Frames" on page 5-1. See Figure 7-1 on page 7-1 for the relationship of FSMs.

This FSM handles ring recovery and is called from the Routing Logic for FSMs for a particular input. Once the input is processed, the logic specifies other actions to take.

Input

AM_FUNCTIONAL_ADDRESS
This flag indicates whether the ring station was the previous active monitor (AM) that failed. If this flag is set to B'1', the ring station was the active monitor that failed; the ring station is prevented from actively participating in token-claiming, and enters Claim Token Repeat mode.

CTFRAMES_CNT
This counter is decremented when a ring station is in Claim Token Transmit and receives its Claim Token MAC frame. When this counter reaches B'0', the token-claiming process is complete, and this ring station is the new active monitor.

FRAMES
These are the frames the ring station copied from the ring.

M_BIT
This counter is decremented when a ring station receives a Beacon MAC frame from its NAUN. When this counter reaches B'0', the ring station activates the function to set the monitor bit in repeated Beacon MAC frames. If a Beacon MAC frame is received that this ring station's NAUN did not transmit, this counter is set to its threshold value.

RCVBEACON
A ring station decrements this counter when it receives a Beacon MAC frame from its nearest active downstream neighbor (NADN). When this counter reaches B'0', the ring station removes itself from the ring and tests the functioning of its adapter.

Note: Because only one item can be transmitted at a time, received information is not repeated during token-claiming or Beacon Transmit. However, a ring station may incorporate information from a received frame into the frames it is queueing for transmission.

States

01 Reset (RESET)
This state indicates the FSM is not active. This state is used as an entry state to this FSM.

02 Claim Token Transmit (CTX)
The ring station has entered token-claiming, is transmitting Claim Token MAC frames, and is waiting for a Claim Token MAC frame whose source address is greater than or equal to the ring station's individual address (SA $\geq$ STA).

03 Claim Token Repeat (CTR)
The ring station has received a Claim Token frame with a higher source address and is waiting for the token-claiming process to complete.

04 Beacon 2 Transmit (B2X) - Signal Loss
The ring station is transmitting type '2' Beacon MAC frames and is awaiting the termination of the error condition.

05 Beacon 3 Transmit (B3X) - Streaming
The ring station is transmitting type '3' Beacon MAC frames and is awaiting the termination of the error condition.

06 Beacon 4 Transmit (B4X) - Intermediate Streaming
The ring station is transmitting type '4' Beacon MAC frames and is awaiting the termination of the error condition.

07 Beacon 1 Repeat (B1R)
The ring station is repeating type '1' Beacon MAC frames.

08 Beacon n Repeat (BnR)
The ring station is repeating type 'n' Beacon MAC frames, where n = 2, 3, or 4.

INPUTS	RESET 01	CTX 02	CTR 03	B2X 04	B3X 05	B4X 06	B1R 07	BnR 08
T(escape) expires	/	/	/	/	/	/	2(A)	2(A)
Detect signal loss	2(A)	-	2(F)	-	4(Z)	4(Z)	2(A1)	2(A1)
CTFRAMES & participant & ATTACHED & AM_FUNCTIONAL_ADDRESS=B'0' (SA < STA)	2(A)	-	-	-	-	-	2(A1)	2(A1)
CTFRAMES & (not participant|not ATTACHED |AM_FUNCTIONAL_ADDRESS= B'1') (SA < STA)	3(B)	-	-	-	-	-	3(B1)	3(B1)
CTFRAMES M=B'0', NAUN=stored NAUN (SA = STA)	3(C)	-(J)	-(R)	-(R)	-(R)	-(R)	-	-
CTFRAMES M=B'1' (SA = STA)	3(C)	-	-(R)	-(R)	-(R)	-(R)	-	-
CTFRAMES, NAUN≠stored NAUN (SA = STA)	3(C)	3(K)	-(R)	-(R)	-(R)	-(R)	-	-
CTFRAMES (SA > STA)	3(B)	3(L)	-	-	-	-	3(B1)	3(B1)
Beacon type '1' (SA ≠ STA)	7(D)	7(M)	7(I1)	7(U)	7(U)	7(U)	-(E1)	7(E1)
Beacon type '2' (SA ≠ STA)	8(D)	8(M)	8(I1)	8(U)	8(U)	8(U)	8(E1)	-(E1)
Beacon type '3' (SA ≠ STA)	8(D)	8(M)	8(I1)	-	8(U)	8(U)	8(E1)	-(E1)
Beacon type '4' (SA ≠ STA)	8(D)	8(M)	8(I1)	-	-	8(U)	8(E1)	-(E1)
Beacon, M=B'1', from NAUN	8(D)	8(M)	8(I1)	-	-	-	2(A1)	2(A1)
Beacon type '2', M=B'0' (SA = STA)	8(E)	8(N)	8(J1)	2(W)	2(W)	2(W)	8(F1)	-(F1)
Beacon type '3', M=B'0' (SA = STA)	8(E)	8(N)	8(J1)	2(W)	2(W)	2(W)	8(F1)	-(F1)
Beacon type '4', M=B'0' (SA = STA)	8(E)	8(N)	8(J1)	2(W)	2(W)	2(W)	8(F1)	-(F1)
Beacon type '2', M=B'1' (SA = STA)	8(E)	8(N)	8(J1)	-	-	-	8(F1)	-(F1)
Beacon type '3', M=B'1' (SA = STA)	8(E)	8(N)	8(J1)	-	-	-	8(F1)	-(F1)
Beacon type '4', M=B'1' (SA = STA)	8(E)	8(N)	8(J1)	-	-	-	8(F1)	-(F1)
M_BIT counter = B'0'	/	/	/	/	/	/	-(H1)	-(H1)
T(beacon_transmit) expires	/	/	/	-(D1)	-(D1)	-(D1)	/	/
RCVBEACON counter = B'0'	/	/	/	/	/	/	-(Y)	-(Y)
T(claim_token) expires, CTFRAMES not received	/	5(O)	/	/	/	/	/	/
T(claim_token) expires, CTFRAMES received	/	6(O)	6(K1)	/	/	/	/	/
T(claim_token) expires & signal loss	/	4(O)	-	/	/	/	/	/
T(good_token) expires	2(A)	/	/	/	/	/	/	/
T(attach) expires	2(A)	-	-	/	/	/	/	/
T(receive_notification) expires	2(G)	/	/	/	/	/	/	/
T(ring_purge) expires	2(A)	/	/	/	/	/	/	/
T(transmit_pacing) expires	/	-(H)	/	-(X)	-(X)	-(X)	/	/
Ring purge (SA ≠ STA)	/	-	1(S)	/	-	/	/	/
Ring purge (SA = STA)	/	-(P)	1(T)	/	-	/	/	/
CTFRAMES_CNT = B'0'	/	1(Q)	/	/	/	/	/	/

OUTPUT CODE	FUNCTION
A	Insert Master Clock. Introduce 24-bit delay. Transmit a Claim Token MAC frame. Set CTFRAMES counter to its threshold value. Start T(claim_token). Start T(transmit_pacing).
B	Start T(claim_token). Reset Active_Monitor_Functional_Address to B'0'
C	Start T(claim_token). Reset Active_Monitor_Functional_Address to B'0' Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
D	Start T(escape).
E	Start T(escape). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
F	Restart T(claim_token). Transmit a Claim Token MAC frame. Set CTFRAMES counter to its threshold value. Start T(transmit_pacing). Insert Master Clock. Introduce 24-bit delay.
G	Purge pending Active Monitor Present and Standby Monitor Present MAC frames. Insert Master Clock. Introduce 24-bit delay. Transmit a Claim Token MAC frame. Set CTFRAMES counter to its threshold value. Start T(claim_token). Start T(transmit_pacing).
H	Transmit a Claim Token MAC frame. Start T(transmit_pacing).
J	Decrement CTFRAMES counter.
K	Restart T(claim_token). Cancel T(transmit_pacing). Queue Report Active Monitor Error MAC frame. Remove Master Clock Remove 24-bit delay
L	Restart T(claim_token). Cancel T(transmit_pacing). Remove Master Clock Remove 24-bit delay
M	Cancel T(claim_token). Cancel T(transmit_pacing). Remove Master Clock. Remove 24-bit delay. Start T(escape).
N	Cancel T(claim_token). Cancel T(transmit_pacing). Remove Master Clock. Remove 24-bit delay. Start T(escape). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.

OUTPUT CODE	FUNCTION
O	Transmit an appropriate Beacon MAC frame. Start T(beacon_transmit). Restart T(transmit_pacing).
P	Queue a Report Active Monitor Error MAC frame.
Q	Cancel T(claim_token). Cancel T(transmit_pacing). Indicate won token-claiming by setting STATION_WON_TOKEN_CLAIMING flag to B'1'. Reset REMOVE_RCV to 0. Reset REMOVE_TRANS to 0.
R	Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
S	Set RING_RECOVERED flag to B'1'. Set REMOVE_RCV flag to B'0'. Set REMOVE_TRANS flag to B'0'. Cancel T(claim_token).
T	Set RING_RECOVERED flag to B'1'. Set REMOVE_RCV flag to B'0'. Set REMOVE_TRANs flag to B'0'. Cancel T(claim_token). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
U	Cancel T(beacon_transmit). Cancel T(transmit_pacing). Remove Master Clock. Remove 24-bit delay. Start T(escape).
V	Cancel T(beacon_transmit). Cancel T(transmit_pacing). Remove Master Clock. Remove 24-bit delay. Start T(escape). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
W	Cancel T(beacon_transmit). Insert Master Clock. Introduce 24-bit delay. Transmit a Claim Token MAC frame. Start T(claim_token). Restart T(transmit_pacing). Set CTFRAMES counter to its threshold value.
X	Transmit a Beacon MAC frame. Start T(transmit_pacing).
Y	If REMOVE_RCV is reset to B'0' then Remove from the ring. Initiate self-test. Stop T(soft_error). Reset SOFT_ERROR. Set REMOVE_RCV flag to B'1'. (*This indicates that the ring station has tested during this beacon process.*)
Z	Transmit a Beacon type '2' MAC frame. Start T(transmit_pacing). Restart T(beacon_transmit).

OUTPUT CODE	FUNCTION
A1	Cancel T(escape). Transmit a Claim Token MAC frame. Start T(claim_token). Start T(transmit_pacing). Set CTFRAMES counter to its threshold value. Insert Master Clock Introduce 24-bit delay.
B1	Cancel T(escape). Start T(claim_token).
C1	Cancel T(escape). Start T(claim_token). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
D1	If REMOVE_TRANS is reset to B'0' then Remove from the ring. Initiate self-test. Stop T(soft_error). Reset SOFT_ERROR. Set REMOVE_TRANS flag to B'1'. (*This indicates that the ring station has tested during this beacon process.*)
E1	Restart T(escape).
F1	Restart T(escape). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
H1	Set MB flag to B'1'. (*This activates the setting of the monitor bit in Beacon from NAUN.*) Set REMOVE_TRANS flag to B'1'. (*The NADN of the beaconing ring station does not remove and test.*)
I1	Cancel T(claim_token). Start T(escape).
J1	Cancel T(claim_token). Start T(escape). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
K1	Transmit appropriate beacon. Start T(beacon_transmit). Restart T(transmit_pacing). Insert Master Clock. Introduce 24-bit delay.

Receiving Finite State Machine

Function This FSM describes the actions performed by the receiving side of a ring station. The outputs of this FSM are passed to the Token Transmission FSM, the MAC Recovery FSM, the Station Attachment FSM, and the Monitor Functions FSM, as illustrated in Figure 7-1 on page 7-1.

This FSM is called from the Routing Logic for FSMs for a particular input. Once the input is processed, control is returned to the Routing Logic, and it specifies other actions to take.

Input The input consists of signal input from the ring and is expressed in terms this ring station recognizes, such as SD and ED.

States

01 Repeat mode (REPEAT)
All inputs, whether modified or unmodified in this state, are repeated (output to the Token Transmission FSM).

02 Starting delimiter received (SD)
Once an SD is received, the machine looks for an indication that this ring station either sent the frame or should receive it. In any case, it continues repeating to the token FSMs.

03 Copying (COPY)
The input, a frame addressed to this ring station, is routed to the processing FSMs, as well as to the token FSMs. Entry into this state represents the passing of the input to the processing FSMs.

04 Stripping (STRIP)
The input, a frame transmitted by this ring station, is routed to the token FSMs (to check for reservations), as well as to the attachment FSM. Like state 03, this state passes the input to the processing FSMs.

05 Copying/Stripping (C/S)
In this state the ring station is both stripping and receiving a frame. This could be the case when a ring station sends a frame to itself (Duplicate Address Test).

INPUTS	REPEAT 01	SD 02	COPY 03	STRIP 04	C/S 05
Starting delimiter	2(A)	-(A)	2(A)	2(I)	2(I)
DA=STA	/	3(C)	/	/	/
SA=STA	/	4	5	/	/
Ending delimiter on byte boundary	-	1(O)	1(E)	1(I)	1(M)
Abort delimiter	-	1	1(F)	1(J)	1(J)
Burst 4 error	-	1	1(F)	1(J)	1(J)
Burst 5 error	-(B)	1(B)	1(G)	1(K)	1(K)
Code violation error	-	-(D)	-(H)	-(L)	-(L)
Ending delimiter on non-byte boundary	-	1(P)	1(P)	1(Q)	1(Q)
Bad CRC	-	-(D)	-(H)	-(L)	-(L)
T(physical_trailer) expires	-(N)	1(N)	/	1(N)	1(N)

OUTPUT CODE	FUNCTION
A	Reset frame synchronization.
B	Convert the signal to idles starting at bit 5 of the burst error.
C	Set indicator (SET_A). (*This specifies that the frame is recognized by this ring station.*)
D	Note error condition to set error-detected bit in ED.
E	Set frame-copied bits (C) in the frame status field to B'1'. If the SET_A indicator is set and the address-recognized bits (A) ≠ B'1' then Set the A bits in the frame status field to B'1'.
F	Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)
G	Convert input to idles starting at bit 5 of the burst error. Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)
H	Note error condition to set error-detected bit in ending delimiter. Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)
I	Cancel T(physical_trailer). Reset frame synchronization.
J	Cancel T(physical_trailer). Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)
K	Convert input to idles starting at bit 5 of the burst error. Cancel T(physical_trailer). Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)
L	Note error condition to set error-detected bit in ending delimiter. Cancel T(physical_trailer). Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)
M	Set frame-copied bits (C) in the frame status field to B'1'. If the SET_A indicator is set and the address-recognized bits (A) ≠ B'1' then Set the A bits in the frame status field to B'1'. Cancel T(physical_trailer).
N	Increment LOST_FRAME counter.
O	If error condition detected, set error-detected bit in ending delimiter.
P	Set error-detected bit in ending delimiter. Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)
Q	Set error-detected bit in ending delimiter. Cancel T(physical_trailer). Reset indicator (SET_A). (*This specifies that the frame is not recognized by this ring station due to a detected error.*)

Station Attachment Finite State Machine

Function

This FSM describes the actions of a ring station attaching to the ring. The relationship of this FSM with the other FSMs is shown in Figure 7-1 on page 7-1.

This FSM is called from the Routing Logic for FSMs for a particular input. Once the input is processed, the logic specifies other actions to take.

Input

ACTIVATE_MAC
Command received from DLC.LAN.MGR to activate the MAC sub-layer and attach to the ring.

DFC
The Duplicate Address Test Frame Counter (DFC) is initialized to DFC = n≥B'1'. When a Duplicate Address Test frame is stripped with AC≠B'00', DFC is decremented and another Duplicate Address Test frame is queued. If DFC reaches B'0', the address is assumed to be a duplicate.

FRAMES
Copied or stripped from the ring.

NDFC
The Non-Duplicate Address Frame Counter (NDFC) is initialized to NDFC = m≥B'1'. When a Duplicate Address Test frame is stripped with AC = B'00', NDFC is decremented and another Duplicate Address Test frame is queued. If NDFC reaches B'0', the address is assumed to be unique.

NEIGHBOR_NOTIFICATION
This is a flag that is set in the Station Attachment FSM to indicate that an Active Monitor Present or Standby Monitor Present MAC frame with AC = B'00' was previously received (and the NAUN stored). In this case, the ring station can respond immediately with a Standby Monitor Present MAC frame, and need not wait for the next neighbor notification cycle. This flag is initialized to B'0' and is set to B'1' when the ring station receives an Active Monitor Present or Standby Monitor Present MAC frame.

NOPSC
The No Parameter Server Counter (NOPSC) is initialized to NOPSC = p≥1. When a Request Parameters frame is stripped with AC = B'00', NOPSC is decremented and another Request Parameters frame is queued. If NOPSC reaches B'0', no ring parameter server is present, the default values are used, and the ring station participates in normal ring operation.

RRTRYC
The Response Retry Counter (RRTRYC) is initialized to RRTRYC = q≥B'1'. If T(response) expires, RRTRYC is decremented and another Request Parameters frame is queued.

States

01 Disconnected (DISC)
Not physically connected to the ring.

02 Self Test (TEST)
To test the ring station for proper functioning.

03 Monitor Check (MCK)
To determine the presence of an active monitor.

04 Claim Token (CLTK)
There is currently no active monitor on this ring; this ring station is participating in token-claiming to choose a new active monitor.

05 Duplicate Address Check (ADCK)
To make sure the ring station's individual address is unique.

06 Neighbor Notification (NNOT)
For the ring station to determine its nearest active upstream neighbor (NAUN) and to identify itself to its nearest active downstream neighbor (NADN).

07 Request Initialization (INIT)
To acquire configuration parameters from a ring parameter server if one exists, and to register its configuration parameters with network management.

08 Reset (RESET)
This state indicates the ring station has attached to the ring and this FSM is no longer active.

INPUTS	DISC 01	TEST 02	MCK 03	CLTK 04	ADCK 05	NNOT 06	INIT 07	RESET 08
Receive ACTIVATE_MAC (DLC.LAN.MGR)	2(A)	-	-	-	-	-	-	-(F)
SELF_TEST_SUCCESSFUL = B'1' (test successful)	/	3(C)	/	/	/	/	/	/
SELF_TEST_SUCCESSFUL = B'0' (test unsuccessful)	/	1(Y)	/	/	/	/	/	/
RING_RECOVERED flag = B'1'	/	/	/	5(E)	-	-	-	-
Copy Active Monitor Present | Standby Monitor Present (A=B'0')	/	/	5(D)	5(D)	-(K)	7(P)	-	-
Copy Active Monitor Present | Standby Monitor Present (A≠B'0')	/	/	5(E)	5(E)	-	-	-	-
Ring station won token-claiming	/	/	/	5(E)	-	7(Q)	-	-
Detect NEIGHBOR_NOTIFICATION flag set	/	/	-	-	-	7(R)	-	/
Copy Ring Purge MAC frame	/	/	5(E)	-	-	-	-	-
T(attach) Expires	/	/	4(X)	-	1(B)	1(B)	1(B)	/
Strip Duplicate Address Check (A=B'0')	/	/	/	/	-(M)	/	/	/
Strip Duplicate Address Check (A≠B'0')	/	/	/	/	-(N)	/	/	/
Detect NDFC=B'0'	/	/	/	-	6(O)	/	-	/
Detect DFC=B'0'	/	/	/	-	1(B)	/	-	/
Copy Set Parameters (DA=STA)	/	/	-	-	-	-	8(S)	-
Strip Request Initialization (A=B'0')	/	/	/	/	/	/	-(T)	/
Strip Request Initialization (A≠B'0')	/	/	/	/	/	/	-(U)	/
Detect NOPSC=B'0'	/	/	-	-	-	-	8(V)	/
T(response) expires	/	/	/	/	/	/	-(W)	-
Detect RRTRYC=B'0'	/	/	/	-	-	-	1(B)	/
Copy Beacon MAC frame	/	/	1(B)	1(B)	1(B)	1(B)	1(B)	-
Detect signal loss	/	/	/	/	1(B)	1(B)	-	-
T(claim_token) expires	/	/	/	1(B)	1(B)	1(B)	-	-
T(ring_purge) expires	/	/	/	1(B)	1(B)	1(B)	1(B)	-

OUTPUT CODE	FUNCTION
A	Initialize NEIGHBOR_NOTIFICATION Flag to B'0', NDFC=m, DFC=n, NOPSC=p, RRTRYC=q. Initiate self-test procedures.
B	Remove from the ring. Report MAC_ACTIVATED failure to DLC.LAN.MGR.
C	Attach to the ring. Start T(attach).
D	Restart T(attach). Record NAUN. Set NEIGHBOR_NOTIFICATION Flag to B'1'. Queue a Duplicate Address Test MAC frame.
E	Restart T(attach). Queue a Duplicate Address Test MAC frame.
F	Report error to DLC.LAN.MGR; ring station is already active.
K	Record NAUN. Set NEIGHBOR_NOTIFICATION Flag to B'1'.
M	Decrement NDFC. If NDFC≠B'0' and DFC≠B'0' then Queue a Duplicate Address Test MAC frame.
N	Decrement DFC. If NDFC≠B'0' and DFC≠B'0' then Queue a Duplicate Address Test frame.
O	Restart T(attach).
P	Record NAUN. If not the active monitor then Queue Standby Monitor Present frame. Queue Report NAUN Change MAC frame. Restart T(attach). Queue a Request Initialization MAC frame.
Q	Queue Active Monitor Present MAC frame. Start T(neighbor_notification).
R	Queue Standby Monitor Present MAC frame. Queue Report NAUN Change MAC frame. Restart T(attach). Queue a Request Initialization MAC frame.
S	Set appropriate parameters in the ring station. Cancel T(response). Cancel T(attach). Queue Response MAC frame. Report MAC_ACTIVATED success to DLC.LAN.MGR. Set ATTACHED flag.
T	Decrement NOPSC. If NOPSC≠0 then Queue a Request Initialization frame.
U	Start T(response).
V	Use station default values. Report MAC_ACTIVATED success to DLC.LAN.MGR. Cancel T(response). Cancel T(attach). Set ATTACHED flag.
W	Decrement RRTRYC. If RRTRYC≠0 then Queue a Request Parameters frame.
X	Insert Master Clock. Introduce 24-bit delay.
Y	Report MAC_ACTIVATED failure to DLC.LAN.MGR

Monitor Functions Finite State Machine

Function	This FSM describes the actions of the active monitor and standby monitor protocols. See Figure 7-1 on page 7-1 for the FSM relationships. This FSM represents the actions taken while a ring station is the active or standby monitor. This FSM is called from the Routing Logic for FSMs for a particular input. Once the input is processed, control is returned to the Routing Logic, and it specifies other actions to take.
Input	**FRAMES** Copied from the ring. **NEIGHBOR_NOTIFICATION** This flag is set to B'0' to indicate that neighbor notification is *not* complete, and to B'1' to indicate that it *is* complete. **RING_RECOVERED** This flag is set by the MAC Recovery FSM to indicate that token-claiming has been successfully completed, and the ring station has become a standby monitor. **STATION_WON_TOKEN_CLAIMING** This flag indicates that this ring station has actively participated in the token-claiming process and has won. After successfully purging the ring, the ring station will activate the active monitor function. The input consists of received frames and timer expirations.
States	**01 Reset (RESET)** This state indicates the FSM is not active. This state is used as an entry state to this FSM. **02 Ring Purge (PURGE)** The active monitor has initiated the ring purge process. This state is unique to the Active Monitor FSM. **03 Active Monitor Functions (ACTIVE)** The active monitor has successfully purged the ring. All active monitor functions (except ring purge) are conducted while in this state. **04 Standby Monitor Functions (STANDBY)** The actions of a standby monitor are described in this state.

INPUTS	RESET 01	PURGE 02	ACTIVE 03	STANDBY 04
Monitor bit = B'1' in access control field of frame or priority token	-	-	2(C)	-
Monitor bit = B'0' in access control field of frame or priority token	-	-	-(Q)	-
Claim Token MAC frame	-	-	1(R)	1(V)
STATION_WON_TOKEN_CLAIMING	2(A)	/	/	/
Ring Purge MAC frame (SA≠STA) & ATTACHED	4(B)	4(H)	4(H)	-
Ring Purge MAC frame (SA≠STA) & not ATTACHED	-	1(I)	1(Y)	/
Ring Purge MAC frame (SA=STA) & ATTACHED	4(N)	3(D)	4(P)	-(U)
Ring Purge MAC frame (SA=STA) & not ATTACHED	-	3(D)	1(Y)	/
T(neighbor_notification) expires and NEIGHBOR_NOTIFICATION flag = B'1'	/	/	-(J)	/
T(neighbor_notification) expires and NEIGHBOR_NOTIFICATION flag = B'0'	/	/	-(K)	/
Active Monitor Present MAC frame (AC=B'00')	-	-	-(W)	-
Active Monitor Present MAC frame (AC≠B'00')	-	-	-(X)	-
Standby Monitor Present MAC frame (AC=B'00')	-	-	-(L)	-
Standby Monitor Present MAC frame (AC≠B'00')	-	-	-(M)	-
Active Monitor Present MAC frame (SA≠STA) & ATTACHED	-	-	4(H)	-
Active Monitor Present MAC frame (SA≠STA) & Not ATTACHED	-	-	1(Y)	/
Receive Beacon MAC frame	-	1(F)	1(Z)	1(V)
T(any_token) expires	/	-	2(O)	/
T(good_token) expires	/	/	/	1(V)
T(ring_purge) expires	/	1	/	/
T(receive_notification) expires	/	/	1(Z)	1(V)
Ring station ATTACHED and ACTIVE_MONITOR	4(B)	/	/	/
Detect signal loss	-	1(F)	1(Z)	1(V)
RING_RECOVERED flag is set and ATTACHED	4(B)	/	/	/

OUTPUT CODE	FUNCTION
A	Transmit a Ring Purge MAC frame. Start T(ring_purge).
B	Restart T(good_token). Restart T(receive_notification).
C	Transmit a Ring Purge MAC frame. Start T(ring_purge).
D	Issue token with the priority reservation of the Ring Purge MAC frame. Queue Active Monitor Present MAC frame. Queue Report New Active Monitor MAC frame. Set Active Monitor functional address. Start T(neighbor_notification). Start T(receive_notification). Start T(any_token). Cancel T(ring_purge).
F	Cancel T(any_token). Cancel T(neighbor_notification). Turn off Active Monitor functional address. Remove Master Clock. Remove 24-bit delay.
H	Remove Master Clock. Remove 24-bit delay. Start T(good_token). Start T(receive_notification). Cancel T(any_token). Cancel T(neighbor_notification). Queue Report Active Monitor Error MAC frame. Turn off Active Monitor functional address.
I	Cancel T(any_token). Cancel T(neighbor_notification). Queue Report Active Monitor Error MAC frame. Turn off Active Monitor functional address.
J	Restart T(neighbor_notification). Queue Active Monitor Present MAC frame. Reset NOT_COMPLETE flag to B'0'.
K	Restart T(neighbor_notification). Queue Active Monitor Present MAC frame. Queue Report Neighbor Notification Incomplete MAC frame.
L	Indicate completion of neighbor notification by setting NOT_COMPLETE flag to B'1'.
M	Note the source address of the frame (for the error report if necessary).
N	Restart T(good_token). Restart T(receive_notification). Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
O	Transmit a Ring Purge MAC frame. Start T(ring_purge). Increment TOKEN_ERROR counter.
P	Start T(good_token). Start T(receive_notification). Cancel T(any_token). Cancel T(neighbor_notification). Queue Report Active Monitor Error MAC frame. Turn off active monitor functional address. Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.

OUTPUT CODE	FUNCTION
Q	Set the monitor bit in repeated frame or priority token to B'1'.
R	Cancel T(any_token). Cancel T(neighbor_notification). Cancel T(receive_notification). Queue Report Active Monitor Error frame. Remove Master Clock. Remove 24-bit delay.
U	Note that another ring station seems to be transmitting with the same source address. While removal from the ring and reattachment to the ring would provide another duplicate address test, no action is mandated.
V	Cancel T(good_token). Cancel T(receive_notification).
W	Set NOTIFICATION_COMPLETE Flag. Restart T(receive_notification). Set SINGLE_STATION Flag.
X	Note the source address of the frame (for the error report, if necessary). Restart T(receive_notification). Reset SINGLE_STATION Flag.
Y	Cancel T(any_token). Cancel T(neighbor_notification). Cancel T(receive_notification). Queue Report Active Monitor Error MAC frame. Turn off Active Monitor functional address. Remove Master Clock. Remove 24-bit delay.
Z	Cancel T(any_token). Cancel T(neighbor_notification). Cancel T(receive_notification). Queue Report Active Monitor Error MAC frame. Turn off Active Monitor functional address. Remove Master Clock. Remove 24-bit delay.

Finite State Machines for Priority Operation

The two finite state machines on the following pages describe the operation of ring stations supporting priority. The Token Transmitting FSM describes the generation of tokens when a ring station receives either a token or one of its own frames, and the repeating of frames that have been transmitted by other ring stations. The Frame Transmission FSM describes how pending frames are transmitted and how reservations are made in tokens and in the frames of other ring stations.

The Token Transmission FSM and the Frame Transmission FSM are distinct and independent, except that outputs of the Token Transmission FSM are used as inputs by the Frame Transmission FSM (Figure 7-1 on page 7-1). If the input to a Token Transmission FSM is a frame transmitted by another ring station, the frame is repeated unaltered. If the input is either a token or a frame being stripped, then a token is issued.

The Frame Transmission FSM receives from the Token Transmission FSM either a token or a repeated frame. (Incoming idle or abort sequences are merely repeated.) It can repeat a token or frame unaltered, transmit either one with a new priority reservation, or capture the token and transmit a pending frame of its own.

Because all ring stations must be able to issue priority tokens, each ring station must implement the Token Transmission FSM.

Token Transmission Finite State Machine

Function

This FSM describes the actions of a ring station to generate tokens and to repeat frames transmitted by other ring stations. Token generation can occur only when a ring station receives a token or is stripping its own frame. Token generation is not related to making reservations for a priority token and is independent of new frame transmission. The frames and tokens that are generated are issued to another FSM in the ring station (see "Frame Transmission Finite State Machine" on page 7-32) along with an indication (the Permissible Token Indicator, or PTI) if the token can be used by the same ring station to transmit a frame. Because a ring station must relinquish a token after using it once, tokens generated after a ring station strips its own frame cannot be used to transmit another frame, unless the new token is of higher priority than the one used to transmit the frame being stripped.

This FSM is called from the Routing Logic for FSMs for a particular input. Once the input is processed, control is returned to the Routing Logic, which specifies other actions to take.

Input

The input consists of received frames that the ring station transmitted (and is now stripping), received frames that other ring stations transmitted (which are repeated unaltered) or received tokens. The input also includes the priority and reservation bits that those frames or tokens carry.

The notation T(x,y), FS(x,y), and FR(x,y) describes a received token (if an input) or an issued token (if an output), the frame being stripped, and the frame being repeated, respectively, with transmission priority x and priority reservation y. The priority and reservation indicators, and additional stored variables, are defined below.

Received Priority (Pr)
The priority of the received token or frame.

Received Reservation (Rr)
The priority from the reservation field of the received token or frame.

State-Associated Stored Received Priority (Spi, i = 1 through 5)
The priority of the token stored upon entering state i, before the origination of a higher priority token. Sp1 is defined as B'000'.

State-Associated Stored Transmitted Priority (Sxi, i = 1 through 5)
The priority of the token originated and transmitted upon entering state i. Sx1 is defined as B'000'.

The values (Spi,Sxi), i = 2 through 5, are stored as a pair, so that when a token of priority Sxi returns to the sender, it will issue a token of the original priority, Spi, and return to state (i-1). Whenever a state transition occurs (up or down), the values of Sxi and Spi used in the output codes assume that the state variable i is that of the higher state.

States		
	1 NPH	Not in Priority-Hold
	2 PH1	Priority-Hold with 1 transition stored
	3 PH2	Priority-Hold with 2 transitions stored
	4 PH3	Priority-Hold with 3 transitions stored
	5 PH4	Priority-Hold with 4 transitions stored

INPUTS	NPH 01	PH1 02	PH2 03	PH3 04	PH4 05
FR(Pr,Rr) & Pr≥Sxi	-(A)	-(A)	-(A)	-(A)	-(A)
FR(Pr,Rr) & Pr<Sxi	/	1(A)	1(A)	1(A)	1(A)
FS(Pr,Rr) & Rr≤Pr	-(B)	-(B)	-(B)	-(B)	-(B)
FS(Pr,Rr) & Rr>Pr	2(C)	3(C)	4(C)	5(C)	/
T(Pr,Rr) & Pr>Sxi	-(D)	-(D)	-(D)	-(D)	-(D)
T(Pr,Rr) & Pr=Sxi & Rr>Spi	-(D)	-(F)	-(F)	-(F)	-(F)
T(Pr,Rr) & Pr=Sxi & Rr≤Spi	-(D)	1(G)	2(G)	3(G)	4(G)
T(Pr,Rr) & Pr<Sxi	/	1(D)	1(D)	1(D)	1(D)

OUTPUT CODE	FUNCTION
A	FR(Pr,Rr) [Repeat the frame unaltered.]
B	T(Pr,Rr) [Issue a token with no change in priority.] Reset Permissible Token Indicator (PTI) to B'0'.
C	Spi=Pr [Store a new transition.] Sxi=Rr [Store a new transition.] T(Rr,0) [Issue a token at a requested, higher priority.] Set Permissible Token Indicator (PTI) to B'0'.
D	T(Pr,Rr) [Repeat the token unaltered.] Set Permissible Token Indicator (PTI) to B'1'.
F	Sxi=Rr [Store a new transition.] T(Rr,0) [Issue a token at a requested, higher priority.] Set Permissible Token Indicator (PTI) to B'0'.

G	If this ring station has no frames pending transmission at priority Pr then
	T(Spi,Rr) [Issue a token at a stored, lower priority.]
	Reset Permissible Token Indicator (PTI) to B'0'.
	Else
	Set Permissible Token Indicator (PTI) to B'1'.

Frame Transmission Finite State Machine

Function

This FSM describes the actions of a ring station to transmit frames or make reservations for priority tokens, when a frame is pending transmission. When no frame is pending, the ring station acts as a repeater of all tokens and frames.

Although not every ring station will be configured to transmit priority frames, this FSM applies to all ring stations. If the ring station does not transmit priority frames, then all of its pending frames are of priority B'000', and the FSM continues to be correct.

This FSM assumes that the ring station has already dealt with any received tokens, as described by the finite state machine "Token Transmission Finite State Machine" on page 7-30. Therefore, the tokens taken as input have come from another FSM in the ring station, not directly off the ring, as illustrated in Figure 7-1 on page 7-1.

This FSM is called from the Routing Logic for FSMs for a particular input. Once the input is processed, control is returned to the Routing Logic, and it specifies other actions to take.

Input

The input consists of either received frames that other ring stations have transmitted, or tokens issued by the token-handling FSM, and the priority and reservation bits that they carry. The FSM also receives a Permissible Token Indicator (PTI) from the preceding token FSM. The PTI must equal B'1' for a pending frame to be transmitted. The notation T(x,y), FR(x,y), and NF(x,y) describes a received token (if an input) or a transmitted token (if an output), the frame being repeated (from another ring station), and the new frame (originated by this station), respectively, with transmission priority x and priority reservation y. The priority, reservation, and permissible token are defined below.

Received Priority (Pr)
The priority of the received token or frame.

Received Reservation (Rr)
The priority from the reservation field of the received token or frame.

Pending Priority (Pp)
The priority of a frame pending transmission (waiting for a usable token).

Permissible Token Indicator (PTI)
An indication, when PTI= B'1', that the token may be used to transmit a pending frame.

State

1 FXFP
Frame Transmission: Full Priority

INPUTS	FXFP 01
T(Pr,Rr) & Pr>Pp & Rr≥Pp	-(A)
T(Pr,Rr) & Pr>Pp & Rr<Pp	-(B)
T(Pr,Rr) & Pr<Pp & PTI=B'0'	-(A)
T(Pr,Rr) & Pr<Pp & PTI=B'1'	-(C)
T(Pr,Rr) & Pr=Pp & PTI=B'0'	-(A)
T(Pr,Rr) & Pr=Pp & PTI=B'1'	-(C)
FR(Pr,Rr) & Rr<Pp	-(E)

OUTPUT CODE	FUNCTION
A	T(Pr,Rr) [Repeat the token unaltered.]
B	T(Pr,Pp) [Repeat the token with a new reservation.]
C	NF(Pr,0) [Transmit the pending new frame.]
E	FR(Pr,Pp) [Repeat the frame with a new reservation.]

Part 3. The Logical Link Control (LLC) Sub-Layer

Chapter 8. LLC Frames

The variable-length information field contains data in an integral number of bytes, exchanged between higher-level layers (above the MAC sub-layer), or LAN-level management information. For LLC transmissions, the information field is called an LLC Protocol Data Unit (LPDU); its format is shown below:

LPDU Format

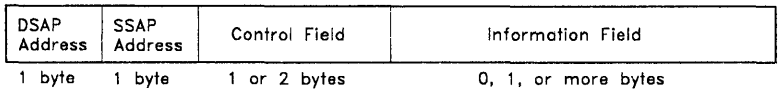
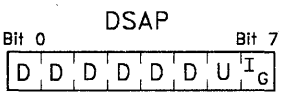


Figure 8-1. LPDU Format

DSAP Address Field

The destination service access point (DSAP) address field identifies the service access point (SAP) for which the LPDU is intended. (In this reference, a service access point is the logical point at which a Path Control component acquires the services of the Data Link Control layer [see page 4-1].) The DSAP address field is a single byte with the format shown below:

DSAP



D = DSAP Address Bits
U = User-Defined Address Bit
I/G = Individual/Group Bit

Figure 8-2. DSAP Address

The 6 DSAP Address Bits (D) and the User-Defined Address Bit (U) form the address of the SAP for which the LPDU is intended. The U bit indicates whether the address is defined by the user (B'0') or by the IEEE (B'1').

The Individual/Group Bit (I/G) indicates whether the address is an individual address (B'0') or group address (B'1').

IEEE-Defined DSAP Addresses

Null SAP (X'00')

This address provides some ability to respond to remote nodes even when no service access point (SAP) has been activated. This SAP supports only connectionless service, and responds only to XID and TEST command LPDUs.

LLC Sub-Layer Management Individual SAP (X'02')

The individual LLC sub-layer management SAP is reserved for future use by IEEE-standardized network management entities.

Network Management Function SAPs (X'x2', where x ≠ 0)

These SAPs are reserved for future use by IEEE-standardized network management entities.

LLC Sub-Layer Management Group SAP (X'03')

The group LLC sub-layer management SAP is reserved for future use by IEEE-standardized network management entities.

Department of Defense Internet Protocol SAP (X'06'; military standard 1777)

This SAP is reserved for the use of the US Department of Defense Internet Protocol.

National Standards Bodies SAP (X'x6', where x ≠ 0)

These SAPs are reserved for assignment by national standards bodies.

ISO Network Layer SAP (X'FE')

This SAP is reserved for the use of the ISO network layer.

Global SAP (X'FF')

The global SAP address, when used as the DSAP address, indicates that copies of the LPDU are to go to each active SAP in this DLC.LAN.

Bridge Spanning Tree Protocol SAP (X'42')

This SAP is used by the Spanning Tree Protocol.

User-Defined DSAP Addresses

SNA Path Control Individual SAP (X'04')

This is a default individual SAP address, used by SNA nodes, that identifies Path Control as the data link user. In a request to DLC.LAN.MGR to establish a link with an adjacent link station, if a SAP address (either local or remote) is not specified, this SAP is used. When multiple links between two SNA nodes using the same adapters are required, only one link may use this SAP as both the local and remote SAP addresses, because links are uniquely defined by the destination and source MAC station addresses, and the DSAP and SSAP addresses.

SNA Path Control Group SAP (X'05')

This group SAP provides a way to route an LPDU to all SNA SAPs without identifying them beforehand. Each individual SAP that has been identified to the LLC as belonging to this group SAP receives a copy of the LPDU and the addressing and routing information.

NETBIOS SAP (X'F0')

This SAP is used for all LLC communication that is driven by NETBIOS emulation.

Local Area Network Management Individual SAP (X'F4')

This SAP provides the default individual SAP address for LAN management functions when they communicate at the LLC level. For information on network management, see Chapter 13, "Network Management."

Local Area Network Management Group SAP (X'F5')

This group SAP provides the default group SAP address for LAN management functions.

Remote Program Load (RPL) SAP (X'F8')

This SAP provides a default SAP address to be used for a remote program load (RPL) procedure.

User Defined SAPs (X'8y' – X'9C', where y≠ B'xx1x')

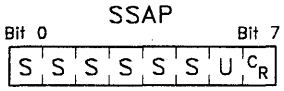
These SAPs are for non-standard, non-architectural higher-layer protocols and are to prevent inter-operability problems with standard and architectural protocols.

Reserved SAP (X'Fy', where y ≠ B'xx1x')

These SAPs are reserved by IBM for future use.

SSAP Address Field

The source service access point (SSAP) address field identifies the SAP that originated the LPDU. The SSAP address field is a single byte with the format shown below:



S = SSAP Address Bits
U = User-Defined Address Bit
C/R = Command/Response Bit

Figure 8-3. SSAP Address

The six SSAP Address Bits (S) and the User-Defined Address Bit (U) identify the address of the SAP that originated the LPDU. The U bit indicates whether the address is defined by the user (B'0') or by the IEEE (B'1').

The Command/Response Bit (C/R) indicates whether the LPDU is a command (B'0') or a response (B'1'). When processing the SSAP address in received frames, this bit is not considered part of the address.

Control Field

The LPDU Control Field contains the commands and responses, sequence numbers, and the Poll/Final bit for extended asynchronous balanced mode of operation.

Connection-Oriented Command and Response Repertoire

LLC link stations can send and receive the following command and response LPDUs:

COMMANDS	COMMANDS/RESPONSES	RESPONSES
I [1]		I [1]
	REJ [2]	
	RNR [2]	
	RR [2]	
SABME [3]		UA
DISC [3]		DM
TEST [3] [5]		TEST [5]
XID [3] [5]		XID [5]
		FRMR [4]

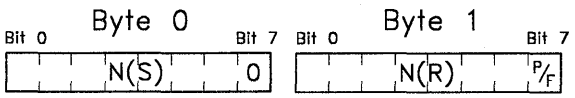
Figure 8-4. Connection-Oriented LPDU Command and Response Repertoire

Notes:

1. Link stations must accept an I-format LPDU sent as a command or as a response, with the P/F bit set to either B'0' or B'1', provided the LPDU is in sequence and is otherwise valid.
2. S-format command LPDUs are always sent with the P bit set to B'1'. When sent as responses, the F bit is set to B'0' or B'1', as appropriate. Link stations must accept S-format command LPDUs sent with the P bit set to either B'0' or B'1'.
3. U-format command LPDUs are always sent with the P bit set to B'1'. When sent as responses, the F bit is set to B'0' or B'1', as appropriate. Link stations must be able to receive U-format command LPDUs with the P bit set to either B'0' or B'1'.
4. The transmission of an FRMR response LPDU is required when Frame Reject exception conditions occur.
5. Support of XID and TEST LPDUs is required for SNA. It is acceptable for XID and TEST LPDUs to be implemented in such a way that they are decoupled from LPDUs sent by link stations.

Information Transfer Format

The information transfer format (I-format) command and response LPDU transfer sequentially-numbered LPDUs containing information fields across links. The format of the I-format LPDU control field is shown below:



N(S) = Transmitter Send Sequence Number
N(R) = Transmitter Receive Sequence Number
P/F = Poll/Final Bit

Figure 8-5. I-Format LPDU Control Field

- The link station sets the Transmitter Send Sequence Number [N(S)] to the value of the Send State Variable [V(S)] for the specified link (see "Send State Variable, V(S)" on page 8-13).

- The link station sets the Transmitter Receive Sequence Number [N(R)] to the current value of the Receive State Variable [V(R)] for the specified link (see "Receive State Variable, V(R)" on page 8-13). N(R) is therefore the expected sequence number of the next received I-format LPDU on that link. N(R) indicates that the link station has received correctly all I-format LPDUs numbered up through N(R)-1 on the specified link.
- The Poll/Final (P/F) bit is called the P bit in command LPDUs and the F bit in response LPDUs. (Whether the LPDU is a command or a response is indicated by bit 7 of the SSAP address field.) It is set in command LPDUs (P=B'1') to request that the remote link station send a response with this bit set (F=B'1'). There should be only one response received with F=B'1' for every command sent with P=B'1'. All link stations must be able to receive an I-format LPDU as a command and as a response, with the P/F bit set to either B'0' or B'1'.

In most cases, link stations send I-format LPDUs as commands with the P bit set to B'0'. However, link stations can also send the following I-format LPDUs:

- A link station can send an I-format LPDU as a command with the P bit set to B'1' during the dynamic window algorithm (see "The Dynamic Window Algorithm" on page 11-23), to force other stations to acknowledge received frames before timer T2 expires, and before count limit N3 is reached. (See "System Parameters" on page 11-3 for descriptions of T2 and N3.)
- A link station can send an I-format LPDU as a response with the F bit set to B'1' to acknowledge the receipt of a command LPDU with the P bit set to B'1', provided the station has an I-format LPDU waiting to be transmitted. This I-format response LPDU indicates the clearance of a busy condition at the sending link station.

Only one I-format or S-format command LPDU with the P bit set to B'1' can be outstanding in a given direction at a given time on the link. The link station must receive an I-format or S-format response LPDU with the F bit set to B'1' before it can issue another I-format or S-format command LPDU with the P bit set to B'1' on the same link. If no I-format or S-format response LPDU with the F bit set to B'1' is received before system-defined timer T1 expires, the link station can send an S-format command LPDU with the P bit set to B'1', for recovery purposes.

Transmission of an I-format or S-format command LPDU with the P bit set to B'1' initiates a *checkpoint cycle*. Link stations cannot send I-format LPDUs during checkpoint cycles that are:

- Initiated by the expiration of Ti (the inactivity timer)
- Initiated by the expiration of T1 (the reply timer).

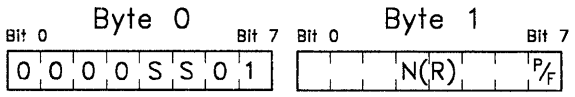
Link stations can send I-format LPDUs during checkpoint cycles initiated for other reasons (for example, during dynamic window algorithm).

Each I-format LPDU is sequentially numbered between 0 and 127. The maximum number of sequentially numbered I-format LPDUs that can be outstanding (unacknowledged) in a given direction on a link at any given time cannot exceed 127. This prevents ambiguity in the association of sent I-format LPDUs with sequence numbers during normal operation and error recovery actions.

Supervisory Format

Supervisory format (S-format) LPDUs perform supervisory control functions, such as acknowledging I-format LPDUs, requesting retransmission of I-format LPDUs, and requesting temporary suspension of transmission of I-format LPDUs. S-format LPDUs do not contain an information field and therefore do not affect V(S) in the sending link station or V(R) in the receiving link station.

The format of the S-format LPDU control field is shown below:



S = Supervisory Function Bit
N(R) = Transmitter Receive Sequence Number
P/F = Poll/Final Bit

Figure 8-6. S-Format LPDU Control Field

- The supervisory function bits indicate the type of S-format command or response being sent:
  - B'00' = Receiver Ready (RR) Command and Response

    A link station uses the Receiver Ready (RR) S-format LPDU to indicate that the station is ready to receive an I-format LPDU. I-format LPDUs numbered up through [N(R) - 1] have been received correctly by the link station. This S-format LPDU indicates the clearance of a busy condition at the sending link station (indicated in a previous RNR LPDU).
  - B'01' = Receiver Not Ready (RNR) Command and Response

    A link station uses the Receiver Not Ready (RNR) S-format LPDU to indicate a busy or slow-down condition (for example, a temporary inability to accept subsequent I-format LPDUs). I-format LPDUs numbered up through [N(R) - 1] are considered to have been accepted by the station prior to the busy condition. I-format LPDUs numbered N(R), and any subsequent I-format LPDUs received, are not considered to have been accepted by the station; the acceptance status of these I-format LPDUs is indicated in subsequent exchanges.
  - B'10' = Reject (REJ) Command and Response

    A link station uses the Reject (REJ) S-format LPDU to request the retransmission of I-format LPDUs starting with number N(R). I-format LPDUs numbered up through [N(R) - 1] are considered to have already been accepted by the station; they need not be sent again. The link station that receives the REJ LPDU, after retransmitting the requested I-format LPDU, can then send any additional I-format LPDUs it has queued for transmission. Only one "sent REJ" condition can be outstanding in a given direction at a given time on the link. The "sent REJ" condition ends when the link station receives an I-format LPDU with N(S) equal to the N(R) of the REJ LPDU. The "sent REJ" condition can also be cleared according to the procedures described in "Receiving an Out-of-Sequence I-Format LPDU" on page 11-20.

    A REJ LPDU indicates the clearance of a busy condition at the sending link station (indicated in a previous RNR LPDU).

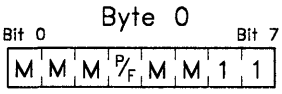
- The link station sets the Transmitter Receive Sequence Number [N(R)] to the current value of the Receive State Variable [V(R)] for the specified link (see "Receive State Variable, V(R)" on page 8-13). N(R) is therefore the expected sequence number of the next received I-format LPDU on that link. N(R) indicates that the link station has received correctly all I-format LPDUs numbered up through N(R)-1 on the specified link.
- The Poll/Final (P/F) bit is called the P bit in command LPDUs and the F bit in response LPDUs. (Whether the LPDU is a command or a response is indicated by bit 7 of the SSAP address field.) It is set in command LPDUs (P=B'1') to request that the remote link station send a response with this bit set (F=B'1'). There should be only one response received with F=B'1' for every command sent with P=B'1'. All link stations must be able to receive an S-format LPDU as a command and as a response, with the P/F bit set to either B'0' or B'1'.

  Link stations send all S-format command LPDUs with the P bit set to B'1'. However, link stations can send S-format responses with the F bit set to either B'0' or B'1':

  - A link station can send an appropriate asynchronous S-format response LPDU with the F bit set to B'0' without waiting for a command LPDU.
  - A link station can send an S-format LPDU as a response with the F bit set to B'1' to acknowledge the receipt of a command LPDU with the P bit set to B'1'. This response LPDU must be sent as soon as possible after receiving the command LPDU.

Unnumbered Format

Unnumbered format (U-format) LPDUs provide additional control functions and, in some cases, control data transfer functions (for example, XID). The format of the U-format LPDU control field is shown below:



M = Modifier Function Bit
P/F = Poll/Final Bit

Figure 8-7. U-Format LPDU Control Field

- The Modifier Function Bits indicate the type of U-format command or response being sent:

M Bit Values	Command or Response
0 0 0 1 1	DM Response
0 1 0 0 0	DISC Command
0 1 1 0 0	UA Response
0 1 1 1 1	SABME Command
1 0 0 0 1	FRMR Response
1 0 1 1 1	XID Command or Response
1 1 1 0 0	TEST Command or Response

Figure 8-8. U-Format Commands and Responses

- The Poll/Final (P/F) bit is called the P bit in command LPDUs and the F bit in response LPDUs. (Whether the LPDU is a command or a response is indicated by bit 7 of the SSAP address field.) It is set in command LPDUs (P=B'1') to request that the remote link station send a response with this bit set (F=B'1'). There should be only one response received with F=B'1' for every command sent with P=B'1'. Link stations always send U-format command LPDUs with the P bit set to B'1'. However, all link stations must be able to receive U-format command LPDUs with the P bit set to either B'0' or B'1'. A link station that receives a U-format command LPDU must respond with a U-format response LPDU with the F bit set to the value of the P bit in the command LPDU.

Disconnected Mode (DM) Response

A link station sends the Disconnected Mode (DM) response LPDU to report that it is in asynchronous disconnected mode and is logically disconnected from the link. The DM response LPDU does not contain an information field.

Disconnect (DISC) Command

A link station sends the Disconnect (DISC) command LPDU to terminate an asynchronous balanced mode of operation previously set by an SABME command LPDU. The DISC command LPDU informs the remote link station that the local link station is suspending operation of the link, and the remote link station should assume the asynchronous disconnected mode. The remote link station, upon receiving the DISC command LPDU, sends a UA response LPDU (if it is in asynchronous balanced mode) or a DM response LPDU (if it is in asynchronous disconnected mode).

I-format LPDUs that are unacknowledged when this command is sent or received remain unacknowledged. Any queued BTUs that are unacknowledged or pending transmission are discarded.

The DISC response LPDU does not contain an information field.

Unnumbered Acknowledgment (UA) Response

A link station uses the Unnumbered Acknowledgment (UA) response LPDU to acknowledge the receipt and acceptance of SABME and DISC command LPDUs. The UA response LPDU does not contain an information field.

Set Asynchronous Balanced Mode Extended (SABME) Command

A link station uses the SABME command LPDU to initiate data transfer in the extended asynchronous balanced mode of operation with a remote link station. When DLC.LAN.MGR receives a contact signal from the Physical Unit, DLC.LAN.MGR directs the link station identified in the signal to either send the SABME command LPDU, or to respond to a received SABME command LPDU with a UA response LPDU.

The remote link station that receives the SABME command LPDU must send a UA response LPDU to the local link station as soon as possible. The remote link station then sets its send and receive state variables to X'00' and assumes the asynchronous balanced mode (extended). When the local link station receives the UA response LPDU, it sets its own send and receive state variables to X'00', and also assumes the asynchronous balanced mode—extended.

The remote link station can reject the SABME command LPDU by responding with the DM response LPDU.

I-format LPDUs that are unacknowledged when this command is sent or received remain unacknowledged. Any queued BTUs that are unacknowledged or pending transmission are discarded.

The SABME command LPDU does not contain an information field.

Frame Reject (FRMR) Response

A link station uses the Frame Reject (FRMR) response LPDU to report that it has detected an abnormality in an incoming LPDU.

The abnormality is described in the 5-byte information field carried by the FRMR response LPDU.

Bytes 0 and 1, shown below, carry the contents of the control field as received in the LPDU that caused the frame-reject condition. If the rejected control field was in a U-format LPDU, byte 1 is set to X'00'.

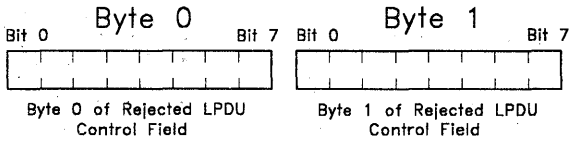


Figure 8-9. Frame Reject Information Field (Bytes 0 and 1)

Bytes 2, 3, and 4 of the FRMR response LPDU information field are shown below:

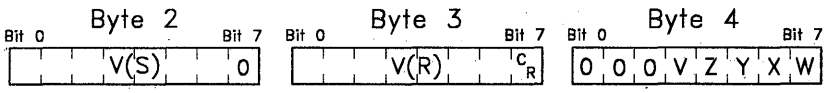


Figure 8-10. Frame Reject Information Field (Bytes 2, 3, and 4)

Byte 2 carries the send state variable for this link at the rejecting link station.

Byte 3 carries the receive state variable for this link at the rejecting link station. The C/R bit is set to B'0' if the LPDU causing the frame reject condition was a command LPDU and to B'1' if it was a response LPDU.

The last 5 bits of byte 4 give the reason for the frame reject condition:

- Bit V indicates that the send sequence number carried by the control field indicated in bytes 0 and 1 is invalid. A send sequence number is invalid if it is greater than or equal to the last sent receive sequence number plus the maximum receive window size (indicated in the information field of an XID LPDU).

  Note: When this condition is detected, the link station sends an REJ LPDU, not an FRMR response LPDU. This condition is specified here only for purposes of interpretation upon receipt of the FRMR response LPDU.

- Bit Z indicates that the receive sequence number carried by the control field indicated in bytes 0 and 1 does not refer to either the next I-format LPDU to be transmitted or to an I-format LPDU that has been transmitted but not acknowledged.

  The same receive sequence number may be received in multiple successive I-format or S-format LPDUs, provided it does not call for the retransmission of a confirmed I-format LPDU, or for transmission of an I-format LPDU that has not been transmitted and is not the next sequential I-format LPDU. The receive sequence number count is invalid only if it "regresses" to reference an I-format LPDU that has already been confirmed, or if it "skips" one or more values to reference an I-format LPDU *beyond* the next one to be transmitted.

- Bit Y indicates that the length of the information field in the received I-format LPDU exceeded the available buffer capacity, so the LPDU was not accepted.

However, this does not require establishment of a frame reject condition; the frame may be truncated, passed to path control with the lost data indicator, and acknowledged. The incoming frame must have passed the validity checks before any of these options are realized; otherwise the frame is simply discarded.

This bit is mutually exclusive with bit W.

- Bit X indicates one of two conditions:
  - The detected LPDU contained an information field, and it should not have.
  - The detected LPDU was an FRMR response LPDU with an information field that was not 5 bytes long.
- Bit W indicates that the control field indicated in bytes 0 and 1 represents an invalid or unsupported LPDU. Examples of unsupported LPDUs include Set Normal Response Mode (SNRM) and Set Asynchronous Response Mode (SARM). Invalid LPDUs include:
  - An S-format or U-format LPDU that contains an information field
  - An unexpected UA response LPDU (no SABME LPDU or DISC LPDU was sent).

    Note: An I-format or S-format response LPDU received with the F bit set to B'1', when no command LPDU with the P bit set to B'1' is unanswered, is processed as if the F bit were set to B'0'.

  Whenever bit V or bit X is set to B'1', bit W must also be set to B'1'.

The frame reject condition established when such a problem is detected is maintained until it is reset by receiving or sending an SABME command LPDU or DISC command LPDU, or by receiving a DM response LPDU. Except for the SABME or DISC command LPDU, any frames that arrive while the frame reject condition is active are ignored, except to respond to command LPDUs with the same FRMR response LPDU.

Exchange Identification (XID) Command

A link station uses the Exchange Identification (XID) command LPDU to convey identification and characteristics of the sending node, and to cause the remote link station to respond with the XID response LPDU. This command does not affect the mode and state variables maintained by the remote link station.

To maintain compatibility with IEEE Standard 802.2, link stations must be able to respond to XID LPDUs with the following information field format:

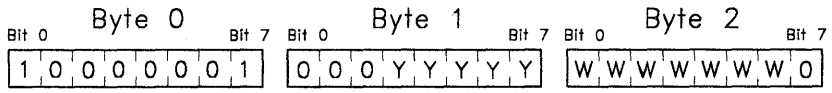


Figure 8-11. IEEE Standard 802.2 XID Information Field

The subfields of the information field are as follows:

Byte 0 This byte has the value X'81', indicating a standards-defined XID information field with the IEEE 802.2 Basic Format.

Byte 1 Bits 0 through 2 are reserved for future use by the IEEE; they are transmitted as B'000'.

Bits 3 through 7 indicate the class of service supported by the XID sender, either connectionless only (B'00001') or both connectionless and connection-oriented (B'00011'). If the SSAP address is X'00', the entire LLC sub-layer has the indicated service classes. Any other SSAP address indicates that the service class applies only to the SSAP.

Byte 2 Bits 0 through 6 contain the XID sender's maximum receive window size (RW).

Bit 7 is reserved for future use by the IEEE and is set to B'0'.

The proper response to a received 802.2 XID command LPDU is always an 802.2 XID response LPDU with the above information.

Note: Link stations can also exchange XID LPDUs that have other information field formats, including SNA formats.

Exchange Identification (XID) Response

A link station uses the XID response LPDU to respond to an XID command LPDU. The information field is the same as that described in "Exchange Identification (XID) Command."

Note: Regardless of the value of the P bit, the receiver of the XID command LPDU must respond as soon as possible. The Physical Unit handles all SNA XID processing, and must be notified as soon as possible that the XID command LPDU has been received.

Test (TEST) Command

A link station uses the TEST command LPDU to cause the remote link station to respond with the TEST response LPDU as soon as possible, thereby performing a basic test of the link station-to-link station transmission path. This command does not affect the mode and state variables maintained by the remote link station.

The TEST command LPDU can contain an optional information field. If the information field is present, the remote link station returns it, if possible, in the TEST response LPDU.

Test (TEST) Response

A link station uses the TEST response LPDU to reply to the TEST command LPDU. The F bit is set to either B'0' or B'1', corresponding to the value in the received TEST command LPDU.

Control Field State Variables

Each link station maintains a send state variable, V(S), for the I-format LPDUs it sends and a receive state variable, V(R), for the I-format LPDUs it receives. These two state variables operate independently. It is also necessary for the link station to maintain variables to control XID and TEST exchanges, busy conditions, transmit window size, and checkpointing operations. The following sections describe these variables.

Send State Variable, V(S)

V(S) denotes the sequence number of the next in-sequence I-format LPDU to be transmitted on the link. V(S) takes on values sequentially between 0 and 127. The value of V(S) is incremented by one with each successive I-format LPDU transmission on the associated link, but does not exceed the N(R) value of the last received I-format or S-format LPDU by more than the window size (which has a maximum value of 127).

Receive State Variable, V(R)

V(R) denotes the sequence number of the next in-sequence I-format LPDU to be received on a specific link. V(R) takes on values sequentially between 0 and 127. The value of the V(R) associated with a specific link is incremented by one whenever an error-free, in-sequence I-format LPDU is received whose send sequence number, N(S), equals the value of V(R) for that link.

Last Received N(R), V(A)

V(A) contains the N(R) value from the last valid I-format or S-format LPDU received. This is used to determine the upper transmit window edge, that is, the value of V(S) at which the link station should stop sending because the window has been reached.

Poll State Variable, V(P)

V(P) denotes the value of V(S) at the time the last I-format or S-format command LPDU with the P bit set to B'1' was transmitted; it is set to the value of V(S). V(P) is used to determine if a frame must be transmitted again when a response LPDU with the F bit set to B'1' is received. If N(R) of the received response LPDU does not acknowledge all frames up to (but not including) the value of V(P), the frames that are not acknowledged must be transmitted again. This variable is necessary only when I-frame transmission is allowed during a checkpointing operation (see "The Dynamic Window Algorithm" on page 11-23).

Busy State Variable, V(B)

V(B) denotes the ready/busy condition of the local (Lb) or remote (Rb) link stations, or both (LRb) as indicated by RNR, RR, REJ and other related LPDUs. (See Chapter 12, "State Tables" on page 12-1 for more information.)

Final State Variable, V(F)

V(F) is set to the value of the final bit required in the response to the last received command LPDU when the response must be delayed. (See Chapter 12, "State Tables" on page 12-1 for more information.)

Initialization State Variable, V(I)

V(I) denotes the link setup status for local (LIp) or remote (RIp) link stations, or both (LRIp) as determined by the source of link setup initialization. It also denotes that setup is not complete (ISp). (See Chapter 12, "State Tables" on page 12-1 for more information.)

Test State Variable, T(S)

T(S) denotes the link test status (ITp) when a locally initiated test is in process. This variable is required only for link stations that support TEST LPDUs. (See Chapter 12, "State Tables" on page 12-1 for more information.)

ID_Exchange State Variable, X(S)

X(S) denotes the ID_Exchange status for local (IXp) or remote (OXp) ID exchanges, or both (IOXp), as determined by the source of the XID initiation. This variable is required only for link stations that support XID LPDUs. (See Chapter 12, "State Tables" on page 12-1 for more information.)

Working Window Size, Ww

Ww is the maximum number of sequentially numbered I-format LPDUs that the link station may have outstanding (unacknowledged) at any given time. Ww is initialized to the value of TW (see "Maximum Number of Outstanding I-Format LPDUs (TW)" on page 11-4) when a link is established. When the dynamic window algorithm is invoked, Ww varies from 1 up to its maximum TW (see "The Dynamic Window Algorithm" on page 11-23).

Information Field

The information field, when present, consists of an integral number of bytes of data. I-format LPDUs do not need to contain an information field.

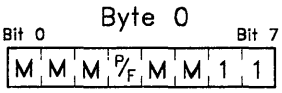
Chapter 9. Connectionless Service

The IBM Token-Ring Network LLC sub-layer supports *connectionless service*.

Connectionless service does not require the establishment of data links. Once the service access point (SAP) has been enabled within the LLC sub-layer, the SAP can send information to and receive information from a remote SAP that is also using connectionless service. Connectionless service does not have any mode-setting commands and does not require the LLC sub-layer to maintain state information about LPDU transfers.

Connectionless Command and Response Repertoire

The three unnumbered LPDU formats allowed in connectionless service are UI, XID, and TEST. The control field formats for these three frame types are shown below:



M = Modifier Function Bit
P/F = Poll/Final Bit

Figure 9-1. Connectionless LPDU Control Field

- The Modifier Function Bits indicate the type of U-format command or response LPDU being sent:
  - 1 0 1 1 1 = XID Command/Response
    These LPDUs are described in detail on page 8-12 .
  - 1 1 1 0 0 = TEST Command/Response
    These LPDUs are described in detail on page 8-12 .
  - 0 0 0 0 0 = UI Command
    The UI command LPDU is used to transport unsequenced data. The LLC sub-layer does not acknowledge UI command LPDUs, nor does it verify their sequence numbers. Therefore, UI command LPDUs can be lost if an exception (for example, a transmission error or a receiver-busy condition) occurs during transmission.

    A UI command LPDU is always sent with the P bit set to B'0'.
- The Poll/Final (P/F) bit is called the P bit in command LPDUs, and the F bit in response LPDUs. (Whether the LPDU is a command or a response is indicated by bit 7 of the SSAP address field.) Regardless of the setting of the P bit, the receiver of an XID or TEST command LPDU must respond with an XID or TEST response LPDU. The F bit in the response LPDU is set to the value of the P bit in the command LPDU.

IEEE 802.2 Service Specification

The IEEE 802.2 service specification contains service primitives that invoke LLC sub-layer functions. (In this reference, a *primitive* is an abstract, implementation-independent interaction between the user of a service and the provider of the service.) Three of these primitives control the sending of UI, XID, and TEST LPDUs:

- L_DATA.request
- L_MXID.request
- L_MTEST.request.

These primitives require the LLC sub-layer user to specify the destination and source MAC and SAP addresses, as well as any routing information. The LLC sub-layer performs no error recovery for these LPDUs and does not retry the transmission.

The primitives that indicate the arrival of UI command LPDUs and TEST and XID response LPDUs include:

- L_DATA.indication
- L_MXID.confirm
- L_MTEST.confirm.

The information field of the UI, TEST, and XID LPDUs, and the addressing information, are parameters of the indication and confirm primitives listed above.

Notes:

1. The primitives for the sending and receiving of XID and TEST LPDUs are not included in the approved IEEE 802.2 standard. Primitives for the sending of XID and TEST are part of the system and layer management work currently being addressed in both the IEEE 802.1 and IEEE 802.2 subcommittees.
2. The higher layer, or some management entity, determines the routing information to use for connectionless LPDUs. In some cases, this can be accomplished by sending an IEEE 802.2 XID or TEST command frame to the null SAP of the remote station. This is equivalent to the procedure used to determine routing information during link activation.

Chapter 10. Operation of the Access Channel Control

The DLC.LAN access channel control is the multiplexing element of the LLC sub-layer. It builds LLC Protocol Data Units (LPDUs) from the addressing information in its active link station routing table. The access channel control then passes the LPDU, along with the appropriate IBM Token-Ring Network addresses and routing information, to the appropriate MAC sub-layer.

The access channel control routes LPDUs it receives from MAC sub-layers to the appropriate link station or to DLC.LAN.MGR.

Notes:

1. In DLC.LANs with more than one medium access port, the access channel control maintains the mapping between link station and port. This mapping is established by DLC.LAN.MGR when the link is established, and is passed to the access channel control in the attach_ls signal.
2. This specification is specific for connection-oriented service. It does not specify primitives for connectionless service.
3. The access channel must have the ability to exchange frames for the NULL SAP with the DLC.LAN.MGR.

Reception of Invalid LPDUs

When the access channel control or link station receives an invalid LPDU or an LPDU with an unrecognized DSAP address, the LPDU is discarded and the event is logged. Invalid LPDUs that are detected by the access channel control include:

- An LPDU that is identified as invalid by the MAC sub-layer
- A U-format LPDU with a total length less than 3 bytes
- An I-format or S-format LPDU with a total length less than 4 bytes.

Routing in the LLC Access Channel Control

Consider the configuration illustrated below, with node A (MAC address 006896) having links with nodes B (MAC address 000050) and C (MAC address 000049).

Note: For simplicity, the segment numbers in the routing information fields used in the following examples contain the ring numbers illustrated in Figure 10-1. In actual practice, they would contain a common segment portion and an individual bridge portion as described on page 2-10.

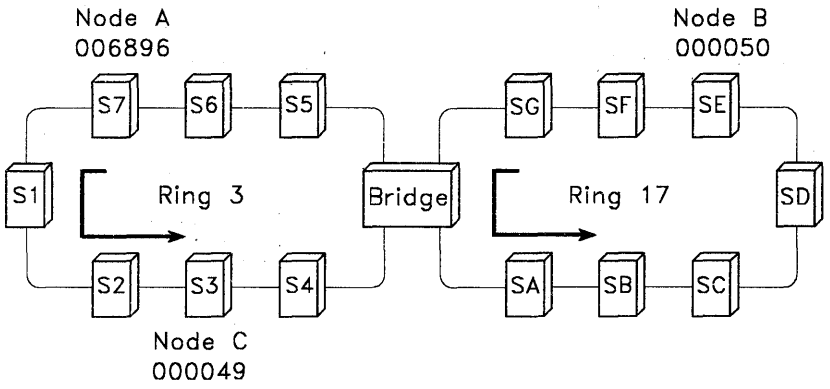


Figure 10-1. Rings Connecting Nodes A, B, and C. S1 through S7 and SA through SG are ring stations.

Control Block	Remote LAN	Routing Information	DSAP/ SSAP
31	000050	cc0317	4/4
51	000049	—	4/4

Figure 10-2. Control Block Table for Node A

In this example, both node A and node B use SAP address X'04' for the link. The DLC.LAN access channel control in node A has its information about links to nodes B and C stored in a table such as the one in Figure 10-2. It is assumed that such a table exists for each ring station, thus eliminating the need to store the local LAN address in the routing table.

The transfer of data from node B to node A starts with the link station passing an access channel control service data unit (ACSDU) to the access channel control. The access channel control adds the appropriate DSAP and SSAP address fields and passes the data unit, along with the appropriate MAC addresses, to the MAC sub-layer.

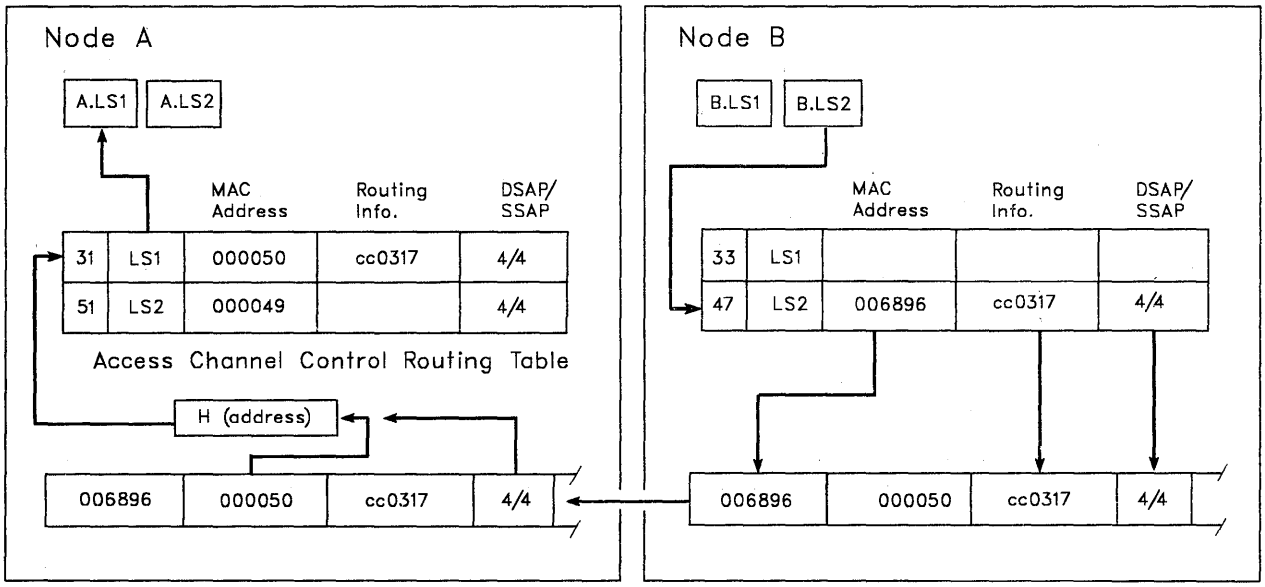


Figure 10-3. Locating Link Station Control Blocks

When link station B.LS2 sends a PDU to A.LS1, the access channel control in B.DLC.LAN finds the MAC destination address, the routing information, and the DSAP and SSAP from its routing table.

When the Basic Link Unit (BLU) arrives in node A, A uses the MAC source address, the DSAP, and the SSAP (with C/R masked out) as the input to a hash function, the output of which is a pointer to the proper link station control block. (A hash function is an algorithm that is used to map input data to the output data without having to search a table. The choice of hash function is left to the implementers.)

BLUs from A to B on the same link would have a destination address of 000050, source address of 006896, and DSAP and SSAP each equal to X'04'.

Chapter 11. Operation of Link Stations

Path control passes basic transmission units (BTUs) to the DLC.LAN link stations, which then add DLC.LAN control field information to the BTUs. On each link, the local and remote link stations use the HDLC asynchronous balanced mode of operation to keep LPDUs in sequence, and to detect and correct, by retransmission, LPDUs that are out of sequence. Link stations pass the BTU and the control field to the access channel control.

Link Station Creation and Termination

A link station does not exist before resources for it have been allocated by DLC.LAN.MGR in response to a CONNECT_OUT_ALS or ASSIGN_ALS_ADDRESS prompt from the Physical Unit. Once resources have been allocated for it, the link station assumes the Asynchronous Disconnected Mode described below.

Similarly, the link station resources are de-allocated by DLC.LAN.MGR in response to DISCONNECT_ALS.

Asynchronous Balanced Mode — Extended

Asynchronous balanced mode — extended (ABME) is a balanced operational mode used between two combined link stations. Either link station can send commands at any time and initiate appropriate response transmissions independent of the other link station. Each such asynchronous transmission contains a single LPDU for transfer of data and to indicate a change in status at the sending link station. ABME includes an activation phase, a data transfer phase, and a termination phase.

Asynchronous Disconnected Mode

Asynchronous disconnected mode (ADM) differs from the operational mode (ABME) in that the link is logically disconnected. That is, no I- or S-format LPDUs can be sent or received. Asynchronous disconnected mode is defined to prevent a link from appearing in an operational mode during unusual or exception conditions, because such operation could cause:

- Sequence number mismatch between the link stations
- Ambiguity in one link station as to the other station's condition.

In addition to receipt of a DISC command LPDU. any of the following conditions causes the link station to enter asynchronous disconnected mode:

- DLC.LAN.MGR activates the link station.
- The link station receives a DM response LPDU.
- The retry limit is exhausted during checkpointing operations.

A link station in asynchronous disconnected mode monitors LPDUs received from the access channel control for the purpose of:

- Accepting and responding to mode-setting command LPDUs (SABME, DISC)
- Accepting an XID command LPDU and returning an XID response LPDU
- Accepting a TEST command LPDU and returning a TEST response LPDU
- Sending a DM response LPDU, when required.

In addition, the link station may send an appropriate mode-setting, XID, or TEST command LPDU when directed by DLC.LAN.MGR. A link station in asynchronous disconnected mode that receives a DISC command LPDU must respond with a DM response LPDU. Also, a link station in asynchronous disconnected mode cannot establish a frame reject exception condition (see "FRMR Exception Conditions" on page 11-26).

System Parameters

Reply Timer (T1)

A link station uses T1 to detect a failure to receive a required acknowledgment or response from the remote link station.

- The link station starts T1 when it transmits one of the following:
  - An I-format LPDU (if T1 is not already running).
  - A command LPDU with the P bit set to B'1'. If T1 was already running, this causes the link station to reset and restart T1.
- The link station resets T1 when it receives one of the following:
  - A REJ LPDU, provided a command LPDU with the P bit set to B'1' is not outstanding.
  - A response LPDU with the F bit set to B'1'.
  - An I-format or S-format LPDU with an N(R) greater than the last N(R) received, and less than or equal to the link station's V(S), provided a command LPDU with the P bit set to B'1' is not outstanding.
- After resetting T1, the link station restarts T1 if additional LPDUs have been sent and acknowledgments or responses are still outstanding. If no additional acknowledgments or responses are outstanding, the link station starts Ti, the inactivity timer (see page 11-3).

When T1 expires, the link station sends one of the following:

- An S-format command LPDU with the P bit set to B'1', to solicit remote link station status
- Any U-format command LPDUs that were not responded to the first time they were sent.

The link station then starts T1 again. If there is no recovery after N2 tries, the link station declares the link inoperative and informs DLC.LAN.MGR (see "Maximum Number of Transmissions (N2)" on page 11-4). DLC.LAN.MGR then informs the Physical Unit that the link is inoperative.

The duration of T1 must take into account any delays introduced by the MAC sub-layer (for example, queuing). The suggested default value for T1 is 1 second.

Inactivity Timer (Ti)

A link station uses Ti to detect an inoperative condition in either the remote link station or in the transmission medium. To ensure that the link station detects such a condition, if the link station does not receive an LPDU before Ti expires, it must solicit the status of the remote link station, using an S-format command LPDU with the P bit set to B'1'. Recovery then proceeds as described in "Reply Timer (T1)." The suggested default value for Ti is 30 seconds.

Receiver Acknowledgment Timer (T2)

The Receiver Acknowledgment Timer (T2) is used in conjunction with counter N3 (see "Number of I-Format LPDUs Received before Sending Acknowledgment (N3)" on page 11-4). A link station uses T2 to delay the sending of an acknowledgment for a received I-format LPDU. The link station starts T2 when it receives an I-format LPDU, and resets T2 when it sends an acknowledgment in an I-format or

S-format LPDU. If T2 expires, the link station must send an acknowledgment as soon as possible.

The value of T2 must be less than that of T1, to ensure that the remote link station will receive the delayed acknowledgment before its T1 expires. The suggested default value for T2 is 100 milliseconds.

Maximum Length of I-Field (N1)

N1 indicates the maximum number of information bytes in an I-format, TEST, or XID LPDU. The LLC sub-layer places no restrictions on the value of N1. An I-format, XID, or TEST LPDU includes the DSAP and SSAP address fields, the control field, and the information field, as shown in Chapter 8, "LLC Frames" on page 8-1; it does not include the frame header or the routing information field.

Maximum Number of Transmissions (N2)

N2 is a system parameter that defines the maximum number of times that an LPDU is sent following the expiration of the reply timer, T1. This count applies also to I-format LPDUs sent again when a checkpoint operation is successful. This prevents loops in the checkpoint procedure.

Number of I-Format LPDUs Received before Sending Acknowledgment (N3)

N3 is a system parameter used in conjunction with T2 to allow stations to reduce acknowledgment traffic by not immediately acknowledging received I-format LPDUs. A counter, Ir_Ct, is initialized to N3. Each time a valid, in-sequence I-format LPDU is received, the counter is decremented. When the counter reaches 0, an acknowledgment is sent. The counter is reset each time an acknowledgment (an I- or S-format LPDU) is sent. (See Chapter 12, "State Tables" on page 12-1 for more information concerning the use of Ir_Ct.)

Number of Acknowledgments Needed to Increment Ww (Nw)

When the working window (Ww) is not equal to the maximum transmit window size (TW), Nw is the number of transmitted I-format LPDUs that must be acknowledged before Ww can be incremented by 1. Nw controls the gradual incrementing of Ww in congestion situations.

Maximum Number of Outstanding I-Format LPDUs (TW)

The maximum number (TW) of sequentially numbered I-format LPDUs that the link station may have outstanding at any given time is a system parameter that must be equal to or less than 127. TW is the maximum number by which the link station's V(S) can exceed the N(R) of the last received LPDU.

Receive Window Size (RW)

The receive window size (RW) denotes the maximum number of unacknowledged sequentially numbered I-format LPDUs that the link station can receive from the remote link station. RW must be less than or equal to 127. It is transmitted in the information field of an SNA XID3 LPDU (used by T2.1 nodes) and applies to the XID sender.

Note: RW is also transmitted in the IEEE 802.2 XID information field. The XID receiver should set its TW to a value less than or equal to the RW of the XID sender to avoid overrunning the XID sender.

Use of the P/F Bit

A link station receiving a command LPDU with the P bit set to B'1' sends a response LPDU with the F bit set to B'1'. This applies to all command-response combinations described below.

The response to an SABME or DISC command LPDU is a UA or DM response LPDU with the F bit set to the received P bit value. The response to an I-format, RR, RNR, or REJ command LPDU with the P bit set to B'1' is an I-format, RR, REJ, RNR, DM, or FRMR response LPDU with the F bit set to B'1'.

Note: The P/F bit is used in conjunction with timer recovery (see "Reply Timer (T1)" on page 11-3).

Link Activation

A link station can initialize the link after receiving the CONTACT_ALS(primary) signal from the Physical Unit. To initialize the link, the link station sends the SABME command LPDU and starts T1. When the UA response LPDU is received, the link station sets V(S) and V(R) to X'00', stops T1, and assumes the data transfer phase.

Note: The primary always sends an RR or RNR command LPDU with the P bit set to B'1' when the UA is received. DLC.LAN.MGR sends CONTACTED to the control point only after sending the RR or RNR command LPDU. Conversely, the secondary does not assume it has been contacted until it receives the RR or RNR command LPDU.

When the DM response LPDU is received, the link station that originated the SABME command stops its T1, does not enter data transfer phase, and reports the response to DLC.LAN.MGR.

The actions to be taken when an SABME or DISC command LPDU is received are described in "Contention of Unnumbered Mode-Setting Commands" on page 11-19. Other command and response LPDUs that are received are ignored, except UA response LPDUs.

If T1 expires before the UA or DM response LPDU is received, the link station sends the SABME command again, restarts T1, and continues this process as described in "Reply Timer (T1)" on page 11-3.

When an SABME command LPDU is received, the link station sends a UA response LPDU to the remote link station, sets its V(S) and V(R) to X'00', and enters the data transfer phase. Sending the UA response LPDU takes precedence over any other pending response LPDUs at that link station, but other LPDUs can be sent after the UA response LPDU has been sent.

If the link station determines that it cannot enter the data transfer phase when the SABME command LPDU is received, it sends the DM response LPDU and remains in asynchronous disconnected mode.

DLC.LAN.MGR controls the activation of links by interfacing with the Physical Unit. Link establishment may be initiated upon request from the Physical Unit, or when a connection request from a remote station is received (for example, when an SNA XID LPDU or SABME LPDU is received). DLC.LAN.MGR is responsible for acti-

vating link stations and informing the access channel control of associated addressing information.

Link activation requires that the initiating node know the destination node's specific station (MAC) address. The examples used in the remainder of this reference assume that the addresses are all from a fixed 6-byte address space, as shown below:

Note: For simplicity, the segment numbers in the routing information field used in this example contain the ring numbers illustrated in Figure 11-1. In actual practice, they would contain a common segment portion and an individual bridge portion as described on page 2-10.

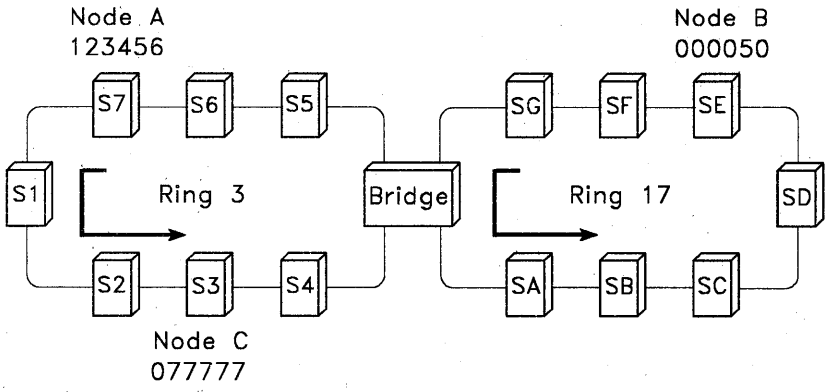


Figure 11-1. Example of Link Initiation. S1 through S7 and SA through SG are ring stations.

Assume no current link is established between node A and node B that uses SAP X'04' in both nodes. To initiate the establishment of a link with node B, the Physical Unit of node A passes a connect_out_als record to its DLC.LAN.MGR. If the signaling_information parameter in connect_out_als contains the MAC address (000050) of the destination node, but does not specify the local and remote SAP addresses, DLC.LAN.MGR sets these addresses to the default value (X'04').

To determine the remote node's routing information, DLC.LAN.MGR sends a TEST or IEEE 802.2 XID command LPDU to the destination node.

Note: When an IEEE 802.2 XID command LPDU is used, the information field must conform to the format specified in the IEEE 802.2 standard. The format does not provide for a correlator and one cannot be included. In a TEST command LPDU, an information field is optional; if it is included, it is returned without change in the TEST response LPDU. There are no rules governing the format of the information field of a TEST command LPDU. The sender may include a correlator for use in associating TEST response LPDUs with TEST command LPDUs. All implementations must be capable of responding to both TEST and IEEE 802.2 XID command LPDUs.

Because a link station has not yet been allocated to the new link, DLC.LAN.MGR in node A is responsible for building the command frame and running a response timer after the frame is transmitted. The DSAP address field of the command frame is set to the null SAP (X'00') address. This address is used because it is always active and must respond to XID and TEST command LPDUs. The SSAP address field may contain the individual SAP address that will be used as the local SAP once the link is established.

The TEST or XID command LPDU is first sent only on the local ring. No routing information field is present in the first TEST or XID command LPDU sent. If a response LPDU is received, then node B resides on the local ring and the resolve procedure is complete.

In our example however, node B is on another ring, and DLC.LAN.MGR's response timer expires without a response being received. DLC.LAN.MGR appends a routing information field indicating an all-routes broadcast to the command frame and sends it out again. The TEST or IEEE 802.2 XID command LPDU is copied and forwarded by all bridges. When the command reaches node B, the access channel control routes the frame to the DSAP, which in an SNA node is in DLC.LAN.MGR, because the combination of the destination and source addresses and DSAP and SSAP does not match the addressing information for an active link station. The routing information field in the received frame identifies the sequence of rings through which the frame passed. DLC.LAN.MGR, on behalf of the null SAP, responds with a TEST or XID response LPDU. The routing information field of the command frame is included in the response LPDU. The direction bit in the routing information field is inverted, and the broadcast mode is reset to B'0'. When node A receives the response LPDU, the access channel control passes the frame to DLC.LAN.MGR for processing.

More than one response LPDU may be received in installations with multiple rings and multiple active paths between rings; once a response LPDU has been received, subsequent response LPDUs can be discarded. The routing information field in the response LPDU is appended to all frames to be exchanged on the link currently being established.

Successful completion of the resolve procedure causes DLC.LAN.MGR to allocate link station resources and inform the access channel control of the link station's addressing information (local and remote MAC and SAP addresses, and routing information). DLC.LAN.MGR then informs the Physical Unit that the connection is complete. The Physical Unit then polls with a null XID (if the characteristics of the adjacent node are unknown) or an XID3 (if the adjacent node is known to support Physical Unit T2.1 XID exchanges).

The access channel control in node B, on receiving an XID for which no link station exists, passes the XID to DLC.LAN.MGR. If no in-bound connections have been authorized by a prior connect_in_als, DLC.LAN.MGR discards the XID and takes no further action. If a connect_in_als has been issued to B's DLC.LAN.MGR (authorizing DLC.LAN.MGR to allocate link station resources to connections), DLC.LAN.MGR uses als_connected_in to notify the Physical Unit that a connection has been established. After the Physical Unit returns an assign_als_addr record to DLC.LAN.MGR, a link station is activated, and the access channel control is informed of the new link station's addressing information. DLC.LAN.MGR then passes the received XID information field to the Physical Unit.

Connection and Contact Flows for DLC.LAN.MGR

The following figure shows in detail an example of the link activation process:

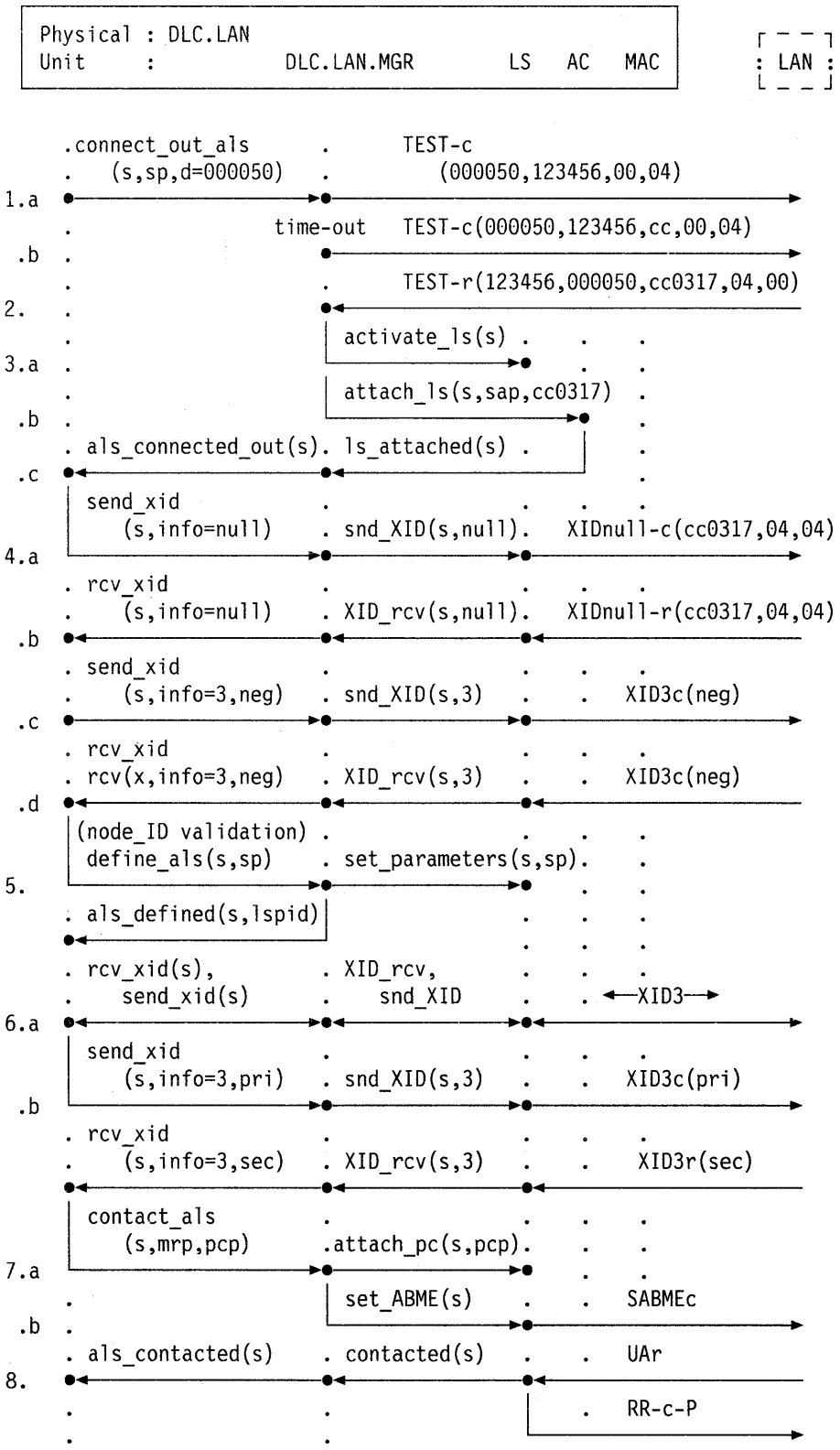


Figure 11-2. Connect Out

c = LPDU command frame
r = LPDU response frame
s = adjacent link station identifier
sp = link station parameters
d = signaling information ("dial digits")
cc = routing information control field
sap = local and remote LSAP addresses
lspid = link station process identifier
pcp = path control connection parameters
mrp = mode/role parameters

The following notes are keyed to Figure 11-2 on page 11-8.

1. The connect_out_als from the Physical Unit causes DLC.LAN.MGR to start the connection phase of link activation. DLC.LAN.MGR sends a TEST command LPDU to the null SAP of the remote station to initiate the resolve procedure. DLC.LAN.MGR uses the signaling information (in this case the address 000050) as the MAC destination address. The MAC source address is 123456. The DSAP is the null SAP (X'00') and the SSAP is X'04'.

   a. The *first* TEST command LPDU is sent only to the local ring (that is, with no routing information field) to limit the amount of inter-ring traffic during the resolve procedure.
   b. If no response LPDU is received within a predefined time period determined by DLC.LAN.MGR, the TEST command LPDU is broadcast to all rings. The routing information is accumulated en route to the destination node.

2. DLC.LAN.MGR in the remote node sends a TEST response LPDU containing the routing information in the routing information field. The access channel control of the local node receives the TEST response LPDU and routes it to DLC.LAN.MGR, because the combination of the destination and source addresses and DSAP and SSAP does not match the addressing information for an active link station. DLC.LAN.MGR obtains the routing information from the TEST response LPDU.

   Note: For simplicity, the segment numbers in the routing information field used in this example contain the ring numbers illustrated in Figure 11-1 on page 11-6. In actual practice, they would contain a common segment portion and an individual bridge portion as described on page 2-10.

3. This flow sequence represents setting up the link station and the access channel setting up its control blocks.

   a. The link station resources are allocated and assigned.
   b. DLC.LAN.MGR requests the access channel control to add the link station to its routing table. The access channel control uses the remote MAC address and the local and remote SAP addresses to determine the search key for the link station control block. Attach_ls results in the access channel control, based on the search key, finding an empty slot in its table for a pointer to the new link station. The access channel control stores the provided routing information (cc0317) in the link station control block.
   c. Once the access channel control bookkeeping is done, DLC.LAN.MGR signals als_connected_out to the Physical Unit.

4. This flow sequence represents the initial SNA XID exchange. XID exchange finite state machines are used by the Physical Unit to determine when to send an XID and what type of XID to send. The information field in each XID received by the link station is passed to the DLC.LAN.MGR, which passes it to

the Physical Unit. When the Physical Unit decides to send an XID, it passes the information field to the DLC.LAN.MGR, which passes it to the link station. The link station keeps track of whether an XID should be sent as a command or response LPDU. When a link station receives a send_XID record from DLC.LAN.MGR, it checks a status variable to see if a response LPDU needs to be sent to a previously received XID command LPDU. If so, an XID response LPDU with the F bit set to the value of the P bit in the previously received XID command LPDU is sent. Otherwise, an XID command LPDU with the P bit set to B'1' is sent.

   a. The first XID sent is a null XID command LPDU, that is, no information field.
   b. The first XID received from the remote link station is a null XID response LPDU. The link station must be able to handle either an XID command or response LPDU. During a link activation race, it is possible to receive an XID command LPDU at this point. (As in the previous example, the type of the first XID sent or received can be either a null or format 3 XID.)
   c. When the Physical Unit receives a null XID, it responds with its own XID information field.
   d. When the Physical Unit receives the first non-null XID information field, it performs the node ID validation.

5. The Physical Unit then sends any adjacent link station parameters to the DLC with define_als. The link station process ID in the als_defined reply is used by the Physical Unit to connect the link station to the proper path control component.

6. Node-role negotiation

   Note: Node role is not relevant to ABME link station operation, but it is required for determining the value of the OAF-DAF Assignment Indicator (ODAI). It is also used to determine which link station sends the SABME command LPDU.

   a. After the link station definition has completed, the node-role negotiation starts.
   b. The negotiation ends when one node (in this case the local node) has sent an XID with the node role set to "pri" (primary only) and receives an XID with a node role of "sec" (secondary only). The XID information field is always provided by the Physical Unit. Thus the node role negotiation is performed by the Physical Unit, not by DLC.LAN.MGR.

7. When the Physical Unit determines that the node-role negotiation is over, it sends the contact_als record to DLC.LAN.MGR.

   a. The contact_als contains two parameters. The first, path control connection, tells DLC.LAN.MGR what path control component is to be associated with the link for purposes of routing data from the link station to path control.
   b. The second parameter, mode/role, tells DLC.LAN.MGR whether to have the link station send SABME or not. In this case, the mode/role is "pri," so the link station sends SABME. Prior to sending an SABME command LPDU, the link station must reset its status variable that keeps track of the state of the XID exchanges. If an XID response LPDU is expected, the response timer should be stopped and the unanswered XID command LPDU discarded.

8. The receipt of the UA response LPDU indicates the contact procedure was successful. However, to provide the UA sender (that is, the role loser) with a timely indication that the UA was received, the link station that sends the SABME, upon receiving the UA, sends an RR or RNR command LPDU with the P bit set to B'1'.

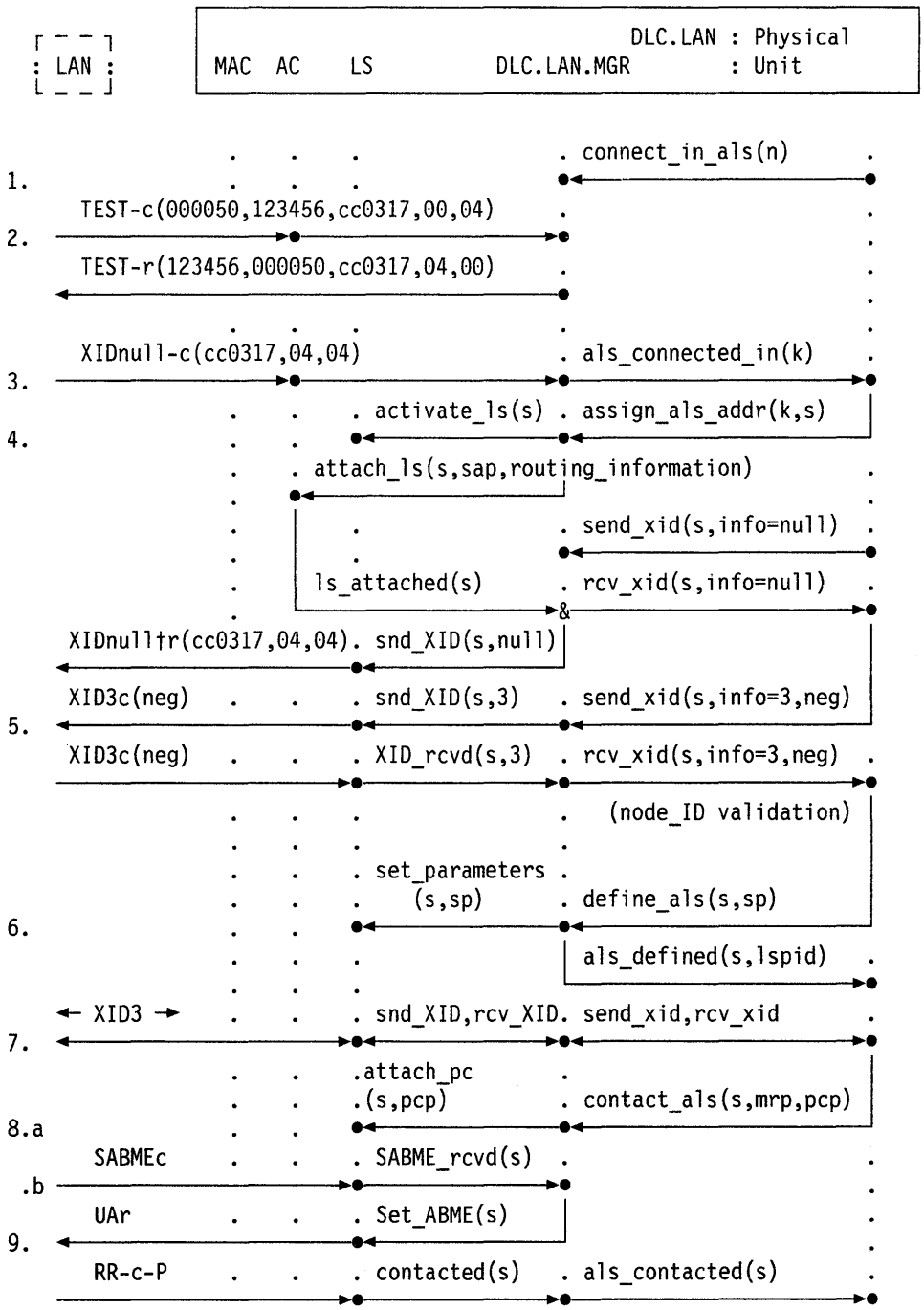


Figure 11-3. Connect In

c = LPDU command frame
r = LPDU response frame
k = request correlator
n = number of port or adjacent link station requests
s = adjacent link station identifier
sp = link station parameters
cc = routing information control field
routing_information = routing information (control field and segment numbers)
sap = local and remote LSAP addresses
lspid = link station process identifier
pcp = path control connection parameters
mrp = mode/role parameters

The following notes are keyed to Figure 11-3 on page 11-11.

1. The connect_in_als from the Physical Unit causes DLC.LAN.MGR to reserve link station resources for n links pending the initiation of link activation by remote nodes.
2. The access channel control receives a TEST command LPDU and routes it to DLC.LAN.MGR because the combination of the destination and source addresses and DSAP and SSAP does not match the addressing information for an active link station. DLC.LAN.MGR responds with a TEST response LPDU. DLC.LAN.MGR in the remote node obtains the routing information from the TEST response LPDU.
3. Once again, the access channel control is unsuccessful in searching for a control block based on the destination and source addresses and DSAP and SSAP of the received XID frame. The access channel control passes the null XID command LPDU to DLC.LAN.MGR. DLC.LAN.MGR interprets the null XID as a request for a connection and notifies the Physical Unit with als_connected_in. A correlator is included to keep the subsequent address assignments associated with the proper connections.

   Note: This operation assumes that the routing information (if any) accompanying the XID is passed to the access channel control by the MAC sub-layer. The access channel control will pass the entire frame, including the routing information, to DLC.LAN.MGR.
4. The Physical Unit passes the adjacent link station address to DLC.LAN.MGR in the assign_als_address record. DLC.LAN.MGR then activates the new link station and updates the access channel control with addressing information about the new link. Once the access channel control bookkeeping is complete, DLC.LAN.MGR passes the previously received null XID to the Physical Unit. Simultaneously, the Physical Unit, after passing assign_als_address to DLC.LAN.MGR, sends a null XID. Again, as stated in the previous example, the type of the first XID sent is dependent upon configuration. For T2.1 nodes, it may be either a null or format 3 XID. The XID being sent in this example is a null XID response LPDU.
5. Having received a null XID, the Physical Unit responds with its own XID (for T2.1 nodes, this is XID3). When a non-null XID is received, the Physical Unit validates the adjacent node's identification.
6. Assuming the adjacent node is a valid partner, the Physical Unit completes the definition of the link station by passing any remaining parameters to the DLC in define_als. DLC.LAN.MGR passes the parameters on to the link station.

7. At this point the Physical Unit is ready to negotiate the node role with the adjacent node. This involves exchanging XIDs until one node determines it is the "primary" and the other determines it is the "secondary."

   Note: Node role is not relevant to ABME link station operation, but it is required for determining the value of the OAF-DAF Assignment Indicator (ODAI). It is also used to determine which link station sends the SABME command LPDU. The contents of the XID information field are always provided by the Physical Unit. Thus the node role negotiation is performed by the Physical Unit, not by DLC.LAN.MGR.

8. When the Physical Unit determines that the node-role negotiation is over, it issues the contact_als record to DLC.LAN.MGR.

   a. The contact_als contains two parameters. One, path control connection, tells the DLC.LAN.MGR what path control component is to be associated with the link for purposes of routing data from the link station to path control.
   b. The other parameter, mode/role, tells DLC.LAN.MGR whether to have the link station send SABME or not. In this case, the mode/role is "sec," and once a SABME has been received, the DLC.LAN.MGR sets the link station to acknowledge an SABME command LPDU with a UA response LPDU. After having received an SABME command LPDU and prior to sending a UA response LPDU, the link station must reset its status variable that keeps track of the state of the XID exchanges. It is possible for the XID exchange procedure to complete with one or both link stations waiting for a response to a previously sent XID command LPDU. If an XID response LPDU is expected, the response timer should be stopped and the unanswered XID command LPDU discarded.

9. The UA sender ("secondary") does not send contacted_als to the Physical Unit until it has received some indication that the UA was received. This indication could be an RR or RNR command LPDU or an information LPDU. Here, the SABME sender has sent an RR command LPDU with the P bit set to B'1', verifying the contact success in the remote link station.

Link Activation Races

At times two nodes will each attempt to activate a link when only one link is desired. For the first link between two nodes, the use of X'04' as the default DSAP and SSAP addresses guarantees that only one link will be established.

However, for subsequent links this method will not work, because the simplest way for a node to activate another link is to set the DSAP address to X'04' and the SSAP address to X'08'. Because both nodes will use this mechanism, two links will be established: one with the first node's SAP address set to X'04' and the second node's SAP address set to X'08', and the other with the first node's set to X'08' and the second node's set to X'04'.

The solution is modeled after the support for dial connections. Specifically, the Physical Unit checks, during concurrent link activation, to see whether the block ID or CP name from the remote node's XID matches that for another link activation. If so, the Physical Unit terminates one of the links.

The remainder of this section describes the link activation races that occur when two nodes simultaneously start to activate a link to each other using the same SAP addresses. Although only one link will be established, contention situations exist that require resolution.

If two nodes in different segments simultaneously start to activate a link to each other, the possibility arises that the routing information in one direction will not match that for the reverse direction. If each node invokes the resolution procedure, determines a different route to the other node, and adds the link station with its associated routing information to its access channel control routing table, a link can be established that will use different routes between the two nodes.

To ensure that frames traveling in either direction on a link use the same route, there are defined conditions in which the routing information contained in a received frame is either adopted as the routing information for the link, or must be compared with the current stored routing information. If routing information is compared and a difference is detected, the following rules determine which route to use:

- If one routing information field is shorter than the other, the shortest route is chosen.
- If the routes are of equal length, the routing information field sent by the station with the higher address is used.

For link stations that implement support of XID and that are in asynchronous disconnected mode (see "Asynchronous Disconnected Mode" on page 11-2), the routing information from an XID received when no XID response LPDU is pending (either incoming or outgoing) is adopted as the routing information for the link. For an XID received in asynchronous disconnected mode when an incoming XID response LPDU is pending, the routing information field of the received XID is compared with the routing information currently stored for the link. If a difference is detected, the resolution rules described above determine which route to use. It is necessary to adopt or compare the routing information of only the first XID received after entering asynchronous disconnected mode. This eliminates the need to continually verify routing information during an XID negotiation sequence.

For SNA links, the sender of the SABME command LPDU is determined by the XID negotiation exchange. This eliminates the possibility of an SABME LPDU collision when a link is activated. For non-SNA links, there may not be an XID exchange sequence prior to one or both of the link stations sending an SABME command LPDU to place the link in asynchronous balanced mode. To ensure reversible routes for all links, there are defined conditions in which the routing information contained in a received SABME command LPDU is adopted as the routing information for the link. There are also defined conditions in which the routing information contained in received SABME command LPDUs or UA response LPDUs must be compared with the current routing information stored for the link.

If a link station receives an SABME command LPDU and neither local nor remote initialization is pending (that is, an SABME command LPDU was sent and the link station is waiting for a UA response LPDU, or an SABME command LPDU was received and the station is waiting for information from the higher layer), the routing information from the SABME command LPDU is adopted as the route for the link.

Routing information must be compared if either of the following occurs:

- The link station receives an SABME command LPDU when local initialization is pending (that is, an SABME command LPDU was sent, and the station is waiting for a UA response LPDU).
- The link station receives a UA response LPDU when local initialization is pending.

If a difference is detected, the resolution rules described above determine which route to use.

If the access channel control changes the routing information for a link, it notifies DLC.LAN.MGR by sending an LS_MODIFIED record. The figure below shows a link activation race where the routing information of a received XID is different from the routing information currently stored for the link.

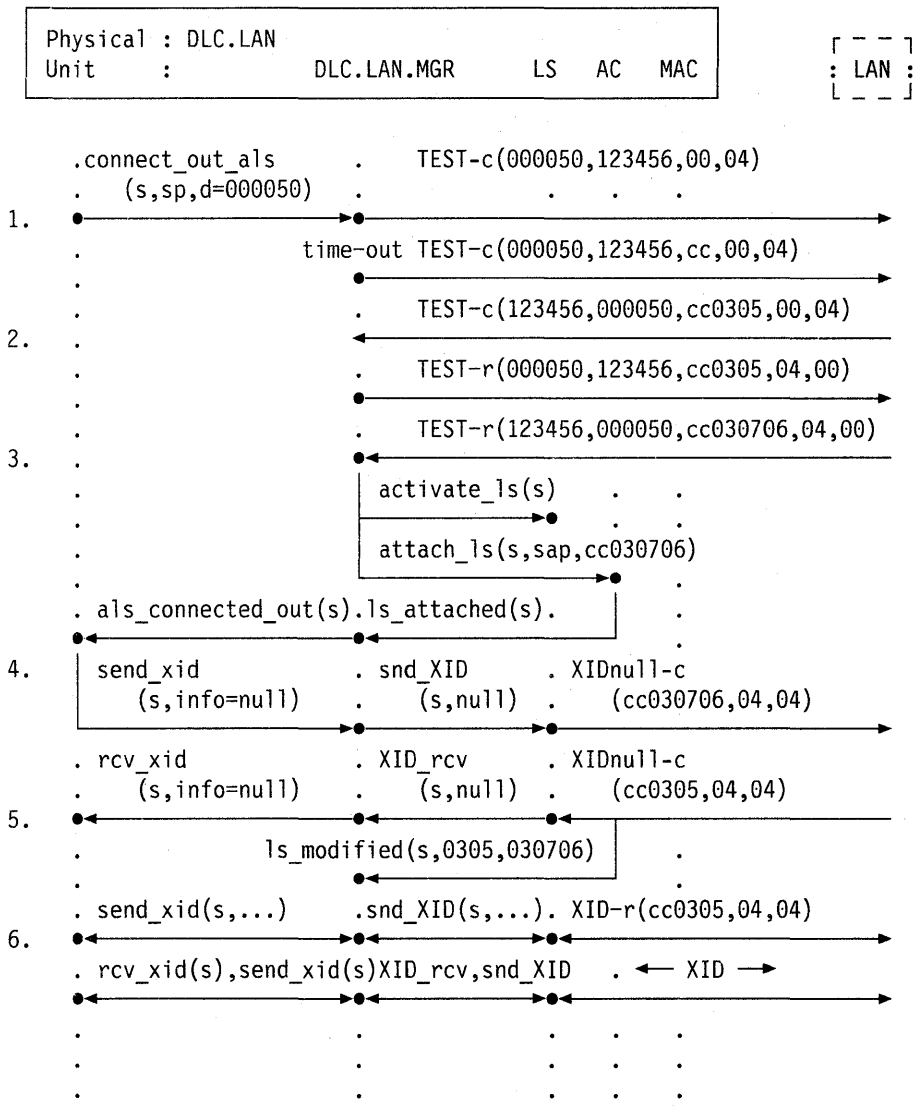


Figure 11-4. Connection Flow for Simultaneous Activation of a Link. Link Activation Race for DLC.LAN Component

The following legend explains the symbols in Figure 11-4:

c = LPDU command frame
r = LPDU response frame
s = adjacent link station identifier
sp = link station parameters
d = signaling information ("dial digits")
cc = routing information control field
sap = local and remote LSAP addresses

The following notes are keyed to Figure 11-4.

1. The connect_out_als causes DLC.LAN.MGR to start the connection phase of the link activation. DLC.LAN.MGR uses the signaling information (in this case the address 000050) as the MAC destination address in the TEST command LPDU sent as part of the resolve procedure. The MAC source address is 123456. The DSAP is the null SAP (X'00') and the SSAP is X'04'. The first TEST command LPDU is sent only on the local segment. No response LPDU is received, so the TEST command LPDU is broadcast to all segments.

2. A TEST command LPDU addressed to the null SAP is received. The MAC source address contains 000050. Both nodes are in the process of determining a route between them. DLC.LAN.MGR responds with a TEST response LPDU. The routing information carried in the frame is cc0305.

   Note: For simplicity, the segment numbers in the routing information fields used in the following examples contain ring numbers. In actual practice, they would contain a common segment portion and an individual bridge portion as described on page 2-10.

3. DLC.LAN.MGR receives a TEST response LPDU from the remote node with the routing information cc030706. DLC.LAN.MGR activates the link station and requests the access channel control to add the link station to its routing table.

4. After receiving als_connected_out, the Physical Unit passes a send_XID with a null information field to the DLC.LAN.MGR. A null XID command LPDU is sent to the remote link station with routing information cc030706.

5. An XID command LPDU is received from the remote link station with routing information cc0305. The routing information is compared with that stored for the link station and a difference detected. The rules for resolving a difference are applied and it is determined that cc0305 should be used because it is the shorter of the two routing information fields. The access channel control notifies the DLC.LAN.MGR of the change by sending an LS_MODIFIED record.

6. The next XID frame is sent to the remote link station with routing information cc0305. The link activation process then proceeds normally with the same routing information being used in both directions of the link.

Data Transfer

After sending the UA response to an SABME command, or after receiving a UA response to an SABME command, the link station can send and receive I- and S-format LPDUs according to the procedures described in "Procedures for Information Transfer" on page 11-19. When an SABME command LPDU is received while in the data transfer phase, the link station follows the resetting procedure described in "Procedures for Resetting" on page 11-25.

Link Deactivation

The Physical Unit can initiate disconnection of the link at any time during the data transfer phase by issuing the DISCONNECT_ALS record to DLC.LAN.MGR. Figure 11-5 and Figure 11-6 on page 11-19 show the flows associated with link disconnection.

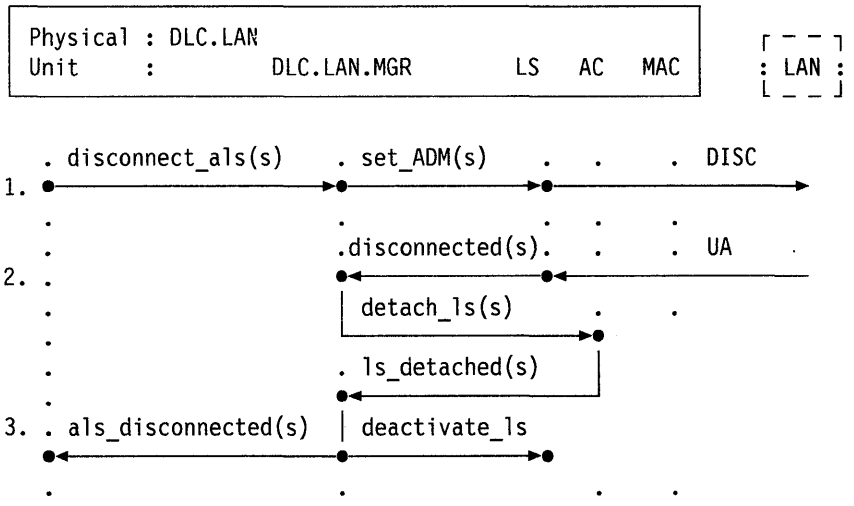


Figure 11-5. Disconnection Flows for DLC.LAN Link Stations

The following notes are keyed to Figure 11-5:

1. The Physical Unit starts the disconnection procedure by issuing DISCONNECT_ALS with the adjacent link station identifier of the link to be disconnected. DLC.LAN.MGR translates that to a set_ADM signal to the link station.
2. If a DM or UA response LPDU is received, DLC.LAN.MGR is notified. If a DM or UA response LPDU is not received within the T1 time-out period, the DISC command LPDU is retransmitted. This is repeated until the retry count is reached, at which point the link station enters the asynchronous disconnected mode and notifies DLC.LAN.MGR of the results. In either case, once the link station has notified DLC.LAN.MGR that the link is inactive, DLC.LAN.MGR notifies the access channel control that the link station no longer exists and releases the link station resources.
3. Once the link station resources are released, DLC.LAN.MGR notifies the Physical Unit that the disconnection procedure has completed.

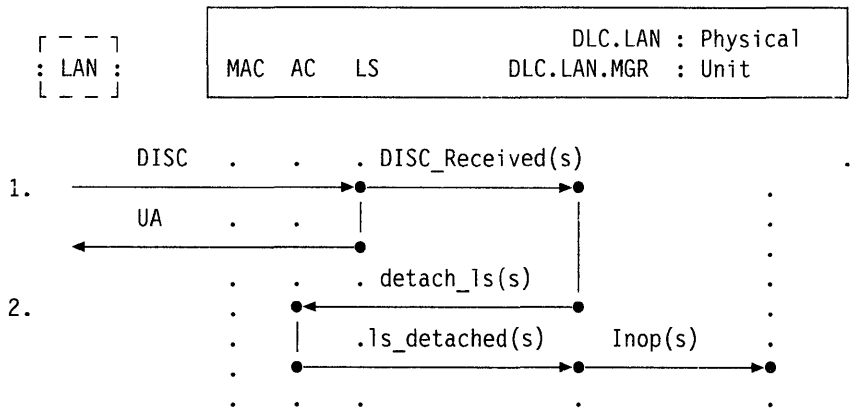


Figure 11-6. Disconnection Flows for DLC.LAN Link Stations

The following notes are keyed to Figure 11-6.

1. When a DISC command LPDU is received, the link station notifies DLC.LAN.MGR and sends a UA response LPDU to the remote link station.
2. DLC.LAN.MGR requests the access channel control to delete the link station from its routing table. Upon completion of this procedure, DLC.LAN.MGR notifies the Physical Unit of the link disconnection.

Contention of Unnumbered Mode-Setting Commands

A contention situation (that is, both link stations send a mode-setting command) is resolved as follows:

- If the two mode-setting command LPDUs are the same, each link station sends a UA response LPDU at the earliest opportunity and enters the indicated mode after receiving the UA response LPDU, or after its T1 timer expires.
- If the two mode-setting command LPDUs are different, each link station enters asynchronous disconnected mode and sends the DM response at the earliest opportunity.

Procedures for Information Transfer

This section describes the procedures that apply to exchanging I-format LPDUs in the data transfer phase.

Sending I-Format LPDUs

A link station sends an I-format LPDU with N(S) set to its current V(S), and with N(R) set to its current V(R). After sending the I-format LPDU, the link station increments its V(S) by one. If T1 is not already running, it is started.

If V(S) is equal to V(A) + Ww, the link station cannot send any new I-format LPDUs. The link station can send RR or RNR LPDUs (as appropriate) to request status, and can send unaccepted I-format LPDUs again.

In most cases, I-format LPDUs are sent as commands with the P bit set to B'0'. Transmission of an I-format LPDU as a command with the P bit set to B'1' is allowed during the dynamic window algorithm (see "The Dynamic Window Algorithm" on page 11-23) when the link station has reached the end of its working

window [V(S) = V(A) + Ww - 1]. The link station sets the P bit to B'1' in the last I-format command LPDU it sends.

A link station can send an I-format LPDU as a response with the F bit set to B'1' if the station has received a command with the P bit set to B'1', and if the station is waiting to send an I-format LPDU. The sending of an I-format LPDU response with the F bit set to B'1' indicates the clearance of any busy condition at the sending link station.

When a link station is in the busy condition, it is still able to send I-format LPDUs, provided that the remote link station is not itself in a busy condition (described in "Busy Condition" on page 11-21). When the link station is in the FRMR exception condition, it stops initiating LPDU transmission on that link until the link is reset.

Receiving I-Format LPDUs

When the link station has buffers available and receives an I-format LPDU whose N(S) is equal to the link station's V(R), the link station accepts the information field of the I-format LPDU, increments its V(R) by 1, and does the following:

- If the I-format LPDU was a command with the P bit set to B'1', the link station sends an RR response LPDU with the F bit set to B'1' and with N(R) equal to the value of the link station's V(R). The link station can also acknowledge the I-format command LPDU by sending an I-format response LPDU (if an I-format LPDU is available to be sent) with the F bit set to B'1'.
- If the I-format LPDU was a command with the P bit set to B'0', or a response with the F bit set to B'0' or B'1', and if an I-format LPDU is available to be sent, the link station acts as in "Sending I-Format LPDUs" on page 11-19 and acknowledges the received I-format LPDU by setting N(R) in the next sent I-format LPDU to the value of V(R). The station can also acknowledge the I-format LPDU by sending an RR LPDU with N(R) equal to V(R).
- If receiving the I-format LPDU caused the link station to enter the busy condition (with regard to subsequent I-format LPDUs), the link station sends an RNR LPDU with N(R) equal to its V(R). If an I-format LPDU is available, the link station can send it as defined in "Sending I-Format LPDUs" on page 11-19 before or after sending the RNR LPDU.

Receiving an Out-of-Sequence I-Format LPDU

When a link station receives an out-of-sequence I-format LPDU, — for example, one whose send sequence number N(S) is not equal to the link station's receive state variable V(R), — the link station discards the I-format LPDU and sends an REJ response LPDU with the N(R) set to V(R). The link station then discards the information field of all received I-format LPDUs until the expected I-format LPDU is received. When receiving the expected I-format LPDU, the link station acknowledges reception as described above in "Receiving I-Format LPDUs." The link station uses the N(R) and P/F bit indications in the discarded I-format LPDUs.

Only one "sent REJ" exception condition can be established at one time from one link station to another (the condition can exist in both directions). The "sent REJ" condition is cleared only when the next in-sequence I-format LPDU is received.

It is the sender's responsibility to ensure that I-format LPDUs are delivered. For example, a remote link station detects an out-of-sequence condition and sends an REJ LPDU. If either the REJ LPDU or the retransmitted I-format LPDU is lost, the sender's T1 expires on that I-format LPDU. The sender then sends an RR

command LPDU with the P bit set to B'1'. If the N(R) in the next supervisory response LPDU with the F bit set to B'1' is not equal to the sender's V(S) (which is the case in this example), retransmission is started with the I-format LPDU whose N(S) equals N(R). Thus the link station that detects the out-of-sequence condition does not need to retransmit the REJ LPDU.

Receiving Acknowledgment

When correctly receiving an I- or S-format LPDU, even in the busy condition (see "Busy Condition"), the receiving link station considers the N(R) in the LPDU an acknowledgment for all I-format LPDUs it has sent with an N(S) up to and including the received N(R) minus 1. The link station resets the timer T1 when it correctly receives an I- or S-format LPDU with an N(R) higher than the last received N(R) (actually acknowledging some I-format LPDUs).

If the timer has been reset and outstanding I-format LPDUs are still unacknowledged, the link station restarts timer T1. If T1 expires, the link station follows the retransmitting procedure (described in "Waiting Acknowledgment" on page 11-22) for the unacknowledged I-format LPDUs.

Receiving an REJ LPDU

When receiving an REJ LPDU, the link station sets its send state variable V(S) to the N(R) received in the REJ LPDU control field. The link station then (re)sends the corresponding I-format LPDU. If other unacknowledged I-format LPDUs had been sent on that link following the one indicated in the REJ LPDU, then those I-format LPDUs are retransmitted by the link station following the retransmitting of the requested I-format LPDU.

Receiving an RNR LPDU

A link station receiving an RNR LPDU stops sending I-format LPDUs on that link. If timer T1 expires, the link station follows the procedure described in "Reply Timer (T1)" on page 11-3.

In any case, the link station does not send any other I-format LPDUs on that link until it receives an RR or REJ LPDU (indicating that the busy condition at the remote link station is clear), or an I-format response LPDU with the F bit set to B'1' (in response to a command LPDU with the P bit set to B'1'), or until the completion of a resetting procedure.

Busy Condition

A link station enters the busy condition on a link when it is temporarily unable to receive I-format LPDUs due to internal constraints (for example, buffering limitations). When a link station enters the busy condition, it sends an RNR response LPDU at the earliest opportunity. The link station may send I-format LPDUs on that link before or after sending the RNR LPDU. While in the busy condition, the link station accepts and processes S-format LPDUs and returns an RNR response LPDU with the F bit set to B'1' if it receives an I-format or S-format command LPDU with the P bit set to B'1'.

To indicate the clearance of the busy condition, the link station sends either an RR or REJ LPDU, or an I-format response LPDU with the F bit set to B'1' (in response to a command LPDU with the P bit set to B'1'), with the N(R) set to the current receive state variable V(R). Additionally, the sending of an SABME command LPDU or a UA response LPDU (to a DISC or SABME command LPDU) indicates the clearing of a busy condition at the sending link station.

Waiting Acknowledgment

The conditions for starting T1 are described in "Reply Timer (T1)" on page 11-3. If the timer T1 expires, one of the following events has occurred:

- An acknowledgment for an outstanding I-format LPDU has not been received.
- The link station sent an S-format or I-format command LPDU with the P bit set to B'1' and did not receive an S-format or I-format response LPDU with the F bit set to B'1'. This condition should exist only in one of the checkpointing states.

Two retry counters (S_retry and I_retry) are kept to prevent continuous cycling from I-format LPDU retry to checkpoint retry and back. If the reply or inactivity timer has expired and no poll was outstanding, the loop on page 11-22 is entered at step 1. If the reply timer expires and a supervisory or information poll was outstanding, the loop is entered at step 4. If the reply timer expires and no poll was outstanding but a retransmitted I-format LPDU remains unacknowledged, the loop is entered at step 2.

1. Set I_retry to N2.
2. If the I_retry counter is equal to X'0', the link station notifies DLC.LAN.MGR that the link has failed and assumes the asynchronous disconnected mode.
3. Set S_retry to N2.
4. If S_retry is equal to X'0', the link station notifies DLC.LAN.MGR that the link has failed and assumes the asynchronous disconnected mode.
5. If S_retry is not X'0', the link station sends an RR (if not busy) or RNR (if busy) command LPDU with the P bit set to B'1', resets and starts timer T1, and decrements the S_retry counter.
6. If T1 expires again, the procedure goes to step 4.
7. If the link station receives an I-format or S-format response LPDU with the F bit set to B'1' and with an N(R) equal to or greater than the link station's V(P), all outstanding I-format LPDUs transmitted before sending the RR or RNR command LPDU with the P bit set to B'1' have been acknowledged. The S_retry and I_retry counters are reset, last_received._Nr is set to the received N(R) value, and the procedure exits from the loop.

   If the received N(R) is greater than the last_received_Nr but less than V(P), indicating that some I-format LPDUs have been acknowledged, the link station sets V(S) and last_received_Nr to the received N(R) value, resets I_retry and S_retry, and the procedure exits from the loop.

   Note: If the link station receives an I-format or S-format command or response LPDU with the P/F bit set to B'0', it can use the received N(R) as an acknowledgment of sent I-format LPDUs but does not reset T1.
8. If the received N(R) is equal to the last_received_Nr and is not equal to or greater than V(P), indicating that the oldest outstanding I-format LPDU must be retransmitted, the I_retry counter is decremented, and the I-format LPDU is retransmitted. If I_retry is equal to X'00' (after decrementing) and a routing information field is required to communicate with the adjacent link station (for example, the adjacent link station resides in a node on a different segment), bit 1 in the routing control field (see "Path Trace Function" on page 18-1) should be set to B'1'. After retransmitting the I-format LPDU, the procedure exits from the loop.

Note: Link stations should use Poll/Final checkpointing in data transfer only when the response timer or the inactivity timer expires, or when the transmit window limit has been reached during activation of the dynamic window algorithm. I-format command LPDUs with the P bit set to B'1' are sent only when the transmit window has been reached. S-format command LPDUs with the P bit set to B'1' are sent only when either T1 or Ti expires; S-format response LPDUs are sent in any other situations requiring the sending of S-format LPDUs.

The Dynamic Window Algorithm

The bridges that interconnect multiple-ring local area networks have finite receive buffer capacity. When these buffers are full, the bridge cannot copy and forward subsequent frames; they are discarded. Link stations that detect this condition and retransmit lost frames may only be causing the congestion to increase.

The dynamic window algorithm is a way to control bridge congestion in multiple-ring local area networks. This algorithm is mandatory on links where the link stations reside in nodes on different rings. The algorithm modifies the transmitting link station's send window when congestion is first detected, and again as the congestion decreases. The receiving link station does not participate in the algorithm and does not require knowledge of the sending link station's participation. Congestion in one direction of a link is treated independently of congestion in the other direction.

If a link station's transmit window (TW) parameter is set to 1, the working window (Ww) parameter will always have the value 1 and the algorithm need not be invoked. If the link station's TW is greater than 1, the station uses a Ww equal to TW in the absence of congestion.

Congestion is indicated by the loss of I-format LPDUs. A link station detects this loss:

- When the link station receives an REJ LPDU
- When T1 expires and the link station transmits an RR command LPDU with the P bit set to B'1', and subsequently receives an I-format or S-format response LPDU in which the F bit is set to B'1', but in which the value of N(R) is less than the link station's poll state variable V(P).

When a link station detects either of these events, it invokes the dynamic window algorithm, setting its Ww to 1. With a send window size of 1, the link station must wait for an acknowledgment after every I-format LPDU transmission. When Nw consecutive I-format LPDUs are successfully transmitted and acknowledged, Ww is increased by 1. As I-format LPDUs are successfully transmitted, Ww gradually increases until it reaches its maximum value, TW. At this point, the dynamic window algorithm ends. The algorithm is shown in the following figure. For illustration purposes, Ia_Ct is introduced to count the number of I-format LPDUs acknowledged since Ww was last incremented; q and r are the quotient and remainder of the indicated division. (See Chapter 12, "State Tables" on page 12-1 for the complete definition and use of Ia_Ct.)

```
If (a REJ is received) or (a solicited I-format or S-format response with
   the F bit set to B'1' is received and the receive sequence number,
   N(R), is less than the link station's poll state variable, V(P)) then

     If the transmit window size, TW, is greater than 1 then

        Set the working window size, Ww, to 1.
        Set the information acknowledged counter, Ia_Ct, to B'0'.
        Set the send state variable, V(S), to the receive sequence
           number, N(R).
        Retransmit the I-format frame whose send sequence number
           equals the receive sequence number, N(R).

     Do while the working window size, Ww, is less than the maximum
        transmit window size, TW, holds.

         Do while acknowledgment not received holds.
            If (REJ is received) or (solicited I-format or S-format
            response with the F bit set to B'1' is received
            and the receive sequence number, N(R), is less than the
            link station's poll state variable, V(P)) then
               Exit the algorithm and start again.

         Increment I-format frame acknowledge count by the receive
            sequence number, N(R), minus the last received N(R) variable,
            V(A), MOD 128.
            (* Ia_Ct := Ia_Ct + ((N(R) - V(A)) *)

         If the I-format frame acknowledged count, Ia_Ct, is
            greater than the number of I-format LPDUs to be
            acknowledged before incrementing Ww, Nw, then

            Set the remainder of I-format frame acknowledged count,
               Ia_Ct, divided by number of I-format LPDUs to be
               be acknowledged before incrementing Ww, Nw, to I-format
               frame acknowledged count, Ia_Ct.
               (* Ia_Ct := remainder (Ia_Ct / Nw) *)

            Set the working window size, Ww, to the minimum of working
               window plus the quotient of the above divide, or the
               maximum transmit window size, TW.
```

Figure 11-7. Dynamic Window Algorithm. If a lost LPDU is detected during the time Ww < TW, the link station restarts the algorithm.

T2/N3 Interaction with the Dynamic Window Algorithm

The algorithm works well when each link station acknowledges every valid I-format LPDU it receives. However, when T2 and N3 are used to reduce the number of RR LPDUs sent as acknowledgments (see "Receiver Acknowledgment Timer (T2)" on page 11-3 and "Number of I-Format LPDUs Received before Sending Acknowledgment (N3)" on page 11-4), the remote link station's N3 must be less than the local link station's Ww, or the remote link station will wait until T2 expires before sending an acknowledgment.

To force the receiver to acknowledge frames when the transmitter has invoked the algorithm and reached the end of its Ww, the transmitter sets the P bit to B'1' in

the last I-format command sent. The remote link station must then send a response LPDU with the F bit set to B'1' as soon as possible. Because the receiving station must acknowledge before either T2 or N3 expires, the receiver does not require knowledge of the transmitting station's activation of the algorithm.

If acknowledgments are received after an I-format command LPDU with the P bit set to B'1' is sent, but before a response LPDU with the F bit set to B'1' is received, the link station can transmit I-format LPDUs until the end of Ww is again reached. The P/F bit of the last I-format LPDU allowed to be sent is set to B'0' when a response with the F bit set to B'1' is expected.

Procedures for Resetting

The resetting phase is used to initialize both directions of data transfer and only applies when both link stations are in the asynchronous balanced mode. With the IEEE 802.2 standard, either node may initiate a resetting of the link by issuing contact, which causes the link station to send the SABME command LPDU and start its T1 timer.

On receiving an SABME command, the link station notifies DLC.LAN.MGR, which notifies the Physical Unit. The Physical Unit can then proceed with the resetting procedure by issuing a contact signal, or it can terminate the resetting procedure by issuing disconnect_als. Contact results in the sending of a UA response and the resetting of the state variables. If the initiating link station receives the UA response, it sets its send and receive state variables V(S) and V(R) to X'0' and stops timer T1. This also clears all exception conditions *and* any busy condition that might have been present at either of the link stations involved in the reset.

If a DM response is received, the link station enters asynchronous disconnected mode, stops its T1 timer, and notifies DLC.LAN.MGR for appropriate action.

If timer T1 expires, the retry procedure described in "Reply Timer (T1)" on page 11-3 is followed. If no response is received after the retry procedure ends, the link station stops sending the SABME command LPDU, enters the asynchronous disconnected mode and notifies DLC.LAN.MGR.

Any other LPDUs (with the exception of SABME and DISC command LPDUs) received before completion of the reset procedure are discarded.

Under certain FRMR conditions listed in "FRMR Exception Conditions" on page 11-26 below, it is possible for the link station to request the remote link station to reset the link by sending the FRMR response LPDU. After sending an FRMR response LPDU, the link station enters the FRMR exception condition. This condition is cleared when the link station receives or sends an SABME or DISC command or DM response. Any other command LPDU received while in the FRMR exception condition causes the link station to resend the FRMR response LPDU with the same information field as sent originally. In the FRMR exception condition, additional I-format LPDUs cannot be sent, and received I-format and S-format LPDUs are discarded.

FRMR Exception Conditions

The conditions that cause an FRMR exception condition are:

- The received control field represents an invalid or unsupported command or response. This includes the following:
  - SABME response
  - DISC response
  - FRMR command
  - DM command
  - UA command
  - Other unsupported commands and responses.
- The received frame contains an information field that is not allowed with that control field, for example, with SABME.
- The length of the information field in the received I-format LPDU exceeds the available buffer capacity. (This is an implementation option.)
- The received sequence number in the received control field does not refer to either the next I-format LPDU to be transmitted or to an I-format LPDU that has been transmitted but not acknowledged.
- The received FRMR response LPDU has an information field that is not 5 bytes in length.

Upon recognition and establishment of an FRMR exception condition, a link station sends an FRMR response LPDU with an information field as defined in "FRMR Exception Conditions" and informs DLC.LAN.MGR.

Until the FRMR exception condition is reset as described in "Procedures for Resetting" on page 11-25, no other LPDUs are sent or accepted except for the retransmission of FRMR upon receipt of a command LPDU.

Link Station Protocol Boundaries

Link stations have protocol boundaries with path control, DLC.LAN.MGR, and the access channel control. BTUs are exchanged between a link station and path control, while access channel control service data units (ACSDUs) are passed between the access channel control and link stations. (The ACSDU consists of the control field and information field of the LPDU.) The protocol boundary with DLC.LAN.MGR handles flows for link activation and deactivation, and status and error reporting. (For a description of this DLC.LAN.MGR protocol boundary, see "DLC.LAN.MGR-to- Link Station Records" on page 4-7.)

Link Station-to-Path Control

The signals exchanged between path control and DLC to accomplish the transfer of BTUs are described in this section (with parameters passed in parentheses). The details of the link station-to-path control interface are implementation-dependent.

- SEND_BTU(ALS_identifier, BTU): SEND BTU contains a BTU that is being sent out of the node via a particular link station.
- BTU_RECEIVED(ALS_identifier, BTU, lost_data_indicator): RECEIVE BTU contains a BTU received by a local link station from a remote link station. (Although MAC does not pass up data units that are too large for its own buffers, the data units may be too large for the access channel control's buffers. In this situation, the lost_data_indicator may be used.)
- REQUEST_BTU(ALS_identifier, number_of_BTUs_requested): REQUEST BTU is sent to path control when a link station is ready to send data, in I-format LPDUs, to the remote link station. The number_of_BTUs_requested parameter is used to control the number of SEND_BTU signals path control can issue at a given time.

Link Station-to-Access Channel Control

The data unit exchanged between link stations and the access channel control is called an access channel control service data unit (ACSDU). The ACSDU consists of the control field and information field of the LPDU.

The only information exchanged between link stations and the access channel control are ACSDUs. The protocol boundary consists of two signals, one in each direction:

- SEND_ACSDU(ALS identifier, ACSDU): Signals the access channel control to build an LPDU for the given station using the given ACSDU. The access channel control keeps track of the routing information, addresses, and output port for each ALS identifier.
- RCV_ACSDU(ALS identifier, ACSDU): Indicates to the link station that an LPDU for that link station has been received. The ACSDU contains the pertinent control and information fields.

Chapter 12. State Tables

The finite state machines in this chapter specify the actions taken by DLC.LAN.LINK_STATION components as a result of inputs originated by DLC.LAN.MGR, by SNA Path Control, and as a result of LPDUs received from adjacent link stations. The charts cover all combinations of states and events that can occur. The protocols defined comply with ISO-HDLC ABM standards, that is, ABM plus options 1 (XID), 2 (REJ), 10 (modulo 128), and 12 (TEST). The protocols are also compatible with IEEE 802.2, and conform to IEEE 802 Logical Link Control Type 2 (connection-oriented) procedures. The state tables for Type 1 (connectionless) procedures are not included, but support of Type 1 procedures is required.

The state tables illustrate one method of implementing support for XID and TEST LPDUs. It is acceptable for XID and TEST LPDUs to be implemented in such a way that they are decoupled from LPDUs sent by link stations.

The state tables include the dynamic window algorithm, even though its implementation is only mandatory for links whose link stations reside in nodes on different rings.

Notation and Terminology

States and events are given descriptive names with several of the state names being derived by combining the names of two or more states to provide meaningful definition. Events that are controlled by higher-level protocols are referred to as function requests. From an implementation point of view some of these function requests may be issued dynamically or may be static conditions that are preset in program or hardware. For example, entering a busy condition may be automatic if link-level buffers are depleted, or it may be initiated by higher layers for flow-control purposes.

LPDU transfers are identified by abbreviated identifiers as shown in parentheses under "HDLC-ABM Command and Response Repertoire" on page 12-4 for the standard HDLC/SDLC commands and responses used. S-format command and response LPDUs (RR, RNR, and REJ) are differentiated by -c and -r, respectively (for example, REJ-c is the S-format Reject command LPDU). The value of the P/F bit is denoted by (abbreviation)^0 or (abbreviation)^1. In cases where no command/response or P/F bit differentiation need be made, the -c/-r and $^0/^1$ are omitted. When a command LPDU is received and the corresponding action requires the sending of a response LPDU, the F bit in the sent response LPDU is set to the same value (^{0} or ^{1}) as the P bit in the received command.

A series of charts is included for each of the various states that a LINK_STATION can assume. Each chart is divided into three columns: Inputs, Action(s) — [LPDU Transfer], and New State, where:

- Inputs are signals from DLC.LAN.MGR or from Path Control, or they are LPDUs received from the remote link station, or they are internally generated timing events. Predicate conditions that affect the actions taken in the current state are shown in parentheses; an example is (Ts=Vc=0). This is described more fully in "Predicate Conditions" on page 12-10.
- Action(s) specifies the action taken by the link station (if any) and [LPDU Transfer] specifies the LPDU transferred across the logical link (if any).
- New State is the state assumed by the LINK_STATION as a result of the specific stimulus and predicate condition in the particular state.

State Table Abbreviations

P_Ct	Poll retry count (used in checkpointing)
Ia_Ct	Number of I-format LPDUs acknowledged since Ww was last incremented
Ir_Ct	Received I-format LPDU count (used to delay acknowledgments)
Is_Ct	I-format LPDU retry count (used to limit number of I-format LPDU retries)
IH	Inform higher layers
IS_Ic	In-sequence I-format command
IS_Ir	In-sequence I-format response
Nr	LPDU receive sequence number
Ns	LPDU send sequence number
Nw	Number of consecutive I-format LPDUs to be acknowledged before incrementing Ww (used with Ia_Ct)
N2	Number of retries allowed (same for poll and I-format LPDU retries)
N3	Number of I-format LPDUs between acks (used with Ir_Ct)
OS_Ic	Out-of-sequence I-format command
OS_Ir	Out-of-sequence I-format response
Pf	Value of P bit in last command (other than XID or TEST) received
Pt	Value of P bit in last received TEST command LPDU
Px	Value of P bit in last received XID command LPDU
Ti	Inactivity timer
Ts	Link test status
T1	Reply timer
T2	Acknowledgment delay timer
TW	Maximum transmit window size
Va	Acknowledge state variable (last valid Nr received)
Vb	Busy state variable (local, remote, or both)
Vc	Stacked command variable (DISC, TEST-c, or XID-c)
Vi	Initialization state variable
Vp	Poll state variable (value of Ns when last command with P bit set to B'1' was sent)
Vr	Receive state variable (next in-sequence Ns to be accepted)
Vs	Send state variable (Ns value for next I-format LPDU to be transferred)
Ww	Working transmit window size
Xs	Exchange identification information status

Notes:

1. Ts and Xs are required for support of responses to TEST and XID by ring stations.
2. T2, N3, and Ia_Ct are required for support of T2.
3. Nw, TW, and Ww are required for support of the dynamic window algorithm.
4. Vp is required if I-frame transmission is allowed during checkpointing initiated by an action other than time out.

HDLC-ABM Command and Response Repertoire

The HDLC-ABM commands and responses used by DLC.LAN components for logical link control and the defined constraints are as follows:

COMMANDS	*RESPONSES*
DISC (DISC-c)	DM (DM-r)
I (I-c)	I (I-r)
REJ (REJ-c)	REJ (REJ-r)
RNR (RNR-c)	RNR (RNR-r)
RR (RR-c)	RR (RR-r)
SABME (SABME-c)	UA (UA-r)
TEST (TEST-c)	TEST (TEST-r)
XID (XID-c)	XID (XID-r)
	FRMR (FRMR)

Notes on the above commands and responses applicable to DLC.LAN:

1. In states where the receipt of in-sequence or out-of-sequence I-format LPDUs makes a difference, in-sequence I-format LPDUs are denoted by IS_Ic (IS_Ir) and out-of-sequence I-format LPDUs are denoted by OS_Ic (OS_Ir).
2. All S-format commands (RR, RNR, REJ) and the U-format commands SABME, DISC, XID, and TEST are always transmitted with the P bit set to B'1'.

State Descriptions

(01) LINK_CLOSED

The link station does not exist, that is, it has no resources (control blocks) and is not known to the system. The only input allowed in this state is ACTIVATE_LS.

(02) DISCONNECTED

The link station has resources and is known to DLC.LAN.MGR and to the access channel control. In this state, the link station can send and receive XID, TEST, SABME, and DISC commands and XID, TEST, UA, and DM responses.

(03) LINK_OPENING

The link station changes from DISCONNECTED to LINK_OPENING after either sending SABME or after sending UA in response to a received SABME. Vi keeps track of which action caused the transition. Exit from the LINK_OPENING state (following the normal sequence of events) occurs when the remote link station responds to an SABME (with UA) or to a UA (with I, RNR, or RR); transition to LINK_OPENED causes the link station to pass the CONTACTED signal to DLC.LAN.MGR.

(04) DISCONNECTING

Link stations enter this state upon sending the DISC command LPDU. This occurs once the link station has been in any of the opened states (states 5 through 21) and received the SET_ADM signal from DLC.LAN.MGR. This state can also be entered from FRMR_RCVD if Ti expires N2 times.

(05) FRMR_SENT

The link station has entered the frame reject exception state and has sent an FRMR response LPDU across the link. The only inputs that cause a change to another state are the receipt of mode-setting command LPDUs (SABME or DISC), the sending of mode-setting command LPDUs (after receiving SET_ABME or SET_ADM signals from DLC.LAN.MGR), or receiving the FRMR response.

(06) LINK_OPENED

The link station is in the data transfer phase of the asynchronous balanced mode of operation and no error or exception conditions are pending.

(07) LOCAL_BUSY

Link stations in this state are unable to receive additional I-format LPDUs, due to internal resource constraints, and have transferred an RNR response LPDU to the remote link station. No other error or exception conditions are present in this state.

(08) REJECTION

Link stations having received one or more out-of-sequence I-format LPDUs establish a REJECTION condition by transferring an REJ response to the remote link station. The condition is corrected by receipt of the next in-sequence I-format LPDU.

(09) CHECKPOINTING

This state indicates that a poll (either RR, RNR, XID, TEST, or I-format command LPDU with the P bit set to B'1') has been sent and the link station is waiting for the appropriate response LPDU with the F bit set to B'1'. The

state tables do not verify that the frame received in response to a poll is of the same type frame as the poll although implementations should include such a check.

(10) CHECKPOINTING + LOCAL_BUSY

A combination of the CHECKPOINTING and LOCAL_BUSY states described above. The state tables do not verify that the frame received in response to a poll is of the same type frame as the poll although implementations should include such a check.

(11) CHECKPOINTING + REJECTION

A combination of the CHECKPOINTING and REJECTION states described above. The state tables do not verify that the frame received in response to a poll is of the same type frame as the poll although implementations should include such a check.

(12) RESETTING

Transition to this state occurs when the link station receives an SABME command LPDU, indicating the remote link station wishes to reset the link after the initial link establishment. The link station (under control of DLC.LAN.MGR) normally sends either a DM or UA response LPDU in this state.

(13) REMOTE_BUSY

This state is entered when the link station has received the RNR LPDU from the adjacent link station, indicating it is temporarily unable to receive additional I-format LPDUs due to internal resource constraints. The remote busy condition is cleared by the receipt of an RR or REJ LPDU, or an I-format LPDU response with the F bit set to B'1'.

(14) LOCAL_BUSY + REMOTE_BUSY

A combination of the LOCAL_BUSY and REMOTE_BUSY states described above.

(15) REJECTION + LOCAL_BUSY

A combination of the REJECTION and LOCAL_BUSY states described above.

(16) REJECTION + REMOTE BUSY

A combination of the REJECTION and REMOTE_BUSY states described above.

(17) CHECKPOINTING + REJECTION + LOCAL_BUSY

A combination of the CHECKPOINTING, REJECTION and LOCAL_BUSY states described above. In this state, the state tables do not verify that the frame received in response to a poll is of the same type frame as the poll although implementations should include such a check.

(18) CHECKPOINTING + CLEARING

A combination state resulting from the termination of a LOCAL_BUSY condition while the link station is in the CHECKPOINTING + LOCAL_BUSY state. Normally the remote link station is informed (that the local busy condition has cleared) by entering a checkpointing operation. However, in CHECKPOINTING + LOCAL_BUSY, this is not possible, hence the different state to remember that the remote link station must be informed after the checkpointing has completed. In this state, the state tables do not verify that the

frame received in response to a poll is of the same type frame as the poll although implementations should include such a check.

(19) CHECKPOINTING + REJECTION + CLEARING

A combination state resulting from the transfer of an unconfirmed local busy clear (RR-r^0) while the link station is in the CHECKPOINTING + REJECTION + LOCAL_BUSY state described above. In this state, the state tables do not verify that the frame received in response to a poll is of the same type frame as the poll although implementations should include such a check.

(20) REJECTION + LOCAL_BUSY + REMOTE_BUSY

A combination state of the REJECTION, LOCAL_BUSY and REMOTE_BUSY states described above.

(21) FRMR_RECEIVED

The link station has received an FRMR response LPDU from the adjacent link station and awaits action by the higher layer.

Input Descriptions

Inputs that result in LINK_STATION actions or LPDU transfers include events, resulting from local conditions or higher-level function requests, or the receipt of command/response LPDUs from the adjacent link station.

Internal Events

The following signals are initiated by DLC.LAN.MGR:

ACTIVATE_LS (Activate Link Station)

Receipt of an activate link station function request from DLC.LAN.MGR.

DEACTIVATE_LS (Deactivate Link Station)

Receipt of a deactivate link station function request from DLC.LAN.MGR.

SABME (Set Asynchronous Balanced Mode — Extended)

Receipt of a function request from DLC.LAN.MGR to place the link in the asynchronous balanced mode of operation.

TEST_LINK (Test Link Connection)

Receipt of a function request from DLC.LAN.MGR to execute a test of the link.

SEND_XID (Exchange Link Station Identification)

Receipt of a function request from DLC.LAN.MGR to exchange link station identification information with the adjacent link station.

SET_ADM (Set Asynchronous Disconnected Mode)

Receipt of a function request from DLC.LAN.MGR to place the link in the asynchronous disconnected mode.

The following signals are generated by internal interactions between the link station and the run-time environment:

ENTER_LCL_Busy (Enter Local Busy Condition)

Receipt of a function request from DLC.LAN.MGR or an internal signal to inform the adjacent link station that further I-format LPDUs temporarily cannot be accepted.

EXIT_LCL_Busy (Leave Local Busy Condition)

Receipt of a function request from DLC.LAN.MGR or an internal signal to inform the adjacent link station that further I-format LPDUs can again be accepted.

SEND_I_POLL (Send I-frame Command Poll)

An I-format LPDU is sent as a command frame with the P bit set to B'1' when the dynamic window algorithm is active and the end of the working window is reached. After sending an I-format LPDU with the P bit set to B'1', the link station enters a checkpointing state.

T1_Expired (Expiration of Reply Timer, T1)

T1 is started and restarted at appropriate times to monitor for expected responses containing the acknowledgment of information frames or for an F bit response expected as the result of sending a P bit command. If T1 expires before the expected response is received, recovery action is initiated (for example, SABME, DISC, RR-c, RNR-c, XID-c, or TEST-c is

[re]transmitted). The recovery may be attempted N2 times, where N2 is a system-defined parameter. When T1 expires, it is considered stopped.

Ti_Expired (Expiration of Inactivity Timer, Ti)

The inactivity timer, Ti, is started at appropriate times to monitor idle periods to maintain the integrity of the link. If Ti expires, the link station initiates a checkpointing action or informs higher layers, depending on the current state. When Ti expires, it is considered stopped.

T2_Expired (Expiration of ACK Delay Timer, T2)

The T2 timer is used in conjunction with the Ir_Ct to reduce the number of acknowledgments that must be sent. When the link station is able to receive I-format LPDUs, the expiration of T2 causes an acknowledgment to be sent (an RR response with the F bit set to B'0'). The timer is reset whenever an I- or S-format LPDU is sent. When T2 expires, it is considered stopped.

LPDU_INVALID (Receipt of Invalid LPDU)

Invalid LPDUs are discarded and may result in the establishment of an FRMR exception condition and the sending of the FRMR response with appropriate information field. The following table lists the set of FRMR reason codes, in terms of the VWXYZ bits in the FRMR information field, and the invalid LPDU definitions associated with each.

VWXYZ Definition

01000 The control field either is not implemented or is invalid. It is invalid if an unsolicited UA response LPDU was received.

01100 An LPDU was received with an information field present but the control field does not permit one.

00010 The information field length exceeded the allowed system value, N1.

Note: An I-format LPDU containing an information field that exceeds buffering capacity may optionally be passed to path control in BTU_RECEIVED with the "lost data" indicator. This option allows nodes that already have such a capability to continue to support it.

00001 The received I- or S-format LPDU contained an Nr value that was not in the range $Va \leq Nr \leq Vs$.

The following signal is passed by path control to the link station:

SEND_BTU (Send Basic Transmission Unit)

Path control uses this signal to pass user data to the link station for transmission to the adjacent path control element.

Received LPDUs

The LPDUs that can be received are described in "HDLC-ABM Command and Response Repertoire" on page 12-4. In addition, the internal event "LPDU_INVALID" is actually caused by receipt of an LPDU.

Predicate Conditions

For some inputs in some states, specific link station actions or LPDU transfers are determined by predicate conditions, shown in parentheses in the state tables. These predicate conditions function as flags or simple two-state or four-state FSMs; that is, they "remember" the occurrence of certain events until particular subsequent events occurs. Many of the predicates are related to variables or status flags defined in "State Table Abbreviations" on page 12-3. For example, the Xs status flag indicates whether an XID command has been sent or received, or both, with the predicate values IXp, OXp, and IOXp.

Predicate Values for Initialization State Variable (Vi)

LIp (Local Initialization Pending)

Vi = LIp indicates that an SABME command LPDU has been sent as a result of a SET_ABME signal from DLC.LAN.MGR. When a corresponding UA response LPDU is received and Vi = LIp, Vi is reset to X'0'. If an SABME command LPDU is received and Vi = LIp, the link station sends a UA response LPDU and sets Vi to LRIp.

RIp (Remote Initialization Pending)

Vi = RIp indicates that an SABME command LPDU has been received from the adjacent link station when Vi was set to X'0'. When DLC.LAN.MGR issues the SET_ABME signal, the link station sends a UA response LPDU and sets Vi to Op.

LRIp (Local and Remote Initialization Pending)

Vi = LRIp indicates that an SABME command LPDU has been sent as a result of a SET_ABME signal from DLC.LAN.MGR, and that a SABME command LPDU has been received and a UA response LPDU sent to the adjacent link station. When a UA response LPDU is received or the T1 timer expires, Vi is reset to X'0'.

Op (Operational Mode Pending)

Vi = Op indicates that the link station has received an SABME command LPDU and sent a UA response LPDU. This value is necessary to ensure that the UA was received by the adjacent link station. When an S- or I-format LPDU is received from the adjacent link station, indicating that the adjacent link station has received the UA, Vi is set to X'0'. If Ti expires, Vi is set to ISp.

ISp (I/S Format Frame Pending)

Vi = ISp indicates that the link has entered the link opened state, but that a UA frame may still be received without causing a FRMR condition. Once any I or S format frames have been received, Vi will be set to not ISp and receipt of a UA will cause a FRMR to be sent.

Predicate Values for Exchange ID Status Variable (Xs):

IXp (Incoming XID Response Pending)

Xs = IXp indicates that an XID command LPDU has been sent and the link station is awaiting an XID response LPDU. If the next input is the receipt of an XID response, the link station sets Xs to X'0'. If the next input is the receipt of an XID command LPDU, the link station sets Xs = IOXp.

OXp (Outgoing XID Response Pending)

Xs = OXp indicates that an XID command LPDU has been received and the next XID sent by the link station should be sent as a response. If the link station receives a SEND_XID signal from DLC.LAN.MGR, that XID is sent as a response and Xs is reset to X'0'.

IOXp (Incoming and Outgoing XID Responses Pending)

Xs = IOXp indicates that the link station sent and then received an XID command LPDU. If the next input is the receipt of an XID response LPDU, the link station sets Xs = OXp. If the next input is the receipt of a SEND_XID signal from DLC.LAN.MGR, the link station sets Xs = IXp.

Predicate Values for Link Test Status Variable (Ts)

ITp (Incoming TEST Response Pending)

Set ON (Ts = ITp) when a U-format LPDU containing a TEST command is sent, and set OFF (Ts≠ITp) when a U-format LPDU containing the corresponding TEST response is received.

OTp (Outgoing TEST Response Pending)

Ts = OTp indicates that a TEST command LPDU has been received and the next TEST LPDU sent should be sent as a response.

IOTp (Incoming and Outgoing TEST Responses Pending)

Ts = IOTp indicates that a TEST command has been both sent and received, and corresponding TEST responses are pending both inbound and outbound. Ts is reset to ITp when the next TEST LPDU is sent (as a response), or to OTp when a TEST response LPDU is received.

Predicate Values Associated with Acknowledgments:

Va = Nr<Vs (Outstanding Acknowledgment)

The Nr in the received LPDU does not acknowledge any outstanding (previously transmitted) I-format LPDUs.

Va<Nr<Vs (Further Acknowledgment Required)

The Nr in the received LPDU acknowledges one or more previously transmitted I-format LPDUs but additional I-format LPDUs remain unacknowledged.

Nr = Vs (Acknowledgment Complete)

The Nr in the received LPDU acknowledges all previously transmitted I-format LPDUs.

Predicate Values for Busy State Variable (Vb)

Lb (Local Busy)

Vb = Lb indicates the existence of a local busy condition.

Rb (Remote Busy)

Vb = Rb indicates the existence of a remote busy condition.

LRb (Local and Remote Busy)

Vb = LRb indicates the existence of a combined local busy and remote busy condition.

Action and LPDU Transfer Descriptions

The link station actions for each state and input, shown under the "Action(s) [LPDU Transfer]" column in the state tables that begin on page 12-15, are intended to be self-explanatory. They include actions taken (if any) by the link station sub-component relative to the higher layer, disposition for exception conditions, and the resultant LPDU (if any) to be transmitted.

Actions

The terms and abbreviations used in the state tables are listed below:

Disable Link Station

Release resources allocated to the link station

Enable Link Station

Allocate link station resources sufficient to support the asynchronous balanced mode of operation.

IT1 (Initiate T1)

Signifies the initialization and starting of reply timer T1.

TT1 (Terminate T1)

Signifies the stopping of reply timer T1, if it is not already stopped.

RT1 (Restart T1)

Signifies the reinitialization and restarting of reply timer T1, or its initialization and starting if it is not already running.

CIT1 (Conditionally Initiate T1)

Signifies the starting of reply timer T1 if it is not already running.

IT2 (Initiate T2)

Signifies the initialization and starting of ack delay timer T2.

TT2 (Terminate T2)

Signifies the stopping of ack delay timer T2 if it is not already stopped.

CIT2 (Conditionally Initiate T2)

Signifies the starting of the ack delay timer T2 if it is not already running.

ITi (Initiate Ti)

Signifies the initialization and starting of inactivity timer Ti.

TTi (Terminate Ti)

Signifies the stopping of inactivity timer Ti, if it is not already stopped.

RTi (Restart Ti)

Signifies the reinitialization and restarting of inactivity timer Ti, or its initialization and starting if it is not already running.

Ignore_LPDU

Signifies the received LPDU is ignored in its entirety.

Dsc_I-fld (Discard I-field)

The information field contained in the received I-format LPDU is discarded. However, some portions of the control field may be used.

IH (Inform Higher Layer)

An action has been taken or a state change has occurred that should be made known to the higher layer.

IH_(RE) (Inform Higher Layer and Send FRMR)

The remote link station has violated the protocol, and a frame reject exception condition has been established. The higher layer should be informed.

Logical Error (Local)

The higher layer has initiated a function request that is not appropriate for the current state of the link station.

Rcv_BTU (Receive Basic Transmission Unit)

The link station performs the following operations on receiving an in-sequence I-format LPDU:

- Sets Vr:=Vr+1 so that the next I- or S-format LPDU sent acknowledges the just-received I-format LPDU
- Issues BTU_RECEIVED to path control with the received BTU.

Update_Va (Update last received Nr state variable)

This action combines the Va updating and timer actions for normal operation, using as inputs the Nr value from the received I- or S-format LPDU and the Va and Vs state variables. Vi is reset to 0 to indicate any UA received will cause a FRMR condition. This action also updates the dynamic window algorithm variables if the algorithm is active and one or more I-format LPDUs have been acknowledged.

Update_Va_Chkpt

Updates Va and other parameters when an I-format or S-format response with the F bit set to B'1' is received in a checkpointing state (9, 10, 11, 17, 18, or 19). Special actions need to be taken because the Nr value in the response to a poll indicates whether retransmission must be started. Vi is reset to 0 to indicate any UA received will cause a FRMR condition. The dynamic window algorithm variables are updated if the algorithm is active and one or more I-format LPDUs have been acknowledged. The dynamic window algorithm is activated if an I-format LPDU must be retransmitted.

Adjust_Ww

Adjusts the working window (Ww) if the dynamic window algorithm is active, and at least Nw acknowledgments have been received since the last increment or setting of Ww. The count of consecutive I-format LPDUs acknowledged is updated.

LPDU Transfers

LPDU transfers are either explicitly specified by including the LPDU name in brackets —for example, [REJ-r^0]— or specified by referring to a procedure that decides whether an LPDU transfer should take place. The following identifies the LPDU transfer descriptions found in the state tables and the procedures.

[IS_Ic^0]

Specifies transmission of the next in-sequence BTU in the information field of an I-format command LPDU with the P bit set to B'0', Ns=Vs, and Nr=Vr.

[**Send_ACK**]
Specifies whether to send an acknowledgment to a received in-sequence I-format LPDU, depending on the acknowledgment delay strategy. If an acknowledgment delay strategy determines that an acknowledgment should not be sent, then T2 is started if it is not already running.

[**Queue_I_LPDU**]
Signifies the queueing of an I-format LPDU by the link station for eventual transmission. This procedure inserts the BTU from path control in the list of BTUs to be sent; however, transmission cannot be assumed because the link station may be in a checkpointing or retransmission state, and therefore may be unable to send the BTU immediately.

[**(LPDU)-c^0|(LPDU)-c^1|(LPDU)-r^0|(LPDU)-r^1**]
Specifies transmission of a command or response LPDU with the P/F bit set to either B'0' or B'1' and Nr=Vr, where (LPDU) is the abbreviated form of the LPDU name (for example, RNR).

Send Process

The send process is a process that runs concurrently with all the actions described above and takes I-format LPDUs from the transmit (or retransmit) queue and sends them. This process is indicated in the state tables only as being stopped (for example, when entering a checkpoint state or when the remote station is busy) or started (for example, when transferring to the LINK_OPENED state). When the state tables indicate the send process should be stopped, it is stopped if it is currently running. When the state tables indicate the send process should be started, it is started if it is not already running.

The following is intended to define the actions taken by the send process:

```
Process Send_Proc
  Do while Va ≤ Vs < Va+Ww                  /* Only use allowed window        */
    If Length(I_LPDU_queue) ≠ empty then do
      If Ww≠TW & Vs=Va+Ww-1 then            /* If dynamic window active &     */
          SEND_I_POLL                       /* window reached, send Ic-poll   */
      Else do                               /* Otherwise, send Ic-no poll,    */
          [IS_Ic⁰]; TTi, CIT1               /*  update timers,                */
          Ir_Ct:=N3; TT2; end               /* and reset count of acks delayed*/
      Vs:=Vs+1; end                         /*   update Vs                    */
    Else request_BTU                        /* Otherwise request BTUs from    */
    End-do-while                            /*   path control                 */
End-process
```

LINK_CLOSED (01)

State: LINK_CLOSED (01) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS	Enable Link Station, Vi = Vb = Xs = Ts = Vc = 0, ITi		DISCONNECTED
DEACTIVATE_LS ENTER_LCL_Busy EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID	Ignore_LPDU		
SEND_BTU|SEND_XID SET_ABME|SET_ADM TEST_LINK Ti_Expired T1_Expired|T2_Expired	Logical Error (Local)		

State: LINK_CLOSED (01) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC DM FRMR SABME TEST-c|TEST-r UA XID-c|XID-r	Ignore_LPDU		

State: LINK_CLOSED (01) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
I-c|I-r	Ignore_LPDU		

State: LINK_CLOSED (01) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ|RNR|RR	Ignore_LPDU		

DISCONNECTED (02)

State: DISCONNECTED (02) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS	Logical Error (Local)		
DEACTIVATE_LS	Disable Link Station		LINK_CLOSED
ENTER_LCL_Busy	Vb = Lb		
EXIT_LCL_Busy	Vb = 0		
LPDU_INVALID	Ignore_LPDU		
SEND_BTU	Logical Error (Local)		
SEND_XID (Ts = Vc = 0, Xs = 0)	Xs = IXp, TTi, IT1, P_Ct = N2	[XID-c[1]]	
SEND_XID (Ts|Vc≠0, Xs = IXp)	Logical Error (Local)		
SEND_XID (Ts = Vc = 0, Xs = OXp)	Xs = 0	[XID-r(F = Px)]	
SEND_XID (Ts = Vc = 0, Xs = IOXp)	Xs = IXp	[XID-r(F = Px)]	
SET_ABME (Ts = Vc = 0, Vi = 0)	Vi = Ilp, Xs = 0, TTi, CIT1, P_Ct = N2, Is_Ct = N2	[SABME[1]]	LINK_OPENING
SET_ABME (Ts = Vc = 0, Vi = RIp)	Vi = Op, Xs = 0, Is_Ct = N2, Ir_Ct = N3, RTi, Va = Vs = Vr = Vp = 0	[UA(F = Pf)]	LINK_OPENING
SET_ABME (Ts|Vc≠0)	Logical Error (Local)		
SET_ADM (Vi = 0)	RTi	[DM[0]]	
SET_ADM (Vi = RIp)	Vi = 0, RTi	[DM(F = Pf)]	
TEST_LINK (Xs = Vc = 0, Ts = 0)	Ts = ITp, TTi, IT1, P_Ct = N2	[TEST-c[1]]	
TEST_LINK (Xs|Vc≠0, Ts = ITp)	Logical Error (Local)		
TEST_LINK (Xs = Vc = 0, Ts = IOTp)	Ts = ITp	[TEST-r(F = Pt)]	
TEST_LINK (Xs = Vc = 0, Ts = OTp)	Ts = 0	[TEST-r(F = Pt)]	
Ti_Expired	IH, ITi		
T1_Expired (P_Ct = 0)	IH		
T1_Expired (P_Ct≠0, Ts = 0|OTp)	Logical Error (Local)		
T1_Expired (P_Ct≠0, Ts = ITp|IOTp)	IT1, Decrement P_Ct	[TEST-c[1]]	
T1_Expired (P_Ct≠0, Xs = 0|OXp)	Logical Error (Local)		
T1_Expired (P_Ct≠0, Xs = IXp|IOXp)	IT1, Decrement P_Ct	[XID-c[1]]	
T2_Expired	Logical Error (Local)		

State: DISCONNECTED (02) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH	[DM^0|DM^1]	
DM FRMR	Ignore_LPDU		
SABME	IH, Vi = RIp, Pf = P, RTi		
TEST-c (Ts = 0|OTp)	IH, Ts = OTp, Pt = P, RTi		
TEST-c (Ts = ITp)	IH, Ts = IOTp, Pt = P		
TEST-c (Ts = IOTp)	IH, Pt = P		
TEST-r^0 TEST-r^1 (Ts = 0|OTp)	Ignore_LPDU		
TEST-r^1 (Ts = ITp)	IH, Ts = 0, TT1, ITi		
TEST-r^1 (Ts = IOTp)	IH, Ts = OTp, TT1, ITi		
UA	Ignore_LPDU		
XID-c (Xs = 0|OXp)	IH, Xs = OXp, Px = P, RTi		
XID-c (Xs = IXp)	IH, Xs = IOXp, Px = P		
XID-c (Xs = IOXp)	IH, Px = P		
XID-r^0	IH		
XID-r^1 (Xs = 0|OXp)	Ignore_LPDU		
XID-r^1 (Xs = IXp)	IH, Xs = 0, TT1, ITi		
XID-r^1 (Xs = IOXp)	IH, Xs = OXp, TT1, ITi		

State: DISCONNECTED (02) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
I-c^0|I-r^0	Ignore_LPDU		
I-c^1		[DM^1]	
I-r^1	Ignore_LPDU		

State: DISCONNECTED (02) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|RNR-c^0|RR-c^0	Ignore_LPDU		
REJ-c^1|RNR-c^1|RR-c^1		[DM^1]	
REJ-r^0|REJ-r^1 RNR-r^0|RNR-r^1 RR-r^0|RR-r^1	Ignore_LPDU		

LINK_OPENING (03)

State: LINK_OPENING (03) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb		
EXIT_LCL_Busy	Vb = 0		
LPDU_INVALID (Vi = Op, Nr≠0, I-c^0|I-r^0 |RNR-c^0|RNR-r^0|RR-c^0 |RR-r^0|REJ-c^0|REJ-r^0)	IH_(RE), RTi, VWXYZ = 00001	[FRMR-r^0]	FRMR_SENT
LPDU_INVALID (Vi = Op, Nr≠0, I-c^1|RNR-c^1|RR-c^1|REJ-c^1)	IH_(RE), RTi, VWXYZ = 00001	[FRMR-r^1]	FRMR_SENT
LPDU_INVALID (Vi≠Op)	Ignore_LPDU		
SEND_BTU|SEND_XID SET_ABME	Logical Error (Local)		
SET_ADM (Vi = LIp|LRIp)	Vi = 0, TT1, ITi	[DM^0]	DISCONNECTED
SET_ADM (Vi = Op)	Vi = 0, TTi, IT1, P_Ct = N2	[$DISC^1$]	DISCONNECTING
TEST_LINK	Logical Error (Local)		
Ti_Expired (Vb = 0)	IH, Vi = ISp, IT1, P_Ct = N2	[RR-c^1]	CHECKPOINTING
Ti_Expired (Vb = Lb)	IH, Vi = ISp, IT1, P_Ct = N2	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
T1_Expired (P_Ct = 0, Vi = LIp)	IH, Vi = 0, ITi		DISCONNECTED
T1_Expired (P_Ct≠0, Vi = LIp)	IT1, Decrement P_Ct	[$SABME^1$]	
T1_Expired (Vi = LRIp, Vb = 0)	IH, Va = Vs = Vr = Vp = 0, Vi = ISp, Ts = Xs = Vc = 0, IT1, P_Ct = N2, Is_Ct = N2, Ir_Ct = N3	[RR-c^1]	CHECKPOINTING
T1_Expired (Vi = LRIp, Vb = Lb)	IH, Va = Vs = Vr = Vp = 0, Vi = ISp, Ts = Xs = Vc = 0, IT1, P_Ct = N2, Is_Ct = N2, Ir_Ct = N3	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
T2_Expired	Logical Error (Local)		

State: LINK_OPENING (03) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, Vi = 0, TT1, RTi	[DM^0|DM^1]	DISCONNECTED
DM	IH, Vi = 0, TT1, RTi		DISCONNECTED
FRMR	Ignore_LPDU		
SABME^0|SABME^1 (Vi = Op)	RTi	[UA^0|UA^1]	
SABME^0|SABME^1 (Vi = LIp|LRip)	Vi = LRIp	[UA^0|UA^1]	
TEST-c|TEST-r UA^0	Ignore_LPDU		
UA^1 (Vi = LIp|LRIp, Vb = 0)	IH, Va = Vs = Vr = Vp = 0, Vi = ISp, Ts = Xs = Vc = 0, RT1, P_Ct = N2, Is_Ct = N2, Ir_Ct = N3	[RR-c^1]	CHECKPOINTING
UA^1 (Vi = LIp|LRIp, Vb = Lb)	IH, Va = Vs = Vr = Vp = 0, Vi = ISp, Ts = Xs = Vc = 0, RT1, P_Ct = N2, Is_Ct = N2, Ir_Ct = N3	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
UA^1 (Vi = Op)	Ignore_LPDU		
XID-c|XID-r	Ignore_LPDU		

State: LINK_OPENING (03) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, Rcv_BTU, RTi, Start Send_Proc	[Send_ACK]	LINK_OPENED
IS_I-c^0|IS_I-r^0 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, Dsc_I-fld, RTi, Start Send_Proc	[RNR-r^0]	LOCAL_BUSY
IS_I-c^1 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, Rcv_BTU, RTi, Start Send_Proc	[RR-r^1]	LINK_OPENED
IS_I-c^1 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, Dsc_I-fld, RTi, Start Send_Proc	[RNR-r^1]	LOCAL_BUSY
OS_I-c^0|OS_I-r^0 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, Dsc_I-fld, RTi, Start Send_Proc	[REJ-r^0]	REJECTION
OS_I-c^0|OS_I-r^0 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, Dsc_I-fld, RTi, Start Send_Proc	[RNR-r^0;]	LOCAL_BUSY
OS_I-c^1 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, Dsc_I-fld, RTi, Start Send_Proc	[REJ-r^1]	REJECTION
OS_I-c^1 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, Dsc_I-fld, RTi, Start Send_Proc	[RNR-r^1]	LOCAL_BUSY

State: LINK_OPENING (03) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
RNR-c^0|RNR-r^0 (Vi = Op, Nr = 0, Vb = 0)	IH, Vb = Rb, Vi = 0, RTi, Is_Ct = N2		REMOTE_BUSY
RNR-c^0|RNR-r^0 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vb = LRb, Vi = 0, RTi, Is_Ct = N2		LOCAL_BUSY + REMOTE_BUSY
RNR-c^1 (Vi = Op, Nr = 0, Vb = 0)	IH, Vb = Rb, Vi = 0, RTi, Is_Ct = N2	[RR-r^1]	REMOTE_BUSY
RNR-c^1 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vb = LRb, Vi = 0, RTi, Is_Ct = N2	[RNR-r^1]	LOCAL_BUSY + REMOTE_BUSY
RR-c^0|RR-r^0 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, RTi, Start Send_Proc		LINK_OPENED
RR-c^0|RR-r^0 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, RTi, Start Send_Proc		LOCAL_BUSY
RR-c^1 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, RTi, Start Send_Proc	[RR-r^1]	LINK_OPENED
RR-c^1 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, RTi, Start Send_Proc	[RNR-r^1]	LOCAL_BUSY
REJ-c^1 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, RTi, Start_Send_Proc	[RR-r^1]	LINK_OPENED
REJ-c^1 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, RTi, Start_Send_Proc	[RNR-r^1]	LOCAL_BUSY
REJ-c^0|REJ-r^0 (Vi = Op, Nr = 0, Vb = 0)	IH, Vi = 0, RTi, Start_Send_Proc		LINK_OPENED
REJ-c^0|REJ-r^0 (Vi = Op, Nr = 0, Vb = Lb)	IH, Vi = 0, RTi, Start_Send_Proc		LOCAL_BUSY

DISCONNECTING (04)

State: DISCONNECTING (04) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb		
EXIT_LCL_Busy	Vb = 0		
LPDU_INVALID	Ignore_LPDU		
SEND_BTU|SEND_XID SET_ABME|SET_ADM TEST_LINK Ti_Expired	Logical Error (Local)		
T1_Expired (P_Ct = 0)	IH, ITi		DISCONNECTED
T1_Expired (P_Ct≠0)	IT1, Decrement P_Ct	[DISC^1]	
T2_Expired	Logical Error (Local)		

State: DISCONNECTING (04) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1		[UA^0|UA^1]	
DM^0|DM^1	IH, TT1, ITi		DISCONNECTED
FRMR	Ignore_LPDU		
SABME^0|SABME^1	IH, TT1, ITi	[DM^0|DM^1]	DISCONNECTED
TEST-c|TEST-r UA^0	Ignore_LPDU		
UA^1	IH, TT1, ITi		DISCONNECTED
XID-c|XID-r	Ignore_LPDU		

State: DISCONNECTING (04) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
I-c|I-r	Ignore_LPDU		

State: DISCONNECTING (04) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ|RNR|RR	Ignore_LPDU		

FRMR_SENT (05)

State: FRMR_SENT (05) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb		
EXIT_LCL_Busy	Vb = 0		
LPDU_INVALID	Ignore_LPDU		
SEND_BTU|SEND_XID	Logical Error (Local)		
SET_ABME	TTi, IT1, P_Ct = N2, Vi = LIp	[SABME^1]	LINK_OPENING
SET_ADM	TTi, IT1, P_Ct = N2	[DISC^1]	DISCONNECTING
TEST_LINK	Logical Error (Local)		
Ti_Expired	IH, ITi		
T1_Expired|T2_Expired	Logical Error (Local)		

State: FRMR_SENT (05) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, RTi	[UA^0|UA^1]	DISCONNECTED
DM	IH, RTi		DISCONNECTED
FRMR	IH, P_Ct = N2, RTi		FRMR_RECEIVED
SABME	IH, Vi = RIp, Pf = P, RTi		RESETTING
TEST UA XID	Ignore_LPDU		

State: FRMR_SENT (05) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
I-c^0|I-c^1	IH, RTi	[FRMR^0|FRMR^1]	
I-r^0|I-r^1	Ignore_LPDU		

State: FRMR_SENT (05) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-c^1 RNR-c^0|RNR-c^1 RR-c^0|RR-c^1	IH, RTi	[FRMR^0|FRMR^1]	
REJ-r^0|REJ-r^1 RNR-r^0|RNR-r^1 RR-r^0|RR-r^1	Ignore_LPDU		

LINK_OPENED (06)

State: LINK_OPENED (06) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb, TT2, Ir_Ct = N3	[RNR-r^0]	LOCAL_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, TT2, RTi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, TT2, RTi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_I_POLL	TTi, RT1, P_Ct = N2, Vp = Vs, TT2, Ir_Ct = N3	[I-c^1]	CHECKPOINTING
SEND_XID (Xs = 0)	Xs = IXp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[XID-c^1]	CHECKPOINTING
SEND_XID (Xs = IXp|IOXp)	Logical Error (Local)		
SEND_XID (Xs = OXp)	IH, Xs = 0	[XID-r(F = Px)]	
SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct = N2, TT2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts = 0)	Ts = ITp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[TEST-c^1]	CHECKPOINTING
TEST_LINK (Ts = ITp|IOTp)	Logical Error (Local)		
TEST_LINK (Ts = OTp)	IH, Ts = 0	[TEST-r(F = Pt)]	
Ti_Expired	IT1, P_Ct = N2, TT2, Ir_Ct = N3, Stop Send_Proc, Vp = Vs	[RR-c^1]	CHECKPOINTING
T1_Expired Is_Ct≠0	IT1, P_Ct = N2, TT2, Ir_Ct = N3, Stop Send_Proc, Vp = Vs	[RR-c^1]	CHECKPOINTING
T1_Expired Is_Ct=0	IH,TT2, P_Ct=N2, IT1	[DISC^1]	DISCONNECTING
T2_Expired	Ir_Ct = N3	[RR-r^0]	

State: LINK_OPENED (06) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, RTi, TT2	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, RTi, TT2		DISCONNECTED
FRMR	IH, TT1, RTi, P_Ct=N2, TT2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, RTi, TT2		RESETTING
TEST-c^0|TEST-c^1	IH, Ts=OTp, Pt=P		
TEST-r^0|TEST-r^1	Ignore_LPDU		
UA	IH_(RE), TT1, RTi, TT2	[FRMR^0]	FRMR_SENT
XID-c^0|XID-c^1	IH, Xs=OXp, Px=P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: LINK_OPENED (06) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0|IS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU	[Send_ACK]	
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU, TT2, Ir_Ct=N3	[RR-r^1]	
OS_I-c^0|OS_I-r^0|OS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld, TT2, Ir_Ct=N3	[REJ-r^0]	REJECTION
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld, TT2, Ir_Ct=N3	[REJ-r^1]	REJECTION

State: LINK_OPENED (06) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr=Vs)	Update_Va		
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr<Vs)	Update_Va, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0		
REJ-c^1 (Va≤Nr=Vs)	Update_Va, TT2	[RR-r^1]	
REJ-c^1 (Va≤Nr<Vs)	Update_Va, TT2, Ir_Ct=N3, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0	[RR-r^1]	
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va, Vb=Rb, Is_Ct=N2, Stop Send_Proc		REMOTE_BUSY
RNR-c^1 (Va≤Nr≤Vs)	Update_Va, Vb=Rb, TT2, Ir_Ct=N3, Is_Ct=N2 Stop Send_Proc	[RR-r^1]	REMOTE_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr≤Vs)	Update_Va		
RR-c^1 (Va≤Nr≤Vs)	Update_Va, TT2, Ir_Ct=N3	[RR-r^1]	

LOCAL_BUSY (07)

State: LOCAL_BUSY (07) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS ENTER_LCL_Busy	Logical Error (Local)		
EXIT_LCL_Busy	Vb = 0, TTi, RT1, P_Ct = N2, Vp = Vs, Stop Send_Proc	[RR-c^1]	CHECKPOINTING
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Xs = 0)	Xs = IXp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[XID-c^1]	CHECKPOINTING + LOCAL_BUSY
SEND_XID (Xs = IXp|IOXp)	Logical Error (Local)		
SEND_XID (Xs = OXp)	IH, Xs = 0	[XID-r(F = Px)]	CHECKPOINTING + LOCAL_BUSY
SEND_I_POLL	TTi, RT1, P_Ct = N2, Vp = Vs	[I-c^1]	CHECKPOINTING + LOCAL_BUSY
SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts = 0)	Ts = ITp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[TEST-c^1]	CHECKPOINTING + LOCAL_BUSY
TEST_LINK (Ts = ITp|IOTp)	Logical Error (Local)		
TEST_LINK (Ts = OTp)	IH, Ts = 0	[TEST-r(F = Pf)]	CHECKPOINTING + LOCAL_BUSY
Ti_Expired	IT1, P_Ct = N2, Stop Send_Proc, Vp = Vs	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
T1_Expired Is_Ct≠0	IT1, P_Ct = N2, Stop Send_Proc, Vp = Vs	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
T1_Expired Is_Ct=0	TT2, P_Ct=N2, IT1	[DISC^1]	DISCONNECTING
T2_Expired	Logical Error (Local)		

State: LOCAL_BUSY (07) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, RTi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, RTi		DISCONNECTED
FRMR	IH, TT1, RTi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, RTi		RESETTING
TEST-c	IH, Ts=OTp, Pt=P		
TEST-r	Ignore_LPDU		
UA	IH_(RE), TT1, RTi, VWXYZ=01000	[FRMR^0]	FRMR_SENT
XID-c	IH, Xs=OXp, Px=P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: LOCAL_BUSY (07) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0|IS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^0]	
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	
OS_I-c^0|OS_I-r^0|OS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^0]	
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	

State: LOCAL_BUSY (07) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr=Vs)	Update_Va		
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr<Vs)	Update_Va, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0		
REJ-c^1 (Va≤Nr=Vs)	Update_Va, TT2	[RR-r^1]	
REJ-c^1 (Va≤Nr<Vs)	Update_Va, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0	[RNR-r^1]	
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va, Vb=LRb, Is_Ct=N2, Stop Send_Proc		LOCAL_BUSY + REMOTE_BUSY
RNR-c^1 (Va≤Nr≤Vs)	Update_Va, Vb=LRb, Is_Ct=N2, Stop Send_Proc	[RNR-r^1]	LOCAL_BUSY + REMOTE_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr≤Vs)	Update_Va		
RR-c^1 (Va≤Nr≤Vs)	Update_Va	[RNR-r^1]	

REJECTION (08)

State: REJECTION (08) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb	[RNR-r^0]	REJECTION + LOCAL_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_I_POLL	TTi, RT1, P_Ct = N2, Vp = Vs	[I-c^1]	CHECKPOINTING + REJECTION
SEND_XID (Xs = 0)	Xs = IXp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[XID-c^1]	CHECKPOINTING + REJECTION
SEND_XID (Xs = IXp|IOXp)	Logical Error (Local)		
SEND_XID (Xs = OXp)	Xs = 0	[XID-r(F = Px)]	CHECKPOINTING + REJECTION
SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts = 0)	Ts = ITp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[TEST-c^1]	CHECKPOINTING + REJECTION
TEST_LINK (Ts = ITp|IOTp)	Logical Error (Local)		
TEST_LINK (Ts = OTp)	Ts = 0	[TEST-r(F = Pt)]	CHECKPOINTING + REJECTION
Ti_Expired	IT1, P_Ct = N2, Stop Send_Proc, Vp = Vs	[RR-c^1]	CHECKPOINTING + REJECTION
T1_Expired Is_Ct≠0	IT1, P_Ct = N2, Stop Send_Proc, Vp = Vs	[RR-c^1]	CHECKPOINTING + REJECTION
T1_Expired Is_Ct=0	P_Ct = N2, IT1	[DISC^1]	DISCONNECTING
T2_Expired	Logical Error (Local)		

State: REJECTION (08) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, RTi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, RTi		DISCONNECTED
FRMR	IH, TT1, RTi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, RTi		RESETTING
TEST-c	IH, Ts=OTp, Pt=P		
TEST-r	Ignore_LPDU		
UA	IH_(RE), TT1, RTi	[FRMR^0]	FRMR_SENT
XID-c	IH, Xs=OXp, Px=P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: REJECTION (08) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0|IS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU	[Send_ACK]	LINK_OPENED
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU	[RR-r^1]	LINK_OPENED
OS_I-c^0|OS_I-r^0|OS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld		
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RR-r^1]	

State: REJECTION (08) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr=Vs)	Update_Va		
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr<Vs)	Update_Va, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0		
REJ-c^1 (Va≤Nr=Vs)	Update_Va, TT2	[RR-r^1]	
REJ-c^1 (Va≤Nr<Vs)	Update_Va, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0	[RR-r^1]	
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va, Vb=Rb, Is_Ct=N2, Stop Send_Proc		REJECTION + REMOTE_BUSY
RNR-c^1 (Va≤Nr≤Vs)	Update_Va, Vb=Rb, Is_Ct=N2, Stop Send_Proc	[RR-r^1]	REJECTION + REMOTE_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr≤Vs)	Update_Va		
RR-c^1 (Va≤Nr≤Vs)	Update_Va	[RR-r^1]	

CHECKPOINTING (09)

State: CHECKPOINTING (09) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb, TT2, Ir_Ct = N3	[RNR-r^0]	CHECKPOINTING + LOCAL_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, TT2, ITi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, TT2, ITi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Vc = 0, Xs = 0)	Vc = XIDp		
SEND_XID (Vc = 0, Xs = IXp)	Logical Error (Local)		
SEND_XID (Vc = 0, Xs = OXp)	IH, Xs = 0	[XID-r(F = Px)]	
SEND_XID (Vc = 0, Xs = IOXp)	IH, Xs = IXp	[XID-r(F = Px)]	
SEND_XID (Vc≠0) SET_ABME	Logical Error (Local)		
SET_ADM (Vc = 0)	Vc = DISCp		
SET_ADM (Vc≠0)	Logical Error (Local)		
TEST_LINK (Vc = 0, Ts = 0)	Vc = TESTp		
TEST_LINK (Vc = 0, Ts = ITp)	Logical Error (Local)		
TEST_LINK (Vc = 0, Ts = IOTp)	IH, Ts = ITp	[TEST-r(F = Pt)]	
TEST_LINK (Vc = 0, Ts = OTp)	IH, Ts = 0	[TEST-r(F = Pt)]	
TEST_LINK (Vc≠0) Ti_Expired	Logical Error (Local)		
T1_Expired (P_Ct = 0)	IH, TT2, ITi		DISCONNECTED
T1_Expired (P_Ct≠0, Ts = Xs = 0)	IT1, Decrement P_Ct, Vp = Vs, TT2, Ir_Ct = N3, Stop Send_Proc	[RR-c^1]	
T1_Expired (P_Ct≠0, Ts = ITp|IOTp)	IT1, Decrement P_Ct	[TEST-c^1]	
T1_Expired (P_Ct≠0, Xs = IXp|IOXp)	IT1, Decrement P_Ct	[XID-c^1]	
T2_Expired	Ir_Ct = N3	[RR-r^0]	

State: CHECKPOINTING (09) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
$DISC^0$|$DISC^1$	IH, TT1, ITi, TT2	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, ITi, TT2		DISCONNECTED
FRMR	IH, TT1, ITi, P_Ct=N2, TT2, Xs=Ts=Vc=0		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, ITi, TT2		RESETTING
TEST-c (Ts=0|OTp)	IH, Ts=OTp, Pt=P		
TEST-c (Ts=ITp|IOTp)	IH, Ts=IOTp, Pt=P		
$TEST\text{-}r^0$ $TEST\text{-}r^1$ (Vc=0, Ts=0|OTp)	Ignore_LPDU		
$TEST\text{-}r^1$ (Vc=0, Ts=ITp)	IH, Ts=0, TT1, ITi, Start Send_Proc		LINK_OPENED
$TEST\text{-}r^1$ (Vc=0, Ts=IOTp)	IH, Ts=OTp, TT1, ITi, Start Send_Proc		LINK_OPENED
$TEST\text{-}r^1$ (Vc=DISCp)	Vc=0, RT1, P_Ct=N2, TT2	[$DISC^1$]	DISCONNECTING
$TEST\text{-}r^1$ (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[$TEST\text{-}c^1$]	
$TEST\text{-}r^1$ (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[$XID\text{-}c^1$]	
UA (Vi≠ISp)	IH_(RE), TT1, ITi, TT2, Xs=Ts=Vc=0	[$FRMR^0$]	FRMR_SENT
UA (Vi=ISp)	Ign_LPDU		
XID-c (Xs=0|OXp)	IH, Xs=OXp, Px=P		
XID-c (Xs=IXp|IOXp)	IH, Xs=IOXp, Px=P		
$XID\text{-}r^0$	IH		
$XID\text{-}r^1$ (Vc=0, Xs=0|OXp)	Ignore_LPDU		
$XID\text{-}r^1$ (Vc=0, Xs=IXp)	IH, Xs=0, TT1, ITi, Start Send_Proc		LINK_OPENED
$XID\text{-}r^1$ (Vc=0, Xs=IOXp)	IH, Xs=OXp, TT1, ITi, Start Send_Proc		LINK_OPENED
$XID\text{-}r^1$ (Vc=DISCp)	Vc=0, RT1, P_Ct=N2, TT2	[$DISC^1$]	DISCONNECTING
$XID\text{-}r^1$ (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[$TEST\text{-}c^1$]	
$XID\text{-}r^1$ (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[$XID\text{-}c^1$]	

State: CHECKPOINTING (09) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Va=Nr≤Vs)	Rcv_BTU	[Send_ACK]	
IS_I-c^0|IS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Rcv_BTU, Is_Ct=N2	[Send_ACK]	
IS_I-c^1 (Va=Nr≤Vs)	Rcv_BTU, TT2, Ir_Ct=N3	[RR-r^1]	
IS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Rcv_BTU, Is_Ct=N2, TT2, Ir_Ct=N3	[RR-r^1]	
IS_I-r^1 (Vc=0)	Update_Va_Chkpt, Rcv_BTU, Start Send_Proc	[Send_ACK]	LINK_OPENED
IS_I-r^1 (Vc=DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, TTi, RT1, P_Ct=N2, TT2	[DISC^1]	DISCONNECTING
IS_I-r^1 (Vc=TESTp)	Update_Va_Chkpt, Rcv_BTU, Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
IS_I-r^1 (Vc=XIDp)	Update_Va_Chkpt, Rcv_BTU, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	
OS_I-c^0|OS_I-r^0 (Va=Nr≤Vs)	Dsc_I-fld, TT2, Ir_Ct=N3	[REJ-r^0]	CHECKPOINTING + REJECTION
OS_I-c^0|OS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I-fld, Is_Ct=N2, TT2, Ir_Ct=N3	[REJ-r^0]	CHECKPOINTING + REJECTION
OS_I-c^1 (Va=Nr≤Vs)	Dsc_I-fld, TT2, Ir_Ct=N3	[REJ-r^1]	CHECKPOINTING + REJECTION
OS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I-fld, Is_Ct=N2, TT2, Ir_Ct=N3	[REJ-r^1]	CHECKPOINTING + REJECTION
OS_I-r^1 (Vc=0)	Update_Va_Chkpt, Dsc_I-fld, TT2, Ir_Ct=N3, Start Send_Proc	[REJ-r^0]	REJECTION
OS_I-r^1 (Vc=DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, RT1, P_Ct=N2, TT2	[DISC^1]	DISCONNECTING
OS_I-r^1 (Vc=TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Tx=ITp, RT1, P_Ct=N2	[TEST-c^1]	
OS_I-r^1 (Vc=XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING (09) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0		
REJ-c^1 (Va=Nr≤Vs)	TT2	[RR-r^1]	
REJ-c^1 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, TT2, Ir_Ct=N3, Ww=1, Ia_Ct=0	[RR-r^1]	
REJ-r^1 (Vc=0)	Update_Va_Chkpt, Vs=Nr, Start_Send_Proc		LINK_OPENED
RNR-c^0|RNR-r^0 (Va=Nr≤Vs)	Vb=Rb, Is_Ct=N2		
RNR-c^0|RNR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Vb=Rb, Is_Ct=N2		
RNR-c^1 (Va=Nr≤Vs)	Vb=Rb, TT2, Ir_Ct=N3, Is_Ct=N2	[RR-r^1]	
RNR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Vb=Rb, TT2, Is_Ct=N2, Ir_Ct=N3	[RR-r^1]	
RNR-r^1 (Vc=0)	Update_Va_Chkpt, Is_Ct=N2		REMOTE_BUSY
RR-c^0|RR-r^0 (Va=Nr≤Vs)			
RR-c^0|RR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2		
RR-c^1 (Va=Nr≤Vs)	TT2, Ir_Ct=N3	[RR-r^1]	
RR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, TT2, Ir_Ct=N3	[RR-r^1]	
RR-r^1 (Vc=0)	Update_Va_Chkpt, Start Send_Proc		LINK_OPENED
REJ-r^1|RNR-r^1|RR-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
REJ-r^1|RNR-r^1|RR-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
REJ-r^1|RNR-r^1|RR-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

CHECKPOINTING + LOCAL_BUSY (10)

State: CHECKPOINTING + LOCAL_BUSY (10) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS ENTER_LCL_Busy	Logical Error (Local)		
EXIT_LCL_Busy	Vb=0	[RR-r^0]	CHECKPOINTING + CLEARING
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ=00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ=00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Vc=0, Xs=0)	Vc=XIDp		
SEND_XID (Vc=0, Xs=IXp)	Logical Error (Local)		
SEND_XID (Vc=0, Xs=OXp)	IH, Xs=0	[XID-r(F=Px)]	
SEND_XID (Vc=0, Xs=IOXp)	IH, Xs=IXp	[XID-r(F=Px)]	
SEND_XID (Vc≠0) SET_ABME	Logical Error (Local)		
SET_ADM (Vc=0)	Vc=DISCp		
SET_ADM (Vc≠0)	Logical Error (Local)		
TEST_LINK (Vc=0, Ts=0)	Vc=TESTp		
TEST_LINK (Vc=0, Ts=ITp)	Logical Error (Local)		
TEST_LINK (Vc=0, Ts=OTp)	IH, Ts=0	[TEST-r(F=Pt)]	
TEST_LINK (Vc=0, Ts=IOTp)	IH, Ts=ITp	[TEST-r(F=Pt)]	
TEST_LINK (Vc≠0) Ti_Expired	Logical Error (Local)		
T1_Expired (P_Ct=0)	IH, ITi		DISCONNECTED
T1_Expired (P_Ct≠0, Ts=Xs=0)	IT1, Decrement P_Ct, Vp=Vs, Stop Send_Proc	[RNR-c^1]	
T1_Expired (P_Ct≠0, Ts=ITp|IOTp)	IT1, Decrement P_Ct	[TEST-c^1]	
T1_Expired (P_Ct≠0, Xs=IXp|IOXp)	IT1, Decrement P_Ct	[XID-c^1]	
T2_Expired	Logical Error (Local)		

State: CHECKPOINTING + LOCAL_BUSY (10) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, ITi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, ITi		DISCONNECTED
FRMR	IH, TT1, ITi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, ITi		RESETTING
TEST-c (Ts=0|OTp)	IH, Ts=OTp, Pt=P		
TEST-c (Ts=ITp|IOTp)	IH, Ts=IOTp, Pt=P		
TEST-r^0 TEST-r^1 (Vc=0, Ts=0|OTp)	Ignore_LPDU		
TEST-r^1 (Vc=0, Ts=ITp)	IH, Ts=0, TT1, ITi, Start Send_Proc		LOCAL_BUSY
TEST-r^1 (Vc=0, Ts=IOTp)	IH, Ts=OTp, TT1, ITi, Start Send_Proc		LOCAL_BUSY
TEST-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2,	[DISC^1]	DISCONNECTING
TEST-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
TEST-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	
UA (Vi≠ISp)	IH_(RE), TT1, ITi	[FRMR^0]	FRMR_SENT
UA (Vi=ISp)	Ign_LPDU		
XID-c (Xs=0|OXp)	IH, Xs=OXp, Px=P		
XID-c (Xs=IXp|IOXp)	IH, Xs=IOXp, Px=P		
XID-r^0	IH		
XID-r^1 (Vc=0, Xs=0|OXp)	Ignore_LPDU		
XID-r^1 (Vc=0, Xs=IXp)	IH, Xs=0, TT1, ITi, Start Send_Proc		LOCAL_BUSY
XID-r^1 (Vc=0, Xs=IOXp)	IH, Xs=OXp, TT1, ITi, Start Send_Proc		LOCAL_BUSY
XID-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2,	[DISC^1]	DISCONNECTING
XID-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
XID-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING + LOCAL_BUSY (10) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Va=Nr≤Vs)	Dsc_I-fld	[RNR-r^0]	
IS_I-c^0|IS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I=fld. Is_Ct=N2	[RNR-r^0]	
IS_I-c^1 (Va=Nr≤Vs)	Dsc_I-fld	[RNR-r^1]	
IS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I-fld, Is_Ct=N2	[RNR-r^1]	
IS_I-r^1 (Vc=0)	Update_Va_Chkpt, Dsc_I-fld, Start Send_Proc	[RNR-r^0]	LOCAL_BUSY
IS_I-r^1 (Vc=DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
IS_I-r^1 (Vc=TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Ts=ITp, TTi, RT1, P_Ct=N2	[TEST-c^1]	
IS_I-r^1 (Vc=XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Xs=IXp, TTi, RT1, P_Ct=N2	[XID-c^1]	
OS_I-c^0|OS_I-r^0 (Va=Nr≤Vs)	Dsc_I-fld		
OS_I-c^0|OS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I-fld, Is_Ct=N2		
OS_I-c^1 (Va=Nr≤Vs)	Dsc_I-fld	[RNR-r^1]	
OS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I-fld, Is_Ct=N2	[RNR-r^1]	
OS_I-r^1 (Vc=0)	Update_Va_Chkpt, Dsc_I-fld, Start Send_Proc		LOCAL_BUSY
OS_I-r^1 (Vc=DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
OS_I-r^1 (Vc=TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Tx=ITp, TTi, RT1, P_Ct=N2	[TEST-c^1]	
OS_I-r^1 (Vc=XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Xs=IXp, TTi, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING + LOCAL_BUSY (10) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0		
REJ-c^1 (Va=Nr≤Vs)		[RNR-r^1]	
REJ-c^1 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0	[RNR-r^1]	
REJ-r^1 (Vc=0)	Update_Va_Chkpt, Vs=Nr, Start Send_Proc		LOCAL_BUSY
RNR-c^0|RNR-r^0 (Va=Nr≤Vs)	Vb=LRb, Is_Ct=N2		
RNR-c^0|RNR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=LRb		
RNR-c^1 (Va=Nr≤Vs)	Vb=LRb, Is_Ct=N2	[RNR-r^1]	
RNR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=LRb	[RNR-r^1]	
RNR-r^1 (Vc=0)	Update_Va_Chkpt, Vb=LRb, Is_Ct=N2		LOCAL_BUSY + REMOTE_BUSY
RR-c^0|RR-r^0 (Va=Nr≤Vs)			
RR-c^0|RR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2		
RR-c^1 (Va=Nr≤Vs)		[RNR-r^1]	
RR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2	[RNR-r^1]	
RR-r^1 (Vc=0)	Update_Va_Chkpt, Start Send_Proc		LOCAL_BUSY
REJ-r^1|RNR-r^1|RR-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vc=0, TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
REJ-r^1|RNR-r^1|RR-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vc=0, Ts=ITp, TTi, RT1, P_Ct=N2	[TEST-c^1]	
REJ-r^1|RNR-r^1|RR-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vc=0, Xs=IXp, TTi, RT1, P_Ct=N2	[XID-c^1]	

CHECKPOINTING + REJECTION (11)

State: CHECKPOINTING + REJECTION (11) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb	[RNR-r^0]	CHECKPOINTING + REJECTION + LOCAL_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Vc = 0, Xs = 0)	Vc = XIDp		
SEND_XID (Vc = 0, Xs = IXp)	Logical Error (Local)		
SEND_XID (Vc = 0, Xs = OXp)	IH, Xs = 0	[XID-r(F = Px)]	
SEND_XID (Vc = 0, Xs = IOXp)	IH, Xs = IXp	[XID-r(F = Px)]	
SEND_XID (Vc≠0) SET_ABME	Logical Error (Local)		
SET_ADM (Vc = 0)	Vc = DISCp		
SET_ADM (Vc≠0)	Logical Error (Local)		
TEST_LINK (Vc = 0, Ts = 0)	Vc = TESTp		
TEST_LINK (Vc = 0, Ts = ITp)	Logical Error (Local)		
TEST_LINK (Vc = 0, Ts = OTp)	IH, Ts = 0	[TEST-r(F = Pt)]	
TEST_LINK (Vc = 0, Ts = IOTp)	IH, Ts = ITp	[TEST-r(F = Pt)]	
TEST_LINK (Vc≠0) Ti_Expired	Logical Error (Local)		
T1_Expired (P_Ct = 0)	IH, ITi		DISCONNECTED
T1_Expired (P_Ct≠0, Ts = Xs = 0)	IT1, Decrement P_Ct, Vp = Vs, Stop Send_Proc	[RR-c^1]	
T1_Expired (P_Ct≠0, Ts = ITp|IOTp)	IT1, Decrement P_Ct	[TEST-c^1]	
T1_Expired (P_Ct≠0, Xs = IXp|IOXp)	IT1, Decrement P_Ct	[XID-c^1]	
T2_Expired	Logical Error (Local)		

State: CHECKPOINTING + REJECTION (11) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, ITi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, ITi		DISCONNECTED
FRMR	IH, TT1, ITi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, ITi		RESETTING
TEST-c (Ts=0|OTp)	IH, Ts=OTp, Pt=P		
TEST-c (Ts=ITp|IOTp)	IH, Ts=IOTp, Pt=P		
TEST-r^0 TEST-r^1 (Vc=0, Ts=0|OTp)	Ignore_LPDU		
TEST-r^1 (Vc=0, Ts=ITp)	IH, Ts=0, TT1, ITi, Start Send_Proc		REJECTION
TEST-r^1 (Vc=0, Ts=IOTp)	IH, Ts=OTp, TT1, ITi, Start Send_Proc		REJECTION
TEST-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
TEST-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
TEST-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	
UA	IH_(RE), TT1, ITi	[FRMR^0]	FRMR_SENT
XID-c (Xs=0|OXp)	IH, Xs=OXp, Px=P		
XID-c (Xs=IXp|IOXp)	IH, Xs=IOXp, Px=P		
XID-r^0	IH		
XID-r^1 (Vc=0, Xs=0|OXp)	Ignore_LPDU		
XID-r^1 (Vc=0, Xs=IXp)	IH, Xs=0, TT1, ITi, Start Send_Proc		REJECTION
XID-r^1 (Vc=0, Xs=IOXp)	IH, Xs=OXp, TT1, ITi, Start Send_Proc		REJECTION
XID-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
XID-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
XID-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING + REJECTION (11) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Va=Nr≤Vs)	Rcv_BTU	[Send_ACK]	CHECKPOINTING
IS_I-c^0|IS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Rcv_BTU, Is_Ct=N2	[Send_ACK]	CHECKPOINTING
IS_I-c^1 (Va=Nr≤Vs)	Rcv_BTU	[RR-r^1]	CHECKPOINTING
IS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Rcv_BTU, Is_Ct=N2	[RR-r^1]	CHECKPOINTING
IS_I-r^1 (Vc=0)	Update_Va_Chkpt, Rcv_BTU, Start Send_Proc	[Send_ACK]	LINK_OPENED
IS_I-r^1 (Vc=DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
IS_I-r^1 (Vc=TESTp)	Update_Va_Chkpt, Rcv_BTU, Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	CHECKPOINTING
IS_I-r^1 (Vc=XIDp)	Update_Va_Chkpt, Rcv_BTU, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	CHECKPOINTING
OS_I-c^0|OS_I-r^0 (Va=Nr≤Vs)	Dsc_I-fld		
OS_I-c^0|OS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I-fld, Is_Ct=N2		
OS_I-c^1 (Va=Nr≤Vs)	Dsc_I-fld	[RR-r^1]	
OS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Dsc_I-fld, Is_Ct=N2	[RR-r^1]	
OS_I-r^1 (Vc=0)	Update_Va_Chkpt, Dsc_I-fld, Start Send_Proc		REJECTION
OS_I-r^1 (Vc=DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
OS_I-r^1 (Vc=TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Tx=ITp, RT1, P_Ct=N2	[TEST-c^1]	
OS_I-r^1 (Vc=XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING + REJECTION (11) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0		
REJ-c^1 (Va=Nr≤Vs)		[RR-r^1]	
REJ-c^1 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0	[RR-r^1]	
REJ-r^1 (Vc=0)	Update_Va_Chkpt, Vs=Nr, Start_Send_Proc		REJECTION
RNR-c^0|RNR-r^0 (Va=Nr≤Vs)	Vb=Rb, Is_Ct=N2		
RNR-c^0|RNR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=Rb		
RNR-c^1 (Va=Nr≤Vs)	Vb=Rb, Is_Ct=N2	[RR-r^1]	
RNR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=Rb	[RR-r^1]	
RNR-r^1 (Vc=0)	Is_Ct=N2, Update_Va_Chkpt		REJECTION + REMOTE_BUSY
RR-c^0|RR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2		
RR-c^1 (Va=Nr≤Vs)		[RR-r^1]	
RR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2	[RR-r^1]	
RR-r^1 (Vc=0)	Update_Va_Chkpt, Start Send_Proc		REJECTION
REJ-r^1|RNR-r^1|RR-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
REJ-r^1|RNR-r^1|RR-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
REJ-r^1|RNR-r^1|RR-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

RESETTING (12)

State: RESETTING (12) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb		
EXIT_LCL_Busy	Vb = 0		
LPDU_INVALID	Ignore_LPDU		
SEND_BTU|SEND_XID	Logical Error (Local)		
SET_ABME	Vi = Op, RTi, Va = Vs = Vr = Vp = 0, Is_Ct = N2, Ir_Ct = N3	[UA(F = P_f)]	LINK_OPENING
SET_ADM	Vi = 0, RTi	[DM(F = P_f)]	DISCONNECTED
TEST_LINK	Logical Error (Local)		
Ti_Expired	IH, ITi		
T1_Expired|T2_Expired	Logical Error (Local)		

State: RESETTING (12) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
$DISC^0$|$DISC^1$	IH, Vi = 0, RTi	[UA^0|UA^1]	DISCONNECTED
DM	IH, Vi = 0, RTi		DISCONNECTED
FRMR	IH, Vi = 0, RTi, P_Ct = N2		FRMR_RECEIVED
SABME	IH, RTi		
TEST-c|TEST-r UA XID-c|XID-r	Ignore_LPDU		

State: RESETTING (12) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I|OS_I	Ignore_LPDU		

State: RESETTING (12) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ|RNR|RR	Ignore_LPDU		

REMOTE_BUSY (13)

State: REMOTE_BUSY (13) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = LRb, TT2, Ir_Ct = N3	[RNR-r^0]	LOCAL_BUSY + REMOTE_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, TT2, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, TT2, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Xs = 0)	Xs = IXp, TTi, RT1, P_Ct = N2	[XID-c^1]	CHECKPOINTING
SEND_XID (Xs = IXp|IOXp)	Logical Error (Local)		
SEND_XID (Xs = OXp)	Xs = 0	[XID-r(F = Px)]	CHECKPOINTING
SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct = N2, TT2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts = 0)	Ts = ITp, TTi, RT1, P_Ct = N2	[TEST-c^1]	CHECKPOINTING
TEST_LINK (TS = ITp|IOTp)	Logical Error (Local)		
TEST_LINK (Ts = OTp)	Ts = 0	[TEST-r(F = Pt)]	CHECKPOINTING
Ti_Expired	IT1,P_Ct = N2, Vp = Vs, TT2, Ir_Ct = N3	[RR-c^1]	CHECKPOINTING
T1_Expired Is_Ct≠0	IT1,P_Ct = N2, Vp = Vs, TT2, Ir_Ct = N3	[RR-c^1]	CHECKPOINTING
T1_Expired Is_Ct=0	TT2, P_Ct=N2, IT1	[DISC^1]	DISCONNECTING
T2_Expired	Ir_Ct = N3	[RR-r^0]	

State: REMOTE_BUSY (13) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, RTi, TT2	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, RTi, TT2		DISCONNECTED
FRMR	IH, TT1, RTi, P_Ct = N2, TT2		FRMR_RECEIVED
SABME	IH, Vi = RIp, Pf = P, TT1, RTi, TT2		RESETTING
TEST-c	IH, Ts = OTp, Pt = P		
TEST-r	Ignore_LPDU		
UA	IH_(RE), TT1, RTi, TT2	[FRMR^0]	FRMR_SENT
XID-c	IH, Xs = OXp, Px = P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: REMOTE_BUSY (13) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU	[Send_ACK]	
IS_Ir^1 (Va≤Nr≤Vs)	Vb=0, Update_Va, Rcv_BTU	[Send_ACK.]	LINK_OPENED
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU, TT2, Ir_Ct=N3	[RR-r^1]	
OS_I-c^0|OS_I-r^0 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld, TT2, Ir_Ct=N3	[REJ-r^0]	REJECTION + REMOTE_BUSY
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld, TT2, Ir_Ct=N3	[REJ-r^1]	REJECTION + REMOTE_BUSY
OS_Ir^1 (Va≤Nr≤Vs)	Vb=0, Update_Va, Dsc_I-fld, TT2	[REJ-r&sup.0]	REJECTION

State: REMOTE_BUSY (13) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr<Vs)	Vb=0, Update_Va, Vs=Nr, Ww=1, Ia_Ct=0, Decrement Is_Ct, Start Send_Proc		LINK_OPENED
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr=Vs)	Vb=0, Update_Va, Start Send_Proc		LINK_OPENED
REJ-c^1 (Va≤Nr<Vs)	Vb=0, Update_Va, Vs=Nr, Ww=1, Ia_Ct=0, TT2, Decrement Is_Ct, Ir_Ct=N3, Start Send_Proc	[RR-r^1]	LINK_OPENED
REJ-c^1 (Va≤Nr=Vs)	Vb=0, Update_Va, TT2, Start Send_Proc	[RR-r^1]	LINK_OPENED
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va		
RNR-c^1 (Va≤Nr≤Vs)	Update_Va, TT2, Ir_Ct=N3	[RR-r^1]	
RR-c^1 Va≤Nr<Vs	Vb=0, Update_Va, TT2, Ir_Ct=N3	[RR-c^1] [RR-r^1]	CHECKPOINTING
RR-c^1 Va≤Nr=Vs	Vb=0, Update_Va, TT2, Start Send_Proc, Ir_Ct=N3	[RR-r^1]	LINK_OPENED
RR-c^0|RR-r^0 (Va≤Nr<Vs)	Vb=0, Update_Va	[RR-c^1]	CHECKPOINTING
RR-c^0|RR-r^0 Va≤Nr=Vs	Vb=0, Update_Va, Start Send_Proc		LINK_OPENED
RR-r^1 Va≤Nr<Vs	Vb=0, Update_Va	[RR-c^1]	CHECKPOINTING
RR-r^1 Va≤Nr=Vs	Vb=0, Update_Va, Start Send_Proc		LINK_OPENED

LOCAL_BUSY + REMOTE_BUSY (14)

State: LOCAL_BUSY + REMOTE_BUSY (14) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS ENTER_LCL_Busy	Logical Error (Local)		
EXIT_LCL_Busy	Vb=Rb, TTi, RT1, P_Ct=N2, Vp=Vs	[RR-c^1]	CHECKPOINTING
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ=00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ=00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Xs=0)	Xs=IXp, TTi, RT1, P_Ct=N2	[XID-c^1]	CHECKPOINTING + LOCAL_BUSY
SEND_XID (Xs=IXp|IOXp)	Logical Error (Local)		
SEND_XID (Xs=OXp)	Xs=0	[XID-r(F=Px)]	
SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts=0)	Ts=ITp, TTi, RT1, P_Ct=N2	[TEST-c^1]	CHECKPOINTING + LOCAL_BUSY
TEST_LINK (Ts=ITp|IOTp)	Logical Error (Local)		
TEST_LINK (Ts=OTp)	Ts=0	[TEST-r(F=Pt)]	
Ti_Expired	IT1, P_Ct=N2, Vp=Vs	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
T1_Expired	IT1, P_Ct=N2, Vp=Vs	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
T2_Expired	Logical Error (Local)		

State: LOCAL_BUSY + REMOTE_BUSY (14) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, RTi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, RTi		DISCONNECTED
FRMR	IH, TT1, RTi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=Rlp, Pf=P, TT1, RTi		RESETTING
TEST-c	IH, Ts=OTp, Pt=P		
TEST-r	Ignore_LPDU		
UA	IH_(RE), TT1, RTi	[FRMR^0]	FRMR_SENT
XID-c	IH, Xs=OXp, Px=P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: LOCAL_BUSY + REMOTE_BUSY (14) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0|IS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^0]	
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	
OS_I-c^0|OS_I-r^0|OS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^0]	
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	

State: LOCAL_BUSY + REMOTE_BUSY (14) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr=Vs)	Update_Va		
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr<Vs)	Vb=Lb, Update_Va, Start Send_Proc, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0		LOCAL_BUSY
REJ-c^1 (Va≤Nr=Vs)	Update_Va, TT2	[RR-r^1]	
REJ-c^1 (Va≤Nr<Vs)	Vb=Lb, Update_Va, Start Send_Proc, Vs=Nr, Decrement Is_Ct, Ww=1, Ia_Ct=0	[RNR-r^1]	LOCAL_BUSY
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va		
RNR-c^1 (Va≤Nr≤Vs)	Update_Va	[RNR-r^1]	
RR-c^1 (Va≤Nr<Vs)	Vb=Lb, Update_Va	[RNR-c^1] [RNR-r^1]	CHECKPOINTING + LOCAL_BUSY
RR-c^1 (Va≤Nr=Vs)	Vb=Lb, Update_Va, Start Send_Proc	[RNR-r^1]	LOCAL_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr<Vs)	Vb=Lb, Update_Va	[RNR-c^1]	CHECKPOINTING + LOCAL_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr=Vs)	Vb=Lb, Update_Va, Start Send_Proc		LOCAL_BUSY

REJECTION + LOCAL_BUSY (15)

State: REJECTION + LOCAL_BUSY (15) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS ENTER_LCL_Busy	Logical Error (Local)		
EXIT_LCL_Busy	Vb = 0, TTi, IT1, P_Ct = N2, Vp = Vs, Stop Send_Proc	[RR-c^1]	CHECKPOINTING + REJECTION
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_I_POLL	TTi, RT1, P_Ct = N2, Vp = Vs	[I-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
SEND_XID (Xs = 0)	Xs = IXp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[XID-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
SEND_XID (Xs = IXp|IOXp)	Logical Error (Local)		
SEND_XID (Xs = OXp)	Xs = 0	[XID-r(F = Px)]	
SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts = 0)	Ts = ITp, TTi, RT1, P_Ct = N2, Stop Send_Proc	[TEST-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
TEST_LINK (Ts = ITp|IOTp)	Logical Error (Local)		
TEST_LINK (Ts = OTp)	Ts = 0	[TEST-r(F = Pt)]	
Ti_Expired	IT1, P_Ct = N2, Stop Send_Proc, Vp = Vs	[RNR-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
T1_Expired Is_Ct≠0	IT1, P_Ct = N2, Stop Send_Proc, Vp = Vs	[RNR-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
T1_Expired Is_Ct=0	TT2, P_Ct=N2, IT1	[DISC^1]	DISCONNECTING
T2_Expired	Logical Error (Local)		

State: REJECTION + LOCAL_BUSY (15) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, RTi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, RTi		DISCONNECTED
FRMR	IH, TT1, RTi		FRMR_RECEIVED
SABME	IH, Vi = RIp, Pf = P, TT1, RTi		RESETTING
TEST-c	IH, Ts = OTp, Pt = P		
TEST-r	Ignore_LPDU		
UA	IH_(RE), TT1, RTi	[FRMR^0]	FRMR_SENT
XID-c	IH, Xs = OXp, Px = P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: REJECTION + LOCAL_BUSY (15) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0|IS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^0]	LOCAL_BUSY
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	LOCAL_BUSY
OS_I-c^0|OS_I-r^0|OS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld		
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	

State: REJECTION + LOCAL_BUSY (15) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr≤Vs)	Update_Va, Vs = Nr, Ww = 1, Ia_Ct = 0		
REJ-c^1 (Va≤Nr≤Vs)	Update_Va, Vs = Nr, Ww = 1, Ia_Ct = 0	[RNR-r^1]	
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va, Vb = LRb, Is_Ct = N2, Stop Send_Proc		REJECTION + LOCAL_BUSY + REMOTE_BUSY
RNR-c^1 (Va≤Nr≤Vs)	Update_Va, Vb = LRb, Is_Ct = N2, Stop Send_Proc	[RNR-r^1]	REJECTION + LOCAL_BUSY + REMOTE_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr≤Vs)	Update_Va		
RR-c^1 (Va≤Nr≤Vs)	Update_Va	[RNR-r^1]	

REJECTION + REMOTE_BUSY (16)

State: REJECTION + REMOTE_BUSY (16) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = LRb	[RNR-r^0]	REJECTION + LOCAL_BUSY + REMOTE_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, RTi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Xs = 0)	Xs = IXp, TTi, RT1, P_Ct = N2	[XID-c^1]	CHECKPOINTING + REJECTION
SEND_XID (Xs = IXp|IOXp)	Logical Error (Local)		
SEND_XID (Xs = OXp)	Xs = 0	[XID-r(F = Px)]	
SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts = 0)	Ts = ITp, TTi, RT1, P_Ct = N2	[TEST-c^1]	CHECKPOINTING + REJECTION
TEST_LINK (Ts = ITp|IOTp)	Logical Error (Local)		
TEST_LINK (Ts = OTp)	Ts = 0	[TEST-r(F = Pt)]	
Ti_Expired	IT1, P_Ct = N2, Vp = Vs	[RR-c^1]	CHECKPOINTING + REJECTION
T1_Expired Is_Ct≠0	IT1, P_Ct = N2, Vp = Vs	[RR-c^1]	CHECKPOINTING + REJECTION
T1_Expired Is_Ct=0	TT2, P_Ct=N2, IT1	[DISC^1]	DISCONNECTING
T2_Expired	Logical Error (Local)		

State: REJECTION + REMOTE_BUSY (16)
Input Class: Received U-format LPDUs

Input	Action	[LPDU Transfers]	New State
DISC^0|DISC^1	IH, TT1, RTi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, RTi		DISCONNECTED
FRMR	IH, TT1, RTi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, RTi		RESETTING
TEST-c	IH, Ts=OTp, Pt=P		
TEST-r	Ignore_LPDU		
UA	IH_(RE), TT1, RTi	[FRMR^0]	FRMR_SENT
XID-c	IH, Xs=OXp, Px=P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: REJECTION + REMOTE_BUSY (16)
Input Class: Received I-format LPDUs

Input	Action	[LPDU Transfers]	New State
IS_I-c^0|IS_I-r^0|IS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU	[Send_ACK]	REMOTE_BUSY
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Rcv_BTU	[RR-r^1]	REMOTE_BUSY
OS_I-c^0|OS_I-r^0|OS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld		
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RR-r^1]	

State: REJECTION + REMOTE_BUSY (16)
Input Class: Received S-format LPDUs

Input	Action	[LPDU Transfers]	New State
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr=Vs)	Update_Va		
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr<Vs)	Vb=0, Update_Va, Vs=Nr, Ia_Ct=0, Ww=1, Decrement Is_Ct, Start Send_Proc,		REJECTION
REJ-c^1 (Va≤Nr=Vs)	Update_Va, TT2	[RR-r^1]	
REJ-c^1 (Va≤Nr<Vs)	Vb=0, Update_Va, Vs=Nr, Ia_Ct=0, Ww=1, Decrement Is_Ct, Start Send_Proc,	[RR-r^1]	REJECTION
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va		
RNR-c^1 (Va≤Nr≤Vs)	Update_Va	[RR-r^1]	
RR-c^1 (Va≤Nr<Vs)	Vb=0, Update_Va	[RR-c^1] [RR-r^1]	CHECKPOINTING + REJECTION
RR-c^1 (Va≤Nr=Vs)	Vb=0, Update_Va, Start Send_Proc	[RR-r^1]	REJECTION
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr<Vs)	Vb=0, Update_Va	[RR-c^1]	CHECKPOINTING + REJECTION
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr=Vs)	Vb=0, Update_Va, Start Send_Proc		REJECTION

CHECKPOINTING + REJECTION + LOCAL_BUSY (17)

State: CHECKPOINTING + REJECTION + LOCAL_BUSY (17) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS ENTER_LCL_Busy	Logical Error (Local)		
EXIT_LCL_Busy (Vb=Lb)	Vb=0	[RR-r^0]	CHECKPOINTING + REJECTION + CLEARING
EXIT_LCL_Busy (Vb=LRb)	Vb=Rb	[RR-r^0]	CHECKPOINTING + REJECTION + CLEARING
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ=00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ=00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Vc=0, Xs=0)	Vc=XIDp		
SEND_XID (Vc=0, Xs=IXp)	Logical Error (Local)		
SEND_XID (Vc=0, Xs=OXp)	IH, Xs=0	[XID-r(F=Px)]	
SEND_XID (Vc=0, Xs=IOXp)	IH, Xs=IXp	[XID-r(F=Px)]	
SEND_XID (Vc≠0) SET_ABME	Logical Error (Local)		
SET_ADM (Vc=0)	Vc=DISCp		
SET_ADM (Vc≠0)	Logical Error (Local)		
TEST_LINK (Vc=0, Ts=0)	Vc=TESTp		
TEST_LINK (Vc=0, Ts=ITp)	Logical Error (Local)		
TEST_LINK (Vc=0, Ts=OTp)	IH, Ts=0	[TEST-r(F=Pt)]	
TEST_LINK (Vc=0, Ts=IOTp)	IH, Ts=ITp	[TEST-r(F=Pt)]	
TEST_LINK (Vc≠0) Ti_Expired	Logical Error (Local)		
T1_Expired (P_Ct=0)	IH, ITi		DISCONNECTED
T1_Expired (P_Ct≠0, Ts=Xs=0)	IT1, Decrement P_Ct, Vp=Vs, Stop Send_Proc	[RNR-c^1]	
T1_Expired (P_Ct≠0, Ts=ITp|IOTp)	IT1, Decrement P_Ct	[TEST-c^1]	
T1_Expired (P_Ct≠0, Xs=IXp|IOXp)	IT1, Decrement P_Ct	[XID-c^1]	
T2_Expired	Logical Error (Local)		

State: CHECKPOINTING + REJECTION + LOCAL_BUSY (17) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, ITi	[UA^0|UA^1]	DISCONNECTED
DM^0|DM^1	IH, TT1, ITi		DISCONNECTED
FRMR	IH, TT1, ITi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, ITi		RESETTING
TEST-c (Ts=0|OTp)	IH, Ts=OTp, Pt=P		
TEST-c (Ts=ITp|IOTp)	IH, Ts=IOTp, Pt=P		
TEST-r^0 TEST-r^1 (Vc=0, Ts=0|OTp)	Ignore_LPDU		
TEST-r^1 (Vc=0, Ts=ITp)	IH, Ts=0, TT1, ITi, Start Send_Proc		REJECTION + LOCAL_BUSY
TEST-r^1 (Vc=0, Ts=IOTp)	IH, Ts=OTp, TT1, ITi, Start Send_Proc		REJECTION + LOCAL_BUSY
TEST-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
TEST-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
TEST-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	
UA	IH_(RE), TT1, ITi	[FRMR^0]	FRMR_SENT
XID-c (Xs=0|OXp)	IH, Xs=OXp, Px=P		
XID-c (Xs=IXp|IOXp)	IH, Xs=IOXp, Px=P		
XID-r^0	IH		
XID-r^1 (Vc=0, Xs=0|OXp)	Ignore_LPDU		
XID-r^1 (Vc=0, Xs=IXp)	IH, Xs=0, TT1, ITi, Start Send_Proc		REJECTION + LOCAL_BUSY
XID-r^1 (Vc=0, Xs=IOXp)	IH, Xs=OXp, TT1, ITi, Start Send_Proc		REJECTION + LOCAL_BUSY
XID-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
XID-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
XID-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING + REJECTION + LOCAL_BUSY (17) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Va = Nr≤Vs)	Dsc_I-fld	[RNR-r^0]	CHECKPOINTING + LOCAL_BUSY
IS_I-c^0|IS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2	[RNR-r^0]	CHECKPOINTING + LOCAL_BUSY
IS_I-c^1 (Va = Nr≤Vs)	Dsc_I-fld	[RNR-r^1]	CHECKPOINTING + LOCAL_BUSY
IS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2	[RNR-r^1]	CHECKPOINTING + LOCAL_BUSY
IS_I-r^1 (Vc = 0)	Update_Va_Chkpt, Dsc_I-fld, Start Send_Proc	[RNR-r^0]	LOCAL_BUSY
IS_I-r^1 (Vc = DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, TTi, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
IS_I-r^1 (Vc = TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Ts = ITp, TTi, RT1, P_Ct = N2	[TEST-c^1]	CHECKPOINTING + LOCAL_BUSY
IS_I-r^1 (Vc = XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Xs = IXp, TTi, RT1, P_Ct = N2	[XID-c^1]	CHECKPOINTING + LOCAL_BUSY
OS_I-c^0|OS_I-r^0 (Va = Nr≤Vs)	Dsc_I-fld		
OS_I-c^0|OS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2		
OS_I-c^1 (Va = Nr≤Vs)	Dsc_I-fld	[RNR-r^1]	
OS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2	[RNR-r^1]	
OS_I-r^1 (Vc = 0)	Update_Va_Chkpt, Dsc_I-fld, Start Send_Proc		REJECTION + LOCAL_BUSY
OS_I-r^1 (Vc = DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, TTi, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
OS_I-r^1 (Vc = TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Tx = ITp, TTi, RT1, P_Ct = N2	[TEST-c^1]	
OS_I-r^1 (Vc = XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Xs = IXp, TTi, RT1, P_Ct = N2	[XID-c^1]	

State: CHECKPOINTING + REJECTION + LOCAL_BUSY (17) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0 (Va<Nr≤Vs)	Va=Nr, Ww=1, Ia_Ct=0		
REJ-c^1 (Va=Nr≤Vs)		[RNR-r^1]	
REJ-c^1 (Va<Nr≤Vs)	Va=Nr, Ww=1, Ia_Ct=0	[RNR-r^1]	
REJ-r^1 (Vc=0, Va≤Nr≤Vs)	Update_Va_Chkpt, Vs=Nr, Start_Send_Proc		REJECTION + LOCAL_BUSY
REJ-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vc=0, TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
REJ-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vc=0, Ts=ITp, TTi, RT1, P_Ct=N2	[TEST-c^1]	
REJ-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vc=0, Xs=IXp, TTi, RT1, P_Ct=N2	[XID-c^1]	
RNR-c^0|RNR-r^0 (Va=Nr<Vs)	Vb=LRb, Is_Ct=N2		
RNR-c^0|RNR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Vb=LRb, Va=Nr, Is_Ct=N2		
RNR-c^1 (Va=Nr≤Vs)	Vb=LRb, Is_Ct=N2	[RNR-r^1]	
RNR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Vb=LRb, Va=Nr, Is_Ct=N2	[RNR-r^1]	
RNR-r^1 (Vc=0, Va≤Nr≤Vs)	Update_Va_Chkpt, Vb=LRb, Is_Ct=N2		REJECTION + LOCAL_BUSY + REMOTE_BUSY
RNR-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vb=Rb, Vc=0, TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
RNR-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vb=Rb, Vc=0, TTi, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
RNR-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vb=Rb, Vc=0, TTi, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	
RR-c^0|RR-r^0 (Va=Nr≤Vs)	—		
RR-c^0|RR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr		
RR-c^1 (Va=Nr≤Vs)	—	[RNR-r^1]	
RR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr	[RNR-r^1]	
RR-r^1 (Vc=0)	Update_Va_Chkpt, Start_Send_Proc		REJECTION + LOCAL_BUSY
RR-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vc=0, TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
RR-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vc=0, TTi, RT1, P_Ct=N2, Ts=ITp	[TEST-c^1]	
RR-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vc=0, TTi, RT1, P_Ct=N2, Xs=IXp	[XID-c^1]	

CHECKPOINTING + CLEARING (18)

State: CHECKPOINTING + CLEARING (18) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb	[RNR-r^0]	CHECKPOINTING + LOCAL_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r|RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Vc = 0, Xs = 0)	Vc = XIDp		
SEND_XID (Vc = 0, Xs = IXp)	Logical Error (Local)		
SEND_XID (Vc = 0, Xs = OXp)	IH, Xs = 0	[XID-r(F = Px)]	
SEND_XID (Vc = 0, Xs = IOXp)	IH, Xs = IXp	[XID-r(F = Px)]	
SEND_XID (Vc≠0) SET_ABME	Logical Error (Local)		
SET_ADM (Vc = 0)	Vc = DISCp		
SET_ADM (Vc≠0)	Logical Error (Local)		
TEST_LINK (Vc = 0, Ts = 0)	Vc = TESTp		
TEST_LINK (Vc = 0, Ts = ITp)	Logical Error (Local)		
TEST_LINK (Vc = 0, Ts = OTp)	IH, Ts = 0	[TEST-r(F = Pt)]	
TEST_LINK (Vc = 0, Ts = IOTp)	IH, Ts = ITp	[TEST-r(F = Pt)]	
TEST_LINK (Vc≠0) Ti_Expired	Logical Error (Local)		
T1_Expired (P_Ct = 0)	IH, ITi		DISCONNECTED
T1_Expired (P_Ct≠0, Ts = Xs = 0)	Decrement P_Ct, IT1, Vp = Vs, Stop Send_Proc	[RR-c^1]	
T1_Expired (P_Ct≠0, Ts = ITp|IOTp)	Decrement P_Ct, IT1	[TEST-c^1]	
T1_Expired (P_Ct≠0, Xs = IXp|IOXp)	Decrement P_Ct, IT1	[XID-c^1]	
T2_Expired	Logical Error (Local)		

State: CHECKPOINTING + CLEARING (18) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, ITi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, ITi		DISCONNECTED
FRMR	IH, TT1, ITi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, ITi		RESETTING
TEST-c (Ts=0|OTp)	IH, Ts=OTp, Pt=P		
TEST-c (Ts=ITp|IOTp)	IH, Ts=IOTp, Pt=P		
TEST-r^0 TEST-r^1 (Vc=0, Ts=0|OTp)	Ignore_LPDU		
TEST-r^1 (Vc=0, Ts=ITp)	IH, Ts=0, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING
TEST-r^1 (Vc=0, Ts=IOTp)	IH, Ts=OTp, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING
TEST-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
TEST-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
TEST-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	
UA (Vi≠ISp)	IH_(RE), TT1, ITi	[FRMR^0]	FRMR_SENT
UA (Vi=ISp)	Ign_LPDU		
XID-c (Xs=0|OXp)	IH, Xs=OXp, Px=P		
XID-c (Xs=IXp|IOXp)	IH, Xs=IOXp, Px=P		
XID-r^0	IH		
XID-r^1 (Vc=0, Xs=0|OXp)	Ignore_LPDU		
XID-r^1 (Vc=0, Xs=IXp)	IH, Xs=0, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING
XID-r^1 (Vc=0, Xs=IOXp)	IH, Xs=OXp, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING
XID-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
XID-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
XID-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING + CLEARING (18) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Va = Nr≤Vs)	Rcv_BTU	[Send_ACK]	CHECKPOINTING
IS_I-c^0|IS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Rcv_BTU, Is_Ct = N2	[Send_ACK]	CHECKPOINTING
IS_I-c^1 (Va = Nr≤Vs)	Rcv_BTU	[RR-r^1]	CHECKPOINTING
IS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Rcv_BTU, Is_Ct = N2	[RR-r^1]	CHECKPOINTING
IS_I-r^1 (Vc = 0)	Update_Va_Chkpt, Rcv_BTU, Start Send_Proc	[Send_ACK]	LINK_OPENED
IS_I-r^1 (Vc = DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, TTi, RT1, P_Ct = N2	[$DISC^1$]	DISCONNECTING
IS_I-r^1 (Vc = TESTp)	Update_Va_Chkpt, Rcv_BTU, Vc = 0, Ts = ITp, TTi, RT1, P_Ct = N2	[TEST-c^1]	CHECKPOINTING
IS_I-r^1 (Vc = XIDp)	Update_Va_Chkpt, Rcv_BTU, Vc = 0, Xs = IXp, TTi, RT1, P_Ct = N2	[XID-c^1]	CHECKPOINTING
OS_I-c^0|OS_I-r^0 (Va = Nr≤Vs)	Dsc_I-fld	[REJ-r^0]	CHECKPOINTING + REJECTION
OS_I-c^0|OS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2	[REJ-r^0]	CHECKPOINTING + REJECTION
OS_I-c^1 (Va = Nr≤Vs)	Dsc_I-fld	[REJ-r^1]	CHECKPOINTING + REJECTION
OS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2	[REJ-r^1]	CHECKPOINTING + REJECTION
OS_I-r^1 (Vc = 0)	Update_Va_Chkpt, Dsc_I-fld, Start Send_Proc	[REJ-r^0]	REJECTION
OS_I-r^1 (Vc = DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, TTi, RT1, P_Ct = N2	[$DISC^1$]	DISCONNECTING
OS_I-r^1 (Vc = TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Tx = ITp, TTi, RT1, P_Ct = N2	[TEST-c^1]	CHECKPOINTING
OS_I-r^1 (Vc = XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Xs = IXp, TTi, RT1, P_Ct = N2	[XID-c^1]	CHECKPOINTING

State: CHECKPOINTING + CLEARING (18) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0		
REJ-c^1 (Va=Nr≤Vs)		[RR-r^1]	
REJ-c^1 (Va<Nr≤Vs)	Ww=1, Ia_Ct=0, Va=Nr, Is_Ct=N2	[RR-r^1]	
REJ-r^1 (Vc=0)	Update_Va_Chkpt, Vp=Nr, RT1	[RR-c^1]	CHECKPOINTING
RNR-c^0|RNR-r^0 (Va=Nr≤Vs)	Vb=Rb		
RNR-c^0|RNR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=Rb		
RNR-c^1 (Va=Nr≤Vs)	Vb=Rb	[RR-r^1]	
RNR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=Rb	[RR-r^1]	
RNR-r^1 (Vc=0)	Update_Va_Chkpt, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING
RR-c^0|RR-r^0 (Va=Nr≤Vs)			
RR-c^0|RR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2		
RR-c^1 (Va=Nr≤Vs)		[RR-r^1]	
RR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2	[RR-r^1]	
RR-r^1 (Vc=0)	Update_Va_Chkpt, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING
REJ-r^1|RNR-r^1|RR-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
REJ-r^1|RNR-r^1|RR-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
REJ-r^1|RNR-r^1|RR-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

CHECKPOINTING + REJECTION + CLEARING (19)

State: CHECKPOINTING + REJECTION + CLEARING (19) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb = Lb	[RNR-r^0]	CHECKPOINTING + REJECTION + LOCAL_BUSY
EXIT_LCL_Busy	Logical Error (Local)		
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r|RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ = 00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ = 00001	[FRMR^1]	FRMR_SENT
SEND_XID (Vc = 0, Xs = 0)	Vc = XIDp		
SEND_XID (Vc = 0, Xs = IXp)	Logical Error (Local)		
SEND_XID (Vc = 0, Xs = OXp)	IH, Xs = 0	[XID-r(F = Px)]	
SEND_XID (Vc = 0, Xs = IOXp)	IH, Xs = IXp	[XID-r(F = Px)]	
SEND_XID (Vc≠0) SET_ABME	Logical Error (Local)		
SET_ADM (Vc = 0)	Vc = DISCp		
SET_ADM (Vc≠0)	Logical Error (Local)		
SEND_BTU		[Queue_I_LPDU]	
TEST_LINK (Vc = 0, Ts = 0)	Vc = TESTp		
TEST_LINK (Vc = 0, Ts = ITp)	Logical Error (Local)		
TEST_LINK (Vc = 0, Ts = OTp)	IH, Ts = 0	[TEST-r(F = Pt)]	
TEST_LINK (Vc = 0, Ts = IOTp)	IH, Ts = ITp	[TEST-r(F = Pt)]	
TEST_LINK (Vc≠0) Ti_Expired	Logical Error (Local)		
T1_Expired (P_Ct = 0)	IH, ITi		DISCONNECTED
T1_Expired (P_Ct≠0, Ts = Xs = 0)	Decrement P_Ct, IT1, Vp = Vs, Stop Send_Proc	[RR-c^1]	
T1_Expired (P_Ct≠0, Ts = ITp|IOTp)	Decrement P_Ct, IT1	[TEST-c^1]	
T1_Expired (P_Ct≠0, Xs = IXp|IOXp)	Decrement P_Ct, IT1	[XID-c^1]	
T2_Expired	Logical Error (Local)		

State: CHECKPOINTING + REJECTION + CLEARING (19) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, ITi	[UA^0|UA^1]	DISCONNECTED
DM	IH, TT1, ITi		DISCONNECTED
FRMR	IH, TT1, ITi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, ITi		RESETTING
TEST-c (Ts=0|OTp)	IH, Ts=OTp, Pt=P		
TEST-c (Ts=ITp|IOTp)	IH, Ts=IOTp, Pt=P		
TEST-r^0 TEST-r^1 (Vc=0, Ts=0|OTp)	Ignore_LPDU		
TEST-r^1 (Vc=0, Ts=ITp)	IH, Ts=0, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING + REJECTION
TEST-r^1 (Vc=0, Ts=IOTp)	IH, Ts=OTp, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING + REJECTION
TEST-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
TEST-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
TEST-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	
UA	IH_(RE), TT1, ITi	[FRMR^0]	FRMR_SENT
XID-c (Xs=0|OXp)	IH, Xs=OXp, Px=P		
XID-c (Xs=IXp|IOXp)	IH, Xs=IOXp, Px=P		
XID-r^0	IH		
XID-r^1 (Vc=0, Xs=0|OXp)	Ignore_LPDU		
XID-r^1 (Vc=0, Xs=IXp)	IH, Xs=0, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING + REJECTION
XID-r^1 (Vc=0, Xs=IOXp)	IH, Xs=OXp, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING + REJECTION
XID-r^1 (Vc=DISCp)	Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
XID-r^1 (Vc=TESTp)	Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
XID-r^1 (Vc=XIDp)	Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

State: CHECKPOINTING + REJECTION + CLEARING (19) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0 (Va = Nr≤Vs)	Rcv_BTU	[Send_ACK]	CHECKPOINTING
IS_I-c^0|IS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Rcv_BTU, Is_Ct = N2	[Send_ACK]	CHECKPOINTING
IS_I-c^1 (Va = Nr≤Vs)	Rcv_BTU	[RR-r^1]	CHECKPOINTING
IS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Rcv_BTU, Is_Ct = N2	[RR-r^1]	CHECKPOINTING
IS_I-r^1 (Vc = 0)	Update_Va_Chkpt, Rcv_BTU, Start Send_Proc	[Send_ACK]	LINK_OPENED
IS_I-r^1 (Vc = DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
IS_I-r^1 (Vc = TESTp)	Update_Va_Chkpt, Rcv_BTU, Vc = 0, Ts = ITp, RT1, P_Ct = N2	[TEST-c^1]	CHECKPOINTING
IS_I-r^1 (Vc = XIDp)	Update_Va_Chkpt, Rcv_BTU, Vc = 0, Xs = IXp, RT1, P_Ct = N2	[XID-c^1]	CHECKPOINTING
OS_I-c^0|OS_I-r^0 (Va = Nr≤Vs)	Dsc_I-fld		CHECKPOINTING + REJECTION
OS_I-c^0|OS_I-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2		CHECKPOINTING + REJECTION
OS_I-c^1 (Va = Nr≤Vs)	Dsc_I-fld	[RR-r^1]	CHECKPOINTING + REJECTION
OS_I-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va = Nr, Dsc_I-fld, Is_Ct = N2	[RR-r^1]	CHECKPOINTING + REJECTION
OS_I-r^1 (Vc = 0)	Update_Va_Chkpt, Dsc_I-fld, Start Send_Proc		REJECTION
OS_I-r^1 (Vc = DISCp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, RT1, P_Ct = N2	[DISC^1]	DISCONNECTING
OS_I-r^1 (Vc = TESTp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Tx = ITp, RT1, P_Ct = N2	[TEST-c^1]	CHECKPOINTING + REJECTION
OS_I-r^1 (Vc = XIDp)	Update_Va_Chkpt, Dsc_I-fld, Vc = 0, Xs = IXp, RT1, P_Ct = N2	[XID-c^1]	CHECKPOINTING + REJECTION

State: CHECKPOINTING + REJECTION + CLEARING (19) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0		
REJ-c^1 (Va=Nr≤Vs)		[RR-r^1]	
REJ-c^1 (Va<Nr≤Vs)	Va=Nr, Is_Ct=N2, Ww=1, Ia_Ct=0	[RR-r^1]	
REJ-r^1 (Vc=0)	Update_Va_Chkpt, Vp=Nr, Decrement Is_Ct, RT1	[RR-c^1]	CHECKPOINTING + REJECTION
RNR-c^0|RNR-r^0 (Va=Nr≤Vs)	Vb=Rb		
RNR-c^0|RNR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=Rb		
RNR-c^1 (Va=Nr≤Vs)	Vb=Rb, Is_Ct=N2	[RR-r^1]	
RNR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2, Vb=Rb	[RR-r^1]	
RNR-r^1 (Vc=0)	Update_Va_Chkpt, RT1, Vp=Vs, Is_Ct=N2	[RR-c^1]	CHECKPOINTING + REJECTION
RR-c^0|RR-r^0 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2		
RR-c^1 (Va=Nr≤Vs)		[RR-r^1]	
RR-c^1 (Va<Nr≤Vs)	Adjust_Ww, Va=Nr, Is_Ct=N2	[RR-r^1]	
RR-r^1 (Vc=0)	Update_Va_Chkpt, RT1, Vp=Vs	[RR-c^1]	CHECKPOINTING + REJECTION
REJ-r^1|RNR-r^1|RR-r^1 (Vc=DISCp)	Update_Va_Chkpt, Vc=0, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
REJ-r^1|RNR-r^1|RR-r^1 (Vc=TESTp)	Update_Va_Chkpt, Vc=0, Ts=ITp, RT1, P_Ct=N2	[TEST-c^1]	
REJ-r^1|RNR-r^1|RR-r^1 (Vc=XIDp)	Update_Va_Chkpt, Vc=0, Xs=IXp, RT1, P_Ct=N2	[XID-c^1]	

REJECTION + LOCAL_BUSY + REMOTE_BUSY (20)

State: REJECTION + LOCAL_BUSY + REMOTE_BUSY (20) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS ENTER_LCL_Busy	Logical Error (Local)		
EXIT_LCL_Busy	Vb=Rb, TTi, RT1, P_Ct=N2, Vp=Vs	[RR-c^1]	CHECKPOINTING + REJECTION
LPDU_INVALID (I-c^0|I-r|REJ-c^0|REJ-r |RNR-c^0|RNR-r|RR-c^0|RR-r and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ=00001	[FRMR^0]	FRMR_SENT
LPDU_INVALID (I-c^1|REJ-c^1|RNR-c^1|RR-c^1 and not Va≤Nr≤Vs)	IH_(RE), TT1, ITi, VWXYZ=00001	[FRMR^1]	FRMR_SENT
SEND_BTU		[Queue_I_LPDU]	
SEND_XID (Xs=0)	Xs=IXp, TTi, RT1, P_Ct=N2	[XID-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
SEND_XID (Xs=OXp)	Xs=0	[XID-r(F=Px)]	
SEND_XID (Xs=IXp|IOXp) SET_ABME	Logical Error (Local)		
SET_ADM	TTi, RT1, P_Ct=N2	[DISC^1]	DISCONNECTING
TEST_LINK (Ts=0)	Ts=ITp, TTi, RT1, P_Ct=N2	[TEST-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
TEST_LINK (Ts=OTp)	Ts=0	[TEST-r(F=Pt)]	
TEST_LINK (Tx=ITp|IOTp)	Logical Error (Local)		
Ti_Expired	IT1, P_Ct=N2, Vp=Vs	[RNR-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
T1_Expired Is_Ct≠0	IT1, P_Ct=N2, Vp=Vs	[RNR-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
T1_Expired Is_Ct=0	P_Ct=N2, IT1	[DISC^1]	DISCONNECTING
T2_Expired	Logical Error (Local)		

State: REJECTION + LOCAL_BUSY + REMOTE_BUSY (20) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC^0|DISC^1	IH, TT1, RTi	[UA^0|UA^1]	DISCONNECTED
DM^0|DM^1	IH, TT1, RTi		DISCONNECTED
FRMR	IH, TT1, RTi, P_Ct=N2		FRMR_RECEIVED
SABME	IH, Vi=RIp, Pf=P, TT1, RTi		RESETTING
TEST-c	IH, Ts=OTp, Pt=P		
TEST-r	Ignore_LPDU		
UA	IH_(RE), TT1, RTi	[FRMR^0]	FRMR_SENT
XID-c	IH, Xs=OXp, Px=P		
XID-r^0	IH		
XID-r^1	Ignore_LPDU		

State: REJECTION + LOCAL_BUSY + REMOTE_BUSY (20) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
IS_I-c^0|IS_I-r^0|IS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^0]	LOCAL_BUSY + REMOTE_BUSY
IS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	LOCAL_BUSY + REMOTE_BUSY
OS_I-c^0|OS_I-r^0|OS_I-r^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld		
OS_I-c^1 (Va≤Nr≤Vs)	Update_Va, Dsc_I-fld	[RNR-r^1]	

State: REJECTION + LOCAL_BUSY + REMOTE_BUSY (20) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ-c^0|REJ-r^0|REJ-r^1 (Va≤Nr≤Vs)	Update_Va, Vs=Nr, Decrement Is_Ct, Vb=Lb, Ww=1, Ia_Ct=0, Start Send_Proc		REJECTION + LOCAL_BUSY
REJ-c^1 (Va≤Nr≤Vs)	Update_Va, Vs=Nr, Decrement Is_Ct, Vb=Lb, Ww=1, Ia_Ct=0, Start Send_Proc	[RNR-r^1]	REJECTION + LOCAL_BUSY
RNR-c^0|RNR-r^0|RNR-r^1 (Va≤Nr≤Vs)	Update_Va		
RNR-c^1 (Va≤Nr≤Vs)	Update_Va	[RNR-r^1]	
RR-c^1 (Va≤Nr<Vs)	Vb=Lb, Update_Va	[RNR-c^1] [RNR-r^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
RR-c^1 (Va≤Nr=Vs)	Vb=Lb, Update_Va	[RNR-r^1]	REJECTION + LOCAL_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr<Vs)	Vb=Lb, Update_Va	[RNR-c^1]	CHECKPOINTING + REJECTION + LOCAL_BUSY
RR-c^0|RR-r^0|RR-r^1 (Va≤Nr=Vs)	Vb=Lb, Update_Va		REJECTION + LOCAL_BUSY

FRMR_RECEIVED (21)

State: FRMR_RECEIVED (21) Input Class: Internal Events			
Input	**Action**	**[LPDU Transfers]**	**New State**
ACTIVATE_LS DEACTIVATE_LS	Logical Error (Local)		
ENTER_LCL_Busy	Vb=Lb		
EXIT_LCL_Busy	Vb=0		
LPDU_INVALID	Ignore_LPDU		
SEND_BTU|SEND_XID	Logical Error (Local)		
SET_ABME	Vi=Llp, TTi, IT1, P_Ct=N2	[SABME[1]]	LINK_OPENING
SET_ADM	TTi, IT1, P_Ct=N2	[DISC[1]]	DISCONNECTING
TEST_LINK	Logical Error (Local)		
Ti_Expired (P_Ct≠0)	Decrement P_Ct IH, ITi		
Ti_Expired (P_Ct=0)	IH, IT1, P_Ct=N2	[DISC[1]]	DISCONNECTING
T1_Expired|T2_Expired	Logical Error (Local)		

State: FRMR_RECEIVED (21) Input Class: Received U-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
DISC[0]|DISC[1]	IH, RTi	[UA[0]|UA[1]]	DISCONNECTED
DM	IH, RTi		DISCONNECTED
FRMR	IH		
SABME	IH, Vi=Rlp, Pf=P, RTi		RESETTING
TEST-c|TEST-r UA XID-c|XID-r	Ignore_LPDU		

State: FRMR_RECEIVED (21) Input Class: Received I-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
I-c|I-r	Ignore_LPDU		

State: FRMR_RECEIVED (21) Input Class: Received S-format LPDUs			
Input	**Action**	**[LPDU Transfers]**	**New State**
REJ|RNR|RR	Ignore_LPDU		

Part 4. Network Management

Chapter 13. Network Management

This chapter describes:

- The protocols used to manage an IBM Token-Ring Network
- The frame formats and flows for the exchange of statistical, error, and control information between the various network components
- The structures used to manage a local area network, how they operate, and how they are interconnected.

Network Management Components

The architectural components used to manage an IBM Token-Ring Network are:

- Ring Station
- Management Servers
  - LAN Reporting Mechanism (LRM)
  - Ring Error Monitor (REM)
  - Configuration Report Server (CRS)
  - Ring Parameter Server (RPS)
  - LAN Bridge Server (LBS)
- LAN Manager
- SNA Control Point.

The participation of ring stations in Token-Ring Network management is described in Chapter 5, "MAC Frames."

The management servers collect data from ring stations in the Token-Ring Network, analyze it, and forward management information to one or more LAN managers. Each of the management servers listed above is described in one of the following chapters.

The LAN manager can log the data, analyze it further, and take management actions based on the information. Also, the LAN manager can notify an SNA control point of error conditions on the Token-Ring Network. These interactions are described in Chapter 19, "Token-Ring Network Alerts."

Token-Ring Network Management Hierarchy

The following figure shows the relationship between ring stations, management servers, a LAN manager, and the SNA control point.

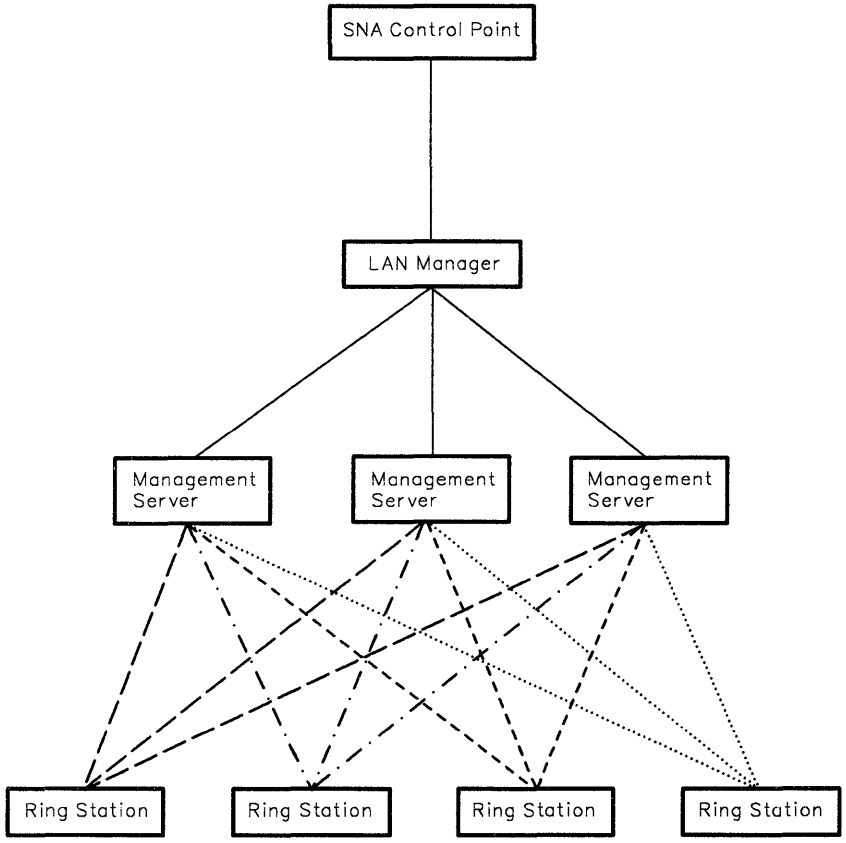


Figure 13-1. Token-Ring Network Management Hierarchy

LAN Components

LAN components perform three LAN management functions which are described in the following sections. Ring stations use only MAC services to send and receive management data to and from management servers. Management servers use the services of connection-oriented LLC for communication between themselves and LAN managers. The following functions are provided by ring stations and management servers.

Status Determination Function

This function is used to request information about another LAN component.

Configuration Control Function

This function is used to change the values of operational parameters of a LAN component and also to change the topology of the network, for example, to remove a ring station from a ring or activate a bridge. Management servers can control the configuration of ring stations and a LAN manager controls the configuration of management servers. The LAN manager can also request management servers to change the configuration of ring stations on their rings.

Notification Function

This function is used by ring stations to inform a management server of events occurring in ring stations or their neighbors. This function is also used by management servers to inform a LAN manager of error conditions or configuration changes it detects on its ring. Notifications flow from ring stations to management servers and from management servers to LAN managers.

LAN Management Frame Format

The LAN management frame is used to carry information between management servers and the LAN manager. LAN management frames are transported between management servers and LAN managers using the services of the logical link control (LLC). For a description of the LLC, see "Part 3. The Logical Link Control (LLC) Sub-Layer."

Management Information Field

The *management information* field is a variable-length field that is contained in the LLC information field (see Chapter 8, "LLC Frames") in frames sent between a management server and a LAN manager. It is a self-defining field containing its own length, identifier, and a list of zero or more contained elements. Each element in the list is a *subvector* and the list itself is a *major vector*. This allows straightforward parsing of variable-length and complex data elements.

Major Vector The major vector identifies the fundamental unit of management information. Its value consists of its length, a function identifier, and zero or more subvectors. Only one major vector is present in a management frame. The format of the major vector is shown below.

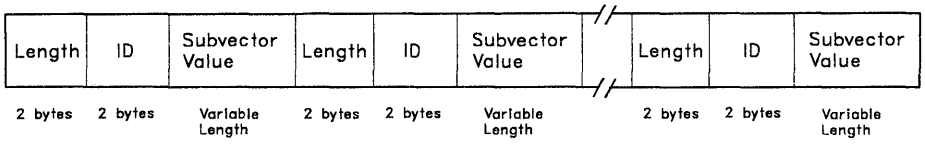


Figure 13-2. Major Vector Format

Major Vector Length Field The value of this field is a 16-bit binary number that gives the length (in bytes) of the major vector, including the length field itself. Its values can be X'0004' to X'FFFF' inclusively (subject to the implementation constraints of the maximum frame length).

Major Vector Identifier Field The value of this field contains a 2-byte code point that identifies the major vector. Bit position zero of this field indicates whether the major vector is universal or is implementation-specific.

- Bit 0 =B'0': Universal
- Bit 0 =B'1': Implementation-Specific

The format of the ID field is shown below.

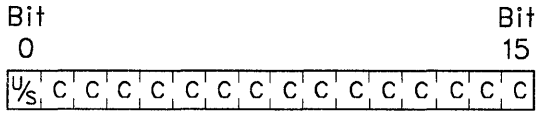


Figure 13-3. Major Vector Identifier Field Format

U/S = Universal/Implementation-Specific Bit
C = Code Point Bits

Note: All major vector identifiers not defined in this reference are reserved by IBM for future use, including those that fall in the implementation-specific range.

Subvectors One subvector is used for each unit of data that is being transmitted. Each subvector is identified by a subvector identifier and is not restricted to a fixed position within a major vector. A subvector contains its length, an identifier, and zero or more subvector values. The formats of the atomic and complex subvectors are shown below.

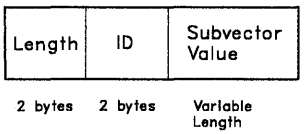


Figure 13-4. Atomic Subvector

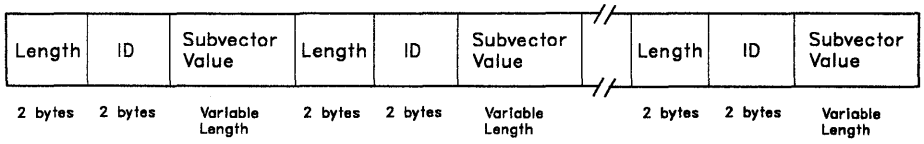


Figure 13-5. Complex Subvector

Subvector Length The value of this field is a 16-bit binary number that gives the length (in bytes) of the subvector, including the length field itself. Its values can be X'0004' to X'FFFF' inclusively (subject to the implementation constraints of the maximum data structure length).

Subvector Identifier Field The value of this field contains a 2-byte code point that identifies the subvector. Bit position zero of this field indicates whether the subvector is universal or implementation-specific.

Bit 0 =B'0': Universal
Bit 0 =B'1': Implementation-specific

Bit position one of this field indicates whether the subvector is an atomic or complex subvector

Bit 1 =B'0': Complex (see Figure 13-5)
Bit 1 =B'1': Atomic (see Figure 13-4)

The format of the ID field is shown below.

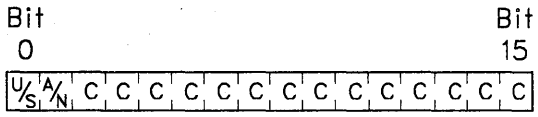


Figure 13-6. Subvector Identifier Field Format

U/S = Universal/Implementation Specific Bit
A/N = Atomic/Non-Atomic Value
C = Code Point Bits

Note: All subvector identifiers not defined in this reference are reserved by IBM for future use, including those that fall in the implementation-specific range.

Subvector Value Field The value of this field is the element of information being carried in this subvector or a set of nested subvectors. A subvector value is always an integral number of bytes in length. Subvectors themselves, at the different nested levels, can contain either atomic data units or other subvectors. These subvectors have the same format as other subvectors. Bit 1 of the subvector identifier specifies whether or not a particular subvector contains nested subvectors (see Figure 13-6) or an atomic value.

Frame Routing in the DLC.LAN.MGR.LMS

This figure shows how the DLC.LAN.MGR.LMS routes frames. The local management services (LMS) component of the DLC.LAN.MGR is the component that handles reports from ring stations and token-ring management.

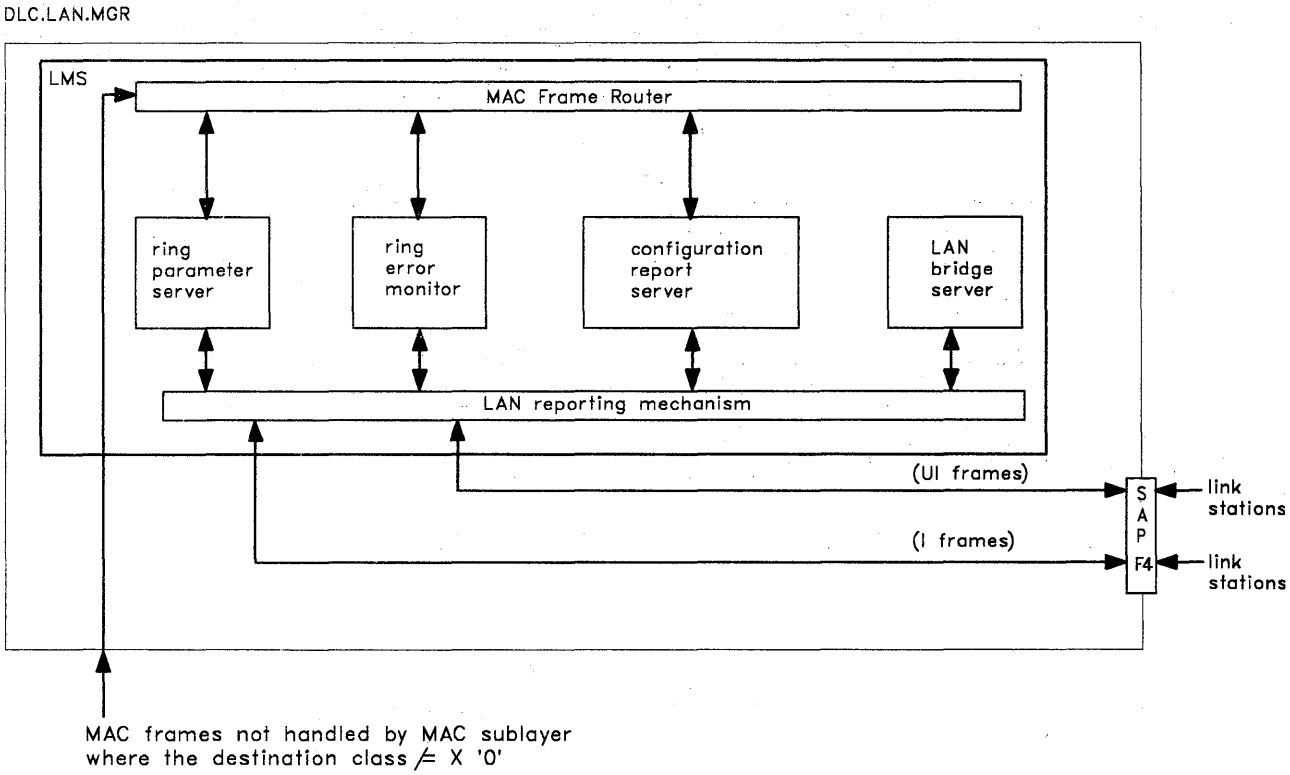


Figure 13-7. Frame Routing in the DLC.LAN.MGR

MAC frames that contain a destination class field that does not indicate a ring station are delivered to the MAC frame router in the DLC.LAN.MGR.LMS component, which forwards MAC frames to the correct management server (either the ring parameter server, the ring error monitor, or the configuration report server depending on the destination class). The LAN bridge server does not receive or process any MAC frames.

LLC frames addressed to SAP X'F4' are processed by the LAN reporting mechanism component. The LAN reporting mechanism routes LLC frames received from remote LAN managers to one of the collocated management servers or acts on the frames itself, depending on the major vector identifier in the frame.

Management Server Functions

The chapters that follow in this section describe the various management functions of the LAN reporting mechanism, the ring error monitor, the configuration report server, the ring parameter server, and the LAN bridge server. The functions of *status determination, configuration control,* and *notification* are described for the management servers.

Server Parameters

The structure of the management information exchanged in LAN management frames is described in the following chapters. The internal representation of this data is implementation specific.

Server Parameter Description

In this reference, management information is shown as data structures consisting of a major vector and various subvectors. The indented form that is used in the following chapters indicates the levels of subvectors. The following figure is representative of the presentation of server parameters in this reference. (See "LAN Management Frame Format" on page 13-3.)

```
Parameter 1
Parameter 2
   Parameter 3
   Parameter 4
      Parameter 5
Parameter 6
```

Figure 13-8. Example of Server Parameters

The first level subvectors are: *Parameter 1*, *Parameter 2*, and *Parameter 6*. *Parameter 3* and *Parameter 4* are nested within *Parameter 2* and *Parameter 5* is nested within *Parameter 4*. For example, in management exchanges between a management server and LAN manager, *Parameter 2* would be represented in the following way:

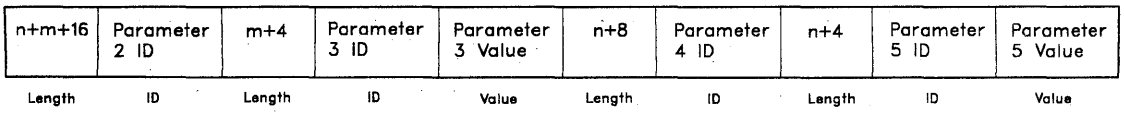


Figure 13-9. Server Parameter Subvector Representation

n = length of Parameter 5 subvector value
m = length of Parameter 3 subvector value

Conditions of Presence

The conditions of presence for each subvector (and nested subvectors) within individual management major vectors are shown in Appendix F, "Conditions of Presence." Each table in that appendix shows all the subvectors that can be present within a major vector or subvector and indicates the conditions for which each vector or subvector is present. The tables also show the major vector and subvector identifier code points and length of each major vector and subvector.

Subvectors

The semantics of each subvector is defined in the *data structures* section in the following chapters. When additional information is required for a subvector in a particular major vector (frame), the information is included in the description of that frame.

Chapter 14. LAN Reporting Mechanism

The LAN reporting mechanism is a management server used to control communication between LAN managers and *remote* management servers.

The LAN reporting mechanism also has a protocol boundary with each LAN management server with which it is *collocated*.

Note: The term *collocated*, as used in this reference, describes those LAN managers and LAN management servers (including the LAN reporting mechanism) that are located so that they can communicate with each other without exchanging frames through the local area network.

Operating Environment Constraints

The LAN reporting mechanism operating environment has the following constraints:

- A LAN reporting mechanism is collocated with each management server that communicates with LAN managers unless that management server and the LAN manager are collocated.
- One LAN reporting mechanism can maintain connections with LAN managers for multiple collocated management servers.
- A management server uses one LAN reporting mechanism to maintain a connection with LAN managers.

LAN Reporting Mechanism Functions

the following sections describe the LAN reporting mechanism functions.

Reporting Link Maintenance Function

This function maintains an LLC connection (called a *reporting link*) for each LAN manager that it communicates with. For a description of LLC connection-oriented service, see Chapter 11, "Operation of Link Stations." Other management servers also use these reporting links to communicate with LAN managers rather than maintaining their own separate connections with the LAN managers.

Control Function

The control function allows the LAN reporting mechanism to define one LAN manager as the *controlling* LAN manager. The controlling LAN manager can use the reporting link to remotely set the values of operational parameters in the LAN servers with which the LAN reporting mechanism is collocated. Also, the controlling LAN manager can request and obtain status or receive unsolicited notifications from a remote LAN reporting mechanism or any of the collocated management servers.

Besides the controlling link, two other types of reporting links are defined:

- Alternate links are used by *alternate* LAN managers. LAN managers that use these reporting links become controlling managers if the controlling reporting link fails, which can occur if the reporting link is terminated normally by the controlling LAN manager or an anomaly in the network forces the connection to terminate. Alternate LAN managers can also use these links to request and obtain the status or receive unsolicited notifications from a LAN reporting mechanism or any of the collocated management servers.
- Observing links that are used by *observing* LAN managers. Observing LAN managers can use these links to request and obtain the status or receive unsolicited notifications from a LAN reporting mechanism or any of the collocated servers. However, they are not eligible to become the controlling LAN manager (even if no other LAN managers have reporting links with the LAN reporting mechanism).

If the link with the controlling LAN manager is closed or lost, the alternate LAN manager using the lowest-numbered reporting link is automatically given control. If no alternate LAN managers have reporting links with the LAN reporting mechanism, control is lost. If there is no controlling LAN manager with a reporting link to the LAN reporting mechanism and a LAN manager establishes a controlling or alternate reporting link, control is automatically shifted to that LAN manager. If an alternate LAN manager has control of the LAN reporting mechanism, and a new LAN manager establishes a reporting link with a lower-numbered identifier, control is shifted to the new LAN manager.

Routing Function

This function routes frames from LAN managers to appropriate collocated management servers and forwards responses and unsolicited notifications from collocated management servers to selected remote LAN managers.

Notification Function

This function notifies the LAN managers when requests for new reporting links are accepted or rejected. It also notifies LAN managers when control is shifted from one LAN manager to another or when control is lost.

If the LAN reporting mechanism cannot deliver data to a LAN manager with which it has a reporting link because its node experiences internal congestion, it reports the fact that it discarded data to the affected LAN manager.

Security Function

This function maintains and checks passwords that are used to authenticate users of reporting links.

Protocol Boundary Function

This function allows communication between the LAN reporting mechanism and its collocated management servers. It also provides a means for management servers to inform the LAN reporting mechanism about their state (for example, active or inactive) and other conditions that might affect the LAN reporting mechanism's capability to route frames between management servers and LAN managers.

Data Structures

The following data structures are sent and received by the LAN reporting mechanism.

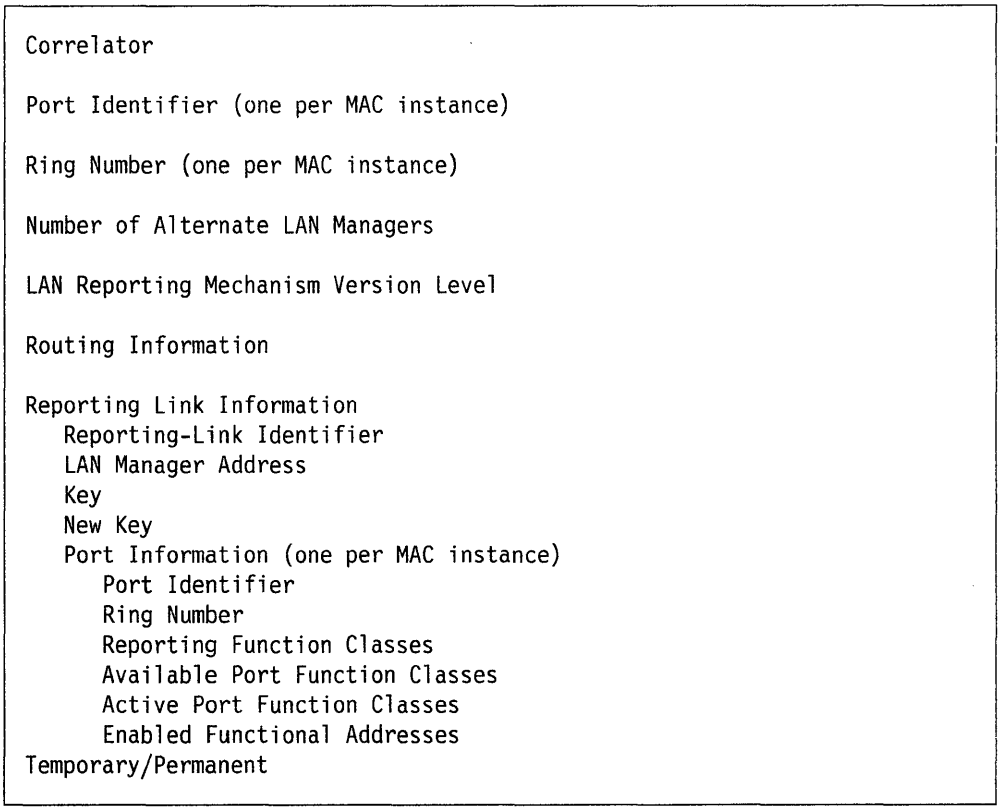


Figure 14-1. LAN Reporting Mechanism Data Structures

Correlator Subvector

This subvector has a 2-byte value field that is used by the LAN manager to associate responses with requests generated by the LAN manager.

Port Identifier Subvector

This subvector has a 6-byte value field containing the MAC address of the ring station attaching the LAN reporting mechanism to the ring (see "Addresses" on page 3-9). There is one port identifier field in the LAN reporting mechanism for each MAC instance used by the LAN reporting mechanism.

Ring Number Subvector

This subvector has a 2-byte integer value field containing the number of the ring into which the station identified in the port identifier is inserted (see "Route Designator Fields" on page 2-10). There is one ring number field in the LAN reporting mechanism for each MAC instance.

Number of Alternate LAN Managers Subvector

This subvector has a 2-byte integer value field indicating the number of reporting links reserved for alternate reporting links. The total number of reporting links supported by the LAN reporting mechanism is: 1 (for the controlling reporting link) plus the number of alternate reporting links plus the number of observing reporting links.

LAN Reporting Mechanism Version Level Subvector

This subvector contains the following information about a LAN reporting mechanism.

Discriminator Field

This field consists of a 1-byte integer that indicates the encoding method used for the rest of the LAN reporting mechanism version level subvector values. Its value is either X'00' (indicating ASCII) or X'01' (indicating EBCDIC).

Common Version Identifier Field

This field consists of a 2-byte numeric version level of the LAN reporting mechanism.

Common Release Identifier Field

This field consists of a 2-byte numeric release level of the LAN reporting mechanism.

Common Modification Identifier Field

This field consists of a 2-byte numeric modification level of the LAN reporting mechanism.

Software Product Program Number Field

This field consists of a 7-byte alphanumeric field that identifies the LAN reporting mechanism.

Routing Information Subvector

This subvector has a value field, up to 18 bytes long, containing the routing information used for a reporting link from a LAN manager to the LAN reporting mechanism. The format of this field is described in "Routing Information Field" on page 2-6.

Reporting-Link Information Subvector

This subvector has a complex value field that contains information relating to a single reporting link. The nested subvectors contained in the reporting link information subvector are listed below.

Reporting-Link Identifier Subvector

This subvector has a 2-byte integer value field that identifies the reporting link within the LAN reporting mechanism. Reporting-link identifiers are sequentially numbered from X'0000'.

The identifiers for alternate reporting links follow the X'0000' reporting-link identifier. Identifiers of observing reporting links follow the identifiers for the alternate reporting links.

LAN Manager Address Subvector

This subvector has a value field that contains the 6-byte MAC address of the LAN manager to which the reporting link is connected (see "Addresses" on page 3-9). A value of X'000000000000' means that no LAN manager has a reporting link with the specified reporting-link identifier.

Key Subvector

This subvector has an 8-byte value field that contains a password maintained by the LAN reporting mechanism for this reporting link. This key is compared against a key that a LAN manager sends when requesting a reporting link with a LAN reporting mechanism.

New Key Subvector

This subvector has an 8-byte value field that contains the value to which the key in the reporting-link information structure is to be set. This subvector is present only when the value of the key kept by the LAN reporting mechanism is being changed.

Port Information Subvector

This subvector has a complex value field that contains information about the types of reports the LAN reporting mechanism sends or forwards to the LAN manager with the reporting link identified in the reporting-link identifier subvector. There is one port information field for each MAC instance used by the LAN reporting mechanism. Each contains the following nested subvectors.

Port Identifier Subvector This subvector has a 6-byte value field that contains the MAC address of the port about which this Port Information Subvector applies. Unique port identifiers are assured by using the individual MAC address to identify a port (see "Addresses" on page 3-9).

Ring Number Subvector This subvector has a 2-byte integer value field that contains the number of the ring into which the station (specified by the port identifier) is inserted.

Reporting Function Classes Subvector This subvector has a 4-byte bit-significant value field that specifies the management servers from which notifications are to be sent to the LAN manager with the reporting link identified in the reporting-link identifier subvector. This subvector is present for each port so that a LAN manager can selectively request reports about any ring being monitored. The significance of the bit positions is defined below. If a bit is set to B'1', the corresponding server reports are sent. All bit positions not shown below are reserved by IBM for future use.

Bit 0: LAN reporting mechanism
Bit 23: LAN bridge server
Bit 27: Configuration report server
Bit 28: Ring error monitor
Bit 30: Ring parameter server

Available Port Function Classes Subvector This subvector has a 4-byte bit-significant value field that indicates the presence of the program code to implement the reporting function specified by the reporting function classes (above) is collocated with this instance of the LAN reporting mechanism. The significance of the bit positions is defined below. If a bit is set to B'1', the corresponding server is collocated with the LAN reporting mechanism. All bit positions not shown are reserved by IBM for future use.

Bit 0: LAN reporting mechanism
Bit 23: LAN bridge server
Bit 27: Configuration report server
Bit 28: Ring error monitor
Bit 30: Ring parameter server

Active Port Function Classes Subvector This subvector has a 4-byte bit-significant value field that indicates if the program code to implement the reporting function specified by the reporting function classes (above) is collocated with this instance of the LAN reporting mechanism and is *currently active*. The significance of the bit positions is defined below. If a bit is set to B'1', the corresponding server is present and active. All bit positions not shown are reserved by IBM for future use.

Bit 0: LAN reporting mechanism
Bit 23: LAN bridge server
Bit 27: Configuration report server
Bit 28: Ring error monitor
Bit 30: Ring parameter server

Enabled Functional Addresses Subvector This subvector has a 4-byte bit-significant value field that indicates the functional addresses that are enabled for the port identified in the port identifier subvector. The functional addresses that are defined and the significance of the bit positions are shown in "Functional Addresses" on page 3-10.

Temporary/Permanent Subvector

This subvector has a 1-byte value field that indicates whether the LRM's parameters have been permanently changed or merely changed until the LRM is no longer operational. That is, a temporary change would not persist when the LRM became operational again. A value of X'00' indicates that the change to the parameter values is intended to be permanent. Any other value for the subvector indicates that the change is intended to be temporary. If this subvector is not present, the change is permanent.

Request LRM Status Frame (X'8601')

A LAN manager may request the status of the LRM process using the "Request LRM Status" frame. The contents of the Request LRM Status frame are listed in Figure 14-2.

```
Request LRM Status
  Correlator
  All
```

Figure 14-2. Request LRM Status Frame

Correlator Subvector

This subvector allows the LAN manager requesting LRM status to correlate the response with its request (described later in this section).

All Subvector

If this subvector is present within the Request LRM Status major vector, the associated response contains:

- Values for LRM version level
- Number of alternate LAN managers
- Reporting link information (all information for each reporting link, including all information for each port).

Conditions of Presence

The conditions of presence for this subvector are shown in "Request LRM Status (X'8601') Conditions of Presence" on page F-18.

Report LRM Status Frame (X'8602')

If no errors are detected in the Request LRM Status, the status requested is returned by LRM to the LAN manager in the "Report LRM Status" frame. The contents of the Report LRM Status are listed in Figure 14-3.

```
Report LRM Status
  Correlator
  LRM Version Level
  Number of Alternate LAN Managers
  Reporting-Link Information
    Reporting-Link Identifier
    LAN Manager Address
    Port Information
      Port Identifier
      Ring Number
```

Figure 14-3. Report LRM Status Frame

Correlator Subvector

This subvector allows the LAN manager requesting LRM status to correlate the response with its request.

Conditions of Presence

The conditions of presence for this subvector are shown in "Report LRM Status (X'8602') Conditions of Presence" on page F-18.

Set Reporting Point (X'8607')

This frame is sent by a LAN manager to a LAN reporting mechanism to establish a reporting link. It is sent as a UI frame (see Chapter 9, "Connectionless Service") to the X'F4' LSAP of the node containing the LAN reporting mechanism.

```
Set Reporting Point
   Correlator
   Number of Alternate LAN Managers
   Routing Information
   Reporting-Link Information
      Reporting-Link Identifier
      Key
      Port Information
         Port Identifier
         Reporting Function Classes
```

Figure 14-4. Set Reporting Point Frame

Correlator Subvector

This subvector allows a LAN manager to correlate a Set Reporting Point frame with its response. For a frame sequence to be valid, the correlator subvector value in the response must match the value that was sent in the Set Reporting Point frame.

Routing Information Subvector

This subvector specifies the routing information from the LAN manager to the LAN reporting mechanism to use for the reporting link being established. The LAN manager obtains the routing information before sending the Set Reporting Point frame (see "Source Routing" on page 3-2) although the LAN reporting mechanism actually initiates the LLC connection using the routing information provided by the LAN manager.

Reporting-Link Identifier Subvector

This subvector identifies the reporting link being requested with the LAN reporting mechanism. Before granting this reporting link to the requesting LAN manager, the LAN reporting mechanism assures that it is not being used by another LAN manager.

Key Subvector

The value of this subvector is checked against the key that is maintained by the LAN reporting mechanism for the requested reporting link. The two keys must match in order to establish the reporting link.

Port Information Subvector

This subvector provides a means for the LAN manager to initialize the reporting function classes parameter for the requested reporting link. This information is used by the LAN reporting mechanism to selectively forward reports from collocated servers to the LAN manager requesting the reporting link.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Set Reporting Point (X'8607') Conditions of Presence" on page F-20.

Parsing Algorithm

Before executing the Set Reporting Point, the LAN reporting mechanism checks it for errors using the algorithm shown in "Parsing Algorithm for Set Parameter Frames" on page E-3.

Parsing Error

If a parsing error is detected, the reporting link is not established and the LAN reporting mechanism responds to the requesting LAN manager with a Set Reporting Point Error frame (see "Set Reporting Point Error (X'8611')" on page 14-14).

Note: If any of the reasons for rejection listed on 14-12 occur, the LAN Manager Rejected frame (not a Set Reporting Point Error frame) is returned to the requesting LAN manager (see "Rejecting the Reporting Link" below).

Establishing the Reporting Link

The LAN reporting mechanism establishes a reporting link (LLC connection) with the requesting LAN manager if: (1) the two key values match, (2) the reporting link table entry is not in use, and (3) other implementation constraints are not violated. The reporting link uses the routing information specified in the routing information subvector of the Set Reporting Link frame.

The LAN reporting mechanism sends a LAN Manager Accepted frame to the LAN manager when it establishes the reporting link (see "LAN Manager Accepted (X'8608')" on page 14-11). The LAN reporting mechanism also notifies the other LAN managers with which it has reporting links that a new reporting link has been established by sending a New Reporting Link Established frame (see "New Reporting Link Established Notification (X'860B')" on page 14-21). Using these notifications, each LAN manager that has a reporting link with the LAN reporting mechanism can maintain information about other LAN managers in the local area network.

Rejecting the Reporting Link

If the LAN reporting mechanism rejects the request to establish a reporting link, it sends a LAN Manager Rejected frame to the requesting LAN manager (see "LAN Manager Rejected (X'8609')" on page 14-12). The LAN reporting mechanism also notifies the other LAN managers (with which it has reporting links) of the rejection by sending a Report LAN Manager Rejected frame (see "Report LAN Manager Rejection (X'860A')" on page 14-22). This notification allows LAN managers with reporting links to the LAN reporting mechanism to detect potential attempts to breach security in the local area network.

LAN Manager Accepted (X'8608')

This is the first frame sent by the LAN reporting mechanism to a LAN manager after a reporting link is established.

```
LAN Manager Accepted
   Correlator
   Number of Alternate LAN Managers
   Reporting-Link Information
      Reporting-Link Identifier
      Port Information
         Port Identifier
         Ring Number
         Reporting Function Classes
         Available Port Function Classes
         Active Port Function Classes
         Enabled Functional Addresses
```

Figure 14-5. LAN Manager Accepted Frame

Correlator Subvector

The LAN manager uses the value of this subvector to correlate a response with the corresponding Set Reporting Point frame. For a frame sequence to be valid, the correlator subvector value in the LAN Manager Accepted frame must match the correlator subvector value that was sent in the Set Reporting Point frame.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "LAN Manager Accepted (X'8608') Conditions of Presence" on page F-21.

Reporting-Link Information Subvector

The information contained in this subvector pertains to the reporting link just established (over which this frame is sent).

Number of Alternate LAN Managers Subvector

The value of this subvector can be used by the LAN manager to determine the role it plays (controlling, alternate, or observing).

LAN Manager Rejected (X'8609')

This frame is sent to the LAN manager by the LAN reporting mechanism if a problem is detected with the Set Reporting Point frame and a reporting link cannot be established (for example, the frame contains an incorrect key or the requested reporting link is in use). This frame is sent using connectionless LLC service (see Chapter 9, "Connectionless Service") to the X'F4' LSAP in the requesting LAN manager.

```
LAN Manager Rejected
   Correlator
   Rejection Code
   Number of Alternate LAN Managers
   Reporting-Link Information
     Reporting-Link Identifier
     Port Information
       Port Identifier
       Ring Number
       Reporting Function Classes
       Available Port Function Classes
       Active Port Function Classes
       Enabled Functional Addresses
```

Figure 14-6. Report LAN Manager Rejected Frame

Correlator Subvector

This subvector allows a LAN manager to correlate a LAN Manager Rejected frame with the corresponding Set Reporting Point frame. For a frame sequence to be valid, the correlator subvector value in the LAN Manager Rejected frame must match the correlator subvector value that was sent in the Set Reporting Point frame.

Rejection Code Subvector

This subvector's value indicates the reason for the rejection:

X'0001' – Invalid key
X'0002' – Reporting link already in use
X'0003' – Invalid reporting-link identifier
X'0004' – Unable to establish connection
X'0005' – Route traverses this node (this node is a bridge)

The value X'0005' indicates that the route specified in the Set Reporting Point frame traverses the node in which the LAN reporting mechanism is present. This situation (which only occurs if the LAN reporting mechanism is located in a MAC-layer bridge) is undesirable, since the reporting link being established would traverse the ring being reported on (and possibly failing). To assure that the LAN manager receives notifications about failing rings, the route specified in the Set Reporting Point frame must not traverse the LAN reporting mechanism's node.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "LAN Manager Rejected (X'8609') Conditions of Presence" on page F-21.

Set Reporting Point Error (X'8611')

This frame is sent to a LAN manager by a LAN reporting mechanism if a syntactic or semantic error is detected in a Set Reporting Point frame sent by the LAN manager. It is a UI frame (see Chapter 9, "Connectionless Service") and is sent to the X'F4' SAP of the requesting LAN manager.

```
Set Reporting Point Error
   Correlator
   Reason Code
   Error Code
   Error Offset
   LRM Version Level
```

Figure 14-7. Set Reporting Point Error Frame

Correlator Subvector

This subvector allows a LAN manager to correlate a Set Reporting Point Error frame with the corresponding Set Reporting point frame. For a frame sequence to be valid, the correlator subvector value in the Set Reporting Point Error frame must match the value that was sent in the Set Reporting Point frame.

Reason Code

This subvector is a 1-byte value that indicates the type of error that occurred. Valid values for the Reason Code subvector are:

- X'00' — An error in the Set Reporting Point frame was detected while executing the parsing algorithm (see "Parsing Algorithm for Request Status Frames" on page E-2).
- X'01' — The length specified in the set reporting point major vector length field does not equal the received frame length.

Error Code and Error Offset Subvectors

If the value of the reason code subvector is X'00', the error code and error offset subvectors further describe the type and position of the error.

The error code subvector contains a 2-byte diagnostic code that describes the reason that an error was detected in the Set Reporting Point frame. This code refers to the *first* error detected. The valid values for syntax errors are shown in Figure E-2 on page E-3.

The error offset subvector indicates the offset (in numbers of bytes) where the *first* error was detected. The offset is a 2-byte integer and is counted from the first byte of the major vector identifier.

If the error code indicates that no correlator subvector was present in the set reporting point frame, then the value of the correlator subvector is X'000000000'.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Set Reporting Point Error (X'8611') Conditions of Presence" on page F-23.

Close Reporting Link (X'860E')

This frame is sent by a LAN manager to request that its reporting link with a LAN reporting mechanism be terminated.

Subvectors

This frame contains no subvectors.

Conditions of Presence

The conditions of presence for this frame's subvector are shown in "Close Reporting Link (X'860E') Conditions of Presence" on page F-22.

Set LRM Parameters Frame (X'8603')

A controlling LAN manager may specify values of operational parameters in LRM using the "Set LRM Parameters" frame. The information contained in the LRM Parameters frame is shown in Figure 14-8.

```
Set LRM Parameters
   Correlator
   Port Identifier
   Ring Number
   Number of Alternate LAN Managers
   Reporting-Link Information
     Reporting-Link Identifier
     New Key
     Port Information
       Port Identifier
       Reporting Function Classes
       Enabled Functional Addresses
   Temporary/Permanent
```

Figure 14-8. Set LRM Parameters Frame

Correlator Subvector

This subvector allows the LAN manager setting the parameters to correlate this set command with the response returned by LRM.

Conditions of Presence

The conditions of presence of the subvectors of the Set LRM Parameters major vector are described in "Set LRM Parameters (X'8603') Conditions of Presence" on page F-19.

LRM Parameters Set Frame (X'8604')

This frame is returned to the LAN manager issuing the request if the Set LRM Parameters frame does not contain syntactic errors and did not encounter any errors during execution of the Set LRM Parameters command. The contents of the LRM Parameters Set frame are shown in Figure 14-9.

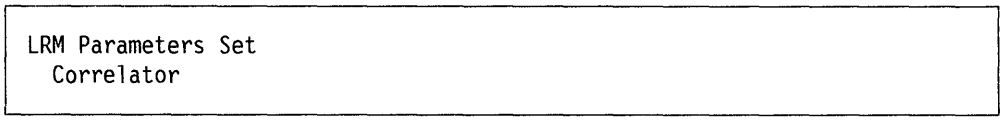


Figure 14-9. LRM Parameters Set Frame

Correlator Subvector

The correlator subvector has the same value as the correlator subvector in the Set LRM Parameters frame for which this is a response.

Conditions of Presence

The conditions of presence of the subvectors of the Set LRM Parameters major vector are described in "LRM Parameters Set(X'8604') Conditions of Presence" on page F-19.

LRM Error Frame (X'8605')

If a syntax error is detected in either the Set LRM Parameters or Request LRM Status frame, the LRM Error frame is returned to the LAN manager which issued the request.

This frame is also sent to the requesting LAN manager if an error is encountered during the execution of the command. The contents of the LRM Error frame are shown in Figure 14-10.

```
LRM Error
  Correlator
  Error Code
  Error Offset
  LRM Version Level
```

Figure 14-10. LRM Error Frame

Correlator Subvector

This subvector has the same value as the Correlator subvector in the Set LRM Parameters frame to which this is a response. The Correlator subvector is present to allow the LAN manager trying to set or request the values of LRM parameters to correlate the response with the original request. If the Correlator subvector was not present in the Request LRM Status or Set LRM Parameters, an error code of "Unrecognized subvector" is returned. The value of the Correlator subvector in the LRM Error frame is X'00000000'.

Error Code Subvector

This subvector contains a 2-byte diagnostic code describing the reason that an error was detected in the corresponding Set LRM Parameters frame. This code refers to the *first* error detected. If the error was syntactic in nature, the valid values for this subvector are listed in Figure E-2 on page E-3. Otherwise, if the error was encountered while executing the command, the Error Code subvector has a value of X'000C'.

Error Offset Subvector

This subvector has a value equal to the offset (in bytes) in the corresponding Set LRM Parameters frame in which the first error was detected. This offset is 2 bytes long and is counted from the first byte of the major-vector identifier.

LRM Version Level Subvector

This subvector may be used by the LAN manager to determine the reason for the error.

Conditions of Presence

The conditions of presence of the subvectors of the LRM Error frame are described in "LRM Error (X'8605') Conditions of Presence" on page F-19.

LRM Parameters Changed Notification Frame (X'8606')

This frame is sent by a LAN reporting mechanism to notify all LAN managers with which it has reporting links that its operational parameters have been changed. This notification is sent in the LRM Parameters Changed Notification frame. The LRM Parameters Changed Notification frame is *not* sent to the LAN manager that caused the values to change. The contents of the LRM Parameters Changed Notification frame are shown in Figure 14-11.

```
LRM Parameters Changed Notification
  Port Identifier
  Ring Number
  Number of Alternate LAN Managers
  Reporting-Link Information
    Reporting-Link Identifier
    New Key
    Port Information
      Port Identifier
      Reporting Function Classes
      Enabled Functional Addresses
```

Figure 14-11. LRM Parameters Changed Notification Frame

Correlator Subvector

This subvector is not present in the LRM Parameters Changed Notification.

This is the only difference from the information in the Set Parameters frame that triggered this frame to be sent.

Conditions of Presence

The conditions of presence of the subvectors of the LRM Parameters Changed are described in "LRM Parameters Changed(X'8606') Conditions of Presence" on page F-20.

Report LAN Manager Control Shift (X'860C')

This frame is sent by the LAN reporting mechanism to all LAN managers with which it has reporting links when a new LAN manager becomes the controlling LAN manager.

This frame is also sent when control of the LAN reporting mechanism is lost; that is, no LAN manager has a controlling reporting link with the LAN reporting mechanism.

```
Report LAN Manager Control Shift
   Status Code
   Reason Code
   Number of Alternate LAN Managers
   Reporting-Link Information
      Reporting-Link Identifier
      LAN Manager Address
```

Figure 14-12. Report LAN Manager Control Shift Frame

Status Code Subvector

This subvector has a 2-byte value field that contains one of the following values:

- X'0000' – Control lost (no new controlling LAN manager)
- X'0001' – New controlling LAN manager

Reason Code Subvector

This subvector specifies whether control was shifted as a result of the controlling LAN manager sending a Close Reporting Link frame or because an anomaly in the LAN caused the LLC connection to terminate, or because a LAN manager established a reporting link with a higher priority. Valid values for this code are:

- X'0000' – Normal termination (a Close Reporting Link frame was sent)
- X'0001' – Error in the network (LLC connection lost)
- X'0002' – A LAN manager has assumed control from an alternate LAN manager

Number of Alternate LAN Managers Subvector

This subvector is present to inform the other LAN managers that their roles may have changed. For example, an alternate LAN manager may have become an observing LAN manager if the new controlling LAN manager changed the value of the number of alternate LAN Managers parameter with the Set Reporting Point frame.

Reporting Link Information Subvector

The information in this subvector frame pertains to the *new* controlling LAN manager and its reporting link with the LAN reporting mechanism.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report LAN Manager Control Shift (X'860C') Conditions of Presence" on page F-22.

New Reporting Link Established Notification (X'860B')

This frame is sent by the LAN reporting mechanism to all LAN managers with which it has reporting links, when a new reporting link is established with the LAN reporting mechanism.

This frame is not sent to the LAN manager that set the new reporting link (a LAN Manager Accepted frame is sent to this LAN manager).

```
New Reporting Link Established Notification
   Reporting-Link Information
      Reporting-Link Identifier
      LAN Manager Address
```

Figure 14-13. New Reporting Link Established Notification Frame

Reporting-Link Information Subvector

This subvector contains information pertaining to the new reporting link.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "New Reporting Link Established Notification (X'860B') Conditions of Presence" on page F-22.

Report LAN Manager Rejection (X'860A')

This frame is sent by a LAN reporting mechanism when it rejects a request by a LAN manager to establish a new reporting link with that LAN manager. It is sent to all LAN managers with reporting links to LAN reporting mechanisms that have requested LRM notifications (in the reporting function class subvector).

```
Report LAN Manager Rejection
   Rejection Code
   LAN Manager Address
```

Figure 14-14. Report LAN Manager Rejection Frame

Rejection Code Subvector

This subvector indicates the reason that a request to establish a reporting link was rejected. The value for this subvector is the same as the value of the rejection code subvector in the corresponding LAN Manager Rejected frame (see "LAN Manager Rejected (X'8609')" on page 14-12), which was sent to the LAN manager that was denied the reporting link.

LAN Manager Address Subvector

This subvector contains the MAC address of the LAN manager that had a request for a reporting link denied.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report LAN Manager Rejection (X'860A') Conditions of Presence" on page F-21.

Report LRM Control Breach Attempt (X'860D')

This frame is sent by a LAN reporting mechanism to all the LAN managers (with which it has reporting links) if any LAN manager (other than the controlling LAN manager) attempts to set the values of the operational parameters in a LAN reporting mechanism or a collocated server (or ring station through a collocated server).

When a LAN reporting mechanism routes a frame to a management server, it indicates the type of LAN manager (controlling or not) that originated the frame. The server notifies the LAN reporting mechanism (using the protocol boundary function) if a manager other than the controlling LAN manager has attempted to set its parameters. If the LAN reporting mechanism receives such an indication, it constructs and sends the Report LRM Control Breach Attempt frame.

```
Report LRM Control Breach Attempt
   Reporting-Link Information
      Reporting-Link Identifier
      LAN Manager Address
```

Figure 14-15. Report LRM Control Breach Attempt Frame

Reporting Link Information Subvector

The information contained in this subvector pertains to the LAN manager that unsuccessfully attempted to set the values of operational parameters in a LAN reporting mechanism or a collocated server (or ring station).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report LRM Control Breach Attempt (X'860D') Conditions of Presence" on page F-22.

Invalid Request (X'8610')

This frame is sent by the LAN reporting mechanism to a LAN manager when it receives a frame from that LAN manager to be routed to a collocated server that is not active.

```
Invalid Request
   Correlator
   Reason Code
   Number of Alternate LAN Managers
   Reporting-Link Information
      Reporting-Link Identifier
      Port Information
         Port Identifier
         Ring Number
         Reporting Function Classes
         Available Port Function Classes
         Active Port Function Classes
         Enabled Functional Addresses
```

Figure 14-16. Invalid Request Frame

Correlator Subvector

This subvector allows a LAN manager to correlate this frame with the frame it sent to the inactive management server.

Reason Code Subvector

This subvector indicates the reason the frame was sent. Valid values for this subvector are:

- X'0000' – The server is not available.
- X'0001' – The server is available but not active.
- X'0002' – The server's state does not allow for reception of this frame (for example, receiving a Set Reporting Point frame over an existing reporting link).

Reporting-Link Information Subvector

The information contained in this subvector pertains to the reporting-link over which the invalid request was received. The reporting-link information subvector returned includes the reporting-link identifier and port information for each MAC instance used by this LAN reporting mechanism. The port information is included to inform the LAN manager about the ring number, the reporting function classes, the available port function classes, and the active port function classes parameters corresponding to each port identifier.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Invalid Request (X'8610') Conditions of Presence" on page F-23.

LRM Terminating (X'860F')

This frame is sent by a LAN reporting mechanism to LAN managers if a local operator terminates the program or if the LAN reporting mechanism detects an internal error that causes it to terminate operations.

On receiving the LRM Terminating frame, the LAN managers should assure that they still have a means to receive information about the ring on which the LAN reporting mechanism and its collocated servers reside and send a Close Reporting Link frame to the LAN reporting mechanism. If the LAN reporting mechanism has no active reporting links (or after a predetermined time), it terminates operation.

Figure 14-17. LRM Terminating Frame

Reason Code Subvector

This subvector indicates the reason that LRM is terminating and has a 1-byte value field containing one of the following values:

- X'00' – Operator-terminated LAN reporting mechanism
- X'01' – Ring station error
- X'02' – Remove command received and LAN reporting mechanism ring station removed.
- X'03' – Internal hardware or software error
- X'04' – Remote bridge telecommunications line failure.

Conditions of Presence

The conditions of presence for this frame's subvector are shown in "LRM Terminating (X'860F') Conditions of Presence" on page F-22.

Report Parsing Error (X'8600')

This frame is sent by the LAN reporting mechanism if it receives a frame in which it does not recognize the major vector identifier. It is sent to the LAN manager from which the unparsable frame was received.

```
Report Parsing Error
   Major Vector Identifier of Error Frame
   LRM Version Level
   Error Code
```

Figure 14-18. Report Parsing Error Frame

Major Vector Identifier Subvector

This subvector contains as its value the major vector identifier (the third and fourth bytes of the frame's management information field) that the LAN reporting mechanism does not recognize (see "LAN Management Frame Format" on page 13-3).

Error Code Subvector

This subvector contains a 1-byte value field that specifies the error type that caused the frame to be sent. The following values have been defined for the error code subvector:

- X'00' – Major vector missing (frame too short)
- X'01' – Unrecognized major vector identifier
- X'02' – Major vector length not equal to actual length of received information field.

If the major vector is missing the value of the major vector identifier of error frame subvector contains the information after the second byte of the exchange-specific information field (if any).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report LRM Parsing Error (X'8600') Conditions of Presence" on page F-18.

LRM Congestion Frame (X'8612')

This frame is sent by the LAN reporting mechanism to a LAN manager (through the LAN reporting mechanism) when its buffers are congested and it has discarded data that should have been sent to the LAN manager. If the remote LAN manager is congested, it may request that the LAN reporting mechanism control the flow of traffic it sends to the LAN manager (see "Procedures for Information Transfer" on page 11-19). In this case, notifications and responses may congest the LAN reporting mechanism's queues on a particular reporting link and data may be discarded. Once a piece of data is discarded, the LRM Congestion frame is queued for transmission to the LAN manager to which the discarded data should have been sent.

Subvectors

This frame contains no subvectors.

Conditions of Presence

The conditions of presence for this frame's subvector are shown in "LRM Congestion (X'8612') Conditions of Presence" on page F-23.

Management Servers Present Frame (X'8613')

This frame is sent by the LAN reporting mechanism to notify all LAN managers that the station containing the management servers is present in the LAN. To improve the chances that it will be successfully received by all the LAN managers in the LAN, this frame is sent multiple times with a pre-determined time interval between transmissions.

```
Management Servers Present
   LRM Version Level
   Reporting-Link Information
      Port Information
         Port Identifier
         Ring Number
```

Figure 14-19. Management Servers Present Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Management Servers Present (X'8613') Conditions of Presence" on page F-23.

Chapter 15. Ring Error Monitor

The ring error monitor (REM) observes, collects, and analyzes hard-error and soft-error reports sent by ring stations on a single ring and assists in fault isolation and correction. The ring error monitor functional address on the ring is the destination address for all soft-error reports (Report Soft Error MAC frames) generated by ring stations. Hard-error reports (Beacon MAC frames) are sent to the all-stations MAC address and examined by the ring error monitor hard-error analysis program.

Ring Error Monitor Functions

The following sections describe the ring error monitor functions.

Hard-Error Processing Function

Hard errors are detected, isolated, and bypassed through the use of a Beacon MAC frame. This frame is generated by any ring station that detects a hard-error condition and is addressed to all the other stations on the ring. The contents of Beacon MAC frames on the ring are made available to the ring error monitor hard-error processing function by the ring station. This information includes:

- The address of the Beacon MAC frame transmitter
- The address of the nearest active upstream neighbor (NAUN) of the Beacon MAC frame transmitter
- The value of the beacon type subvector in the Beacon MAC frame (see "Beacon Type, X'01'" on page 5-17)
- The value of the physical location subvector in the Beacon MAC frame (see "Physical Location, X'0B'" on page 5-21).

Beacon Fault Domain

When a station receives a Beacon MAC frame indicating that it is the upstream station of the fault domain, the station leaves the ring and performs a test on its internal logic and its lobe. If this test does not detect an error within the station or on its lobe, the station re-attaches to the ring.

If the error condition remains in the ring for a predefined time (long enough for the NAUN to perform the self-test and return to the ring), the Beacon MAC frame transmitter removes itself from the ring and performs a test on its internal logic and lobe. Again, if this test does not detect an error within the station or on its lobe, the station re-attaches to the ring.

The two (adjacent) stations involved in this process and the media between them are collectively known as the beacon fault domain.

"Hard-Error Detection and Reporting" on page 3-30 further describes the actions taken by ring stations when they detect hard-error conditions and receive reports of hard-error conditions from another station.

Hard-Error Isolation Function

The ring error monitor hard-error processing function monitors the Beacon MAC frames sent on the ring and uses the information contained in those frames to isolate the location of the error. The ring error monitor can also detect whether fault domain stations are able to automatically bypass the fault by leaving the ring or manual procedures must be instituted to recover the ring.

Once the ring error monitor detects a beaconing condition (Beacon MAC frames are observed on its ring), it starts the permanent-error detection timer. If the beaconing condition does not cease before the permanent-error detection timer expires, the ring error monitor notifies LAN managers (through a LAN reporting mechanism) that an error which cannot be automatically recovered from exists on the ring. When the ring recovers (through some manual recovery process), the ring error monitor sends another notification to LAN managers.

If the beaconing condition ceases before the permanent-error detection timer expires, the ring error monitor may query each station identified in the fault domain indicated in the last received Beacon MAC frame (see "Beacon MAC Frame, X'02'" on page 5-10) by issuing each station a Request Ring Station Address MAC frame (see "Request Ring Station Address MAC Frame, X'0E'" on page 5-13). If both stations respond to the queries, the ring error monitor notifies LAN managers (through a LAN reporting mechanism) that a temporary beaconing condition existed on the ring. If one or both stations do not respond to the queries after a predetermined number of retries, the ring error monitor notifies LAN managers (through a LAN reporting mechanism) that there was a temporary beaconing condition on the ring and that one or both stations left the ring in order to bypass the fault. The address(es) of the station(s) that left the ring are included in the notification to LAN managers.

The querying of stations once a temporary beaconing condition has been detected is an optional function of the ring error monitor. If the ring error monitor does not support this function, it simply notifies LAN managers when it detects either permanent or temporary beaconing conditions on its ring and does not include information about the recovery of temporary beaconing conditions.

Soft-Error Processing Function

This function determines if a non-random or excessive soft-error condition is occurring on the ring to which the ring error monitor is attached. If possible, the ring error monitor isolates the most probable source of the soft errors to a single fault domain, where fault domain consists of a ring station, its nearest active upstream neighbor (NAUN), and the media between them.

Soft-Error Reporting Function

The ring error monitor performs error detection and isolation by analyzing Report Soft Error MAC frames that are periodically sent by ring stations experiencing errors (see "Soft-Error Detection and Reporting" on page 3-29). These MAC frames are sent to the functional address assigned to the ring error monitor (see "Functional Addresses" on page 3-10).

The ring error monitor analyzes the Report Soft Error MAC frames as they arrive and determines if soft errors are occurring at a rate that significantly degrades the performance of the ring. When the ring error monitor detects such a condition, it

notifies the LAN manager and (when possible) provides data indicating the probable source of the error.

The error counters included in a Report Soft Error MAC frame (sent by a station) are divided into two types: isolating and non-isolating, which correspond to the type of error being reported. The source of an isolating error can be isolated to a fault domain; the source of non-isolating errors cannot be isolated to a domain smaller than the ring.

Non-Isolating Soft-Error Processing Function

Non-isolating errors can be isolated only to the ring on which they occur. A counter is kept for each type of non-isolating error that is reported to the ring error monitor in Report Soft Error MAC frames. Some internal error counters are also treated as non-isolating errors. If one of these counters exceeds a threshold value, a notification is sent (through the LAN reporting mechanism) to the network manager and the ring error monitor's counter reset. No other analysis is performed by the ring error monitor for non-isolating errors.

Isolating Soft-Error Processing Function

When a Report Soft Error MAC frame is received by the ring error monitor, the information contained in the isolating error counts is used to accumulate *weight* against two stations on the ring: the reporting station and its NAUN. The weight accumulated for a particular station is an indication of the likelihood that the station is causing excessive soft errors on the ring. Stations with nonzero weights associated with them, along with their weights, are stored in a table by the ring error monitor.

The ring error monitor maintains two weight thresholds: the impending-soft-error threshold and the excessive-soft-error threshold. The impending-soft-error threshold is lower than the excessive-soft-error threshold.

Exceeding the impending-soft-error threshold (also known as the *pre-weight exceeded* condition) is used as an indication of a potential problem on the token ring caused by excessive soft errors. Token-ring performance may soon be degraded once this threshold is exceeded.

Exceeding the excessive-soft-error threshold (also known as the *weight exceeded* condition) is used as an indication of a serious soft-error problem on the token ring that is significantly affecting its performance.

When either soft-error threshold is reached, the ring error monitor sends a report to LAN managers. These frames indicate the fault domain responsible for the error condition.

Since even random errors can cause the accumulated error weight for a station to exceed the threshold eventually, a fixed value is subtracted from the weight for a station at periodic time intervals. As a result of this periodic decrementing of the accumulated weights, only the stations continuously accumulating weight at a rate faster than the decrement rate will have weight accumulations that grow with time.

The ring error monitor also dynamically adjusts the threshold and decrement rate to normalize the algorithm for each ring's environment.

If the station with the highest weight in the ring error monitor's table is in a weight exceeded condition, but is no longer accumulating error weight at a sufficient rate

to continue reporting the error to the LAN manager, the ring error monitor notifies the LAN manager that the error condition is subsiding.

Intensive Mode Reporting Function

Intensive mode reporting is a facility that enables a LAN manager to request a ring error monitor to forward selected Report Soft Error MAC frames it receives to the LAN manager. There are two types of intensive mode operations: ring-intensive mode and auto-intensive mode. Each of the modes can be enabled to generate reports to the LAN manager under certain conditions.

Ring-intensive mode operation enables the ring error monitor to generate reports to the LAN manager for all Report Soft Error MAC frames that contain at least one error of a specified type of soft error (isolating or non-isolating) on its ring.

Auto-intensive mode operation causes the ring error monitor to automatically forward the contents of Report Soft Error MAC frames received from adapters in fault domains that have exceeded one of the error thresholds. The ring error monitor sends reports for the following stations (if auto-intensive mode is enabled):

- The station that caused a fault domain to be in the notification condition (pre-weight exceeded or weight exceeded).
- The nearest active upstream neighbor (NAUN) of the station that caused a fault domain to enter the notification condition.
- The nearest active downstream neighbor (NADN) of the station that caused a fault domain to enter the notification condition.

Receiver-Congestion Error Reporting Function

The ring error monitor also isolates stations that cannot receive a significant number of frames from the ring because their input buffers are full. A counter indicating the number of frames for which the station recognized its own address but could not copy the frame is sent in the Report Soft Error MAC frames received by the ring error monitor. This error is not considered an isolating error, because the problem may not actually be in the station receiving the data from the ring, but in one or more stations *sending* an excessive number of frames to the congested adapter. Nevertheless, the identification of stations experiencing congestion is useful to the LAN manager for problem determination.

The ring error monitor maintains a table, called the *receiver congestion table* (similar to the isolating table used in isolating soft-error analysis), for accumulating data about congested stations on its ring. Each Report Soft Error MAC frame is checked to determine if its receiver-congestion error counter has a non-zero value. If so, a *receiver-congestion state* exists for the reporting station. The number of occurrences of the receiver-congestion state is used to accumulate a receiver-congestion weight against the reporting station.

The receiver-congestion weight for the reporting station is incremented for each occurrence of the receiver-congestion state. If no entry exists in the receiver-congestion table for a station in a receiver-congestion state and there is room in the table, a new entry is created for the station. If the receiver-congestion table is full, the entry cannot be created and a non-isolating error counter is incremented (see "Non-Isolating Soft-Error Processing Function" on page 15-3). If the receiver-congestion weight for a table entry exceeds a predetermined threshold value and the ring error monitor is enabled to send receiver congestion frames, then a Receiver Congestion Notification frame is sent immediately to LAN managers mon-

itoring the ring error monitor (see "Receiver Congestion Notification (X'810E')" on page 15-29).

If an entry has been in the table for a predetermined amount of time and its weight has not exceeded the threshold causing a Receiver Congestion Notification frame to be sent to LAN managers, it is removed (aged) from the table. This aging process prevents stations infrequently experiencing congestion from causing notifications to be sent to LAN managers.

If an entry in the receiver-congestion table has exceeded the threshold causing a Receiver Congestion Notification frame to be sent to LAN managers, but has not had its receiver-congestion weight incremented during the last notification interval, a Receiver-Congestion Ended frame is sent to LAN managers monitoring the ring error monitor at the end of the notification interval (see "Receiver Congestion Ended (X'810F')" on page 15-30). If multiple entries in receiver-congestion table meet these conditions, they are all reported in the same Receiver-Congestion Ended frame.

Status Request Function

The ring error monitor accepts and responds to requests for status from a LAN manager.

Set Parameters Function

The ring error monitor accepts and executes requests to set its parameters from the controlling LAN manager.

Operating Environment Constraints

The ring error monitor operating environment has the following constraints:

- One or more ring error monitors can reside on a ring. If no ring error monitors are on a particular ring, it is not possible to monitor errors for that ring.
- One or more ring error monitors can report to a LAN manager. The LAN manager also sets reporting function classes in the LAN reporting mechanism (see Chapter 14, "LAN Reporting Mechanism") to selectively receive the ring error monitor reports and notifications.

  Note: The ring error monitor functional address must be enabled for the ring error monitor to receive Report Soft Error MAC frames.
- A ring error monitor's parameters can be set only by the controlling LAN manager. This control is enforced by the LAN reporting mechanism and the ring error monitor together. (See "Control Function" on page 14-1.)

Communication Functions

Frames sent between a LAN manager and a ring error monitor are described later in this chapter. These frames are sent using the reporting link established by a LAN manager with the LAN reporting mechanism. (See Chapter 14, "LAN Reporting Mechanism.")

Data Structures

The following data structures are sent and received by the ring error monitor.

```
Correlator
Ring Number
Notification Enable
Non-Isolating Notification
   Lost Frames
   Receiver-Congestion Errors
   Frame-Copied Errors
   Frequency Errors
   Token Errors
   Reserved
   Table-Full Errors
   Minimum-Decrement Errors
   Receiver-Congestion Table-Full Errors
Intensive Mode Data
   Ring-Intensive Mode Mask
   Auto-Intensive Mode Mask
REM Version Level
Isolating Table
REM Isolating Status
   Decrement Interval
   Number Of Entries
   Fault Domain
      Receiver
         Flags
         Error Weight
         Station Address
         NAUN Address
         Physical Location
      Transmitter
         Flags
         Error Weight
         Station Address
         NAUN Address
         Physical Location
Ring Status
      Ring State
      Beacon Data
         Beacon Type
         Beaconing Station Address
         NAUN Address
         Physical Location
Removed Station Address
```

Figure 15-1. Ring Error Monitor Data Structures

Correlator Subvector

This subvector has a 2-byte value field that is used by the LAN manager to associate responses with requests generated by the LAN manager.

Ring Number Subvector

This subvector has a 2-byte integer value field that contains the number of the ring being monitored by the ring error monitor. If more than one ring is being monitored, separate information is maintained for each ring.

Notification Enable Subvector

This subvector has a 2-byte value field that contains two bit-significant (byte-length) fields indicating the reporting modes for which the ring error monitor is currently enabled. These two fields are defined as follows:

- Enabled Notifications: This is a 1-byte bit-significant field that specifies the notifications the ring error monitor will generate. The bits are defined as follows:
  - Bit 0 – Weight exceeded: If bit 0 = B'1', the ring error monitor is enabled to notify LAN managers (through the LAN reporting mechanism) when the ring error monitor detects a weight-exceeded condition (the excessive-soft-error threshold has been exceeded by a fault domain).
  - Bit 1 – Pre-weight exceeded: If bit 1 = B'1', the ring error monitor is enabled to notify LAN managers (through the LAN reporting mechanism) when the ring error monitor detects a pre-weight exceeded condition (the impending-soft-error threshold has been exceeded by a station).
  - Bits 2,3 – Reserved for future use by IBM.
  - Bit 4 – Non-Isolating Threshold Exceeded: If bit 4 = B'1', the ring error monitor is enabled to notify all the LAN managers (through the LAN reporting mechanism) when the ring error monitor detects that one of the non-isolating error counters that it maintains has exceeded its respective threshold value.
  - Bit 5 – Forward Frames: If bit 5 = B'1', the ring error monitor is enabled to forward (through the LAN reporting mechanism) the contents of Report Neighbor Notification MAC frames and Report Monitor Error MAC frames that it receives to LAN managers with reporting links to the LAN reporting mechanism.
  - Bits 6,7 – Reserved for future use by IBM.
- Enable Intensive: This is a 1-byte bit-significant field that specifies the intensive modes that are enabled. The bits are defined as follows:
  - Bit 0 – Reserved for future use by IBM.
  - Bit 1 – Ring-Intensive Error Reporting: If bit 1 = B'1', the ring error monitor is enabled to generate intensive mode reports for Report Soft Error MAC frames that originated on the specified ring and contain error counts of the type enabled for ring-intensive mode reporting. The types of errors that ring-intensive mode is enabled for are specified in the ring-intensive mode mask in the intensive mode data parameter (see "Intensive Mode Data Subvector" on page 15-10).
  - Bit 2 – Auto-Intensive Error Reporting: If bit 2 = B'1', the ring error monitor is enabled to generate intensive mode reports for Report Soft Error MAC frames that originated from stations that have been selected for intensive mode because that station or one of its adjacent neighbors is in a pre-weight exceeded condition and contains error counts of the type enabled for auto-intensive mode. The types of errors for which auto-intensive mode reporting is enabled are specified in the auto-intensive mode mask in the intensive mode data parameter (see "Intensive Mode Data Subvector" on page 15-10).
  - Bits 3-7 – Reserved for future use by IBM.

Non-Isolating Notification Subvector

This subvector has a value field containing nested subvectors that contain information about the non-isolating error counts reported to the ring error monitor (see "Non-Isolating Error Counts, X'2E'" on page 5-20). It includes the accumulated error counts and the error count threshold values. The non-isolating notification subvector contains the following nested subvectors.

Lost Frames Subvector

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Lost-Frame Count Field — This field contains the number of lost-frame errors reported to the ring error monitor since this count last exceeded its threshold or was reset.
- Lost Frame Threshold Field — This field contains the number of the lost frame errors that can be reported to the ring error monitor before the ring error monitor will reset the count to zero and send a Non-Isolating Notification frame to the LAN manager. A value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of non-isolating error.

Receiver Congestion Errors Subvector

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Receiver Congestion Count Field — This field contains the number of receiver-congestion errors reported to the ring error monitor since this count last exceeded its threshold or was reset.
- Receiver Congestion Threshold Field — This field contains the number of the receiver-congestion non-isolating errors that can be reported to the ring error monitor before it will reset the count to 0 and send a Non-Isolating Notification frame to the LAN manager. A value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of non-isolating error.

Frame-Copied Error Subvector

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Frame-Copied Error Count field — This field contains the number of frame-copied errors reported to the ring error monitor since this count last exceeded its threshold or was reset.
- Frame-Copied Threshold field — This field contains the number of the frame-copied that can be reported to the ring error monitor before it resets the count and sends a Non-Isolating Notification frame to the LAN manager. A value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of non-isolating error.

Frequency Error Subvector

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Frequency-Error Count field — This field contains the number of frequency errors reported to the ring error monitor since this count last exceeded its threshold or was reset.
- Frequency-Error Threshold field — This field contains the number of the frequency errors that can be reported to the ring error monitor before it resets the count and sends a Non-Isolating Notification frame to the LAN manager. A

value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of non-isolating error.

Token Error Subvector

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Token-Error Count Field – This field contains the number of token errors reported to the ring error monitor since this count last exceeded its threshold or was reset.
- Token-Error Threshold field – This field contains the number of token errors that can be reported to the ring error monitor before it will reset the count and send a Non-Isolating Notification frame to the LAN manager. A value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of non-isolating error.

Reserved Subvector

This subvector has a 2-byte value field that is reserved by IBM for future use.

Table-Full Errors Subvector

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Table-Full Error Count field – This field contains the number of isolating table-full conditions that the ring error monitor has detected since this count last exceeded its threshold or was reset. An isolating table-full condition is detected by the ring error monitor when all entries in the isolating table are in use and a Report Soft Error MAC frame containing isolating error counts is received from a station for which there is no entry in the table.
- Table-Full Error Threshold field – This field contains the number of the isolating table-full conditions that can be detected by the ring error monitor before it will reset the count and send a Non-Isolating Notification frame to the LAN manager. A value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of error.

Minimum-Decrement Errors Subvector

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Minimum-Decrement Error Count Field – This field contains the number of times the ring error monitor has attempted to set its decrement value below the minimum value allowed (by design) since this count last exceeded its threshold or was reset.
- Minimum-Decrement Error Threshold Field – This field contains the number of minimum-decrement error conditions that can be detected by the ring error monitor before it will reset the count and send a Non-Isolating Notification frame to the LAN manager. A value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of error.

Receiver-Congestion Table-Full Errors

This subvector has a 2-byte value field that consists of two 1-byte integer fields.

- Receiver-Congestion Table-Full Error Count Field – This field contains the number of times a receiver-congestion table-full condition was detected by the ring error monitor. A receiver-congestion table-full condition is detected by the ring error monitor when all entries in the receiver-congestion table are in use and a Report Soft Error MAC frame containing a nonzero value for receiver

congestion errors is received from a station for which there is no entry in the table.

- Receiver-Congestion Table-Full Threshold Field — This field contains the number of receiver-congestion table-full conditions that can be detected by the ring error monitor before it will reset the counter and send a Non-Isolating Notification frame to the LAN manager. A value of X'00' in this field disables the ring error monitor from generating Non-Isolating Notification frames for this type of error.

Intensive Mode Data Subvector

This subvector contains a mask that specifies the types of errors that must be present in a Report Soft Error MAC frame for it to be reported as part of the Intensive Mode operation. There are four subvectors contained in this subvector:

- Ring-intensive mode mask
- Auto-intensive mode mask

Ring-Intensive Mode Mask and Auto-Intensive Mode Mask Subvectors

Each mask is defined as follows:

- Each subvector has a 2-byte, bit-significant value field. The first byte is the *isolating error mask*. Bits 0-4 of this field indicate which of the isolating error counters in the soft-error report must be nonzero for an intensive mode report to be generated.

  Bit 0 = Line errors
  Bit 1 = Internal errors
  Bit 2 = Burst errors
  Bit 3 = A/C errors
  Bit 4 = Aborts transmitted

  If the respective bit is set to B'1', the error counter is checked to see if it is nonzero. If so, an intensive mode report is generated and sent through the LAN reporting mechanism to the LAN managers. If the respective bit is set to B'0', then the error counter is logged but does not cause an intensive mode report to be made.

- The second byte is the *non-isolating error mask*. This field performs the same function as the isolating error mask, but refers to the non-isolating errors that are reported in a Report Soft Error MAC frame. Bits 0-4 of this field indicate which of the non-isolating error counters in the soft-error report must be nonzero for an intensive mode report to be generated.

  Bit 0 = Lost frame
  Bit 1 = Receiver congestion
  Bit 2 = Frame copied
  Bit 3 = Frequency error
  Bit 4 = Token error

Note: Intensive Mode is never active for the isolating table-full, minimum decrement reached, and receiver congestion table-full errors.

REM Version Level Subvector

This subvector contains the following information about a ring error monitor.

Discriminator Field

This field consists of a 1-byte integer that indicates the encoding method used for the rest of the REM version level subvector value. Its value is either X'00' (indicating ASCII) or X'01' (indicating EBCDIC).

Common Version Identifier Field

This field consists of the 2-byte numeric version level of the ring error monitor.

Common Release Identifier Field

This field consists of the 2-byte numeric release level of the ring error monitor.

Common Modification Identifier Field

This field consists of the 2-byte numeric modification level of the ring error monitor.

Software Product Program Number Field

This field consists of a 7-byte alphanumeric field that identifies the ring error monitor.

Isolating Table Subvector

This subvector has a variable-length value containing entries for each station for which the ring error monitor has accumulated error weight. The value of this parameter can be used for ring error monitor error diagnosis, but is not useful to the LAN manager as a part of normal operation. The format of this subvector is implementation-dependent.

REM Isolating Status Subvector

This subvector contains information resulting from the ring error monitor's soft isolating error analysis. The information pertains to the fault domain with the highest error weights in the isolating table. The ring error monitor isolating status subvector contains the following nested subvectors.

Decrement Interval Subvector

This subvector has a 1-byte integer value field that contains the time interval (in seconds) between decrements to the weights associated with stations in the isolating table.

Number of Entries Subvector

This subvector has a 1-byte integer value field that contains the number of entries in the isolating table (the isolating table contains an entry for each station with a nonzero error weight associated with it).

Fault Domain Subvector

This subvector contains a receiver subvector and a transmitter subvector that together identify a fault domain. The receiver subvector contains the isolating table entry for the downstream station. The transmitter subvector contains the isolating table entry for the upstream station.

The entry for the station that caused the fault domain to enter a notification condition can be present in either the receiver or the transmitter subvector. Either the

NAUN or the NADN (whichever has a higher error weight associated with it) is also present in the other subvector (the transmitter or the receiver, respectively).

Receiver Subvector This subvector contains the following nested subvectors, containing information maintained in the isolating table for the downstream station of a fault domain.

- Flags – This subvector has a 1-byte, bit-significant subvector that indicates whether this station is in a weight-exceeded or a pre-weight-exceeded condition. The bits of this value field are defined as follows:
  - Bit 0 indicates that the accumulated error weight for this entry in the isolating table has exceeded the impending-soft-error threshold.
  - Bit 1 indicates that the accumulated error weight for this entry in the isolating table has exceeded the excessive soft-error threshold.
- Error Weight – This subvector has a 1-byte integer subvector that contains the value (in the range of 0 through 127) of the calculated error weight associated with this station in the isolating table. The value of error weight is initialized to 0.
- Station Address – This subvector has a 6-byte (bit-string) subvector that contains the MAC address (see "Addresses" on page 3-9) with which the accumulated error weight is associated.
- NAUN Address – This subvector has a 6-byte (bit-string) subvector that contains the MAC address of the NAUN of the station identified in the station address (see "NAUN, X'02'" on page 5-19).
- Physical Location – This subvector has a value field 4 bytes long that contains the assigned physical location of the station identified in the station address (see "Assign Physical Location, X'04'" on page 5-17).

Transmitter Subvector This subvector contains the following nested subvectors which contain information maintained in the isolating table for the upstream station of a fault domain.

- Flags: This subvector has a 1-byte, bit-significant subvector that indicates whether this station is in a weight-exceeded or a pre-weight-exceeded condition. The bits of this value field are defined as follows:
  - Bit 0 indicates that the accumulated error weight for this entry in the isolating table has exceeded the pre-weight-exceeded condition (it has exceeded the impending-soft-error threshold).
  - Bit 1 indicates that the accumulated error weight for this entry in the isolating table has exceeded the weight-exceeded condition (it has exceeded the excessive-soft-error threshold).
- Error Weight: This subvector has a 1-byte integer subvector containing the value (in the range of 0 through 127) of the calculated error weight associated with this station in the isolating table. The value of error weight is initialized to 0.
- Station Address: This subvector has a 6-byte (bit-string) subvector that contains the MAC address (see "Addresses" on page 3-9) with which the accumulated error weight is associated.
- NAUN Address: This subvector has a 6-byte (bit-string) value field that contains the MAC address of the NAUN of the station identified in the station address (see "NAUN, X'02'" on page 5-19).

- Physical Location: This subvector has a value field 4 bytes long that contains the assigned physical location of the station identified in the station address (see "Assign Physical Location, X'04'" on page 5-17).

Ring Status Subvector

This subvector has a complex value containing two subvectors: the ring state subvector and the beacon data subvector.

Ring State Subvector This subvector has a 2-byte bit-significant value field indicating the hard-error processing state of the token ring to which the ring error monitor is attached. The valid values for this subvector follow:

- X'0000' — Normal: No hard-error condition exists on the token ring.
- X'0001' — Temporary beaconing: A temporary beaconing condition was detected and the beacon processing function was preempted by another beacon condition before it could complete its analysis.
- X'0002' — Temporary beaconing: A temporary beaconing condition was detected and this ring error monitor does not implement the function to query the fault domain stations after temporary beaconing (after the self-test timer expires).
- X'0003' — Temporary beaconing: A temporary beaconing condition was detected. The ring error monitor queried the beacon fault domain stations and received responses from both stations, indicating that neither station left the ring as part of the automatic recovery process.
- X'0004' — Temporary beaconing: A temporary beaconing condition was detected. The ring error monitor queried the beacon fault domain stations and detected that one of the stations in the fault domain has left the ring as part of the automatic recovery process.
- X'0005' — Temporary beaconing: A temporary beaconing condition was detected. The ring error monitor queried the beacon fault domain stations and detected that both of the stations in the fault domain had left the ring as part of the automatic recovery process.
- X'0006' — Permanent beaconing: The ring has been in a beaconing condition for longer than the permanent-error detection timer.

Beacon Data Subvector This subvector has a complex value field containing the following four subvectors:

- Beacon Type — This subvector has a value field 2 bytes long that indicates the reason for the beaconing condition on the ring. The value of this subvector is the same as that of the beacon type subvector in the Beacon MAC frame. For a list of valid values for this subvector, see "Beacon Type, X'01'" on page 5-17.
- Beaconing Station Address — This subvector has a 6-byte bit-string value field that contains the MAC address of the station that is transmitting Beacon MAC frames (the downstream station) of the beacon fault domain.
- NAUN Address — This subvector has a 6-byte bit-string value field that contains the MAC address of the nearest active upstream neighbor or the station that is transmitting Beacon MAC frames (the upstream station of the beacon fault domain).
- Physical Location — This subvector has a 4-byte value field containing the value of the physical location subvector in the received Beacon MAC frames. (See "Assign Physical Location, X'04'" on page 5-17.)

Removed Station Address Subvector

This subvector has a 6-byte bit-string value field that contains the MAC address (see "Addresses" on page 3-9) of a station in the beacon fault domain that ring error monitor has determined was removed as part of the automatic-recovery process. (See "Hard-Error Processing Function" on page 15-1.)

Request REM Status (X'8101')

This frame is sent by the LAN manager to request certain status information about a ring error monitor.

```
Request REM Status
   Correlator
   Ring Number
   Notification Enable
   Intensive Mode Data
      Ring-Intensive Mode Mask
      Auto-Intensive Mode Mask
   Isolating Table
   Ring Status
```

Figure 15-2. Request REM Status Frame

Correlator Subvector

This subvector allows the LAN manager to correlate this frame with its response. For a frame sequence to be valid, the correlator subvector value in the response frame must match the value that is sent in the Request Ring Error Monitor Status frame.

Ring Number Subvector

This subvector identifies the ring error monitor information to which the request applies (if the ring error monitor is monitoring more than one ring).

Subvector Length

The length of each atomic subvector except the correlator subvector and ring number subvector is 4 bytes (a 2-byte subvector length field, a 2-byte subvector identifier field, and a zero-length value field).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Request REM Status (X'8101') Conditions of Presence" on page F-2.

Parsing Algorithm

Before executing the Request REM Status, the ring error monitor checks the frame for errors using the algorithm shown in "Parsing Algorithm for Request Status Frames" on page E-2.

Responses

If the Request REM Status frame does not contain any syntactic or semantic errors and no errors occur while executing the request, the LAN reporting mechanism returns the requested information in a Report REM Status frame (see "Report REM Status (X'8102')" on page 15-16).

If an error is detected in the Request REM Status frame, the LAN reporting mechanism returns an REM Error frame (see "REM Error (X'810C')" on page 15-20).

Report REM Status (X'8102')

This frame is sent by the ring error monitor to a requesting LAN manager in response to a Request REM Status (see "Request REM Status (X'8101')" on page 15-15) if no errors are detected in the Request REM Status frame.

```
Report REM Status
   Correlator
   Ring Number
   Notification Enable
   Intensive Mode Data
      Ring-Intensive Mode Mask
      Auto-Intensive Mode Mask
   Isolating Table
   Ring Status
      Ring State
      Beacon Data
         Beacon Type
         Beaconing Station Address
         NAUN of Beaconing Station
         Physical Location of Beaconing Station
```

Figure 15-3. Report REM Status Frame

Correlator Subvector

The LAN manager uses the value of this subvector to correlate this Report REM Status frame with the corresponding Request REM Status frame. The value of the correlator subvector is the same as the correlator value in the associated Request REM Status frame.

Ring Number Subvector

This subvector identifies the ring to which the information contained in this frame applies (if the ring error monitor is monitoring more than one ring).

Other Subvectors

The remainder of this frame consists of the information requested in the Request REM Status command.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report REM Status (X'8102') Conditions of Presence" on page F-2.

Set REM Parameters (X'8103')

This frame is sent by a controlling LAN manager to change the values of operational parameters maintained in a ring error monitor.

```
Set REM Parameters
   Correlator
   Ring Number
   Notification Enable
   Intensive Mode Data
      Ring-Intensive Mode Mask
      Auto-Intensive Mode Mask
   Reset
```

Figure 15-4. Set REM Parameters Frame

Correlator Subvector

This subvector allows a LAN manager to correlate responses it receives with the Set REM Parameters frame that it sent. For a frame sequence to be valid, the correlator subvector value in the response frame must match the value that was sent in the associated Set REM Parameters frame.

Ring Number Subvector

This subvector identifies the information to which this request applies (if the ring error monitor is monitoring more than one ring).

Note: The value of the ring number parameter maintained by the ring error monitor is not changed when the ring number is present in the Set REM Parameters frame. This subvector's value is used only to select the ring error monitor table to which this request applies.

Reset Subvector

If the reset subvector is present in the Set REM Parameters frame, the ring error monitor resets its internal operational parameters. The exact meaning of this subvector is dependent on the ring error monitor implementation.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Set REM Parameters (X'8103') Conditions of Presence" on page F-3.

Parsing Algorithm

Before executing the Set REM Parameters, the LAN reporting mechanism checks it for errors using the parsing algorithm shown in "Parsing Algorithm for Set Parameter Frames" on page E-3.

Responses

If the Set REM Parameters frame does not contain any parsing errors, the ring error monitor sets the parameter values indicated and sends an REM Parameters Set frame to the requesting LAN Manager (see "REM Parameters Set (X'8104')" on page 15-19). The ring error monitor also notifies the other LAN managers (with which it has reporting links) that the values of its operational parameters have been changed by sending a REM Parameters Changed Notification (see "REM Parameters Changed Notification (X'8105')" on page 15-21).

If an error is detected when parsing the Set REM Parameters frame or while executing the command, the LAN reporting mechanism sends a REM Error Frame (see "REM Error (X'810C')" on page 15-20) to the requesting LAN manager.

REM Parameters Set (X'8104')

This frame is sent (through the LAN reporting mechanism) by the ring error monitor to the LAN manager if the associated Set REM Parameters frame does not contain any errors.

```
REM Parameters Set
   Correlator
```

Figure 15-5. REM Parameters Set Frame

Correlator Subvector

This subvector is always present. It allows a LAN manager to correlate this response frame with the associated Set REM Parameters frame.

Conditions of Presence

The conditions of presence for this frame's subvector are shown in "REM Parameters Set (X'8104') Conditions of Presence" on page F-3.

REM Error (X'810C')

This frame is sent (through the LAN reporting mechanism) by the ring error monitor to the requesting LAN manager if an error is detected in a Request REM Status frame (see "Request REM Status (X'8101')" on page 15-15) or a Set REM Parameters frame (see "Set REM Parameters (X'8103')" on page 15-17).

```
REM Error
   Correlator
   Error Code
   Error Offset
   REM Version Level
```

Figure 15-6. REM Error Frame

Correlator Subvector

This subvector has the same value as the correlator subvector in the Request REM Status frame or Set REM Parameters frame to which this frame is a response. The correlator subvector allows the LAN manager to associate the error response with the original request.

Error Code Subvector

This subvector contains a 2-byte diagnostic code that describes the reason that an error was detected in the corresponding Request REM Parameters frame or Set REM Parameters frame. This code refers to the *first* error detected. The valid values for syntax errors are listed in Figure E-1 on page E-2 and Figure E-2 on page E-3 for the responses to the Request REM Status and Set REM Parameters frames, respectively. If the error was detected while executing the command, the subvector value is X'000C'.

Note: If the correlator subvector was not present in the Request REM Status or Set REM Parameters frame, an error code of *unrecognized subvector* (see "Parsing Algorithm for Request Status Frames" on page E-2 and "Parsing Algorithm for Set Parameter Frames" on page E-3) and a correlator value of X'00000000' is returned in the REM Error frame.

Error Offset Subvector

The value of this subvector is the offset (in number of bytes) into the corresponding Request REM Parameters frame or Set REM Parameters frame where the *first* error was detected. The offset is a 2-byte integer and is counted from the first byte of the major vector identifier. (See "Parsing Algorithm for Request Status Frames" on page E-2 and "Parsing Algorithm for Set Parameter Frames" on page E-3 for details.)

REM Version Level Subvector

This subvector can be used by the LAN manager to assist in determining the reason for the error.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "REM Error (X'810C') Conditions of Presence" on page F-6.

REM Parameters Changed Notification (X'8105')

This frame is sent by the ring error monitor (through the LAN reporting mechanism) to LAN managers when the values of the ring error monitor's operational parameters are changed by the controlling LAN manager. This frame is *not* sent to the controlling LAN manager that requested the change.

The information in this frame is identical to the information in the Set REM Parameters frame that caused it to be sent except that the correlator subvector is not present in the REM Parameters Changed Notification frame.

```
REM Parameters Changed
   Ring Number
   Notification Enable
   Intensive Mode Data
      Ring-Intensive Mode Mask
      Auto-Intensive Mode Mask
   Reset
```

Figure 15-7. REM Parameters Changed Frame

Reset Subvector

If this subvector is present, REM has reset its internal operational parameters. The exact meaning of this subvector is dependent on the REM implementation.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "REM Parameters Changed Notification (X'8105') Conditions of Presence" on page F-3.

Pre-Weight-Exceeded Notification (X'8107')

This frame is sent (through the LAN reporting mechanism) by the ring error monitor to LAN managers when the impending-soft-error threshold is exceeded by the entry in the isolating table with the highest error weight and if bit 1 of the notification enable subvector is set to B'1'. (See "Isolating Soft-Error Processing Function" on page 15-3 for details about this condition.)

```
Pre-Weight-Exceeded Notification
   Ring Number
   Notification Enable
   REM Isolating Status
      Decrement Interval
      Number Of Entries
      Fault Domain
         Receiver
            Flags
            Error Weight
            Station Address
            NAUN Address
            Physical Location
         Transmitter
            Flags
            Error Weight
            Station Address
            NAUN Address
            Physical Location
```

Figure 15-8. Pre-Weight-Exceeded Notification Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Pre-Weight Exceeded Notification (X'8107') Conditions of Presence" on page F-4.

Weight-Exceeded Notification (X'8108')

This frame is sent (through the LAN reporting mechanism) by the ring error monitor to LAN managers when the excessive-soft-error threshold is exceeded by an entry in the isolating table and if bit 0 of the notification enable subvector is set to B'1'. (See "Isolating Soft-Error Processing Function" on page 15-3 for details about this condition.) This notification is repeated periodically (at the rate specified in the notification interval in the installation parameters subvector) as long as:

- A station's accumulated error weight exceeds one-half of the excessive-soft-error threshold value, and
- Report Soft Error MAC frames that are accumulating error weight against the station with the highest error weight are being received at a sufficient rate.

```
Weight-Exceeded Notification
   Ring Number
   Notification Enable
   REM Isolating Status
      Decrement Interval
      Number Of Entries
      Fault Domain
         Receiver
            Flags
            Error Weight
            Station Address
            NAUN Address
            Physical Location
         Transmitter
            Flags
            Error Weight
            Station Address
            NAUN Address
            Physical Location
```

Figure 15-9. Weight-Exceeded Notification Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Weight-Exceeded Notification (X'8108') Conditions of Presence" on page F-5.

Error Rate Decaying Notification (X'8106')

This frame is sent (through the LAN reporting mechanism) by the ring error monitor to LAN managers when the station with the highest error weight is in a weight-exceeded condition, but is no longer accumulating error weight at a sufficient rate to continue sending Weight-Exceeded Notification frames. (See "Isolating Soft-Error Processing Function" on page 15-3 for details about this condition.)

This notification is generated a predetermined number of times (specified in the notification decay limit subvector in the installation parameters subvector) while the condition persists. The purpose of this notification is to give a timely indication of a successful recovery action for a soft-error condition on the ring to LAN managers.

```
Error Rate Decaying Notification
   Ring Number
   Notification Enable
   REM Isolating Status
      Decrement Interval
      Number Of Entries
      Fault Domain
         Receiver
            Flags
            Error Weight
            Station Address
            NAUN Address
            Physical Location
         Transmitter
            Flags
            Error Weight
            Station Address
            NAUN Address
            Physical Location
```

Figure 15-10. Error Rate Decaying Notification Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Error Rate Decaying Notification (X'8106') Conditions of Presence" on page F-4.

Non-Isolating Threshold Exceeded Notification (X'8109')

This frame is sent (through the LAN reporting mechanism) by the ring error monitor to LAN managers. It is sent when the total number of a type of non-isolating errors processed by the ring error monitor exceeds the threshold value for that type of non-isolating error (see "Non-Isolating Soft-Error Processing Function" on page 15-3) and when bit 4 of the Notification Enable parameter in the ring error monitor is set to B'1'. When this notification is sent, the error counters that exceeded the thresholds are reset to X'00'.

```
Non-Isolating Threshold Exceeded Notification
   Ring Number
   Notification Enable
   Non-Isolating Notification
      Lost Frames
      Receiver Congestion Errors
      Frame Copied Errors
      Frequency Errors
      Token Errors
      Reserved
      Table-Full Errors
      Minimum-Decrement Errors
      Receiver-Congestion Table-Full Errors
```

Figure 15-11. Non-Isolating Threshold Exceeded Notification Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Non-Isolating Threshold Exceeded (X'8109') Conditions of Presence" on page F-5.

Forward MAC Frame (X'810A')

The ring error monitor receives the following unsolicited reports in the form of MAC frames from the stations on the ring:

- Report Soft Error MAC frame
- Report Neighbor Notification Incomplete MAC frame
- Report Monitor Error MAC frame.

The Report Soft Error MAC frame is used by the ring error monitor process to perform error analysis for the ring. If instructed by the LAN manager (using a Set REM Parameters frame to set intensive modes), the ring error monitor can also forward (through the LAN reporting mechanism) all received Report Soft Error MAC frames to LAN managers with which it has reporting links, based on the following criteria:

- Nonzero values for specified types of error counters
- Stations with error weights exceeding certain thresholds.

(See "Intensive Mode Reporting Function" on page 15-4 for details.) The information from the Report Soft Error MAC frame is sent (through the LAN reporting mechanism) to LAN managers in the Forward MAC Frame.

The Report Neighbor Notification Incomplete MAC frame and Report Monitor Error MAC frame are also forwarded (through the collocated LAN reporting mechanism) to LAN managers in the Forward MAC Frame.

```
Forward MAC Frame
   Ring Number
   Reporting Station Address
   Report Soft Error
      NAUN
      Physical Location
      Isolating Error Counts
      Non-Isolating Error Counts
   Report Neighbor-Notification Incomplete
      Address of Last Neighbor Notification
   Report Monitor Error
      Error Code
      Physical Location
      NAUN Address
```

Figure 15-12. Forward MAC Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Forward MAC Frame (X'810A') Conditions of Presence" on page F-6.

Beaconing Condition on Ring (X'8110')

The ring error monitor sends this frame (through the LAN reporting mechanism) to LAN managers when it detects either a permanent or temporary beaconing condition on the ring it monitors.

If the ring error monitor hard-error processing function (see "Hard-Error Processing Function" on page 15-1) determines that the beacon condition is temporary, it queries the beacon fault domain stations and determines whether one or both stations have left the token ring. The addresses of the absent station(s) are included in this frame.

If the ring error monitor hard-error processing function detects a hard-error condition on the ring, it notifies LAN managers using the Beaconing Condition on Ring frame.

```
Beaconing Condition on Ring
   Ring Number
   Ring Status
      Ring State
      Beacon Data
         Beacon Type
         Station Address
         NAUN Address
         Physical Location
   Removed Station Address
```

Figure 15-13. Beaconing Condition on Ring Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Beaconing Condition on Ring (X'8110') Conditions of Presence" on page F-7.

Beaconing Condition Recovered (X'8111')

The ring error monitor sends this frame (through the LAN reporting mechanism) to LAN managers when it detects that a permanent beaconing condition has recovered (through manual recovery).

```
Beaconing Condition Recovered
   Ring Number
```

Figure 15-14. Beaconing Condition Recovered Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Beaconing Condition Recovered (X'8111') Conditions of Presence" on page F-7.

Receiver Congestion Notification (X'810E')

The ring error monitor sends this frame (through the LAN reporting mechanism) to LAN managers when it detects that a station on its ring is experiencing excessive congestion (see "Receiver-Congestion Error Reporting Function" on page 15-4).

Figure 15-15. Receiver Congestion Notification Frame

Ring Station Address Subvector

This subvector identifies a station that the receiver-congestion analysis function in the ring error monitor has determined to be reporting excessive congestion errors.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Receiver Congestion Notification (X'810E') Conditions of Presence" on page F-6.

Receiver Congestion Ended (X'810F')

The ring error monitor sends this frame (through the LAN reporting mechanism) to LAN managers when one or more of the stations for which it reported excessive congestion (see "Receiver Congestion Notification (X'810E')" on page 15-29) are no longer experiencing that condition. Multiple stations for which the receiver congestion condition has ended may be identified in this notification.

```
Receiver Congestion Ended
   Ring Number
   Ring Station Address
```

Figure 15-16. Receiver Congestion Ended Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Receiver Congestion Ended (X'810F') Conditions of Presence" on page F-7.

Chapter 16. Configuration Report Server

A configuration report server (CRS) accepts commands from the LAN manager to get station information, set station parameters, and remove stations on its ring. It also collects and forwards configuration reports generated by stations on its ring to the LAN manager.

Configuration Report Server Functions

The configuration report server forwards certain (configuration-related) MAC frames to the LAN manager. The configuration report server functions are identified below.

Station Status Request

This function accepts and executes requests from the LAN manager to get the status of stations on the configuration report server's ring and returns the results to LAN managers (through the LAN reporting mechanism).

Set Station Parameters Function

This function accepts and executes requests from the LAN manager to set the values of operational parameters in stations on the configuration report server's ring. The success or failure of this operation is reported to LAN managers (through the LAN reporting mechanism).

Remove Station Function

This function accepts and executes requests from the LAN manager to force a station to leave the configuration report server's ring. The success or failure of this operation is reported to the LAN manager (through the LAN reporting mechanism).

Notification Function

This function forwards information about configuration changes on the configuration report server's ring to LAN managers (through the LAN reporting mechanism). The configuration changes reported are new active monitor and new Nearest Active Upstream Neighbor. This function also forwards information contained in Report Transmit Forward MAC Frames received by the configuration report server (see "Report Transmit Forward MAC Frame, X'2A'" on page 5-13).

Communication Function

Frames sent between a LAN manager and a configuration report server are described in this chapter. These frames are sent using the reporting link established by a LAN manager with the LAN reporting mechanism (see Chapter 14, "LAN Reporting Mechanism").

Data Structures

The following data structures are sent and received by the configuration report server.

```
Correlator

Ring Station Address

Ring Number

Local Ring Number

CRS Version Level

Addressing Information
   NAUN Address
   Physical Location
   Group Address
   Functional Address

State Information
   Ring Station Microcode Level
   Ring Station Status
   Unique Station ID

Attachments Information
   Product Instance ID
   Enabled Function Classes
   Allowed Access Priority
   Functional Address(es)

Station Error
   Reason Code
   Response Code

Reporting Station Address

NAUN Address

Physical Location of Reporting Station

Product Instance Identifier

Transmit Status Code

Soft-Error Report Timer

Enabled Function Classes

Allowed Access Priority
```

Figure 16-1. Configuration Report Server Data Structures

Correlator Subvector

This subvector has a 2-byte value field that is used by the LAN manager to associate responses with requests generated by the LAN manager.

Ring Station Address Subvector

This subvector has a 6-byte (bit string) value field containing the individual address of a station from which the configuration report server is to request MAC information, set station parameters, or force a station to leave the ring. This address must be an individual address (see "Individual and Group Addresses" on page 3-9).

Ring Number Subvector

This subvector has a 2-byte (bit string) value field containing the ring number of the ring to which the station identified in the ring station address subvector or reporting station address subvector is attached.

Local Ring Number Subvector

This subvector has a 2-byte (bit string) value field containing the ring number to be set in the station identified in the ring station address subvector. This subvector is present only in the Set Ring Station Parameters frame.

CRS Version Level Subvector

This subvector contains the following information about a configuration report server.

Discriminator Field

This field consists of a 1-byte integer that indicates the encoding method used for the rest of the configuration report server version level subvector value. Its value is either X'00' (indicating ASCII) or X'01' (indicating EBCDIC).

Common Version Identifier Field

This field consists of the 2-byte numeric version level of the configuration report server.

Common Release Identifier Field

This field consists of the 2-byte numeric release level of the configuration report server.

Common Modification Identifier Field

This field consists of the 2-byte numeric modification level of the configuration report server.

Software Product Program Number Field

This field consists of a 7-byte alphanumeric field that identifies the configuration report server.

Addressing Information Subvector

This subvector contains the following nested subvectors, whose values are extracted from Report Station Address MAC frames received by the configuration report server.

NAUN Address Subvector

This subvector has a 6-byte (bit string) value field containing the individual address of a station's nearest active upstream neighbor (NAUN).

Physical Location Subvector

This subvector has a 4-byte value field containing the value of the Physical Location Subvector in Report Ring Station Addresses MAC frames (see "Physical Location, X'0B'" on page 5-21).

Group Address Subvector

This subvector has a 4-byte (bit string) value containing the lower-order 4 bytes of the enabled station group address (see "Group Address, X'2B'" on page 5-18).

Functional Address Subvector

This subvector has a 4-byte value field (bit string) value field that specifies the functional addresses recognized by a station. For information on functional addresses, see "Functional Addresses" on page 3-10.

State Information Subvector

This subvector value contains the following nested subvectors, whose values are extracted from Report Station MAC frames received by the configuration report server.

Ring Station Microcode Level Subvector

This subvector has a 10-byte (EBCDIC) value field containing a station's microcode level. For information about a ring station's microcode level, see "Ring Station Microcode Level, X'23'" on page 5-23.

Ring Station Status Subvector

This subvector has a 6-byte (bit string) value field containing the status of a ring station. Its contents are described in "Ring Station Status Subvector, X'29'" on page 5-23.

Unique Station ID Subvector

This subvector has a 6-byte (bit string) value field that contains a unique identifier for the ring station.

Attachments Information Subvector

This subvector has a value field containing the following nested subvectors extracted from Report Station Attachment MAC frames received by the configuration report server.

Product Instance ID Subvector

This subvector has a complex value field up to 18 bytes long that uniquely identifies the hardware product of the station. The value field of this subvector is specified in "Product Instance ID, X'22'" on page 5-21.

Enabled Function Classes Subvector

This subvector has a 2-byte (bit string) value field specifying the function classes that a ring station is permitted to transmit. Its contents are described in "Enabled Function Classes, X'06'" on page 5-17.

Allowed Access Priority Subvector

This subvector has a 2-byte value field specifying the maximum token priority a station is permitted to transmit. Its contents are described in "Allowed Access Priority, X'07'" on page 5-17.

Functional Address(es) Subvector

This subvector has a 4-byte (bit string) value field identifying the functional addresses recognized by a station. Its contents are described in "Functional Address, X'2C'" on page 5-18.

Station Error Subvector

This subvector has a complex value field containing the error information in the case that the configuration report server could not communicate with a station as directed by the LAN manager. It contains two nested subvectors: the reason code subvector and the response code subvector.

Reason Code Subvector

This subvector has a 2-byte (bit string) value field identifying the reason that the configuration report server could not communicate as requested with the station. The following list defines the valid values for this subvector.

- X'0000' — All of the requests to the station indicated that the frame had not been recognized or copied (address recognized indicators = B'0' and frame copied indicators = B'0').
- X'0001' — All of the requests to the station indicated that the frame had been recognized but not copied (address recognized indicators = B'1' and frame copied indicators = B'0').
- X'0002' — The requests indicated that the frame was recognized and copied, but the configuration report server timed out waiting for a response from the station.
- X'0003' — The request was received by the station, and a negative Response MAC frame was generated (and received by the configuration report server). The response code contained in the Response MAC frame is included in the response code subvector.
- X'0004' — The ring to which the configuration report server is attached is inoperative (beaconing, wire fault), so the MAC frames could not be issued on the requested ring to the specified ring station.

Response Code Subvector

This subvector has a 4-byte value field containing a copy of the value of the response code subvector in a received Response MAC frame. For information about the MAC frame response code, see "Response Code, X'20'" on page 5-22.

Reporting Station Address Subvector

This subvector has a 6-byte (bit string) containing the individual address of a ring station reporting a configuration change.

NAUN Address Subvector

This subvector has a 6-byte (bit string) value field containing the individual address of the nearest active upstream neighbor (NAUN) of the station identified in the reporting station address subvector.

Physical Location of Reporting Station Subvector

This subvector has a 4-byte value field containing the physical location of the reporting station (see "Assign Physical Location, X'04'" on page 5-17).

Product Instance ID Subvector

This subvector has a complex value field up to 18 bytes long that uniquely identifies the hardware product of the station. The value field of this subvector is specified in "Product Instance ID, X'22'" on page 5-21.

Transmit Status Code Subvector

This subvector has a 2-byte value field containing the strip status of a transmitted frame. It is sent in the Transmit Forward MAC frame (maintained by a ring station). See "Transmit Status Code, X'2A'" on page 5-23.

Soft-Error Report Timer Subvector

This subvector has a 2-byte value field containing the time interval at which ring stations send Report Soft Error frames when they are experiencing errors. The value of this subvector is specified in "Soft Error Report Timer Value, X'05'" on page 5-23.

Enabled Function Classes Subvector

This subvector has a 2-byte (bit string) value field specifying the function classes that a ring station is permitted to transmit. Its contents are described in "Enabled Function Classes, X'06'" on page 5-17.

Allowed Access Priority Subvector

This subvector has a 2-byte value field specifying the maximum token priority a station is permitted to transmit. Its contents are described in "Allowed Access Priority, X'07'" on page 5-17.

Request Station Information (X'8304')

This frame is sent by the LAN manager to request certain information about one of the ring stations attached to the configuration report server's local ring. Depending on the information requested by the LAN manager, the configuration report server queries the station using one of the following MAC frames:

- "Request Ring Station Address MAC Frame, X'0E'" on page 5-13
- "Request Ring Station Attachments MAC Frame, X'10'" on page 5-13
- "Request Ring Station State MAC Frame, X'0F'" on page 5-13.

If no response is received from the station, the configuration report server retries the request a predetermined number of times or until a response is received.

```
Request Station Information
   Correlator
   Station Address
   Ring Number
   All
   Addressing Information
   State Information
   Attachments Information
```

Figure 16-2. Request Station Information Frame

Correlator Subvector

This subvector is always present and allows a LAN manager to correlate the response frames that it receives with the Request Station Information frames that it sent. For a frame sequence to be valid, the correlator subvector value in the response frame must match the value that is sent in the Request Station Information frame.

Ring Number Subvector

This subvector indicates the ring to which the station identified in the Station Address Subvector is attached, in the case where the configuration report server is using more than one MAC instance (and monitoring multiple rings).

All Subvector

If this subvector is present in the Request CRS Status major vector, the response contains the values for each of the parameters shown in Figure 16-2.

Subvector Length

The length of each atomic subvector (except the correlator and ring number subvectors) is 4 bytes (a 2-byte subvector length field, a 2-byte subvector identifier field, and a zero-length value field).

Parsing Algorithm

Before executing the Request Station Information, the configuration report server checks it for errors using the algorithm shown in "Parsing Algorithm for Request Status Frames" on page E-2.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Request Station Information (X'8304') Conditions of Presence" on page F-9.

Responses

If the Request Station Information frame does not contain syntactic or semantic errors, the configuration report server queries the station and returns the requested information in a Report Station Information frame (see "Report Station Information (X'8306')" on page 16-9).

If a syntactic or semantic error is detected when parsing the Request Station Information frame, the CRS Error frame is returned to the requesting LAN manager (see "CRS Error (X'830B')" on page 16-14).

Report Station Information (X'8306')

This frame is returned in response to a Request Station Information frame (see "Request Station Information (X'8304')" on page 16-7). The information contained in this frame was requested by the LAN manager and is obtained from Report Ring Station Addresses, Report Ring Station State, or Report Ring Station Attachments MAC frames.

If no response is received from a station or the Response MAC frame (see "Response MAC Frame, X'00'" on page 5-13) is returned (indicating an error), the configuration report server sends the LAN manager a Report Station Information frame containing the station error subvector.

```
Report Station Information
   Correlator
   Ring Station Address
   Ring Number
   Addressing Information
      NAUN Address
      Physical Location
      Group Address
      Functional Address(es)
   State Information
      Ring Station Microcode Level
      Ring Station Status
      Unique Station ID
   Attachments Information
      Product Instance ID
      Enabled Function Classes
      Allowed Access Priority
      Functional Address(es)
   Station Error
      Reason Code
      Response Code
```

Figure 16-3. Report Station Information Frame

Correlator Subvector

The LAN manager uses the value of this subvector to correlate a Report Station Information frame with the corresponding Request Station Information frame.

Ring Station Address Subvector

The value of this subvector indicates the individual MAC address of the station to which the information in this frame pertains.

Station Error

This subvector is used by the configuration report server to report that it could not successfully query a station.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report Station Information (X'8306') Conditions of Presence" on page F-10.

Set Station Parameters (X'8307')

This frame is sent by the controlling LAN manager to set the values of operational parameters of a ring station. The configuration report server sets the station parameters by issuing the Change Parameters MAC frame (see "Change Parameters MAC Frame, X'0C'" on page 5-10) to the station. If the station does not respond positively, the configuration report server retries the request a predetermined number of times.

```
Set Station Parameters
   Correlator
   Ring Station Address
   Ring Number
   Local Ring Number
   Physical Location
   Soft-Error Report Timer Value
   Enabled Function Classes
   Allowed Access Priority
```

Figure 16-4. Set Station Parameters Frame

Correlator Subvector

This subvector allows a LAN manager to associate a Set Station Parameters frame with its corresponding response. For a frame sequence to be valid, the correlator subvector value in the response frame must match the value of the correlator subvector in the Set Station Parameters frame.

Ring Station Address Subvector

The value of this subvector indicates the MAC address of the station to which the information in this frame pertains.

Ring Number Subvector

This subvector indicates the ring to which the station identified in the Station Address subvector is attached, in the case where the configuration report server is using more than one MAC instance (and monitoring multiple rings).

Parsing Algorithm

Before executing the Set Station Parameters, the configuration report server checks it for errors using the algorithm shown in "Parsing Algorithm for Set Parameter Frames" on page E-3.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Set Station Parameters (X'8307') Conditions of Presence" on page F-10.

Responses

If the Set Station Parameters frame does not contain syntactic or semantic errors, the configuration report server issues a MAC frame to set the values in the station and returns the result of the operation in a Station Parameters Set Frame (see "Station Parameters Set (X'8308')" on page 16-11).

If a syntactic or semantic error is detected when parsing the Set Station Parameters frame, the CRS Error frame is returned to the requesting LAN manager (see "CRS Error (X'830B')" on page 16-14).

Station Parameters Set (X'8308')

This frame is sent by the configuration report server to indicate the success or failure of a Set Station Parameters command.

```
Station Parameters Set
   Correlator
   Station Error
      Reason Code
      Response Code
```

Figure 16-5. Station Parameters Set Frame

Correlator Subvector

The LAN manager uses the value of this subvector to correlate this Station Parameters Set frame with the associated Set Station Parameters frame.

Station Error

If the station specified in the corresponding Set Station Parameters frame did not respond, then this subvector is present and indicates the reason.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Station Parameters Set (X'8308') Conditions of Presence" on page F-11.

Remove Ring Station (X'8309')

This frame is sent by the LAN manager to request the removal of a ring station from the configuration report server's local ring. The configuration report server issues a Remove Station MAC frame (see "Remove Ring Station MAC Frame, X'0B'" on page 5-11) to the indicated station. The configuration report server then queries the station to ensure its removal. This process is retried a predetermined number of times or until the station's removal is confirmed.

```
Remove Ring Station
   Correlator
   Ring Station Address
   Ring Number
```

Figure 16-6. Remove Ring Station Frame

Correlator Subvector

This subvector allows a LAN manager to associate a Remove Ring Station frame with its corresponding response. For a frame sequence to be valid, the correlator subvector value in the response must match the value of the correlator in this Remove Ring Station frame.

Ring Station Address Subvector

The value of this subvector indicates the MAC address of the station to be removed from the ring.

Ring Number Subvector

This subvector indicates the ring to which the station identified in the Station Address subvector is attached, in the case where the configuration report server is using more than one MAC instance (and monitoring multiple rings).

Parsing Algorithm

Before executing the Remove Ring Station, the configuration report server checks it for errors using the algorithm shown in "Parsing Algorithm for Set Parameter Frames" on page E-3.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Remove Ring Station (X'8309') Conditions of Presence" on page F-11.

Responses

If the Remove Ring Station frame does not contain syntactic or semantic errors, the configuration report server issues one or more Remove Ring Station MAC frames (see "Remove Ring Station MAC Frame, X'0B'" on page 5-11). The configuration report server returns the result of the operation in a Ring Station Removed frame (see "Ring Station Removed (X'830A')" on page 16-13).

If a syntactic or semantic error is detected when parsing the Set Station Parameters frame, the CRS Error frame is returned to the requesting LAN manager (see "CRS Error (X'830B')" on page 16-14).

Ring Station Removed (X'830A')

This frame is sent by the configuration report server to indicate the success or failure of a Remove Ring Station operation requested by the controlling LAN manager. The configuration report server issues the Remove Ring Station command and then queries the ring station a predetermined number of times to ensure that it has actually left the ring before sending the Ring Station Removed frame. If the station is still present, the configuration report server tries again to remove it and confirm its absence from the ring. This process is repeated a predetermined number of times and the result reported in the Ring Station Removed frame.

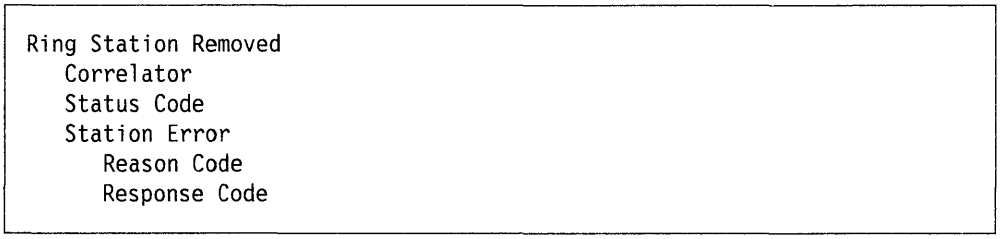


Figure 16-7. Ring Station Removed Frame

Correlator Subvector

The LAN manager uses the value of this subvector to correlate a Remove Ring Station frame with the associated Ring Station Removed frame.

Status Code Subvector

This subvector has a 1-byte value field containing one of the following values:

X'00' — Station removed
X'01' — Station could not be removed.

Station Error Subvector

This subvector gives more information about the reason that the station could not be removed.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Ring Station Removed (X'830A') Conditions of Presence" on page F-11.

CRS Error (X'830B')

This frame is sent by the configuration report server if a syntax, semantic, or execution error is detected in any of these frames:

- Request Station Information
- Set Station Parameters
- Remove Ring Station.

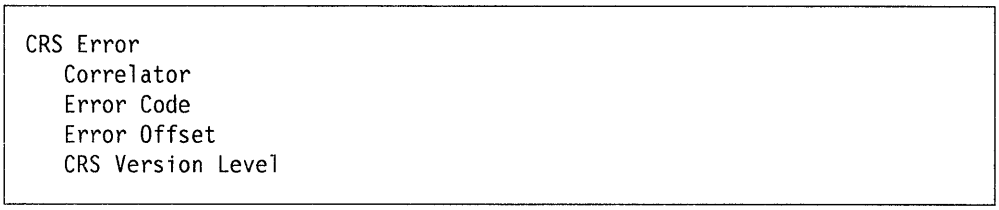


Figure 16-8. CRS Error Frame

Correlator Subvector

This subvector has the same value as the correlator subvector in the frame to which this CRS Error frame is a response. The correlator subvector allows the LAN manager to associate the CRS Error frame with the original request.

If the correlator subvector was not present in the Set CRS Parameters frame or the Request CRS Parameters frame, an error code of *unrecognized subvector* and a correlator value of X'00000000' is returned in the CRS Error frame.

Error Code Subvector

This subvector contains a 2-byte diagnostic code that describes the reason that an error was detected in the corresponding frame. This code refers to the *first* error detected. The valid values for syntax errors are listed in Figure E-1 on page E-2 and Figure E-2 on page E-3. If the error was detected while executing the command, the subvector value is X'000C'.

Error Offset Subvector

The value of this subvector is the offset (in number of bytes) into the corresponding frame where the *first* error was detected. The offset is a 2-byte integer and is counted from the first byte of the major vector identifier.

CRS Version Level Subvector

This subvector can be used by the LAN manager to help determine the reason for the error.

Conditions of Presence

The conditions of presence for this frame's subvectors are the same as for the CRS Error frame shown in "CRS Error (X'830B') Conditions of Presence" on page F-11.

Report New Active Monitor (X'8301')

This frame is sent by the configuration report server (through the LAN reporting mechanism) to inform LAN managers that a new active monitor has been established for the ring into which the server's adapter is inserted. All of the information in this frame (except the ring number) is present in the Report New Active Monitor MAC frame (see "Report New Active Monitor MAC Frame, X'25'" on page 5-12) that triggered this notification.

```
Report New Monitor
   Ring Number
   Station Address
   NAUN Address
   Physical Location
   Product Instance Identifier
```

Figure 16-9. Report New Monitor Frame

Station Address Subvector

This subvector identifies the station from which the Report New Active Monitor MAC frame was received.

NAUN Address, Physical Location, and Product Instance Identifier Subvectors

These subvectors contain the same values as the corresponding subvectors in the received Report New Active Monitor MAC frame (see Table 5-2 on page 5-15).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report New Active Monitor (X'8301') Conditions of Presence" on page F-9.

Report NAUN Change (X'8302')

This frame is sent by the configuration report server (through the LAN reporting mechanism) to inform LAN managers that a change in the configuration of the network has occurred. (For example, the presence of a new station or the disappearance of an old station has been detected.) All of this information (except the ring number) is present in the Report NAUN-Change MAC frame ("Report NAUN Change MAC Frame, X'26'" on page 5-12) that triggered this notification.

```
Report NAUN Change
   Ring Number
   Station Address
   NAUN Address
   Physical Location
```

Figure 16-10. Report NAUN Change Frame

Station Address Subvector

This subvector identifies the station from which the Report NAUN Change MAC frame was received.

NAUN Address and Physical Location Subvectors

These subvectors contain the same values as the corresponding subvectors in the received Report NAUN Change MAC frame (see Table 5-2 on page 5-15).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report NAUN Change (X'8302') Conditions of Presence" on page F-9.

Report Transmit-Forward (X'8303')

This frame is sent by the configuration report server (through the LAN reporting mechanism) to LAN managers after a Transmit-Forward MAC frame has been received and forwarded by a station. The information contained in a Report-Transmit Forward MAC frame (see "Report Transmit Forward MAC Frame, X'2A'" on page 5-13) is forwarded to LAN managers in this frame.

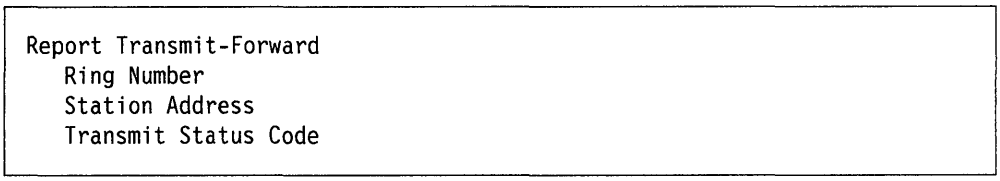


Figure 16-11. Report Transmit Forward Frame

Station Address Subvector

This subvector identifies the station from which the Report Transmit Forward MAC frame is received.

Transmit Status Code Subvector

This subvector contains the same value as the corresponding subvector in the Report Transmit Forward MAC frame (see "Transmit Status Code, X'2A'" on page 5-23).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report Transmit-Forward (X'8303') Conditions of Presence" on page F-9.

Chapter 17. Ring Parameter Server

A ring parameter server (RPS) function resides on each ring in a multiple-ring environment for which operational parameters are being managed and provides three main services:

- Sends initialization information to new stations that are attaching to the ring
- Ensures that stations on the ring have consistent values for operational parameters
- Forwards registration information to LAN managers from stations attaching to the ring.

Ring Parameter Server Functions

The ring parameter server performs the functions listed below.

Status Request Function

The ring parameter server accepts and executes requests for status from a LAN manager.

Ring Station Parameters Maintenance Function

This function issues commands to set parameter values in ring stations. The ring station operational parameters controlled by the ring parameter server are:

- Ring number
- Soft-error report timer.

Registration Function

This function accepts unsolicited registration information from ring stations and forwards this information to LAN managers (through the LAN reporting mechanism).

When a ring station attaches to a ring, it requests values for the ring's operational parameters from the ring parameter server and it also passes information about its own configuration parameters and microcode level to the ring parameter server. This process is described in "Attaching to the Ring" on page 3-27.

If a ring parameter server is present on the ring, it responds to the station's request by sending it the current values for the ring's operational parameters. The ring parameter server then notifies the LAN managers (using the services of a collocated LAN reporting mechanism) that a new station has attached to the ring.

If a ring parameter server is *not* present on the ring, the ring station uses the values assigned by the program using the ring station or the default values for its operational parameters.

Ring Parameter Server Activation

When a ring parameter server becomes active on a ring, it determines if another ring parameter server is active on the ring. If not, the ring parameter server in the attaching station activates its ring station parameters maintenance function. It then broadcasts an Initialize Ring Station MAC frame (see "Initialize Ring Station MAC Frame, X'0D'" on page 5-11) a predetermined number of times to all the ring stations on the ring, thus establishing its default parameter values as the ring's operational parameters. The responses returned by the ring stations for the Initialize Ring Station MAC frames are discarded by the ring parameter server.

However, if another ring parameter server is active on the ring, the new ring parameter server compares the values received for the ring's operational parameters (which it received when its ring station attaches to the ring) with its own stored values. If the values match, the ring parameter server activates its ring parameter server function.

If the values do not match, the new ring parameter server reports to the local operator (if any) that its parameters are inconsistent with the existing parameters for the ring. It then disables the ring parameter server functional address in the station attaching it to the ring to prevent it from receiving requests for initialization from attaching stations. The ring parameter server status request function remains active in order to provide status to the LAN manager on request.

Ring Parameter Server Messages

Messages that are sent between a ring parameter server and LAN managers are described in this section. These messages are sent using reporting links that are established by the collocated LAN reporting mechanism (see Chapter 14, "LAN Reporting Mechanism").

Data Structures

The following data structures are sent and received by the ring parameter server.

```
Correlator

Ring Number

RPS Version Level

Soft-Error Report Timer Value

Ring Station Address

NAUN Address

Product Instance ID

Ring Station Microcode Level

Attachment Status
```

Figure 17-1. Ring Parameter Server Data Structures

Correlator Subvector

This subvector has a 2-byte value field that is used by the LAN manager to associate responses with requests generated by the LAN manager.

Ring Number Subvector

This subvector has a 2-byte integer value field containing the identifier of the ring into which the ring parameter server station is attached.

RPS Version Level Subvector

This subvector has a complex value field containing the following information about a ring parameter server.

Discriminator Field

This field consists of a 1-byte integer that indicates the encoding method used for the rest of the RPS version level subvector value. Its value is either X'00' (indicating ASCII) or X'01' (indicating EBCDIC).

Common Version Identifier Field

This field consists of the 2-byte numeric version level of the ring parameter server.

Common Release Identifier Field

This field consists of the 2-byte numeric release level of the ring parameter server.

Common Modification Identifier Field

This field consists of the 2-byte numeric modification level of the ring parameter server.

Software Product Program Number Field

This field consists of a 7-byte alphanumeric field that identifies the ring parameter server.

Soft-Error Report Timer Value Subvector

This subvector has a 2-byte value field specifying the time-out value for the ring's soft-error report timer (in 10-millisecond increments). The timer controls the frequency at which ring stations send error reports to the ring error monitor (see "Soft Error Report Timer Value, X'05'" on page 5-23).

Ring Station Address Subvector

This subvector has a 6-byte (bit string) value field containing the MAC address of a ring station that is attaching to the ring.

NAUN Address Subvector

This subvector has a 6-byte (bit string) value field containing the address of the nearest active upstream neighbor of the station that is attaching to the ring.

Product Instance ID Subvector

This subvector has a complex value field up to 18 bytes long that uniquely identifies the hardware product of the station attaching to the ring. The value field of this subvector is specified in "Product Instance ID, X'22'" on page 5-21.

Ring Station Microcode Level Subvector

This subvector has a 10-byte value field indicating the microcode version of the station that is attaching to the ring. For information about a ring station's microcode level, see "Ring Station Microcode Level, X'23'" on page 5-23.

Attachment Status Subvector

This subvector has a 2-byte value field containing the status of the attachment process for a station that is attaching to the ring. Valid values are:

- X'0000' — The attachment process completed without error. (See "Attaching to the Ring" on page 3-27.)
- X'0001' — A Request Parameters MAC frame was received by the ring parameter server, the predetermined number of Initialize Ring Station MAC frames were sent to the attaching station, and at least one negative Response MAC frame was received from the station attaching to the ring.
- X'0002' — A well-formed Request Parameters MAC frame was received by the ring parameter server and a predetermined number of Initialize Ring Station MAC frames were sent to the attaching station and no Response MAC frames were received from the attaching station.

Note: The last two status codes imply that a station tried to attach to the ring, but did not succeed.

Request RPS Status (X'8201')

This frame is sent by a LAN manager to a ring parameter server to request certain status information.

```
Request RPS Status
   Correlator
   Ring Number
   All
```

Figure 17-2. Request RPS Status

Correlator Subvector

This subvector is always present and allows a LAN manager to associate the response frames that it receives with the Request RPS Status frames that it sent. For a frame sequence to be valid, the correlator subvector value in the response frame must match the value of the correlator in the associated Request RPS Status frame.

Ring Number Subvector

This subvector is used to identify the information being requested in the case that the ring parameter server is using more than one MAC instance (and monitoring multiple rings).

All Subvector

If this subvector is present in the Request RPS Status major vector, the response contains the values for each of the subvectors shown in Figure 17-3 on page 17-6.

Parsing Algorithm

Before executing the Request RPS Status, the ring parameter server checks it for errors using the algorithm shown in "Parsing Algorithm for Request Status Frames" on page E-2.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Request RPS Status (X'8201') Conditions of Presence" on page F-8.

Responses

If the Request RPS Status frame does not contain syntactic or semantic errors, the ring parameter server returns the requested values in a Report RPS Status frame (see "Report RPS Status (X'8202')" on page 17-6).

If an error is detected, the ring parameter server sends an RPS Error frame (see "RPS Error (X'8203')" on page 17-7).

Report RPS Status (X'8202')

This frame is sent by a ring parameter server to a LAN manager in response to a Request RPS Status frame (see "Request RPS Status (X'8201')" on page 17-5) if no errors are detected in the request frame.

```
Report RPS Status
   Correlator
   Ring Number
   RPS Version Level
   Soft-Error-Report Timer Value
```

Figure 17-3. Report RPS Status

Correlator Subvector

The LAN manager uses the value of this subvector to associate a Report RPS Status frame with the associated Request RPS Status frame.

Ring Number Subvector

This subvector identifies the ring parameter information to which this frame applies.

Other Subvectors

The information requested in the Request RPS Status frame is returned in the remaining subvectors in this frame.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report RPS Status (X'8202') Conditions of Presence" on page F-8.

RPS Error (X'8203')

This frame is sent by a ring parameter server to the requesting LAN manager if an error is detected in a Request RPS Status frame (see "Request RPS Status (X'8201')" on page 17-5).

```
RPS Error
   Correlator
   Error Code
   Error Offset
   RPS Version Level
```

Figure 17-4. RPS Error

Correlator Subvector

This subvector has the same value as the correlator subvector in the Request RPS Status frame for which this frame is a response. The correlator subvector allows the LAN manager to associate the error response with the original request.

If the correlator subvector was not present in the Request RPS Status frame, an error code of *unrecognized subvector* and a correlator value of X'00000000' is returned in the RPS Error frame.

Error Code Subvector

This subvector contains a 2-byte diagnostic code that describes the reason that an error was detected in the corresponding Request RPS Parameters frame. This code refers to the *first* error detected. The valid values for syntax errors are listed in Figure E-1 on page E-2.

Error Offset Subvector

The value of this subvector is the offset (in number of bytes) into the corresponding frame where the *first* error was detected. The offset is a 2-byte integer and is counted from the first byte of the major vector identifier.

RPS Version Level Subvector

This subvector can be used by the LAN manager to help determine the reason for the error.

Conditions of Presence

The conditions of presence for this frame's subvector are shown in "RPS Error (X'8203') Conditions of Presence" on page F-8.

Report Station in Ring (X'8206')

This frame is sent by a ring parameter server to notify LAN managers when a new station attaches to the ring.

The information in this notification is used to identify the newly attached station. LAN managers can use this information, along with the Report NAUN Change frame from the configuration report server (see "Report NAUN Change (X'8302')" on page 16-16) to maintain a configuration data base.

If an error is detected while a station is attaching to the ring, this information is sent to the LAN manager in the attachment status subvector.

```
Report Station in Ring
   Ring Number
   Ring Station Address
   NAUN Address
   Product Instance Identifier
   Ring Station Microcode Level
   Attachment Status
```

Figure 17-5. Report Station in Ring Frame

Subvectors

Besides the ring number, the values of this frame's subvectors are extracted from the Request Initialization MAC frame (see "Request Initialization MAC Frame, X'20'" on page 5-13) sent by the station attaching to the ring (see "Attaching to the Ring" on page 3-27).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report Station in Ring (X'8206') Conditions of Presence" on page F-8.

Chapter 18. LAN Bridge Server

A LAN bridge server (LBS) keeps statistical information about frames forwarded between two or more rings (through a bridge) and sends this information to selected LAN managers (through the LAN reporting mechanism).

LAN Bridge Server Functions

A LAN bridge server provides the functions listed below.

Status Request Function

This function accepts and responds to requests for status from a LAN manager for status.

Set Parameters Function

This function accepts and executes commands from a controlling LAN manager to change the internal configuration of the bridge or set the values of operational parameters.

Notification Function

This function collects inter-ring traffic statistics and forwards this information to LAN managers and optionally displays them locally. This function also notifies LAN managers of LAN bridge server state changes.

Path Trace Function

This function is used to trace the path of non-broadcast frames through a multi-ring network. The LAN bridge server notifies LAN managers when it receives non-broadcast frames for forwarding with bit 1 of the routing control field (see "Routing Information Field" on page 2-6) set to B'1'.

Bridge Performance Monitoring Function

This function maintains counters of the number of frames and bytes forwarded through the bridge and the number of frames not received by a bridge station because the station was congested. It also counts the number of frames copied into the bridge for forwarding to a target ring but not actually forwarded because of an anomaly in the frame, the bridge, or on the target ring. Periodically the percentage of frames lost or discarded by the bridge because of anomalies is calculated and if the percentage exceeds a predefined threshold, the LAN bridge server sends a notification to the LAN manager.

The LAN bridge server can also be configured to periodically send its error and traffic counters to the LAN manager to allow for trend analysis by the LAN manager.

LAN Bridge Server Messages

Messages that are sent between a LAN Bridge Server and LAN managers are described in this chapter. These messages flow over reporting links that are established by a LAN reporting mechanism. (see Chapter 14, "LAN Reporting Mechanism").

Data Structures

The following data structures are sent and received by the LAN bridge server.

```
Correlator
Bridge Type
Bridge Version Level
Number of Ports
Partition Bits
Path Trace
Calculation Interval
Notification Interval
Percent Frames Lost Threshold
Percent Frames Lost
Port Information (one per port)
   Port Identifier
   Ring number
   Port Type
   Ring Data Rate
   Ring Status
   Adapter Status
   Port Hop Count
   Frames Discarded Counter Value
   T-Frames Discarded Counter Value
   Frames Discarded Counter Value (long)
   B-Frames-Transmitted Counter Value
   T-B-Frames-Transmitted Counter Value
   B-Bytes-Transmitted Counter Value
   T-B-Bytes-Transmitted Counter Value
   NB-Frames-Transmitted Counter Value
   T-NB-Frames-Transmitted Counter Value
   NB-Bytes-Transmitted Counter Value
   T-NB-Bytes-Transmitted Counter Value
   Frames-Not-Received Counter Value
   T-Frames-Not-Received Counter Value
   Frames-Not-Received Counter Value (long)
   Frames-Not-Forwarded Counter Value
   T-Frames-Not-Forwarded Counter Value
   Frames-Not-Forwarded Counter Value (long)
   Frames Discarded - Internal Error
   T-Frames Discarded - Internal Error
   Bytes Discarded - Internal Error
   Frames Not Routed Across Bridge
   T-Frames Not Routed Across Bridge
   Single-Route Broadcast Enabled/Disabled
Route Status (one per route)
   Route Identifier
      Port Identifier
      Port Identifier
   Bridge Identifier
   Largest Frame Size
   Route Active Status
   Single-Route Broadcast Mode
Ring Number
Forwarded-Frame Addressing Information
Forwarded-Frame Length
Forwarded-Frame Data
Forwarded-Frame Status
Bridge Internal Status
Temporary/Permanent
```

Figure 18-1. Bridge Server Data Structures

Correlator Subvector

This subvector has a 2-byte value field that is used by LAN managers to associate responses with requests they generate.

Bridge Type Subvector

This subvector has a 2-byte (bit string) value field specifying the type of bridge. The valid values for this parameter are:

- X'0001' for a bridge connecting two token-rings
- X'0002' for a bridge connecting two CSMA/CD LANs
- X'0003' for a bridge connecting a token-ring to a CSMA/CD LAN.

In addition, the leftmost bit indicates whether the bridge is implemented in a single system or is split into two separate halves, separated by some communications media:

- If the bit is B'0', the bridge is implemented in a single system
- If the bit is B'1', the bridge is split.

Bridge Version Level Subvector

This subvector has a complex value field containing the following information about a LAN bridge server.

Discriminator Field

This field consists of a 1-byte integer that indicates the encoding method used for the rest of the bridge version level subvector value field. Its value is X'00' (indicating ASCII) or X'01' (indicating EBCDIC).

Common Version Identifier Field

This field consists of a 2-byte numeric version level of the LAN bridge server.

Common Release Identifier Field

This field consists of a 2-byte numeric release level of the LAN bridge server.

Common Modification Identifier Field

This field consists of a 2-byte numeric modification level of the LAN bridge server.

Software Product Program Number Field

This field consists of a 7-byte alphanumeric field that identifies the LAN bridge server.

Number of Ports Subvector

This subvector has a 2-byte integer value field containing the number of ports (ring stations) in the bridge.

Partition Bits Subvector

This subvector has a 1-byte integer value field specifying the number of bits reserved in the routing information field for each bridge identifier (see "Routing Information Field" on page 2-6).

Path Trace Subvector

This subvector has a 1-byte value field specifying whether or not the LAN bridge server is enabled to send Path Trace Report frames to LAN managers when the LAN bridge server receives a frame for forwarding with bit 1 of the routing control field in the routing information field set to B'1' (see "Routing Information Field" on page 2-6). If the value field of this subvector is not X'00', Path Trace Report frames are sent to LAN managers (through the LAN reporting mechanism) for these frames.

Calculation Interval Subvector

This subvector has a 2-byte integer value field indicating the frequency with which the LAN bridge server calculates the percentage of frames lost or discarded by the bridge. This interval is defined in seconds; that is, a value of X'0001' specifies that the LAN bridge server will calculate the percentage every second.

Notification Interval Subvector

This subvector has a 2-byte integer value field indicating the frequency at which the LAN bridge server sends Bridge Counter Report frames to the LAN manager. This interval is defined in seconds; that is, a value of X'0001' specifies that the LAN bridge server will send the counters every second.

Percent Frames Lost Threshold Subvector

This subvector has a 2-byte integer value field that contains the threshold value for the following ratio: the number of frames lost or discarded by the bridge during the last calculation interval divided by the number of frames destined to be forwarded through the bridge during that interval. This value is expressed as a percentage (it is multiplied by 100) and is used to trigger the bridge performance threshold exceeded notification (see "Bridge Performance Monitoring Function" on page 18-1).

Percent Frames Lost Subvector

This subvector has a 2-byte integer value field indicating the ratio of the sum of the frames lost or discarded by the bridge during the last calculation interval divided by the number of frames destined to be forwarded through the bridge during the last calculation interval. This ratio is expressed as a percentage (it is multiplied by 100).

Port Information Subvector

This subvector has a complex value field containing the following information about a port. One port information subvector is present for each MAC instance used by the LAN bridge server.

Port Identifier Subvector

This subvector has a 6-byte value field containing the MAC address of the port (see "Addresses" on page 3-9). Using the MAC address to identify a port ensures unique port identifiers.

Ring Number Subvector

This subvector has a 2-byte integer value field containing the number of the ring to which the station specified by the port identifier is attached.

Port Type Subvector

This subvector has a 2-byte bit string value field specifying the type of port. Its value is X'0001' for a ring station.

Ring Data Rate Subvector

This subvector has a 2-byte bit-string value field that indicates the data rate of the ring or bus to which this port is attached. Its value is indicated in megabits per second as shown in the following:

- X'0002' is 2 Mbps
- X'0004' is 4 Mbps
- X'0010' is 16 Mbps.

Ring Status Subvector

This subvector has a 2-byte value indicating the operational status of the ring to which the station specified by the port identifier subvector is attached. Its value is one of the following:

X'00' - The ring is operational.
X'02' - The ring is in a beaconing condition.

Adapter Status Subvector

This subvector has a 2-byte bit string value field specifying the status of the station for the port described in this port information structure. Its values are:

X'0000' — Attached to the ring
X'0001' — Not attached to the ring.

Port Hop Count Subvector

This subvector has a 1-byte integer value field that is used to determine if a broadcast frame can be forwarded by a bridge. If the number of bridges that a broadcast frame has traversed before getting to this bridge is greater than or equal to the port hop count limit, the frame is not forwarded by this bridge (see "Source Routing" on page 3-2).

Frames Discarded Counter Value Subvector

This subvector has a 2-byte integer value field containing the number of frames received by this bridge port and not forwarded onto the target ring because the target ring was inoperative (beaconing or wire fault at the bridge port is occurring).

T-Frames Discarded Counter Value Subvector

This subvector has a 2-byte integer value field indicating the number of frames received by this bridge port and not forwarded onto the target ring because the target ring was inoperative *during the last calculation period.*

Frames Discarded Counter Value Subvector (long)

This subvector has a 4-byte integer value field containing the number of frames received by this bridge port and not forwarded onto the target ring because the target ring was inoperative due to beaconing or wire fault at the bridge port.

B-Frames-Transmitted Counter Value Subvector

This subvector has a 4-byte integer value field containing the number of broadcast (B) frames received by this port and forwarded through this bridge.

T-B-Frames-Transmitted Counter Value Subvector

This subvector has a 4-byte integer value field indicating the number of broadcast (B) frames received by this port and forwarded through this bridge *during the last calculation period.*

B-Bytes-Transmitted Counter Value Subvector

This subvector has a 6-byte integer value field containing the number of bytes received by this port and forwarded by this bridge in broadcast frames.

T-B-Bytes-Transmitted Counter Value Subvector

This subvector has a 6-byte integer value field containing the number of bytes received by this port and forwarded by this bridge in broadcast frames *during the last calculation interval.*

NB-Frames-Transmitted Counter Value Subvector

This subvector has a 4-byte integer value field containing the number of non-broadcast (NB) frames received by this port and forwarded through this bridge.

T-NB-Frames-Transmitted Counter Value Subvector

This subvector has a 4-byte integer value field containing the number of non-broadcast (NB) frames received by this port and forwarded through this bridge *during the last calculation interval.*

NB-Bytes-Transmitted Counter Value Subvector

This subvector has a 6-byte integer value field containing the number of bytes received by this port and forwarded by this station in non-broadcast frames.

T-NB-Bytes-Transmitted Counter Value Subvector

This subvector has a 6-byte integer value field containing the number of bytes received by this port and forwarded by this station in non-broadcast frames *during the last calculation interval.*

Frames-Not-Received Counter Value Subvector

This subvector has a 2-byte value field containing a count of the number of frames that were not copied by a bridge station because it was congested. These frames have their frame status field A bits set and their C bits *not* set (see "Frame Status Field" on page 2-14).

T-Frames-Not-Received Counter Value Subvector

This subvector has a 2-byte value integer field indicating the number of frames that were not copied by a bridge port because it was congested *during the last calculation interval.*

Frames-Not-Received Counter Value Subvector (long)

This subvector has a 4-byte value field containing a count of the number of frames that were not copied by a bridge station because it was congested. These frames have their frame status field A bits set and their C bits *not* set (See "Frame Status Field" on page 2-14).

Frames-Not-Forwarded Counter Value Subvector

This subvector has a 2-byte value field containing a count of the number of frames that were received by this port and not forwarded because one of the following conditions was detected:

- Frame too short
- Frame too long (see "Largest Frame Size" below)
- Duplicate ring numbers in routing information field
- Invalid routing information field
  - Length of routing information field is odd
  - Length of routing information field in routed non-broadcast frame is less than 6
  - Length of routing information field in broadcast frame is less than 2
  - Source ring number of broadcast frame is not present in the routing information field.

T-Frames-Not-Forwarded Counter Value Subvector

This subvector has a 2-byte integer value field indicating the number of frames that were received by this port and not forwarded for one of the reasons indicated above in the description of the frames-not-forwarded counter value subvector *during the last calculation interval.*

Frames-Not-Forwarded Counter Value Subvector (long)

This subvector has a 4-byte value field containing a count of the number of frames that were received by this port and not forwarded because one of the following conditions was detected:

- Frame too short
- Frame too long
- Duplicate ring numbers in routing information field
- Invalid routing information field
  - Length of routing information field is odd
  - Length of routing information field in routed non-broadcast frame is less than 6
  - Length of routing information field in broadcast frame is less than 2
  - Source ring number of broadcast frame is not present in the routing information field.

Frames Discarded - Internal Error

This subvector has a 4-byte value field containing a count of the number of frames that were copied into the bridge but not forwarded to the target ring or bus because of an error internal to the bridge.

T-Frames Discarded - Internal Error

This subvector has a 4-byte value field containing a count of the number of frames that were copied into the bridge but not forwarded to the target ring or bus because of an error internal to the bridge *during the last calculation interval.*

Bytes Discarded - Internal Error

This subvector has a 6-byte integer value field containing the number of bytes copied into the bridge but whose entire frame was not forwarded to the target ring or bus because of an internal error in the bridge.

Frames Not Routed Across Bridge Subvector

This subvector has a 4-byte integer value field indicating the number of frames received by an adapter that were not supposed to be routed through the bridge.

Note: This counter applies to bridge ports connected to CSMA/CD LANS, because they copy all frames on the LAN before deciding whether to forward the frame to the target LAN.

T-Frames Not Routed Across Bridge Subvector

This subvector has a 4-byte integer value field indicating the number of frames received by an adapter but were not supposed to be routed through the bridge during the last calculation interval.

Note: This counter applies to bridge ports connected to CSMA/CD LANS, because they copy all frames on the LAN before deciding whether to forward the frame to the target LAN.

Single-Route Broadcast Enabled/Disabled Subvector

This subvector has a 1-byte integer value field that is used to determine whether or not this bridge port is part of a single-route broadcast path. If its value is *not* X'00', this route will accept and forward single-route broadcast frames (see "Source Routing" on page 3-2).

Route Status Subvector

This subvector has a value field containing information about a route through a bridge. A route status subvector is present for each route in a bridge and contains the following nested subvectors:

Route Identifier Subvector

This subvector has a value field and is present for each route through a bridge. The route identifier uniquely identifies a route in a bridge and contains two nested subvectors.

Port Identifier Subvector This subvector has a 6-byte value field containing the MAC address of the entry port in the route. Using the MAC address to identify a port ensures unique port identifiers.

Port Identifier Subvector This subvector has a 6-byte value field containing the MAC address of the exit port in the route.

Note: The route composed of adapters A and B can be represented as (A,B) or (B,A). Routes, by convention, are identified by identifying the port with the numerically lower MAC address first.

Bridge Identifier Subvector

This subvector has a 2-byte integer value field that uniquely identifies a bridge among multiple bridges spanning the same two rings. This allows parallel bridges to be uniquely identified.

Largest Frame Size Subvector

This subvector has a 1-byte integer value field that specifies the largest frame that will be forwarded by a bridge. This value may reflect the largest frame that can be transmitted on an attached segment or the buffer size restrictions in the bridge itself (see "Routing Information Field" on page 2-6).

Route Active Status Subvector

This subvector has a 1-byte bit string value field that indicates the frame-forwarding status of the route. Its values are:

- X'00' — Active status. Frames are being forwarded between the two segments identified in this route.
- X'01' — Inactive status. Frames are not being forwarded between the two segments identified in this route.

Single-Route Broadcast Mode Subvector

This subvector has a 1-byte value field (bit string) indicating if the bridge is participating in the automatic calculation of the single-route broadcast path in the LAN. Its values are:

- X'00' - The bridge is manually configured for single-route broadcast
- *not* X'00' - The bridge is configured to participate in the spanning tree protocols to automatically calculate the single-route broadcast path in the LAN.

Ring Number Subvector

This subvector has a 2-byte value field that contains the number of the ring into which a station is inserted. A Ring Number parameter is maintained for each port in the bridge.

Forwarded-Frame Addressing Information Subvector

This subvector has a value field up to 30 bytes long that consists of the following subfields:

- Forwarded-Frame Destination Address:
  A 6-byte (bit string) value field containing the MAC address contained in the destination address field of a frame that was received by the bridge for routing to another ring.
- Forwarded-Frame Source Address:
  A 6-byte (bit string) value field containing the MAC address contained in the source address field of a frame that was received by the bridge for routing to another ring.
- Forwarded-Frame Routing Information:
  A variable-length value field containing the routing information contained in the routing information field of a frame that was received by the bridge for routing to another ring.

Forwarded-Frame Length Subvector

This subvector has a 2-byte integer value field containing the length of the frame received by the bridge for routing to another ring. This length includes the length of the entire MAC frame including the MAC header and trailer fields.

Forwarded-Frame Data Subvector

This subvector has a 10-byte value field containing the first 10 bytes of the information field of the MAC frame (or the entire information field, if it is less than 10 bytes long). This value includes the LLC DSAP, SSAP, LLC control field, and 6 bytes of the LLC information field, but does not contain the routing information field or the MAC headers. These fields are described in Chapter 2, "MAC Frame Format" and Chapter 8, "LLC Frames." The 6 bytes of the LLC information field are intended to allow the recipient of Path Trace Report frames (that contain this information) to identify the data frame for which this report was generated.

Forwarded-Frame Status Subvector

This subvector has a 2-byte bit-significant value field indicating the strip status of a transmitted frame (see "Transmit Status Code, X'2A'" on page 5-23).

Bridge Internal Status Subvector

This subvector has a variable-length value field containing internal status variables used by the LAN bridge server. The format of this subvector is implementation dependent.

Temporary/Permanent Subvector

This subvector has a 1-byte value field that indicates whether the LAN Bridge Server's parameters have been permanently changed or merely changed until the LAN Bridge Server is no longer operational. That is, a temporary change would not persist when the LAN Bridge Server became operational again. A value of X'00' indicates that the change to the parameter values is intended to be permanent. Any other value for the subvector indicates that the change is intended to be temporary. If this subvector is not present, the change is permanent.

Request Bridge Status (X'8501')

This frame is sent by a LAN manager to a LAN bridge server to request certain status information.

```
Request Bridge Status
   Correlator
   All
   All (long counters)
```

Figure 18-2. Request Bridge Status Frame

Correlator Subvector

This subvector allows a LAN manager to correlate the response frames that it receives with the Request Bridge Status frames that it sends. For a frame sequence to be valid, the correlator subvector value in the response must match the value that was sent in the Request Bridge Status frame.

All Subvector

If this subvector is present within the Request Bridge Status major vector, the response contains all of the values for each of the parameters listed within the Report Bridge Status major vector (shown within Figure 18-3 on page 18-12).

All (long counters) Subvector

The All (long counters) subvector is the same as the All subvector except that the response will contain the long version of the bridge counters.

Parsing Algorithm

Before executing the Request Bridge Status, the LAN bridge server checks it for errors using the parsing algorithm shown in "Parsing Algorithm for Request Status Frames" on page E-2.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Request Bridge Status (X'8501') Conditions of Presence" on page F-13.

Responses

If the Request Bridge Status frame does not contain syntactic or semantic errors and no errors occur while executing the command, the LAN bridge server returns the requested information in a Report Bridge Status frame (see "Report Bridge Status (X'8502')" on page 18-12).

If an error is detected in the Request Bridge Status frame, the Bridge Error Frame is returned to the requesting LAN manager (see "Bridge Error (X'8505')" on page 18-17).

Report Bridge Status (X'8502')

This frame is sent by a LAN bridge server to a LAN manager in response to a Request Bridge Status frame if no errors are detected in the request frame (see "Request Bridge Status (X'8501')" on page 18-11).

```
Report Bridge Status
   Correlator
   Bridge Type
   Bridge Version Level
   Number of Ports
   Partition Bits
   Path Trace
   Calculation Interval
   Notification Interval
   Percent Frames Lost Threshold
   Port Information
      Port Identifier
      Ring Number
      Port Type
      Ring Data Rate
      Ring Status
      Adapter Status
      Port Hop Count
      Frames Discarded Counter Value
      Frames Discarded Counter Value (long)
      B-Frames-Transmitted Counter Value
      B-Bytes-Transmitted Counter Value
      NB-Frames-Transmitted Counter Value
      NB-Bytes-Transmitted Counter Value
      Frames-Not-Received Counter Value
      Frames-Not-Received Counter Value (long)
      Frames-Not-Forwarded Counter Value
      Frames-Not-Forwarded Counter Value (long)
      Frames Discarded - Internal Error
      Bytes Discarded - Internal Error
      Frames Not Routed Across Bridge
      Single-Route Broadcast Enabled/Disabled
   Route Status
      Route Identifier
         Port Identifier
         Port Identifier
      Bridge Identifier
      Largest Frame Size
      Route Active Status
      Single-Route Broadcast Mode
   Bridge Internal Status
```

Figure 18-3. Report Bridge Status Frame

Correlator

The LAN manager uses the value of this subvector to associate a Report Bridge Status frame with the corresponding Request Bridge Status frame.

Other Subvectors

The information requested in the Request Bridge Status frame is returned in the remaining subvectors in this frame.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Report Bridge Status (X'8502') Conditions of Presence" on page F-14.

Set Bridge Parameters (X'8503')

This frame is sent by the controlling LAN manager to set the operational parameters of a LAN bridge server. The subvectors in this frame specify the parameters to be set and the values to which they are to be set.

```
Set Bridge Parameters
   Correlator
   Notification Interval
   Percent Frames Lost Threshold
   Port Information
      Port Identifier
      Port Hop Count
      Single-Route Broadcast Enabled/Disabled
   Route Status
      Route Identifier
         Port Identifier
         Port Identifier
      Bridge Identifier
      Largest Frame Size
      Route Active Status
      Single-Route Broadcast
   Bridge Internal Status
   Temporary/Permanent
```

Figure 18-4. Set Bridge Parameters Frame

Correlator Subvector

This subvector allows a LAN manager to correlate responses it receives with the Set Bridge Parameters frame that it sends. For a frame sequence to be valid, the correlator subvector value in the bridge response frame must match the value that was sent in the associated Set Bridge Parameters frame.

Parsing Algorithm

Before executing the Set Bridge Parameters, the LAN bridge server checks it for errors using the parsing algorithm shown in "Parsing Algorithm for Set Parameter Frames" on page E-3.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Set Bridge Parameters (X'8503') Conditions of Presence" on page F-15.

Responses

If the Set Bridge Parameters frame does not contain any parsing errors, the LAN bridge server attempts to set the values indicated and sends an Bridge Parameters Set frame to the requesting LAN Manager (see "Bridge Parameters Set (X'8504')" on page 18-16).

The LAN bridge server also notifies the other LAN managers that the values of its operational parameters have been changed by sending a Bridge Parameters Changed Notification (see "Bridge Parameters Changed Notification (X'8506')" on page 18-18).

If an error is detected when parsing the Set Bridge Parameters frame or while executing the command, the LAN bridge server sends a Bridge Error frame to the requesting LAN manager (see "Bridge Error (X'8505')" on page 18-17).

Bridge Parameters Set (X'8504')

This frame is sent by a LAN bridge server to the controlling LAN manager to indicate the successful receipt and execution of a Set Bridge Parameters frame.

```
Bridge Parameters Set
   Correlator
```

Figure 18-5. Bridge Parameters Set Frame

Correlator Subvector

The LAN manager uses the value of this subvector to associate a Set Bridge Parameters frame with the corresponding Bridge Parameters Set frame.

Conditions of Presence

The correlator subvector is always present.

Bridge Error (X'8505')

This frame is sent by a LAN bridge server to the requesting LAN manager if an error is detected in a Request Bridge Status (see "Request Bridge Status (X'8501')" on page 18-11) or a Set Bridge Parameters frame (see "Set Bridge Parameters (X'8503')" on page 18-14).

```
Bridge Error
   Correlator
   Error Code
   Error Offset
   Bridge Version Level
```

Figure 18-6. Bridge Error Frame

Correlator Subvector

This subvector has the same value as the correlator subvector in the Request Bridge Status frame or Set Bridge Parameters frame to which this Bridge Error frame is a response. The correlator subvector allows the LAN manager to correlate the error response with the original request.

If the correlator subvector was not present in the Request Bridge Parameters frame or the Set Bridge Parameters frame, an error code of *unrecognized subvector* and a correlator value of X'00000000' are returned in the Bridge Error frame.

Error Code Subvector

This subvector contains a 2-byte diagnostic code that describes the reason that an error was detected in the corresponding Request Bridge Parameters frame or Set Bridge Parameters frame. This code refers to the *first* error detected. The valid values for syntax errors are listed in Figure E-1 on page E-2 and Figure E-2 on page E-3. If the error was detected while executing the command, the subvector value is X'000C'.

Error Offset Subvector

This value of this subvector is the offset (in number of bytes) into the corresponding frame where the *first* error was detected. The offset is a 2-byte integer and is counted from the first byte of the major vector identifier.

Bridge Version Level Subvector

This subvector can be used by the LAN manager to help determine the reason for the error.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Bridge Error (X'8505') Conditions of Presence" on page F-15.

Bridge Parameters Changed Notification (X'8506')

This frame is sent by the LAN bridge server to LAN managers, (through the LAN reporting mechanism) when the values of its operational parameters are changed by the controlling LAN manager. This frame is *not* sent to the controlling LAN manager that requested the change.

The information in this frame is identical to the information in the Set Bridge Parameters frame that caused it to be sent except that the correlator subvector is not present.

```
Bridge Parameters Changed Notification
   Notification Interval
   Percent Frames Lost Threshold
   Port Information
      Port Identifier
      Port Hop Count
      Single-Route Broadcast Enabled/Disabled
   Route Status
      Route Identifier
         Port Identifier
         Port Identifier
      Bridge Identifier
      Largest Frame Size
      Route Active Status
      Single-Route Broadcast Mode
   Bridge Internal Status
```

Figure 18-7. Bridge Parameters Changed Notification Frame

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Bridge Parameters Changed Notification (X'8506') Conditions of Presence" on page F-16.

Bridge Counter Report (X'8509')

This frame is sent periodically by the LAN bridge server to LAN managers (through the LAN reporting mechanism) at the rate specified in the LAN bridge server notification interval subvector.

```
Bridge Counter Report
   Port Information
      Port Identifier
      Ring Number
      Frames Discarded Counter Value
      Frames Discarded Counter Value (long)
      B-Frames-Transmitted Counter Value
      B-Bytes-Transmitted Counter Value
      NB-Frames-Transmitted Counter Value
      NB-Bytes-Transmitted Counter Value
      Frames-Not-Received Counter Value
      Frames-Not-Received Counter Value (long)
      Frames-Not-Forwarded Counter Value
      Frames-Not-Forwarded Counter Value (long)
      Frames Discarded - Internal Error
      Bytes Discarded - Internal Error
      Frames Not Routed Across Bridge
      Frames Filtered at Bridge
```

Figure 18-8. Bridge Counter Report

Port Information Subvector

This subvector contains the values of the bridge error and traffic counters maintained by the LAN bridge server.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Bridge Counter Report (X'8509') Conditions of Presence" on page F-17.

Path Trace Report (X'8508')

This frame is sent by the LAN bridge server to either the controlling LAN manager or the originating station when the bridge receives a frame with bits 1 or 2 of the routing control field set to B'1' and the bridge is configured to send path trace report frames. See "Routing Information Field" on page 2-6 and "Path Trace Subvector" on page 18-4 for details.

If bit 1 of the routing control field is set to B'1', and the path trace parameter in the LAN bridge server is not X'00', the Path Trace Report frame is sent to LAN managers through the collocated LAN reporting mechanism (see Chapter 14, "LAN Reporting Mechanism").

If bit 2 of the routing control field is set to B'1', the Path Trace Report frame is sent using LLC type-1 service (Chapter 9, "Connectionless Service") to the X'F4' LSAP at the originating station of the frame (for the LSAP assignments, see "DSAP Address Field" on page 8-1).

```
Path Trace Report
   Forwarded-Frame Addressing Information
   Forwarded-Frame Length
   Forwarded-Frame Data
   Forwarded-Frame Strip Status
   Partition Bits
   Port Information
      Port Identifier
      Ring Status
      Adapter Status
      Port Hop Count
   Route Status
      Route Identifier
         Port Identifier
         Port Identifier
      Bridge Identifier
      Largest Frame Size
      Route Active Status
```

Figure 18-9. Path Trace Report Frame

Note: LAN bridge servers can be enabled or disabled for sending Path Trace Report frames when the bridge forwards a frame with the bit 1 of the routing-control field set to B'1'. If the bridge receives a frame (for forwarding to a target ring) that has bit 1 of the routing-control field set to B'1' and it has not been configured to send Path Trace Report frames, the frame is forwarded but no report is sent to the LAN manager. The bridge server always sends a Path Trace Report frame to the originating station when it forwards (or receives for forwarding) a frame that has bit 2 of the routing control field set to B'1'.

Subvectors

The Path Trace Report frame contains information about the frame that was forwarded through the bridge (or received for forwarding through the bridge), the route taken within the bridge, and the status of the two rings connected through this route. Information about the status on the target segment is also reported.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Path Trace Report (X'8508') Conditions of Presence" on page F-17.

Bridge Performance Threshold Exceeded (X'8507')

This frame is sent by the LAN bridge server to LAN managers (through the LAN reporting mechanism) when its calculated ratio of frames lost or discarded by the bridge exceeds a predetermined threshold (defined in the percent frames lost threshold subvector).

```
Bridge Performance Threshold Exceeded
   Percent Frames Lost Threshold
   Percent Frames Lost
   Port Information
      Port Identifier
      Ring Number
      T-Frames Discarded Counter Value
      T-B-Frames-Transmitted Counter Value
      T-NB-Frames-Transmitted Counter Value
      T-Frames-Not-Received Counter Value
      T-Frames-Not-Forwarded Counter Value
      T-Frames Discarded - Internal Error
      T-Frames Not Routed Across Bridge
      T-B-Bytes-Transmitted Counter Value
      T-NB-Bytes-Transmitted Counter Value
```

Figure 18-10. Bridge Performance Threshold Exceeded Frame

Percent Frames Lost Subvector

The value of this subvector is the ratio of lost or discarded frames to forwarded frames. For this frame to be sent, this subvector's value must be greater than or equal to the percent frames lost threshold subvector.

Port Information Subvector

This subvector contains the information used by the LAN bridge server to calculate the percent frames lost ratio.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Bridge Performance Threshold Exceeded (X'8507') Conditions of Presence" on page F-16.

Single-Route Broadcast Status Change (X'850A')

The bridge server sends a Single-Route Broadcast Status Changed frame to LAN managers when it begins or ceases forwarding single-route broadcast frames and is participating in the automatic spanning-tree calculation of the single-route broadcast path through the LAN. The contents of the Single-Route Broadcast Status Changed frame are shown in Figure 18-11.

```
Single-Route Broadcast Status Changed
   Port Information
      Port Identifier
      Single-Route Broadcast Enabled/Disabled
```

Figure 18-11. Single-Route Broadcast Status Changed Frame

Note: All of the subvectors must be present in this frame.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Single-Route Broadcast Status Change (X'850A') Conditions of Presence" on page F-17.

Chapter 19. Token-Ring Network Alerts

The LAN manager uses *alerts* to notify an SNA network operator of problems in a token-ring network and to provide information that will assist the operator in isolating and resolving these problems. By notifying a central network operator of problems, multiple local area networks can be managed without requiring a LAN manager operator for each local area network.

Conditions for Sending Alerts

Alerts are notifications that are sent to an SNA control point (see *SNA Format and Protocol Reference Manual: Management Services*, SC30-3346) to notify a network operator of problems and impending problems. The conditions for which alerts are sent must meet the following criteria:

- The condition must be an *error condition*. Information about normal, error-free operations is not transported in alerts.
- The condition must either affect availability (a problem) or threaten to affect availability (an impending problem). Availability is affected not only when a resource is totally inoperative, but also when its performance is severely degraded.
- The condition cannot be resolved by local action. If a node is not attended and requires some action to resolve a problem, an alert is sent to inform the network operator of the problem.

Information Contained in Alerts

Alerts contain error information about the entire local area network, the ring, and, where possible, the station or stations experiencing problems. This Token-Ring Network specific information is transported in the alert in the LAN Link Connection Subsystem Data (X'51') subvector (see *SNA Formats*, GA27-3136).

Single-Ring Alert Conditions

The alert conditions defined in this section pertain to a single ring, even though this single ring may be part of a multi-ring network. In a multi-ring network, additional alert conditions are defined (see "Multiple-Ring Alert Conditions" on page 19-2). The alert conditions for a token ring are:

1. The reporting node's ring station detected a problem on its lobe during the wrap-test portion of the attachment process. (This is an alert condition only if the reporting node is not attended.)
2. The reporting node's ring station detected a beaconing condition on the ring during the attachment process. (This is an alert condition only if the reporting node is not attended.)
3. The reporting node's ring station detected the presence of a station with its individual address on the ring during the attachment process. (This is an alert condition only if the reporting node is not attended.)
4. The reporting node's ring station received a Remove Ring Station MAC frame during the attachment process. (This is an alert condition only if the reporting node is not attended.)

5. The reporting node's ring station detected an error during the attachment process that is not defined in 1, 2, 3, or 4. (This is an alert condition only if the reporting node is not attended.)
6. The reporting node's ring station has detected a wire-fault condition on its lobe.
7. The reporting node's ring station has left the ring as part of the beacon automatic-recovery process (see "Hard-Error Processing Function" on page 15-1). That is, the ring station was a member of the fault domain and left the ring to perform a test of itself and its lobe, which was unsuccessful.
8. The reporting node's ring station received a Remove Ring Station MAC frame and, as a result, left the ring.
9. A ring has been in a beaconing condition for a period longer than the permanent-error detection timer (see "Hard-Error Processing Function" on page 15-1). Manual intervention is required to recover the ring.
10. A ring was in a beaconing condition for a period shorter than the permanent-error detection timer (see "Hard-Error Processing Function" on page 15-1). When the stations in the fault domain were queried, one or both of them had left the ring.
11. A ring was in a beaconing condition for a period shorter than the hard-error detection timer. The reporting node has either queried both fault domain stations and determined that neither left the ring to resolve the beacon condition or does not implement the function of querying the fault domain stations after the beacon condition ceases.
12. The ring error monitor has detected excessive soft errors on a ring.
13. The ring error monitor has detected excessive receiver congestion errors for a station on a ring.
14. The LAN manager did not receive a response from a monitored adapter. The adapter may have left the network or there may be a problem between the LAN manager and the adapter.
15. An optical fiber repeat subsystem has reported that the main ring path has wrapped onto the backup path.
16. An optical fiber repeat subsystem has reported that the backup ring path has been in a beaconing condition for more than six minutes.

Multiple-Ring Alert Conditions

In addition to the alerts described in "Single-Ring Alert Conditions," the *controlling* LAN manager sends alerts for the following conditions if it has reporting links with remote management servers (see Chapter 14, "LAN Reporting Mechanism").

1. An abnormally large percentage of frames are being discarded at a bridge. The LAN bridge server has calculated the ratio of the number of frames it discarded to the number of frames destined to be forwarded through the bridge and the ratio exceeded a predefined threshold.
2. The bridge was taken offline by an operator. The shut-down was orderly and the LAN bridge server issued a frame warning LAN managers that the bridge was being removed from the local area network. Bridge frame-forwarding functions were terminated. That is, either an operator at the bridge or at a controlling LAN manager issued a Set Bridge Parameters frame (see "Set Bridge Parameters (X'8503')" on page 18-14) to set the route active parameter of the LAN bridge server to cause the bridge not to forward frames.

3. The controlling reporting link to a remote management server has been lost. The remote link station does not respond. The inactivity timer (Ti) or acknowledgment timer (T1) has expired, causing the remote station to be polled. The remote station does not respond to the poll. (See Chapter 11, "Operation of Link Stations" for details about connection-oriented LLC operation.)
4. The controlling reporting link to a remote management server has been lost. The remote link station sent a disconnect mode response to the local link station. The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
5. The controlling reporting link to a remote management server has been lost. The local link station sent an invalid or unsupported command or response to the remote link station. This resulted in the remote link station returning a Frame Reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
6. The controlling reporting link to a remote management server has been lost. The local link station sent an I-frame when not permitted to do so to the remote link station. This resulted in the remote link station returning a Frame Reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
7. The controlling reporting link to a remote management server has been lost. The local link station sent a frame with an invalid N(r). This resulted in the remote link station returning a frame reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
8. The controlling reporting link to a remote management server has been lost. The local link station sent a frame with an I-field that was too long. This resulted in the remote link station returning a Frame Reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
9. The controlling reporting link to a remote management server has been lost. The remote link station sent an invalid or unsupported frame to the local link station. This resulted in the local link station returning a frame reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
10. The controlling reporting link to a remote management server has been lost. The remote link station sent an I-frame when not permitted to do so to the local link station. This resulted in the local link station returning a Frame Reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
11. The controlling reporting link to a remote management server has been lost. The remote link station sent a frame with an invalid N(r). This resulted in the local link station returning a frame reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.
12. The controlling reporting link to a remote management server has been lost. The remote link station sent a frame with an I-field that was too long. This

resulted in the local link station returning a frame reject response (see "FRMR Exception Conditions" on page 11-26). The controlling LAN manager tried to re-establish the link after a predetermined time and the attempt was unsuccessful.

13. A remote management server has been unable to receive data on the link and has been sending Receive Not Ready frames continuously for more than 30 seconds (see "Receiving an RNR LPDU" on page 11-21).
14. A remote management server has been congested and, as a result, has disabled some of its management function and probably discarded management data. The LAN reporting mechanism sent a LRM Congestion frame to the LAN manager (see "LRM Congestion Frame (X'8612')" on page 14-27).
15. A LAN manager attempted to establish a reporting link with management services but was rejected because it used an invalid password.

Alert Transport Service

Alert Transport Service is a facility contained within a LAN Manager that provides an "alert passthrough" to an SNA network operator. Devices and software on the LAN that experience error conditions can build alert messages and send them over the LAN to the LAN Manager. The LAN Manager receives the alert messages and forwards them "as is" to the host. The LAN Manager can also log and display the alert message locally.

Alert Transport (X'8701')

This frame is sent by LAN devices to the LAN manager to report error conditions.

Figure 19-1. Alert Transport Frame

Correlator Subvector

This subvector is always present and allows the sending device to correlate the response to this frame.

Alert Major Vector Subvector

This subvector identifies the problem and information related to the problem. The information is provided to allow the recipient of the problem report to diagnose and possibly rectify the problem. These pieces of information are contained in Alert subvectors which make up the actual alert (Alert Major Vector).

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Alert Transport (X'8701') Conditions of Presence" on page F-24.

Alert Transport Received(X'8702')

This frame is sent by a LAN manager to a LAN device in response to a received Alert Transport frame.

```
Alert Transport Received
   Correlator
```

Figure 19-2. Alert Transport Received Frame

Correlator Subvector

The LAN device uses the value of this subvector to correlate an Alert Transport Received frame with the associated Alert Transport frame.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Alert Transport Received (X'8702') Conditions of Presence" on page F-24.

Converter Status Frames

These frames are sent by an optical fiber converter to the LAN manager to indicate status information. The LAN manager uses these frames to generate alerts.

Downstream Converter Presence (X'8402')

This is a unique downstream converter LLC frame to alert the LAN manager that a downstream converter is on the main ring. The downstream converter transmits this frame at 60 second intervals.

Figure 19-3. Downstream Converter Presence Frame

Converter's Partner Address Subvector

This subvector has a field that contains the MAC address of this converter's upstream partner.

Conditions of Presence

The conditions of presence for this frame's subvector are shown in "Downstream Converter Presence (X'8402') Conditions of Presence" on page F-12.

Beaconing Back-up Ring (X'8403')

The upstream converter transmits this frame when the downstream converter is determined to be in a steady beacon transmit state. The upstream converter transmits this frame at 60 second intervals until the problem is resolved.

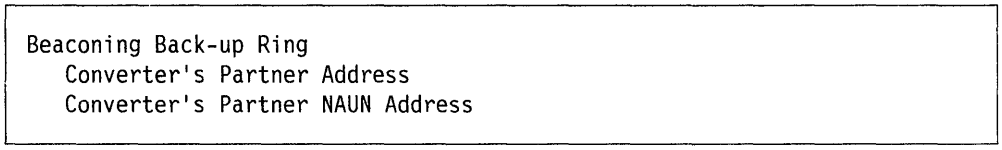


Figure 19-4. Beaconing Back-up Ring Frame

Converter's Partner Address Subvector

This subvector has a field that contains the MAC address of this converter's downstream partner.

Converter's Partner NAUN Address Subvector

This subvector has a field that contains the MAC address of the Nearest Active Upstream Neighbor (NAUN) of the converter's downstream partner.

Conditions of Presence

The conditions of presence for this frame's subvectors are shown in "Beaconing Back-Up Ring (X'8403') Conditions of Presence" on page F-12.

Chapter 20. LAN Configurations

This chapter shows example configurations of LAN managers and management servers in an IBM Token-Ring Network. The sequence progresses from the simplest configuration to more complex configurations.

Single Ring with No LAN Manager

In this configuration, the ring stations (S) use default values for their parameters. Reports generated by the ring stations are not collected because there are no management servers on the ring.

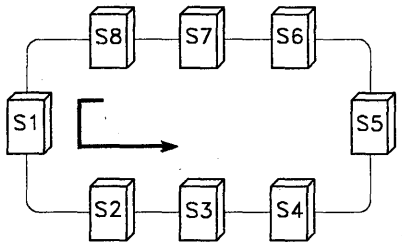


Figure 20-1. A Single Ring with No LAN Manager

Single Ring with One LAN Manager

In this configuration, the configuration report server and the ring error monitor in the LAN manager collect reports generated by the ring stations and can change station parameters and alter the ring configuration.

The ring error monitor (REM) analyzes the information contained in ring station reports and the configuration report server (CRS) collects configuration change notifications sent by ring stations. The configuration report server can also obtain information from stations on the ring, change station parameters, and force ring stations to remove themselves from the ring. The ring error monitor and the configuration report server reside in the same node as the LAN manager application.

The LAN manager may provide an operator interface to the ring error monitor and the configuration report server data and may provide more sophisticated management function.

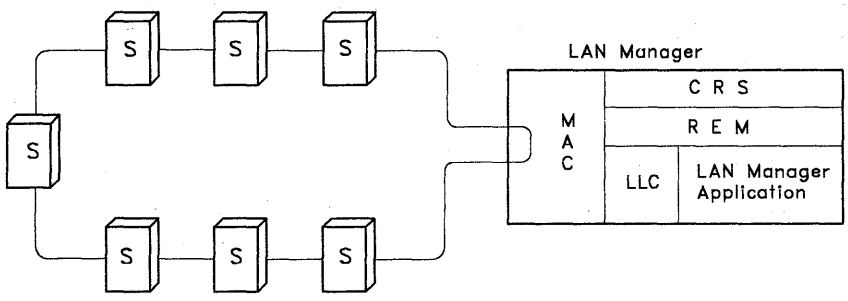


Figure 20-2. A Single Ring with One LAN Manager

Two Rings, Two LAN Managers, and Multiple Servers

In the configuration shown in Figure 20-3 on page 20-3, the *bridge* between the two rings contains five management servers: a LAN reporting mechanism (LRM), a ring error monitor (REM), a configuration report server (CRS), a ring parameter server (RPS), and a LAN bridge server (LBS). Since the ring error monitor is duplicated on ring A, LAN manager A can select one of them from which to receive reports about ring A. The LAN manager should select its own ring error monitor to reduce the traffic on the ring and to avoid relying on a remote node to send error reports across a failing communication path.

The management servers in the bridge maintain two sets of information (except for the LAN reporting mechanism), one set for each ring into which its stations are inserted.

Note: The LAN reporting mechanism (LRM) acts as a remote intermediary between the LAN manager and the other management servers with which the LAN reporting mechanism is collocated. However, the LAN reporting mechanism is not needed to mediate the communication between a management server and a LAN manager if that management server is collocated with the LAN manager. The communication between these two collocated components involves an internal protocol boundary that is specific to each implementation.

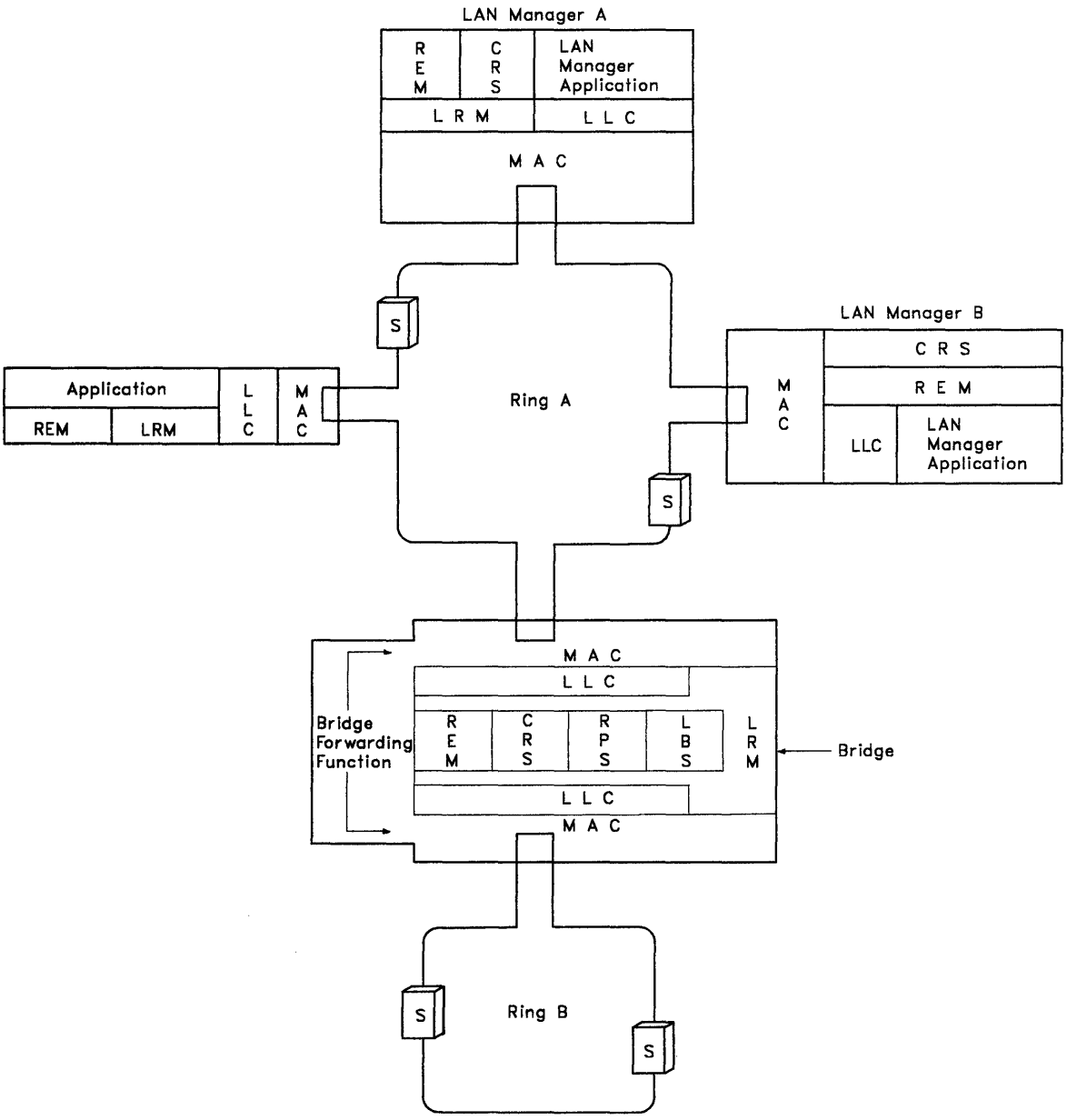


Figure 20-3. Two Rings, Two LAN Managers, and Multiple Servers

Two Rings, Two LAN Managers, Multiple Servers, and a Host

In this configuration, LAN manager A communicates with a host. LAN Manager A can report errors about Ring A, Ring B, any of the ring stations (on either ring), and the management servers in this Token-Ring Network to the SNA control point.

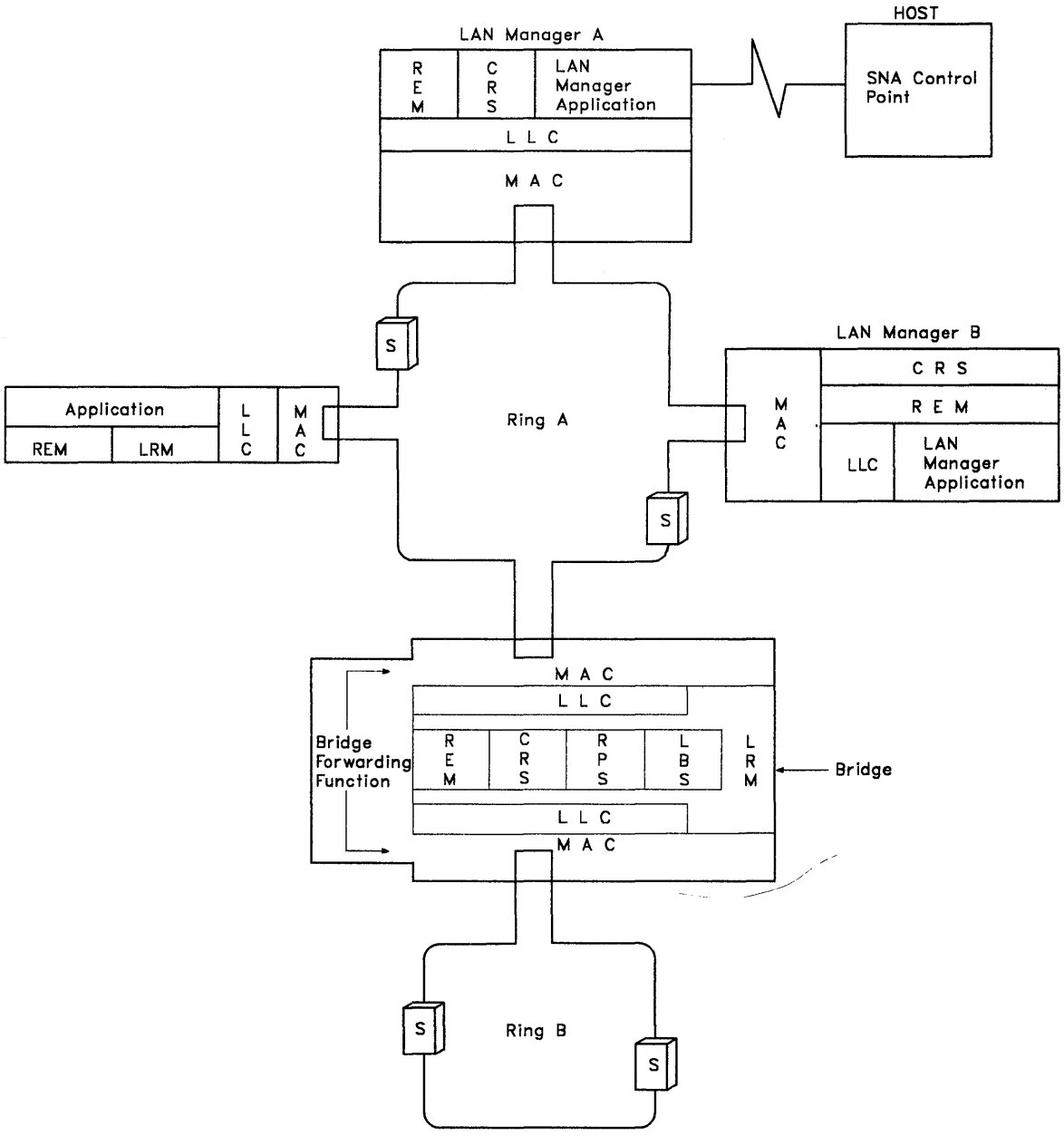


Figure 20-4. Two Rings, Two LAN Managers, Multiple Servers, and a Host

Appendix A. Token-Protocol Timers

This section describes in detail the timers necessary to operate the IBM Token-Ring Network token protocols. The use of each timer is described, along with the following characteristics:

- The *minimum duration* of the timer
- The *value* of the timer
- Actions that *start* the timer (the timer is not currently running)
- Actions that *restart* the timer (the timer is currently running, and is reset to its initial value and started again)
- Conditions that *cancel* the timer (the timer is stopped and reset to its initial value)
- *Actions* taken when the timer expires.

The duration value for the Soft Error Report timer can be changed using either of the Set Parameters MAC frames. The Soft Error Report timer is the only timer on the IBM Token-Ring Network whose value can be set.

In addition to the above characteristics, each timer's IEEE counterpart is given. Because the IEEE defines fewer timers than are required by the IBM Token-Ring Network, some of the IEEE timers serve as counterparts to more than one token-protocol timer.

The Beacon frames and counters referenced in this section, and the network management procedures that involve problems that the timers help to detect, are described in Chapter 5, "MAC Frames" on page 5-1.

T(any_token)

When a station becomes the active monitor, it activates this timer, which specifies how long the active monitor can wait without seeing a starting delimiter sequence.

IEEE COUNTERPART: Valid Transmission Timer (TVX)

MINIMUM DURATION: The greater of the following:

- The time required to propagate a frame around the ring, plus the time required to transmit the longest permitted frame at the nominal data signaling rate on the ring,

  T[physical_trailer] + max[frame length delay]

- The time required to synchronize the ring after a station attaches to it.

VALUE: 10 milliseconds

STARTED BY: The active monitor when it transmits its first token

RESTARTED BY: The active monitor when it detects a starting delimiter

CANCELED BY: The active monitor when it exits active monitor state

TIME-OUT ACTION: The active monitor increments the Token_Error counter and initiates the ring purge process.

T(attach)

This timer specifies the length of time that a station can remain in the various stages of the attachment process. See "Station Attachment Finite State Machine" on page 7-21 for the exact use of this timer.

IEEE COUNTERPART: None

MINIMUM DURATION: The longest stage of the attachment process, which depends on the number of attempts to detect a station with a duplicate address and to acquire configuration parameters. Since these numbers depend on the user's implementation, so does the value of T(attach). Its minimum value, however, must be the maximum possible time waiting for a neighbor notification (T[neighbor_notification]) plus the time a notification takes to circle the ring (also T[neighbor_notification]), that is,

2 x T[neighbor_notification].

VALUE: 18 seconds

STARTED BY: Entering Monitor Check phase of the attachment process

RESTARTED BY: Entering Duplicate Address Check phase, Neighbor Notification phase, or Request Parameters phase of the attachment process

CANCELED BY: Removing from the ring or completing the attachment process

TIME-OUT ACTION:

- In Monitor Check state: The station initiates token-claiming and enters Duplicate Address Check state.
- In Duplicate Address Check state, Neighbor Notification state, or Request Parameters state: The station removes itself from the ring.

T(beacon_transmit)

When a station starts transmitting Beacon MAC frames, it activates this timer, which specifies how long the station can transmit beacons before removing itself from the ring for testing.

IEEE COUNTERPART: None

MINIMUM DURATION: The time required for the NAUN of the beaconing ring station to remove itself from the ring, run a self-test, and attach to the ring

MINIMUM VALUE: 16 seconds

STARTED BY: Detection of the need to transmit Beacon MAC frames

RESTARTED BY: Detection of the need to transmit a different type of beacon

CANCELED BY: Receipt of a Beacon MAC frame whose source address is not the same as the station's individual address, or detection of recovery of the ring

TIME-OUT ACTION: The station removes itself from the ring and runs a self-test.

T(claim_token)

This timer specifies the length of time that a ring station in Claim Token Repeat or Transmit mode can wait for an active monitor to be established. This timer allows detection of streaming frames, streaming tokens, or streaming bits.

IEEE COUNTERPART: No Token Timer (TNT)

MINIMUM DURATION:

max[ring delay] + max[transmit delay]
+ max[transmit interval] + max[hard error detect],

where:

- Ring delay is the time for a MAC frame to circle the ring.
- Transmit delay is the time for the maximum number of ring stations to enter Claim Token Transmit mode.
- Transmit interval is the time for a ring station to transmit three consecutive Claim Token MAC frames.
- Hard-error detect is the time needed to detect a hard error.

However, the minimum duration of this timer must be greater than any one of the following:

- The time for the highest addressed ring station in Claim Token Repeat mode to await resolution of token-claiming,

  [transmit delay] + [transmit interval]

- The time for protocol detection of a hard error while in Claim Token Repeat mode,

  2 x ([ring delay] + [transmit delay])

- The time for hardware detection of a hard error (depends on the user's implementation).

VALUE: 1 second

STARTED BY: Entering Claim Token Transmit mode or Claim Token Repeat mode

RESTARTED BY: Receiving a Claim Token MAC frame

CANCELED BY: Winning token-claiming or receiving a Ring Purge MAC frame

TIME-OUT ACTION:

- In Claim Token Transmit mode: The station enters Beacon Transmit mode and transmits a Beacon MAC frame with an error code of X'0002' (if signal loss initiated token-claiming); of X'0003' (if no Claim Token MAC frames were received); or of X'0004' (if at least one Claim Token MAC frame was received).
- In Claim Token Repeat mode: The station enters Beacon Transmit mode and transmits a Beacon MAC frame with an error code of X'0004'.

T(escape)

This timer specifies how long a station can remain in Beacon Repeat mode, without receiving a Beacon MAC frame, before it must initiate token-claiming. In essence, it times the silent period after a beaconing station has stopped transmitting.

IEEE COUNTERPART: No Token Timer (TNT)

MINIMUM DURATION: The time required by a station in Beacon Repeat mode to detect subsequent Beacon MAC frames

VALUE: 200 milliseconds

STARTED BY: Receipt of a Beacon MAC frame

RESTARTED BY: Receipt of a Beacon MAC frame

CANCELED BY: Receipt of a Claim Token MAC frame, or entry into any mode other than Beacon Repeat mode

TIME-OUT ACTION: The station enters Claim Token Transmit mode.

T(good_token)

Each standby monitor uses this timer to detect frame streaming, bit streaming, and the loss of the active monitor. This timer, whose duration is much longer than T(any_token), ensures that the active monitor's token management functions are active.

IEEE COUNTERPART: No Token Timer (TNT)

MINIMUM DURATION: The maximum length of time for a token to circle a ring once,

T[token_holding] x max[number of stations],

and long enough to allow the ring purge function to fail

VALUE: 2.6 seconds

STARTED BY: Activation of the standby monitor function

RESTARTED BY: Detection of a priority-zero token or a priority-greater-than-zero token followed by a frame

CANCELED BY: Deactivation of the standby monitor function (entry into Claim Token mode or Beacon mode)

TIME-OUT ACTION: The standby monitor enters Claim Token Transmit mode.

T(neighbor_notification)

The active monitor uses this timer to pace the initiation of neighbor notification and to allow detection of a notification failure. Neighbor notification is described on page 3-20.

IEEE COUNTERPART: Active Monitor Timer (TAM)

MINIMUM DURATION: A reasonable maximum time for a notification to circle a ring with a maximum number of stations. The actual maximum is unbounded, since continuous use of a token with a high-access priority can prevent a station from transmitting a Standby Monitor Present MAC frame at priority three. Therefore, a reasonable maximum must be estimated. If the actual maximum, at a given time, is greater than T[neighbor_notification], the transmission of an Active Monitor Present MAC frame at priority B'111' simply interrupts the progression of the token.

VALUE: 7 seconds

STARTED BY: Transmission of the first Active Monitor Present MAC frame after becoming an Active Monitor

RESTARTED BY: Transmission of an Active Monitor Present MAC frame

CANCELED BY: Deactivation of the active monitor function

TIME-OUT ACTION: The active monitor transmits an Active Monitor Present MAC frame. The active monitor also transmits a Neighbor Notification Incomplete MAC frame if the previous neighbor notification process was not completed (the active monitor did not receive a Standby Monitor Present MAC frame from its nearest active upstream neighbor).

T(notification_response)

This timer represents the delay that a ring station must observe between its receipt of an Active Monitor Present or Standby Monitor Present MAC frame in which AC=B'00' and its transmission of a Standby Monitor Present MAC frame. It prevents congestion of the receive buffers in all the ring stations, which could happen if a ring station responded to a received Active Monitor Present or Standby Monitor Present MAC frame using the token appended to it.

IEEE COUNTERPART: Queue PDU Timer (TQP)

MINIMUM DURATION: Time for a second token to arrive at the ring station, following the token appended to the received Standby Monitor Present MAC frame

VALUE: 20 milliseconds

STARTED BY: Receipt of an Active Monitor Present or Standby Monitor Present MAC frame in which AC=B'00'

RESTARTED BY: Receipt of an Active Monitor Present or Standby Monitor Present MAC frame in which AC=B'00'

CANCELED BY: Receipt of a Ring Purge MAC frame

TIME-OUT ACTION: Queueing a Standby Monitor Present MAC frame.

T(physical_trailer)

If a transmitting station fails to detect its transmitted ending delimiter, this timer minimizes the erroneous stripping of ring data.

IEEE COUNTERPART: Return to Repeat Timer (TRR)

MINIMUM DURATION: The maximum time required for the ending delimiter to circle the ring (called maximum ring latency),

max[number of stations, at 2.5 bits delay per station]
+ max[wiring delay] + max[fairness delay]
+ [active monitor delay],

where max[fairness delay] is the number of bits introduced to the ring by the station to allow fair use of the priority token

VALUE: 4.1 milliseconds

STARTED BY: Transmitting the ending delimiter of a frame in which the intermediate frame indicator (I) bit is set to B'0'

RESTARTED BY: Not restarted

CANCELED BY:

- Entering any mode except Strip mode
- Receiving, in Strip mode, a frame with a source address equal to the station's individual address and an ending delimiter at the end
- Receiving, in Strip mode, an abort delimiter (a starting delimiter followed by an ending delimiter)

TIME-OUT ACTION:

- A standby monitor increments the Lost Frame counter and enters Normal Repeat mode. It will transmit a token only if it recognized its returning physical header.
- An active monitor that is not transmitting Ring Purge MAC frames increments the Lost Frame counter and purges the ring.

T(receive_notification)

A standby monitor uses this timer to ensure that neighbor notification occurs often enough. An active monitor uses this timer to ensure that, during neighbor notification, the Active Monitor Present MAC frame is circling the entire ring. Conditions that would prevent an active monitor from successfully conducting the neighbor notification process include:

- The streaming (unbroken transmission) of tokens by the active monitor
- The streaming of tokens by another station
- The streaming of Ring Purge MAC frames by the active monitor.

IEEE COUNTERPART: Standby Monitor Timer (TSM)

MINIMUM DURATION:

N x T[neighbor_notification],

where N is an integer, dependent on the user's implementation

VALUE: 15 seconds

STARTED BY: Activation of the active monitor or standby monitor function

RESTARTED BY: Receipt of an Active Monitor Present MAC frame

CANCELED BY: Deactivation of the active monitor or standby monitor function

TIME-OUT ACTION: The active monitor or standby monitor enters Claim Token Transmit mode.

T(response)

This timer is used to specify how long a ring station, ring parameter server, or LAN manager should wait for a response to a request before assuming failure and retransmitting the request.

IEEE COUNTERPART: None

MINIMUM DURATION: Sufficient time to allow the receiver of the request to respond with the appropriate MAC frame. The receiver may solicit the response MAC frame N times, where N is dependent on the user's implementation.

VALUE: 2.5 seconds

STARTED BY: Stripping a MAC frame that requires a response, in which AC≠00

RESTARTED BY: Not restarted

CANCELED BY: Copying the necessary response MAC frame

TIME-OUT ACTION: Decrementing the Response Retry counter and, if the counter has not gone to zero, queueing the request MAC frame again. If the counter has gone to zero, the request MAC frame is not queued again.

T(ring_purge)

This timer specifies the length of time that an active monitor can continue to transmit Ring Purge MAC frames when purging the ring before assuming a failure and initiating token-claiming.

IEEE COUNTERPART: No Token Timer (TNT)

MINIMUM DURATION:

max[ring delay] + max[hard-error detect],

where:

- Ring delay is the time for a MAC frame to circle the ring.
- Hard-error detect is the time needed to detect a hard error.

VALUE: 1 second

STARTED BY: Entering the ring purge process

RESTARTED BY: Not restarted

CANCELED BY: Successful completion of the ring purge process

TIME-OUT ACTION: The active monitor enters Claim Token Transmit mode.

T(soft_error_report)

This timer specifies the minimum amount of time between the sending of each Report Soft Error MAC frame. Waiting a minimum amount of time allows stations to collect multiple error counts into one transmission during periods of high numbers of errors, thus avoiding additional congestion.

IEEE COUNTERPART: None

MINIMUM DURATION: Depends on the user's implementation. The Soft Error Report Timer Value subvector of a Set Parameters MAC frame (see "Soft Error Report Timer Value, X'05'" on page 5-23) allows a 2-byte specification with a unit increment of 10 milliseconds.

VALUE: 2 seconds

STARTED BY: Incrementing the Soft Error counter if no Report Soft Error MAC frame has been queued for transmission since the previous expiration of the timer

RESTARTED BY: Not restarted

CANCELED BY: Not canceled

TIME-OUT ACTION: The active monitor queues a Report Soft Error MAC frame.

T(transmit_pacing)

This timer specifies the length of time a ring station waits before transmitting another frame. It is used to pace the transmission of Beacon and Claim Token MAC frames. The value of this timer cannot be set by the user.

IEEE COUNTERPART: None

MINIMUM DURATION: 20 milliseconds

VALUE: 20 milliseconds

STARTED BY: Transmitting a Beacon or Claim Token MAC frame

RESTARTED BY: Transmitting a Beacon or Claim Token MAC frame

CANCELED BY: Exiting Beacon Transmit or Claim Token Transmit mode.

TIME-OUT ACTION: The ring station transmits a Beacon or Claim Token MAC frame, depending on the mode:

- In Beacon Transmit mode, when T(transmit_pacing) expires, the ring station transmits another Beacon MAC frame, without waiting for a token, followed by idles. After the frame has been transmitted, the ring station restarts T(transmit_pacing).
- In Claim Token Transmit mode, when T(transmit_pacing) expires, the ring station transmits another Claim Token MAC frame (without waiting for a token) followed by idles (B'0's), and restarts T(transmit_pacing).

Appendix B. The Differential Manchester Code

The IBM Token-Ring Network uses the differential Manchester code to convert binary data into signal elements. These signal elements allow the receiver to derive clocking pulses from the encoded signal.

For each bit of data, the code consists of two signal elements of opposite polarity. This maintains the DC balance in each bit and guarantees a transition for clocking.

The transition from one polarity to another, or the lack of such transition, at the *bit start point* determines whether the bit has a value of B'0' or B'1':

- If there is a transition (the signal elements on both sides of the start point have the *opposite* polarity), the bit is a B'0'.
- If there is no transition (the signal elements on both sides of the start point have the *same* polarity), the bit is a B'1'.

Similarly, the transition from one polarity to another, or the lack of such transition, at the *bit midpoint* determines whether or not the bit is not DC-balanced.

- If there is a transition (the signal elements on both sides of the midpoint have the *opposite* polarity), the bit is a valid B'0' or B'1'.
- If there is no transition (the signal elements on both sides of the midpoint have the *same* polarity), the bit is a *code violation*. There are two categories of code violation:
  - A *J* code violation is also called a *positive* code violation, because there is no transition at the bit start point.
  - A *K* code violation is also called a *negative* code violation, because there is a transition at the bit start point.

The differential Manchester code signal combinations that result in valid bits and code violations are shown below:

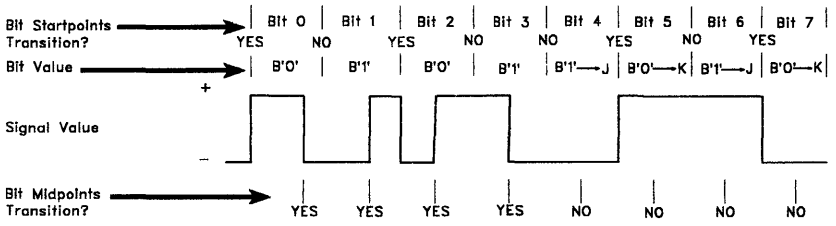


Figure B-1. Differential Manchester Code Signal Combinations

- Bit 0 has a value of B'0', because there is a transition in polarity at the bit start point.
- Bit 1 has a value of B'1', because there is no transition in polarity at the bit start point.
- Bit 2 has a value of B'0', because there is a transition in polarity at the bit start point.
- Bit 3 has a value of B'1', because there is no transition in polarity at the bit start point.
- Bit 4 and bit 6 are J code violations, because there are no transitions in polarity at either the bit midpoints or start points.
- Bit 5 and bit 7 are K code violations, because there are no transitions in polarity at the bit midpoints, but there are transitions at the bit start points.

In the IBM Token-Ring Network, code violations are allowed only in the starting and ending delimiters, to guarantee their uniqueness from the rest of the frame.

The combinations of valid bits and code violations that make up the IBM Token-Ring Network starting delimiter are shown below:

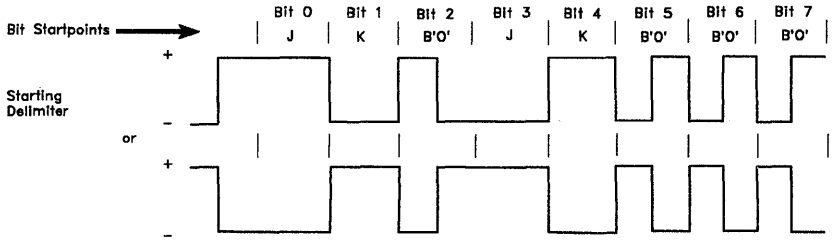


Figure B-2. Signal Combinations for the Starting Delimiter

All valid tokens and frames start with the starting delimiter, exactly as shown above (J K 0 J K 0 0 0). A token or frame that starts with any other byte or combination of bits is invalid.

The combinations of valid bits and code violations that make up the IBM Token-Ring Network ending delimiter are shown below:

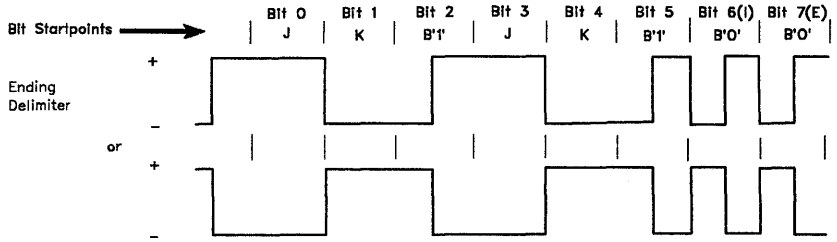


Figure B-3. Signal Combinations for the Ending Delimiter

The ending delimiter is shown as it is transmitted by the originating station, with the error-detected (E) and intermediate frame (I) bits set to B'0'. The first 6 bits of the ending delimiter must be exactly as shown (J K 1 J K 1), or the delimiter is invalid.

Appendix C. Physical Interfaces

This appendix contains tables and figures that describe the physical interfaces of the IBM Token-Ring Network.

Note: For a complete description, please refer to the IEEE 802.5 standard.

Interface at Wall Connector

	4 Mbps		16 Mbps	
Input (to attaching product)	**Minimum**	**Maximum.**	**Minimum**	**Maximum.**
RX level [A] [B]	50 mV p-p	—	150 mV p-p	—
Maximum bit outage time [C]	—	15 msec	—	15 msec
Maximum freq error (operating) [D]	—	0.01 %	—	0.01 %
Maximum freq error (non-operating) [E]	—	400 kHz	—	2.7 MHz
Input impedance [F]	135 Ω	165 Ω	135 Ω	165 Ω
Wire fault conditions [G]	—	—	—	—
DC termination [H]	—	10 Ω	—	10 Ω
Output (from attaching product)	**Minimum**	**Maximum**	**Minimum**	**Maximum**
TX level	3.0 V p-p	4.5 V p-p	3.0 V p-p	4.5 V p-p
Rise times	15 nsec	60 nsec	[O]	[O]
Asymmetry	—	11 nsec	—	3.0 nsec
Jitter [I]	—	2.2 nsec	—	3.2 nsec
Jitter [J]	—	7.0 nsec	—	4.7 nsec
Jitter peaking [K]	—	0.005 dB	—	0.006 dB
Maximum bit outage [L]	—	4 half-bits	—	4 half-bits
Frequency error [E]	—	400 kHz	—	2.7 MHz
Bit error rate [M]	—	10^{-11}	—	10^{-9}
Phantom drive (at 4.1 V)	1 mA	—	1 mA	—
Phantom drive (at 0 V)	4 mA	20 mA	4 mA	20 mA
Output (line) impedance [N]	135 Ω	165 Ω	135 Ω	165 Ω

Physical (see Figure C-2 on page C-4)	
Transmit data (TX)	orange, black
Receive data (RX)	red, green

Notes

[A] Occurring over 140° area at the center of the half-bit time and measured after a passive equalizer with (a pole at 2.7 MHz; a pole at 16 MHz; and a zero at 540 kHz for 4 Mbps operation) and with (a pole at 10.3 MHz; a pole at 25.0 MHz; and a zero at 2.4 MHz for 16 Mbps operation). The signal should be measured with the data wrapped at the connector.

[B] The adapter must function with a peak-to-peak input jitter consisting of a single frequency for the following values:

At 4 Mbps

- Less than 687 ns below 2.0 KHz
- 20 dB/decade derating between 2.0 KHz and 55 KHz
- Less than 25 ns above 55 KHz.

At 16 Mbps

- Less than 937 ns below 1.8 KHz
- 20 dB/decade derating between 1.8 KHz and 270 KHz
- Less than 6.2 ns above 270 KHz.

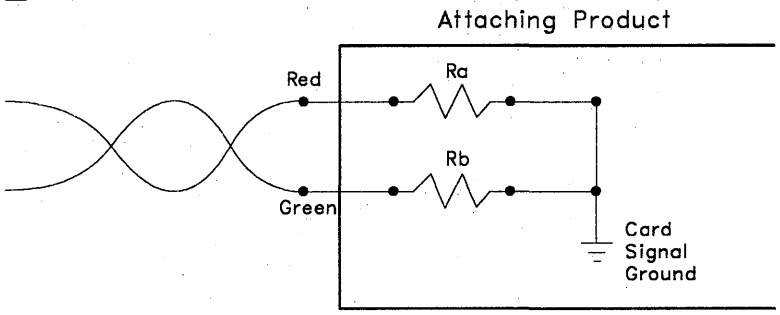
[C] Because of switching time of wiring concentrator.

[D] Averaged for more than 60 seconds.

E Instantaneous frequency can be off by 400 kHz for as long as 50 μsec during certain error conditions. The adapter must be able to recover from this condition.

F Impedance into either the receiver or the driver at 4 MHz.

G (Values are effective resistances as seen from phantom drive.)



Wire fault must be active (fault indicated) when:

(4.3 KΩ ≤ Ra ≤ 5.5 KΩ) AND (Rb > 50.0 KΩ)
OR (4.3 KΩ ≤ Rb ≤ 5.5 KΩ) AND (Ra > 50.0 KΩ)
OR (Ra < 0.1 KΩ) OR (Rb < 0.1 KΩ)

Wire fault must be inactive (fault not indicated) when:

(2.9 KΩ ≤ Ra ≤ 3.82 KΩ) AND (2.9 KΩ ≤ Rb ≤ 3.82 KΩ)
OR (4.3 KΩ ≤ Ra ≤ 5.5 KΩ) AND (4.3 KΩ ≤ Rb ≤ 5.5 KΩ)

These cases are illustrated below, where A indicates an active wire fault, I indicates an inactive wire fault, and D is the region where the wire fault can be either active or inactive.

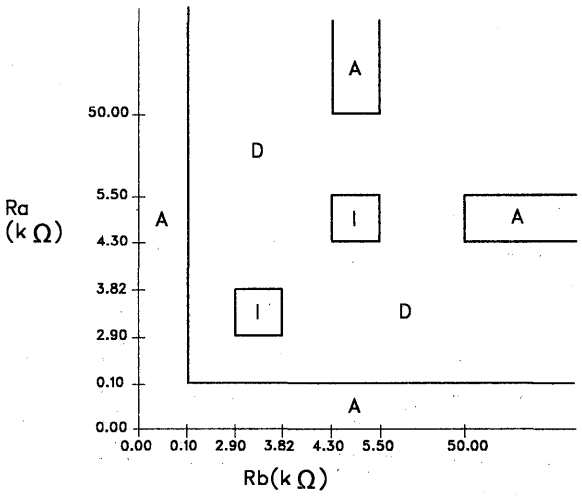


Figure C-1. Active and Inactive Wire Fault Regions

H DC resistance from receive lines to ground.

I Peak-to-peak phase error between the recovered clock and the clock of the upstream station when the data pattern is 500 bytes of B'0's alternating with 500 bytes of B'1's.

J Peak-to-peak phase error when the data pattern is all B'0's.

K Maximum gain of the phase transfer function of the timing recovery circuit.

L Maximum number of Manchester half-bits without a transition.

M Measured with the transmitter repeating the received data.

N Impedance into either the receive or the transmit twisted-pair wire at 4 MHz.

O At 16 Mbps rise times are not adequate to describe transmitter waveform. Please refer to frequency characteristics given in IEEE 802.5 standard.

Interface at Wiring Concentrator (Access Unit)

Input (to wiring concentrator)	**Minimum**	**Maximum**
Phantom voltage	3.9 V	5.2 V
Phantom current (at 4.1 V)	1 mA	—
Phantom current (at 0 V)	4 mA	20 mA
Phantom interruption A	—	50 msec

Output (from wiring concentrator)	**Minimum**	**Maximum**
Attachment time B	—	5 sec
Removal time C	50 msec	200 msec
DC resistance D	4.3 kΩ	5.5 kΩ

Insertion Loss (maximum)

	10 kHz:	0.5 to 4 MHz:	4 to 16 MHz:	16 to 32 MHz:
MRI to MRO	—	0.3 dB	0.6 dB	1.2 dB
TX to RX E	3.0 dB	0.65 dB	1.3 dB	2.6 dB
MRI to RX	3.0 dB	0.55 dB	1.1 dB	2.2 dB
TX to MRO	3.0 dB	0.55 dB	1.1 dB	2.2 dB
TX to RX F	3.0 dB	0.8 dB	1.6 dB	3.2 dB

Crosstalk (maximum)

	10 kHz:	0.5 to 4 MHz:	4 to 16 MHz:
TX to RX	55 dB	43 dB	37 dB
Lobe wrap to MR	55 dB	43 dB	37 dB
MR to BR	59 dB	47 dB	41 dB

Return Loss (minimum)

1 to 6 MHz:	6 to 12 MHz:	12 to 24 MHz:
20 dB	14 dB	11 dB

Common Mode Rejection (minimum)

1 to 6 MHz:	6 to 12 MHz:	12 to 24 MHz:
40 dB	28 dB	25 dB

Physical (see Figure C-2 on page C-4)

Main Ring Path — In (MRI)	red, green
Main Ring Path — Out (MRO)	red, green
Backup Ring Path — In (BRI)	orange, black
Transmit Data (TX)	orange, black
Backup Ring Path — Out (BRO)	orange, black
Receive Data (RX)	red, green

Notes

A Attachment to the ring must not be compromised by phantom interrupts up to the specified time.

B Time from application of phantom drive to attachment to the ring.

C Time from removal of phantom drive to reconfiguration of the ring.

D From transmit wire to receive wire (red-to-orange, green-to-black).

E Transmit to receive of each lobe.

F Each transmit to receive of last lobe.

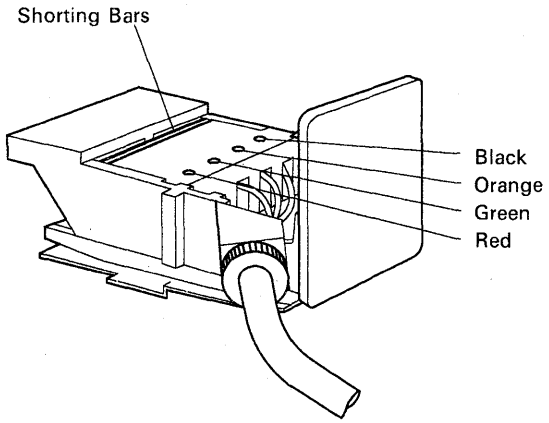


Figure C-2. IBM Cabling System Data Connector (Interior View). This is the connector used throughout the IBM Token-Ring Network.

Appendix D. Protocol Boundary Mapping

The DLC.LAN.MGR protocol boundary (also known as a service specification) is equivalent to the one defined by IEEE 802.2. Only one enhancement to the IEEE 802.2 protocol boundary is required: the LLC sub-layer must pass the information field of received XIDs to the LLC user if the XID format is not that defined by IEEE 802.2. With this enhancement, the discussion that follows describes how an SNA node can use "an IEEE 802.2 LLC."

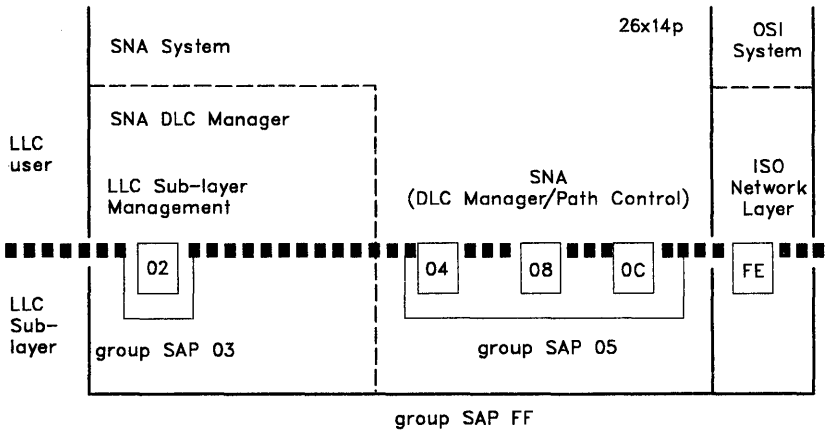


Figure D-1. LLC SAP Addresses and Corresponding LLC Users

Figure D-1 illustrates the IEEE 802.2 LLC as seen by the SNA user. The SNA DLC manager has most or all the IEEE 802.2 service primitives at its disposal, depending on the SAP. In particular, the DLC manager controls all the activation and deactivation primitives for the SNA node; path control manipulates only the L_DATA_CONNECT primitives, as shown in Figure D-2 on page D-2.

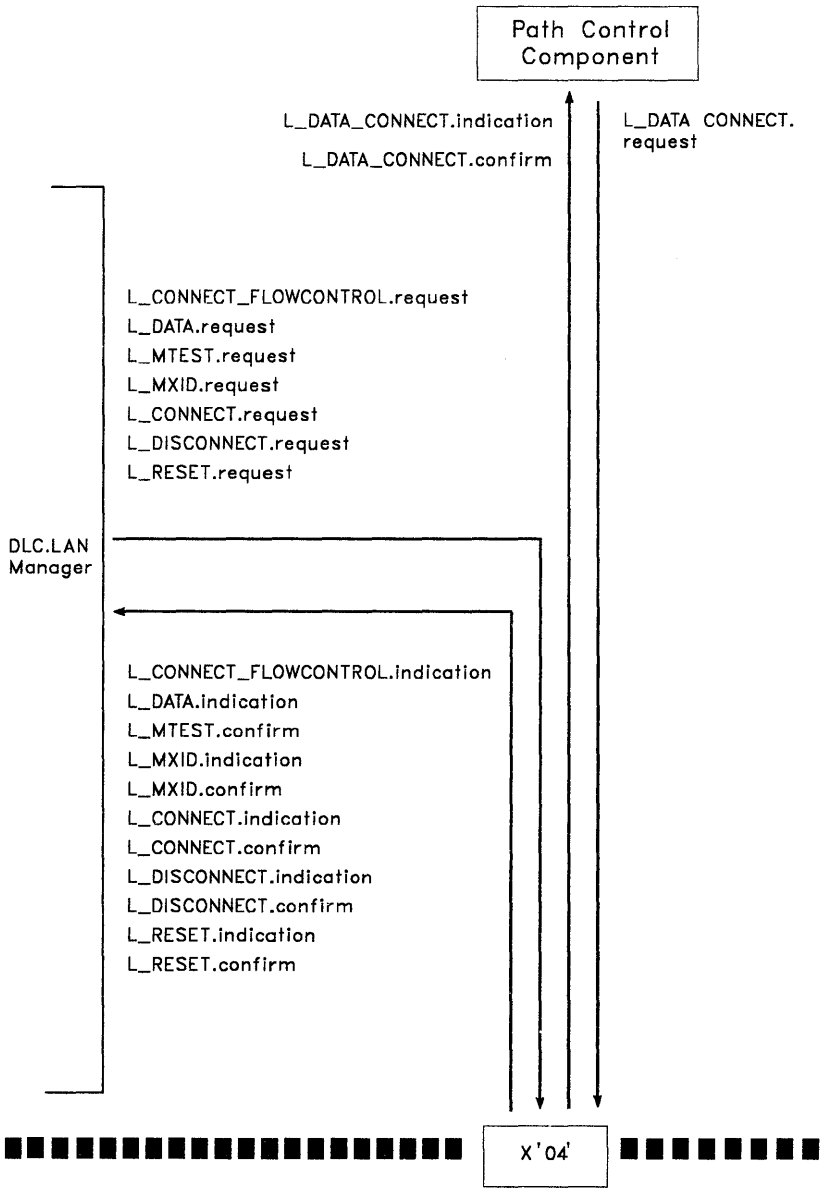


Figure D-2. Use of LLC Service Primitives for SNA SAPs. For more details of the LLC service primitives, consult the IEEE 802.2 standard.

Note: The primitives for the sending and receiving of XID and TEST LPDUs are not included in the approved IEEE 802.2 standard. Primitives for the sending of XID and TEST are part of the system and layer management work currently being addressed in both the IEEE 802.1 and IEEE 802.2 subcommittees.

The following table provides a simplified mapping from the IEEE 802.2 service primitives to the SNA equivalents:

IEEE 802.2	IBM SNA
L_DATA_CONNECT.request	Send_BTU
L_DATA_CONNECT.confirm	—
L_DATA_CONNECT.indication	BTU_received
L_DATA.request	(snd UI)
L_DATA.indication	(rcv UI)
L_CONNECT.request	Set_ABME
L_CONNECT.indication	Rcv_set_mode
L_CONNECT.confirm	Contacted
L_DISCONNECT.request	Set_ADM
L_DISCONNECT.indication	INOP(DISC_received)
L_DISCONNECT.confirm	Disconnected
L_MXID.request	Snd_XID
L_MXID.indication	Rcv_XID
L_MXID.confirm	Rcv_XID
L_MTEST.request	Snd_TEST
L_MTEST.confirm	Rcv_TEST
L_RESET.request	Set_ABME
L_RESET.indication	Rcv_set_mode
L_RESET.confirm	Contacted

The figure below illustrates the relationships between the DLC.LAN components, and how the various SAP addresses are handled by the access channel:

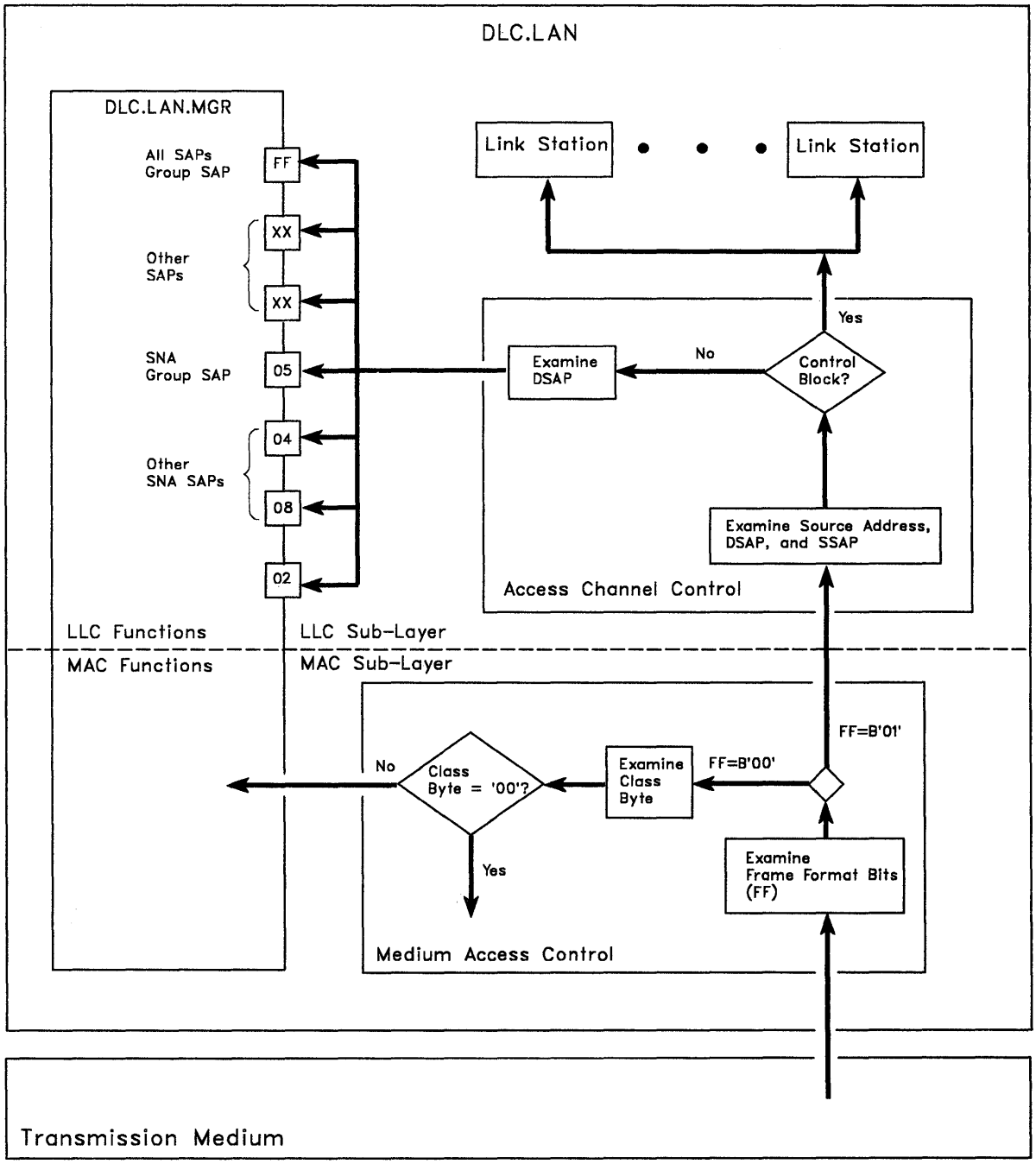


Figure D-3. Incoming Frame Routing in the DLC Component

Appendix E. Parsers

Before executing a Request Status or Set Parameters LLC frame, the management servers check them for errors. This appendix shows the parsing algorithms used to do this.

Parsing Algorithm for Request Status Frames

```
PARSE_REQUEST_VECTOR

   If VECTOR_LENGTH < 4 then
      Set ERROR_CODE to indicate vector length < 4.
      Return OFFSET.
   If vector contains subvectors (as indicated by a bit in the vector identifier) then
      Sum all subvector lengths.
      If sum of subvector lengths is not equal to the vector length - 4 then
         Set the ERROR_CODE to indicate total (subvector lengths) ≠ vector length - 4.
         Return OFFSET.
      Add 4 to OFFSET.
      Do while there are subvectors to scan.
         Look up SUBVECTOR_IDENTIFIER in a table maintained by this server.
         If this SUBVECTOR_IDENTIFIER is not contained in the table then
            Return OFFSET.
         If (SUBVECTOR_IDENTIFIER = all and more subvectors follow within this vector) then
            Set ERROR_CODE to indicate optional subvectors after the all subvector.
            Return OFFSET.
         If this subvector is an unexpected duplicate then
            Set ERROR_CODE to unexpected duplicate subvector.
            Return OFFSET.
         Call PARSE_REQUEST_VECTOR for this subvector.
         Scan for required subvectors.
         If all required subvectors are not present then
            Set the ERROR_CODE to required subvector not present.
            Return OFFSET.
   Else (vector has an atomic value)
      If vector is required and value is invalid then
         Set ERROR_CODE to indicate invalid subvector value.
         Return OFFSET.
      Else if VECTOR_LENGTH ≠ 4 and vector is not required then
         Set ERROR_CODE to indicate invalid length.
         Return OFFSET.
   Set OFFSET to OFFSET + SUBVECTOR_LENGTH.
```

ERROR_CODE Values:

```
X'0002' - Total (subvector lengths) ≠ vector length - 4
X'0003' - Atomic subvector's length ≠ 4
X'0004' - Optional subvectors after the all subvector
X'0006' - Invalid subvector value
X'0008' - Required subvector not present
X'0009' - Vector length < 4.
X'000A' - Unexpected duplicate subvector
```

Note: If the correlator subvector is not present in the request frame, the returned error code = X'0008' (required subvector not present) and the value of the correlator subvector returned is X'00000000'.

Figure E-1. Parsing Algorithm for Request Status Frames

Parsing Algorithm for Set Parameter Frames

```
Procedure PARSE_SET_REQUEST

   If VECTOR_LENGTH < 4 then
      Set ERROR_CODE to indicate vector length < 4.
      Return OFFSET.

   If vector is major vector and sending LAN manager is not controlling
   and vector is only valid from the controlling LAN manager then
      Set ERROR_CODE to indicate unauthorized request.
      Return OFFSET.
   If VECTOR contains subvectors (as indicated by a bit in the vector identifier) then
      Sum all subvector lengths.
      If sum of subvector lengths is not equal to VECTOR_LENGTH - 4 then
         Set ERROR_CODE to indicate total (subvector lengths) ≠ vector length - 4.
         Return OFFSET.
         Add 4 to OFFSET.
         Do while there are subvectors to scan.
            Look up subvector identifier in a table maintained by this server.
            If this subvector identifier is not contained in the table then
               Return OFFSET.
            If this subvector is an unexpected duplicate then
               Set ERROR_CODE to indicate unexpected duplicate subvector.
               Return OFFSET.
            Call PARSE_SET with subvector.
            Scan for required subvectors.
            If all required subvectors are not present then
               Set the ERROR_CODE to indicated required subvector not present.
               Return OFFSET.
   Else (vector has an atomic value)
      Check valid vector length for this subvector.
      If subvector length not valid length then
         Set ERROR_CODE to indicate invalid length.
         Return OFFSET.
      Check valid values for this subvector.
      If subvector value not valid then
         Set ERROR_CODE to indicate invalid subvector value.
         Return OFFSET.
   Set OFFSET to OFFSET + SUBVECTOR_LENGTH.
```

ERROR_CODE Values:

```
X'0002' - Total (subvector lengths) ≠ vector length - 4
X'0004' - Value too high
X'0005' - Value too low
X'0006' - Invalid subvector value
X'0007' - Invalid length for atomic subvector
X'0008' - Required subvector not present
X'0009' - Vector length < 4
X'000A' - Unexpected duplicate subvector
X'000B' - Unauthorized request
```

Note: If the correlator subvector is not present in the request frame, the returned ERROR CODE = X'0008' (required subvector not present) and the value of the correlator subvector returned is X'00000000'.

Figure E-2. Parsing Algorithm for Set Parameter Frames

Appendix F. Conditions of Presence

This appendix shows the *conditions of presence* for the various token-ring management frames described in Part 4 of this reference. Each table in this appendix shows all the subvectors that can be present within a major vector or subvector and indicates the conditions for which each vector or subvector is present. The tables also show the major vector and subvector identifier code points and length of each major vector and subvector.

The hexadecimal major vector and subvector identifiers are listed in this appendix for each frame. In addition, the length of each subvector (in decimal) is included for easy reference. Detailed descriptions of the value fields are given in the Data Structures section of each management server chapter (Chapters 14 through 18).

A "v" in the length field in the following tables means that the value field is variable in length, depending on the presence of optional or conditionally present subvectors within it.

An "i" in the length field in the following tables means that the length of the value field is dependent on the implementation and not specified in the architecture.

Note: The length field values in the following tables contain the two-byte length field and the two-byte ID field of the subvector format.

Ring Error Monitor (REM) Frames

Request REM Status (X'8101') Conditions of Presence

Table F-1. Request REM Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4005	Ring Number	6	Present one time.
C001	Notification Enable	6	Note 1
8004	Intensive Mode Data	v	Note 1
C02C	Ring Intensive Mode Mask	6	Note 1
C02D	Auto Intensive Mode Mask	6	Note 1
C006	Isolating Table	v	Note 1
C016	Ring Status	30	Note 1

Note:

1. Conditionally present if LAN manager is requesting the value for this parameter.

Report REM Status (X'8102') Conditions of Presence

Table F-2. Report REM Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4005	Ring Number	6	Present one time.
C001	Notification Enable	6	Note 1
8004	Intensive Mode Data	v	Note 2
C02C	Ring-Intensive Mode Mask	6	Note 3
C02D	Auto-Intensive Mode Mask	6	Note 3
C006	Isolating Table	v	Note 1
8016	Ring Status	48	Note 1
C026	Ring State	6	Present one time.
8018	Beacon data	38	Note 4
C024	Beacon Type	6	Present one time.
4002	Beaconing Station Address	10	Present one time.
4003	NAUN of Beaconing Station	10	Present one time.
4004	Physical Location of Beaconing Station	8	Present one time.

Note:

1. Present if a subvector with the same identifier was present in the corresponding Request REM Status frame.
2. Present if a subvector with the same identifier was present in the Installation Parameters subvector in the corresponding Request REM Status frame.
3. Present if a subvector with the same identifier was present in the Intensive Mode Data subvector in the corresponding Request REM Status frame.
4. This subvector is present if the value of the Ring State subvector indicates that there is a beaconing (non-normal) condition on the ring.

Set REM Parameters (X'8103') Conditions of Presence

Table F-3. Set REM Parameters Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4005	Ring Number	6	Optionally present one time.
C001	Notification Enable	6	Optionally present one time.
8004	Intensive Mode Data	16	Optionally present one time.
C02C	Ring-Intensive Mode Mask	6	Optionally present one time.
C02D	Auto-Intensive Mode Mask	6	Optionally present one time.
C00E	Reset	4	Optionally present one time.

REM Parameters Set (X'8104') Conditions of Presence

Table F-4. REM Parameters Set Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.

REM Parameters Changed Notification (X'8105') Conditions of Presence

Table F-5. REM Parameters Changed Notification Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
C001	Notification Enable	6	Note 1
8004	Intensive Mode Data	16	Note 1
C02C	Ring-Intensive Mode-Mask	6	Note 1
C02D	Auto-Intensive Mode-Mask	6	Note 1
C00E	Reset	4	Note 1

Note:

1. Conditionally present if this subvector was present in the Set REM Parameters frame that caused this notification to be sent.

Error Rate Decaying Notification (X'8106') Conditions of Presence

Table F-6. Error Rate Decaying Notification Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
C001	Notification Enable	6	Present one time.
800A	REM Isolating Status	v	Present one time.
C023	Decrement Interval	5	Present one time.
C028	Number of Entries	5	Present one time.
800B	Fault Domain	v	Present one time.
8029	Receiver	42	Present one time.
C02F	Flags	5	Present one time.
C030	Error weight	5	Present one time.
4002	Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.
802A	Transmitter	42	Note 1
C02F	Flags	5	Present one time.
C030	Error weight	5	Present one time.
4002	Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.

Note:

1. This subvector is present if there is an entry present in the isolating slot table for the NAUN of the station identified in the receiver subvector.

Pre-Weight Exceeded Notification (X'8107') Conditions of Presence

Table F-7. Pre-Weight Exceeded Notification Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
C001	Notification Enable	6	Present one time.
800A	REM Isolating Status	v	Present one time.
C023	Decrement Interval	5	Present one time.
C028	Number of Entries	5	Present one time.
800B	Fault Domain	v	Present one time.
8029	Receiver	42	Present one time.
C02F	Flags	5	Present one time.
C030	Error weight	5	Present one time.
4002	Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.
802A	Transmitter	42	Note 1
C02F	Flags	5	Present one time.
C030	Error weight	5	Present one time.
4002	Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.

Note:

1. This subvector is present if there is an entry present in the isolating table for the NAUN of the station identified in the receiver subvector.

Weight-Exceeded Notification (X'8108') Conditions of Presence

Table F-8. Weight-Exceeded Notification Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
C001	Notification Enable	6	Present one time.
800A	REM Isolating Status	v	Present one time.
C023	Decrement Interval	5	Present one time.
C028	Number of Entries	5	Present one time.
800B	Fault Domain	v	Present one time.
8029	Receiver	42	Present one time.
C02F	Flags	5	Present one time.
C030	Error weight	5	Present one time.
4002	Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.
802A	Transmitter	42	Note 1
C02F	Flags	5	Present one time.
C030	Error weight	5	Present one time.
4002	Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.

Note:

1. This subvector is present if there is an entry present in the isolating slot table for the NAUN of the station identified in the receiver subvector.

Non-Isolating Threshold Exceeded (X'8109') Conditions of Presence

Table F-9. Non-Isolating Threshold Exceeded Major vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
C001	Notification Enable	6	Present one time.
8003	Non-Isolating Notification	v	Present one time.
C01B	Lost Frames	6	Note 1
C01C	Receiver Congestion Errors	6	Note 1
C01D	Frame-Copied Errors	6	Note 1
C01E	Frequency Errors	6	Note 1
C01F	Token Errors	6	Note 1
C020	Reserved	6	Note 1
C021	Table-Full Errors	6	Note 1
C022	Minimum-Decrement Errors	6	Note 1
C031	Receiver-Congestion Table Full Errors	6	Note 1

Note:

1. Conditionally present one time if the corresponding counter in the ring error monitor has exceeded its threshold and caused this frame to be sent.

Forward MAC Frame (X'810A') Conditions of Presence

Table F-10. Forward MAC Frame Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
4002	Reporting Address	10	Present one time.
8010	Soft-Error Report	42	Note 1
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.
C032	Isolating Error Counts	10	Present one time.
C033	Non-Isolating Error Counts	10	Present one time.
8011	Report Neighbor Notification Failure	14	Note 2
4002	Address of Last Neighbor Notification	10	Present one time.
8012	Report Monitor Error	28	Note 3
4007	Error Code	6	Present one time.
4004	Physical Location	8	Present one time.
4003	NAUN Address	10	Present one time.

Note:

1. This subvector is present if the arrival of a Soft-Error Report MAC frame (along with the appropriate intensive mode being enabled) triggered this frame to be sent.
2. This subvector is present if the arrival of a Report Neighbor-Notification-Incomplete MAC frame triggered this frame to be sent.
3. This subvector is present if the arrival of a Report Monitor-Error MAC frame triggered this frame to be sent.

REM Error (X'810C') Conditions of Presence

Table F-11. REM Error Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4009	Error Code	6	Present one time.
400A	Error Offset	6	Present one time.
400C	REM Version Level	18	Present one time.

Receiver Congestion Notification (X'810E') Conditions of Presence

Table F-12. Receiver Congestion Notification Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
4002	Station Address	10	Present one time.

Receiver Congestion Ended (X'810F') Conditions of Presence

Table F-13. Receiver Congestion Ended Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
4002	Station Address	10	Note 1

Note:

1. Present once for each Station REM has identified as experiencing excessive receiver congestion errors that did not report receiver congestion in the last notification interval.

Beaconing Condition on Ring (X'8110') Conditions of Presence

Table F-14. Beaconing Condition on Ring Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
8016	Ring Status	48	Present one time.
C026	Ring State	6	Present one time.
8018	Beacon Data	38	Present one time.
C024	Beacon type	6	Present one time.
4002	Beaconing Station Address	10	Present one time.
4003	NAUN of Beaconing Station	10	Present one time.
4004	Physical Location of Beaconing Station	8	Present one time.
4002	Removed Station Address	10	Note 1

Note:

1. Conditionally present one time if REM detected a temporary beaconing condition that caused one station to leave the ring. Present two times if REM detected a temporary beaconing condition that caused both fault domain stations to leave the ring.

Beaconing Condition Recovered (X'8111') Conditions of Presence

Table F-15. Beaconing Condition Recovered Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.

Ring Parameter Server (RPS) Frames

Request RPS Status (X'8201') Conditions of Presence

Table F-16. Request RPS Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4005	Ring Number	6	Present one time.
4008	All	6	Present one time.

Report RPS Status (X'8202') Conditions of Presence

Table F-17. Report RPS Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4005	Ring Number	6	Present one time.
400C	RPS Version Number	18	Present one time.
C001	Soft-Error-Report Timer Value	6	Present one time.

RPS Error (X'8203') Conditions of Presence

Table F-18. RPS Error Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4009	Error Code	6	Present one time.
400A	Error Offset	6	Present one time.
400C	RPS Version Level	18	Present one time.

Report Station in Ring (X'8206') Conditions of Presence

Table F-19. Report Station in Ring Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
4002	Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
400B	Product Instance ID	22	Present one time.
C002	Ring Station Microcode Level	14	Present one time.
C004	Attachment Status	6	Present one time.

Configuration Report Server (CRS) Frames

Report New Active Monitor (X'8301') Conditions of Presence

Table F-20. Report New Monitor Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
4002	Reporting Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.
400B	Product Instance ID	22	Present one time.

Report NAUN Change (X'8302') Conditions of Presence

Table F-21. Report NAUN Change Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
4002	Reporting Station Address	10	Present one time.
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.

Report Transmit-Forward (X'8303') Conditions of Presence

Table F-22. Report Transmit-Forward Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4005	Ring Number	6	Present one time.
4002	Reporting Station Address	10	Present one time.
C001	Transmit Status Code	6	Present one time.

Request Station Information (X'8304') Conditions of Presence

Table F-23. Request Station Information Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4002	Station Address	10	Present one time.
4005	Ring Number	6	Present one time.
4008	All	6	Note 1
C002	Addressing Information	4	Note 2
C003	State Information	4	Note 2
C004	Attachments Information	4	Note 2

Note:

1. Conditionally present one time to request addressing, state, and attachments information.
2. Conditionally present one time, if the all subvector is not present, to request the value for this parameter.

Report Station Information (X'8306') Conditions of Presence

Table F-24. Report Station Information Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4002	Station Address	10	Present one time.
4005	Ring Number	6	Present one time.
8002	Addressing Information	38	Note 1
4003	NAUN Address	10	Present one time.
4004	Physical Location	8	Present one time.
C00D	Group Addresses	8	Present one time.
C00F	Functional Addresses	8	Present one time.
8003	State Information	38	Note 2
C010	Ring Station Microcode Level	14	Present one time.
C011	Ring Station Status	10	Present one time.
C012	Unique Station ID	10	Optionally present one time.
8004	Attachments Information	46	Note 3
400B	Product Instance ID	22	Present one time.
C013	Enabled Function Classes	6	Present one time.
C014	Allowed Access Priority	6	Present one time.
C00F	Functional Addresses	8	Present one time.
8005	Station Error	v	Note 4
C015	Reason Code	6	Present one time.
C016	Response Code	8	Note 5

Note:

1. This subvector is present if the addressing information subvector or the all subvector was present in the corresponding Request Station Information frame.
2. This subvector is present if the state information subvector or the all subvector was present in the corresponding Request Station Information frame.
3. This subvector is present if the attachments information subvector or the all subvector was present in the corresponding Request Station Information frame.
4. This subvector is present if the addressing information, state information, attachments information, or all subvectors were present in the corresponding Request Station Information frame and the configuration report server could not successfully obtain the requested information from the station.
5. This subvector is present if the value of the reason code subvector is X'0003', which indicates that a Response MAC frame was received from the station.

Set Station Parameters (X'8307') Conditions of Presence

Table F-25. Set Station Parameters Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4002	Station Address	10	Present one time.
4005	Ring Number	6	Present one time.
C01A	Local Ring Number	6	Optionally present one time.
4004	Physical Location	8	Optionally present one time.
C017	Soft-error Report Timer Value	6	Optionally present one time.
C013	Enabled Function Classes	8	Optionally present one time.
C014	Allowed Access Priority	6	Optionally present one time.

Station Parameters Set (X'8308') Conditions of Presence

Table F-26. Station Parameters Set Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
8005	Station Error	v	Note 1
C015	Reason Code	6	Present one time.
C016	Response Code	8	Note 2

Note:

1. Present if the station either does not respond or responds indicating error to a request to set its parameter values.
2. Present if the value of the above reason code subvector is X'0003', indicating that a Response MAC frame was received from the ring station.

Remove Ring Station (X'8309') Conditions of Presence

Table F-27. Remove Ring Station Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4002	Ring Station Address	10	Present one time.
4005	Ring Number	6	Present one time.

Ring Station Removed (X'830A') Conditions of Presence

Table F-28. Ring Station Removed Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C019	Status Code	5	Present one time.
8005	Station Error	v	Note 1
C015	Reason Code	6	Present one time.
C016	Response Code	8	Note 2

Note:

1. Present if the status code subvector value indicates that the station could not be removed.
2. This subvector is present if the value of the reason code subvector is X'0003', which indicates that a Response MAC frame was received from the station.

CRS Error (X'830B') Conditions of Presence

Table F-29. CRS Error Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4009	Error Code	6	Present one time.
400A	Error Offset	6	Present one time.
400C	CRS Version Level	18	Present one time.

Converter Status Frames

Downstream Converter Presence (X'8402') Conditions of Presence

Table F-30. Downstream Converter Presence Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C001	Converter's Partner Address	10	Always present.

Beaconing Back-Up Ring (X'8403') Conditions of Presence

Table F-31. Beaconing Back-Up Ring Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C001	Converter's Partner Address	10	Always present.
C002	Converter's Partner NAUN Address	10	Always present.

LAN Bridge Server Frames

Request Bridge Status (X'8501') Conditions of Presence

Table F-32. Request Bridge Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4008	All	6	Optionally present one time.
C043	All (long counters)	6	Optionally present one time.

Report Bridge Status (X'8502') Conditions of Presence

Table F-33. Report Bridge Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C001	Bridge Type	6	Note 1
400C	Bridge Version level	18	Note 1
C002	Number of Ports	6	Note 1
C021	Partition Bits	5	Note 1
C026	Path Trace	5	Note 1
C02F	Calculation Interval	6	Note 1
C00C	Notification Interval	6	Note 1
C00A	Percent Frames Lost Threshold	6	Note 4
8003	Port Information	v	Note 2
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Note 4
C004	Port Type	6	Note 4
C035	Ring Data Rate	6	Note 4
C005	Ring Status	6	Note 4
C006	Adapter Status	6	Note 4
C013	Port Hop Count	5	Note 4
C009	Frames Discarded Counter Value	6	Note 4
C032	Frames Discarded Counter Value (long)	8	Note 4
C00B	B-Frames-Transmitted Counter	8	Note 4
C027	B-Bytes-Transmitted Counter	10	Note 4
C029	NB-Frames-Transmitted Counter	8	Note 4
C02B	NB-Bytes-Transmitted Counter	10	Note 4
C022	Frames-Not-Received Counter	6	Note 4
C033	Frames-Not-Received Counter (long)	8	Note 4
C024	Frames-Not-Forwarded Counter	6	Note 4
C034	Frames-Not-Forwarded Counter (long)	8	Note 4
C036	Frames Discarded - Internal Error	8	Note 4
C042	Bytes Discarded - Internal Error	10	Note 4
C044	Frames Not Routed Across Bridge	8	Note 5
C012	Single-Route Broadcast Enabled/Disabled	5	Note 5
800D	Route Status	v	Note 3
800E	Route Identifier	24	Present one time.
C00F	Port Identifier	10	Present one time.
C010	Port Identifier	10	Present one time.
C011	Bridge Identifier	6	Note 4
C015	Largest Frame Size	5	Note 4
C007	Route Active Status	5	Note 4
C041	Single-Route Broadcast Mode	5	Note 4
C02E	Bridge Internal Status	i	Note 1

Note:

1. This subvector is present if the same subvector identifier was present in the corresponding request.
2. One of these subvectors is present for each Port Information subvector in the corresponding Request Bridge Status frame. If the All subvector was present in the Request Bridge Status major vector of the corresponding request frame, one of these subvectors for each Port Information structure maintained by the bridge server is present in the Report Bridge Status frame.
3. If the All subvector was present in the Request Bridge Status major vector of the corresponding request frame, one of these subvectors for each Route Status structure maintained by the bridge server is present in the Report Bridge Status frame.
4. This subvector will be present if the All subvector was present in the Request Bridge Status major vector.
5. This subvector will be present if the All subvector was present in the Request Bridge Status major vector. The value of this subvector indicates whether or not this bridge is currently forwarding single-route broadcast frames. Its value is therefore X'00' or X'01'. The value of this subvector will not indicate whether this bridge is participating in the automatic (spanning tree) calculation of the single-route broadcast paths through the network.

Set Bridge Parameters (X'8503') Conditions of Presence

Table F-34. Set Bridge Parameters Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C00C	Notification Interval	6	Optionally present one time.
C00A	Percent Frames Lost Threshold	6	Optionally present one time.
8003	Port Information	v	Optionally present one or more times.
4002	Port Identifier	10	Present one time.
C013	Port Hop Count	5	Present one time.
C012	Single-Route Broadcast Enabled/Disabled	5	Note 2
800D	Route Status	v	Optionally present one or more times.
800E	Route Identifier	24	Present one time.
C00F	Port Identifier	10	Present one time.
C010	Port Identifier	10	Present one time.
C011	Bridge Identifier	6	Optionally present one time.
C007	Route Active Status	5	Optionally present one time.
C041	Single-Route Broadcast Mode	5	Note 2
C02E	Bridge Internal Status	i	Optionally present one time.
4012	Temporary/Permanent	5	Note 1

Note:

1. If this subvector is present, it's value is used to determine whether the bridge server's parameters have been permanently changed or merely changed until the bridge is no longer operational. That is, a temporary change would not persist when the bridge server became operational again. A value of X'00' indicates that the change to the parameter values is intended to be permanent. Any other value for this subvector indicates that the change is intended to be temporary. If this subvector is not present, the change is permanent.
2. Either the Single-Route Broadcast Enabled/Disabled OR the Single-Route Broadcast Mode subvector (but not both) are present in this frame.

Bridge Parameters Set (X'8504') Conditions of Presence

Table F-35. Bridge Parameters Set Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.

Bridge Error (X'8505') Conditions of Presence

Table F-36. Bridge Error Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4009	Error Code	6	Present one time.
400A	Error Offset	6	Present one time.
400C	Bridge Version Level	18	Present one time.

Bridge Parameters Changed Notification (X'8506') Conditions of Presence

Table F-37. Bridge Parameters Changed Notification Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C00C	Notification Interval	6	Note 1
C00A	Percent Frames Lost Threshold	6	Note 1
8003	Port Information	24	Note 2
4002	Port Identifier	10	Present one time.
C013	Port Hop Count	5	Note 3
C012	Single-Route Broadcast Enabled/Disabled	5	Note 4
800D	Route Status	44	Note 2
800E	Route Identifier	24	Present one time.
C00F	Port Identifier	10	Present one time.
C010	Port Identifier	10	Present one time.
C011	Bridge Identifier	6	Note 4
C007	Route Active Status	5	Note 4
C041	Single-Route Broadcast Mode	5	Note 4
C02E	Bridge Internal Status	6	Note 1

Note:

1. This subvector is present if the subvector with the same identifier was present in the Set Bridge Parameters frame that caused this notification to be sent.
2. One instance of each of these subvectors is present for each subvector with the same identifier that was present in the Set Bridge Parameters frame that caused this notification to be sent.
3. This subvector is present if the subvector with the same identifier was present in the Port Information subvector with the same Port Identifier in the Set Bridge Parameters frame that caused this notification to be sent.
4. This subvector is present if the same identifier was present in the Route Status subvector with the same Route Identifier in the Set Bridge Parameters frame that caused this notification to be sent.

Bridge Performance Threshold Exceeded (X'8507') Conditions of Presence

Table F-38. Bridge Performance Threshold Exceeded Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C00A	Percent Frames Lost Threshold	6	Present one time.
C020	Percent Frames Lost	6	Present one time.
8003	Port Information	90	Present one time.
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Present one time.
C023	T-Frames Discarded Counter Value	6	Note 1
C025	T-B-Frames-Transmitted Counter Value	8	Present one time.
C028	T-NB-Frames-Transmitted Counter Value	8	Present one time.
C02A	T-Frames-Not-Received Counter Value	6	Note 1
C02C	T-Frames-Not-Forwarded Counter Value	6	Note 1
C037	T-Frames Discarded - Internal Error	8	Present one time.
C045	T-Frames Not Routed Across Bridge	8	Note 2
C030	T-B-Bytes-Transmitted Counter Value	10	Present one time.
C031	T-NB-Bytes-Transmitted Counter Value	10	Present one time.

Note:

1. Both the long and short error counters are sent to insure interoperability with different versions of the LAN manager.
2. This subvector is present if it is supported by the bridge server.

Path Trace Report (X'8508') Conditions of Presence

Table F-39. Path Trace Report Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C016	Forwarded-Frame Addressing Information	v	Present one time.
C017	Forwarded-Frame Length	6	Present one time.
C018	Forwarded-Frame Data	14	Present one time.
C02D	Forwarded-Frame Strip Status	6	Present one time.
C021	Partition Bits	5	Present one time.
8003	Port Information	31	Present one time.
4002	Port Identifier	10	Present one time.
C005	Ring Status	6	Present one time.
C006	Adapter Status	6	Present one time.
C013	Port Hop Count	5	Present one time.
800D	Route Status	44	Present one time.
800E	Route Identifier	24	Present one time.
C00F	Port Identifier	10	Present one time.
C010	Port Identifier	10	Present one time.
C011	Bridge Identifier	6	Present one time.
C015	Largest Frame Size	5	Present one time.
C007	Route Active Status	5	Present one time.

Bridge Counter Report (X'8509') Conditions of Presence

Table F-40. Bridge Counter Report Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
8003	Port Information	118	Present one or more times.
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Present one time.
C009	Frames Discarded Counter Value	6	Note 1
C032	Frames Discarded Counter Value (long)	6	Note 1
C00B	B-Frames-Transmitted Counter Value	8	Present one time.
C027	B-Bytes-Transmitted Counter Value	10	Present one time.
C029	NB-Frames Transmitted Counter Value	8	Present one time.
C02B	NB-Bytes-Transmitted Counter Value	10	Present one time.
C022	Frames-Not-Received Counter Value	6	Note 1
C033	Frames-Not-Received Counter (long)	6	Note 1
C024	Frames-Not-Forwarded Counter Value	6	Note 1
C034	Frames-Not-Forwarded Counter (long)	6	Note 1
C036	Frames Discarded - Internal Error	8	Note 1
C042	Bytes Discarded - Internal Error	10	Note 1
C044	Frames Not Routed Across Bridge	8	Note 2

Note:

1. Both the long and short error counters are sent to insure interoperability with different versions of the LAN manager.
2. This subvector is present if it is supported by the bridge server.

Single-Route Broadcast Status Change (X'850A') Conditions of Presence

Table F-41. Single-Route Broadcast Status Change Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
8003	Port Information	19	Present one or more times.
4002	Port Identifier	10	Present one time.
C012	Single-Route Broadcast Enabled/Disabled	5	Present one time.

LAN Reporting Mechanism (LRM) Frames

Report LRM Parsing Error (X'8600') Conditions of Presence

Table F-42. Report LRM Parsing Error Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C002	Major Vector Identifier of Error Frame	8	Present one time.
400C	LRM Version Level	18	Present one time.
C00F	Error Code	5	Present one time.

Request LRM Status (X'8601') Conditions of Presence

Table F-43. Request LRM Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4008	All	6	Optionally present one time.

Report LRM Status (X'8602') Conditions of Presence

Table F-44. Report LRM Status Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
400C	LRM Version Level	18	Present one time.
C004	Number of Alternate LAN Managers	6	Present one time.
8005	Reporting Link Information	v	Note 1
C006	Reporting Link Identifier	6	Present one time.
4002	LAN Manager Address	10	Present one time.
8007	Port Information	20	Note 2
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Present one time.

Note:

1. A reporting-link information subvector is present for ***each*** reporting link maintained by LRM.
2. Present once for each port in the bridge.

Set LRM Parameters (X'8603') Conditions of Presence

Table F-45. Set LRM Parameters Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4002	Port Identifier	10	Note 1
4005	Ring Number	6	Note 1
C004	Number of Alternate LAN Managers	6	Optionally present one time.
8005	Reporting Link Information	v	Optionally present one time.
C006	Reporting Link Identifier	6	Present one time.
C00C	New Key	12	Conditionally present one time.
8007	Port Information	30	Optionally present one or more times.
4002	Port Identifier	10	Present one time.
C008	Reporting Function Classes	8	Optionally present one time.
C011	Enabled Functional Addresses	8	Optionally present one time.
4012	Temporary/Permanent	5	Note 2

Note:

1. If the ring number is being changed, both the Port Identifier and the Ring Number subvectors for the affected MAC instance are present. The Port Identifier subvector identifies the MAC instance whose ring number is being changed. The Ring Number subvector specifies the new ring number for the ring into which the MAC instance identified above is inserted. If either the Port Identifier subvector or the Ring Number subvector is present, both are present.
2. If this subvector is present, its value is used to determine whether LRM's parameters have been permanently changed or merely changed until the LRM is no longer operational. That is, a temporary change would not persist when the LRM became operational again. A value of X'00' indicates that the change to the parameter values is intended to be permanent. Any other value for the subvector indicates that the change is intended to be temporary. If this subvector is not present, the change is permanent.

LRM Parameters Set(X'8604') Conditions of Presence

Table F-46. LRM Parameters Set Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.

LRM Error (X'8605') Conditions of Presence

Table F-47. LRM Error Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
4009	Error Code	6	Present one time.
400A	Error Offset	6	Present one time.
400C	REM Version Level	18	Present one time.

LRM Parameters Changed(X'8606') Conditions of Presence

Table F-48. LRM Parameters Changed Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4002	Port Identifier	10	Note 1
4005	Ring Number (new)	6	Note 1
C004	Number of Alternate LAN Managers	6	Note 1
8005	Reporting Link Information	v	Note 1
C006	Reporting Link Identifier	6	Present one time.
C00C	New Key (2-byte null value)	12	Note 2
8007	Port Information	30	Note 1
4002	Port Identifier	10	Present one time.
C008	Reporting Function Classes	8	Note 1
C011	Enabled Functional Addresses	8	Note 1

Note:

1. These subvectors are present if they were present in the Set LRM Parameters that caused this notification to be sent.
2. This subvector is present if it was present in the Set LRM Parameters that caused this notification to be sent, but has a zero length value field (to keep passwords from flowing any more than possible and to enhance the security of the password-based authentication scheme).

Set Reporting Point (X'8607') Conditions of Presence

Table F-49. Set Reporting Point Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C004	Number of Alternate LAN Managers	6	Note 1
C00C	Routing Information	v	Present one time.
8005	Reporting Link Information	v	Present one time.
C006	Reporting Link Identifier	6	Present one time.
C00B	Key	12	Present one time.
8007	Port Information	v	Note 2
4002	Port Identifier	10	Present one time.
C008	Reporting Function Classes	8	Present one time.

Note:

1. Conditionally present one time if the controlling reporting link is requested (reporting link identifier = X'0000').
2. Can be present one time for each valid port identifier. This subvector specifies the reporting function classes parameter for the requested reporting link (for each port information structure maintained by the LAN reporting mechanism for the reporting link). If it is not included, no unsolicited reports are sent on this reporting link.

LAN Manager Accepted (X'8608') Conditions of Presence

Table F-50. LAN Manager Accepted Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C004	Number of Alternate LAN Managers	6	Present one time.
8005	Reporting Link Information	v	Present one time.
C006	Reporting Link Identifier	6	Present one time.
8007	Port Information (one per port)	52	Note 1
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Present one time.
C008	Reporting Function Classes	8	Present one time.
C009	Available Port Function Classes	8	Present one time.
C00A	Active Port Function Classes	8	Present one time.
C011	Enabled Functional Addresses	8	Present one time.

Note:

1. Present one time for each port information structure maintained by the LAN reporting mechanism for the reporting link information structure.

LAN Manager Rejected (X'8609') Conditions of Presence

Table F-51. LAN Manager Rejected Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C00B	Rejection Code	6	Present one time.
C004	Number of Alternate LAN Managers	6	Note 1
8005	Reporting Link Information	v	Note 2
C006	Reporting Link Identifier	6	Present one time.
8007	Port Information (one per port)	52	Note 3
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Present one time.
C008	Reporting Function Classes	8	Present one time.
C009	Available Port Function Classes	8	Present one time.
C00A	Active Port Function Classes	8	Present one time.
C011	Enabled Functional Addresses	8	Present one time.

Note:

1. Conditionally present one time if Rejection Code is not X'0001' (invalid key) or X'0003' (invalid reporting link identifier).
2. Conditionally present one or more times if Rejection Code is not X'0001' (invalid key) or X'0003' (invalid reporting link identifier). One Reporting Link subvector is present for each reporting link maintained by LRM.
3. Present one time for each port information structure maintained by the LAN reporting mechanism for the reporting link information structure.

Report LAN Manager Rejection (X'860A') Conditions of Presence

Table F-52. Report LAN Manager Rejection Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C00B	Rejection Code	6	Present one time.
4002	LAN Manager Address	10	Present one time.

New Reporting Link Established Notification (X'860B') Conditions of Presence

Table F-53. New Reporting Link Established Notification Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
8005	Reporting Link Information	20	Present one time.
C006	Reporting Link Identifier	6	Present one time.
4002	LAN Manager Address	10	Present one time.

Report LAN Manager Control Shift (X'860C') Conditions of Presence

Table F-54. Report LAN Manager Control Shift Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C00D	Status Code	6	Present one time.
C00E	Reason Code	6	Present one time.
C004	Number of Alternate LAN Managers	6	Present one time.
8005	Reporting Link Information	20	Note 1
C006	Reporting Link Identifier	6	Present one time.
4002	LAN Manager Address	10	Present one time

Note:

1. Present only if the Status Code subvector value is not equal to X'0000' (there is a new controlling LAN manager).

Report LRM Control Breach Attempt (X'860D') Conditions of Presence

Table F-55. Report LRM Control Breach Attempt Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
8005	Reporting Link Information	20	Present one time.
C006	Reporting Link Identifier	6	Present one time.
4002	LAN Manager Address	10	Present one time.

Close Reporting Link (X'860E') Conditions of Presence

Table F-56. Close Reporting Link Major Vector

This major vector contains no subvectors.

LRM Terminating (X'860F') Conditions of Presence

Table F-57. LRM Terminating Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
C010	Reason Code	5	Present one time.

Invalid Request (X'8610') Conditions of Presence

Table F-58. Invalid Request Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C012	Reason Code	6	Present one time.
C004	Number of Alternate LAN Managers	6	Present one time.
8005	Reporting Link Information	v	Present one time.
C006	Reporting Link Identifier	6	Present one time.
8007	Port Information	52	Note 1
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Present one time.
C008	Reporting Function Classes	8	Present one time.
C009	Available Function Classes	8	Present one time.
C00A	Active Function Classes	8	Present one time.
C011	Enabled Functional Addresses	8	Present one time.

Note:

1. Present for each port information structure maintained by the LRM for this reporting link information structure.

Set Reporting Point Error (X'8611') Conditions of Presence

Table F-59. Set Reporting Point Error Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C00F	Reason Code	5	Present one time.
4009	Error Code	6	Note 1
400A	Error Offset	6	Note 1
400C	LRM Version Level	18	Note 1

Note:

1. Conditionally present one time if the value of the reason code subvector is X'00', indicating a parsing error caused the frame to be returned to the LAN manager.

LRM Congestion (X'8612') Conditions of Presence

Table F-60. LRM Congestion Major Vector

This major vector contains no subvectors.

Management Servers Present (X'8613') Conditions of Presence

Table F-61. Management Servers Present Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
400C	LRM Version Level	18	Present one time.
8005	Reporting Link Information	v	Present one time.
8007	Port Information	20	Note 1
4002	Port Identifier	10	Present one time.
4005	Ring Number	6	Present one time.

Note:

1. Present for each port information structure maintained by the LRM for this reporting link information structure.

Alert Transport Frames

Alert Transport (X'8701') Conditions of Presence

Table F-62. Alert Transport Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.
C002	Major Vector Identifier of Error Frame	v	Present one time.

Alert Transport Received (X'8702') Conditions of Presence

Table F-63. Alert Transport Received Major Vector

Subvector ID	Subvector Name	Length	Condition of Presence
4001	Correlator	6	Present one time.

List of Abbreviations

ABM	asynchronous balanced mode
ABME	asynchronous balanced mode — extended
AC	access control field
ACSDU	access channel control service data unit
ADM	asynchronous disconnected mode
ALS	adjacent link station
ANSI	American National Standards Institute
ASCII	American National Standard Code for Information Interchange
BLU	basic link unit
BR	backup ring path
BRI	backup ring path — in
BRO	backup ring path — out
BTU	basic transmission unit
CCITT	International Telegraph and Telephone Consultative Committee
CD	collision detection
CRC	cyclic redundancy check
CRS	configuration report server
CSMA	carrier sense multiple access
CSTAT	command status
DA	destination address
DAF	destination address field
DC	direct current
DCE	data circuit-terminating equipment
DFC	duplicate frame counter
DLC	data link control
DP	data processing
DSAP	destination service access point
DTE	data terminal equipment
EBCDIC	extended binary-coded decimal interchange code
EC	engineering change level
ECMA	European Computer Manufacturers Association
ED	ending delimiter
EIA	Electronic Industries Association
FAI	functional address indicator
F bit	final bit
FC	frame control field
FCS	frame check sequence
FR	frame
FRMR	frame reject
FS	frame status field
FSM	finite state machine
HDLC	high-level data link control
Hz	hertz
IBM PC	IBM Personal Computer
ID	identifier
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Standards Organization
kHz	kilohertz
LAN	local area network
LLC	logical link control
LLID	major vector length and identifier
LPDU	logical link control protocol data unit
LRM	LAN reporting mechanism
LSAP	link service access point
LT	length and type
mA	milliamp
MAC	medium access control
Mbps	megabits per second
MHz	megahertz
MR	main ring path
MRI	main ring path — in
MRO	main ring path — out
MSAP	medium access control service access point
MSDU	medium access control service data unit
msec	millisecond
mV	millivolt
MV	major vector
MVID	major vector identifier
MVL	major vector length
NADN	nearest active downstream neighbor
NAUN	nearest active upstream neighbor
NDFC	non-duplicate frame counter

NOPSC no parameter server counter

N(R) receive sequence number

N(S) send sequence number

nsec nanosecond

OAF origin address field

ODAI OAF-DAF assignment indicator

OSI open systems interconnection

P bit poll bit

PDU protocol data unit

P/F bit poll/final bit

PTI permissible token indicator

PU physical unit

REJ reject

REM ring error monitor

RI routing information

RNR receive not ready

RPS ring parameter server

RR receive ready

RRTRYC response retry counter

RX receive data

SA source address

SABME set asynchronous balanced mode — extended

SAP service access point

SARM set asynchronous response mode

SD starting delimiter

SDU service data unit

sec second

SIF system interface

SNA Systems Network Architecture

SNRM set normal response mode

SSAP source service access point

SSB system status block

STA ring station's individual address

SVID subvector identifier

SVL subvector length

TNT no token timer

TX transmit data

UA unnumbered acknowledgment

UI unsequenced information

XID exchange identifier

Glossary

This glossary defines the Token-Ring Network terms and abbreviations used in this publication. It includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699. Definitions from draft proposals and working papers under development by the International Standards Organization, Technical Committee 97, Subcommittee 1 are identified by the symbol **(TC97)**.

A

access channel control. The collection of logic and protocol machines that manages the transfer of data from the link stations to medium access control and vice versa.

access priority. The maximum priority that a received token can have for the protocol handler to use it for transmission.

access procedure. The procedure or protocol used to gain access to a shared resource. In a local network the shared resource is the medium. The medium access procedures specified by the IEEE 802 standard are CSMA/CD, token bus, and token ring.

access unit. In the IBM Token-Ring Network, a wiring concentrator. See wiring concentrator.

active monitor. A function in a single adapter on a ring network that initiates the transmission of tokens and provides token error recovery facilities. Any active adapter on a network has the ability to provide the active monitor function if the current active monitor fails. Also known as "token monitor."

adapter. The circuit card within a communicating device, and its associated software, that enable the device to communicate over a local area network.

adapter address. The address of the Media Access Control Service Access Point (MSAP).

adjacent link station. From the perspective of a link station in an SNA node, the communicating link station in the adjacent SNA node.

all-rings broadcast frame. A frame that has the B-bit in the routing information field set to 1. In a network of interconnected rings (or MAC segments of all kinds), bridges forward all such frames to all rings (or MAC segments). The destination address is not examined and plays no role in bridge routing.

all-stations broadcast frame. A frame whose destination address is set to all ones. All stations on any ring on which the frame appears will copy it. Which rings it appears on is determined by the routing information, not the destination address. All-stations broadcasting is independent of all-rings broadcasting; the two can be done simultaneously or one at a time.

application program. A program written for or by a user that applies to the user's work.

attach. To attach a device logically to a ring network.

attaching device. Any device that is physically connected to a network and can communicate over the network.

B

basic link unit. The unit of data and control information transmitted over a link by data link control. In token-ring local area networks, the BLU is the unit of information exchanged between MAC entities (that is, ring stations) in different nodes or between ring stations in a node and a bridge. The IBM Token-Ring Network BLU consists of the starting and ending delimiters, the MSDU, the frame check sequence, and the frame status field.

beacon. A frame sent by an adapter indicating a serious ring problem, such as a broken cable. An adapter is said to be *beaconing* if it is sending such a frame.

bridge. A device that links networks that use the same logical link protocols.

broadcast topology. A network topology in which all stations are capable of simultaneously receiving a signal transmitted by any other station on the network.

C

code violation. In differential Manchester code encoding, a bit that does not have a state transition at the bit mid-point. See differential Manchester code.

cyclic redundancy check. A system of error checking performed at both the sending and receiving station after a block check character sequence has been accumulated.

D

data frame. See frame.

data link. Any physical link, such as a wire or a telephone circuit, that connects one or more devices or communication controllers.

data link layer (or level). (TC97) In open systems architecture, the layer that provides the functions and procedures used to establish, maintain, and release data link connections between elements of the network.

delimiter. A bit pattern that defines the limits of a frame or token on a ring.

destination address. The address of the Medium Access Control Service Access Point (MSAP) for which a Medium Access Control (MAC) frame is intended. Also, a field in the MAC frame.

Destination Service Access Point (DSAP) Address. The address of the Link Service Access Point (LSAP) for which a Link Protocol Data Unit (LPDU) is intended. Also, a field in the LPDU.

device. An input/output unit such as a terminal, display, or printer. See attaching device.

differential Manchester code. A data encoding method used by the IBM Token-Ring Network. In this method, a bit has the value of binary '0' if there is a state transition (polarity change) at the beginning of the bit boundary. The bit has the value of binary '1' if there is no transition. In a properly encoded bit, there must always be a transition at every *mid-point* position. If the transition is absent, a code violation results. Since only the presence or absence of the state transition (and not the actual polarity) determine a bit's value, this method of coding is polarity independent.

downstream. On a ring, the direction of data flow. Contrast with upstream.

E

express buffering. A method of improving the likelihood that a ring station will copy a MAC frame immediately, when the ring station's normal receive buffers are full.

F

faceplate. A plate for connecting data and voice connectors to a cabling system. It may be wall mounted or surface mounted.

frame. The unit of transmission in the IBM Token-Ring Network. It includes delimiters, control characters, information, and checking characters. A frame is created when a token has data appended to it by a node.

functional address. A subset of group addresses (MSAPs) that is encoded in bit-significant format, thereby allowing multiple individual groups to be designated by a single address.

G

group address. An address assigned to a collection of service access points (SAPs), either LSAPs or MSAPs.

group SAP. A single address assigned to a group of Service Access Points (SAPs).

H

hard error. An error condition on a ring that requires that the ring be reconfigured or that the source of the error be removed before the ring can resume reliable operation.

I

idles. Signals sent along a ring when neither frames nor tokens are being transmitted. In the IBM Token-Ring Network, idles are B'0's.

L

layer. One of the seven layers of the Open Systems Interconnection (OSI) reference model.

link. The logical connection between nodes, including the end-to-end link control procedures.

link connection. All physical components and protocol machines that lie between the communicating link stations of a link. The link connection may include a switched or leased physical data circuit, a local area network, or an X.25 virtual circuit.

link station. A protocol machine in an SNA node that manages the elements of procedure required for the exchange of data traffic with a communicating link station in an adjacent SNA node.

LLC protocol data unit (LPDU). The unit of information exchanged between link stations in different nodes. The LPDU consists of the DSAP and SSAP address fields, the control field, and the information field (if present).

lobe. In the IBM Token-Ring Network, the section of cable (which may consist of several segments) that attaches a device to an access unit.

local area network. A network in which communications are limited to a moderate-sized geographic area such as a single office building, warehouse, or campus, and which do not generally extend across public rights-of-way. Compare wide area network.

Logical Link Control (LLC). The DLC.LAN sub-layer that provides two types of Data Link Control (DLC) operation. The first type is connectionless service, which allows information to be sent and received without establishing a link. The LLC sub-layer does not perform error recovery or flow control for connectionless service. The second type is connection-oriented service, which requires the establishment of a link prior to the exchange of information. Connection-oriented service provides sequenced information transfer, flow control, and error recovery.

logical link control (LLC) procedure. (TC97) In a local area network, the part of the protocol that governs the assembling of transmission frames and their exchange between data stations independently of how the transmission medium is shared.

M

MAC segment. An individual local area network communicating through the MAC layer within this network.

MAC service data unit (MSDU). The MSDU consists of the LPDU (the DSAP and SSAP address fields, the control field, and the LPDU information field, if present) and the routing information field (if the destination station is located on a different ring).

medium. A physical carrier of electrical energy.

Medium Access Control (MAC). The sub-layer of DLC that supports medium-dependent functions and uses the services of the physical layer to provide services to Logical Link Control (LLC). The MAC sub-layer includes the medium access port.

medium access control (MAC) procedure. (TC97) In a local area network, the part of the protocol that governs access to the transmission medium independently of the physical characteristics of the medium, but taking into account the topological aspects of the network, in order to enable the exchange of data between data stations.

medium access port. A hardware-addressable component (such as a communication adapter) of an SNA node by which the node has access to a transmission medium and through which data passes into and out of the node.

monitor. The *function* required to initiate the transmission of a token on the ring and to provide soft-error recovery in case of lost tokens, circulating frames, or other difficulties. The capability is present in all ring stations.

N

nearest active upstream neighbor (NAUN). For any given ring station on a ring, the ring station directly upstream that is participating in the ring protocols.

network. The assembly of equipment through which connections are made between data stations. See ring network.

node. In SNA, an end point of a link or a junction common to two or more links in a network.

P

port. See medium access port.

primitive. An abstract, implementation-independent interaction between a user of a service and the provider of the service.

protocol. (TC97) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved.

R

remove. To take an attaching device off the ring.

ring. See ring network.

ring network. A network configuration where a series of attaching devices are connected by unidirectional transmission links to form a closed path.

ring station. The functions necessary for connecting to the local area network and for operating with the token-ring protocols. These include token handling, transferring copied frames from the ring to the using node's storage, maintaining error counters, observing Medium Access Control (MAC) sub-layer protocols (for address acquisition, error reporting, or other duties), and (in the full-function native mode) directing frames to the correct Data Link Control link station. A ring station is an instance of a MAC sub-layer in a node attached to a ring.

ring status. The condition of the ring.

ring topology. A logically circular, unidirectional transmission path without defined ends. Control can be distributed or centralized.

S

server. A device on a network dedicated to specific functions.

service access point. The logical point at which an n+1-layer entity acquires the services of the n-layer. In this reference, the layer is assumed to be DLC.LAN. A single SAP can have many links terminating in it. These link "end-points" are represented in DLC.LAN by link stations.

soft error. An intermittent error on a network that causes data to have to be transmitted more than once to be received. A soft error does not, by itself, affect the network's overall reliability. If the number of soft errors reaches the ring error limit, reliability is affected.

source address. The address of the Media Access Control Service Access Point (MSAP) from which a Medium Access Control (MAC) frame is originated. Also, a field in the MAC frame.

Source Service Access Point (SSAP) Address. The address of the Link Service Access Point (LSAP) from which a Link Protocol Data Unit (LPDU) is originated. Also, a field in the LPDU.

subvector. A subcomponent of the MAC major vector.

T

token. A sequence of bits passed from node to node along the network. It consists of a starting delimiter, a frame control field, and an ending delimiter. The frame control field contains a token bit that indicates to a receiving node that the token is ready to accept information. If the node has data to send along the network, it appends the data to the token. The token then becomes a *frame*.

token-claiming. A line-control scheme in which stations on a line compete for the use of that unused line; the station that is successful in gaining control of the line is able to transmit.

token ring. A network with a ring topology that passes tokens from one attaching device to another. The IBM Token-Ring Network is a token-ring local area network.

transmission medium. A physical carrier of electrical energy or electromagnetic radiation.

U

upstream. On a ring network, the direction opposite to that of data flow. Contrast downstream.

V

vector. The MAC frame information field.

W

wide area network. A network that provides data communication capability in geographic areas larger than those serviced by local networks.

wire fault. An error condition caused by a break in the wires or a short between the wires (or shield) in a segment of cable.

wiring concentrator. A lobe concentrator that allows multiple attaching devices access to the ring at a central point such as a wiring closet or in an open work area.

Index

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

Numerics

IBM Token-Ring Network
Architecture Reference

Publication No. SC30-3374-2

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, date, and location of your local IBM branch office:

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Dept. E02
P.O. Box 12195
Research Triangle Park, N.C. 27709-9990

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

IBM Token-Ring Network
Architecture Reference

Publication No. SC30-3374-2

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, date, and location of your local IBM branch office:

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

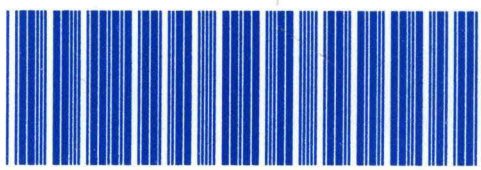
POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Dept. E02
P.O. Box 12195
Research Triangle Park, N.C. 27709-9990

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

SC30-3374-02



Printed in U.S.A.