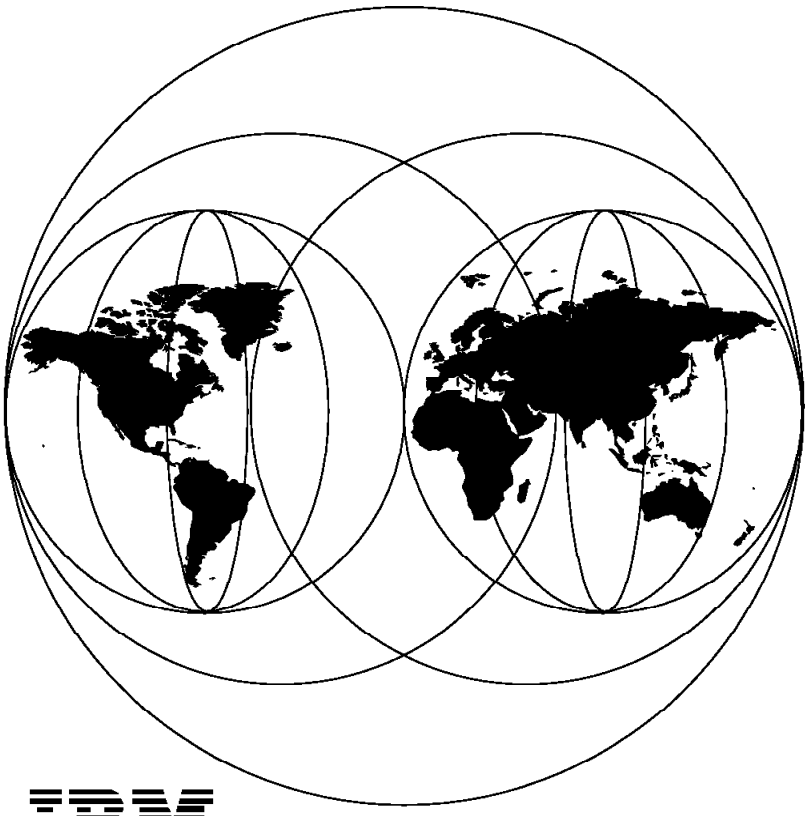
SG24-4919-00

DCE and DFS Performance Tuning and Problem Determination on AIX and OS/2 Warp

June 1997



IBM

International Technical Support Organization

Austin Center

International Technical Support Organization

SG24-4919-00

DCE and DFS Performance Tuning and Problem Determination on AIX and OS/2 Warp

June 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix F, “Special Notices” on page 259.

First Edition (June 1997)

This edition applies to the IBM Directory and Security Servers for AIX, Version 4, and the Directory and Security Server for OS/2 Warp, Version 4, family of products.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures

Tables

Preface

This redbook lists and describes ways and methods for evaluating and improving overall system performance in a heterogeneous Distributed Computing Environment (DCE) installation. It covers the various components, such as the network, network interfaces, the operating systems, and the standard software, involved in DCE-related distributed applications and describes their influence on the overall system performance. Tools that can provide help in the evaluation and tuning process are explained as well.

As a second part, this redbook addresses the problem determination and resolution process and describes possible causes and solutions for many common problems in DCE installations.

Several practical examples are presented to demonstrate the use of the tools and the configuration changes being explained throughout the book.

Some sound knowledge of the DCE principles and DCE configuration and administration is assumed. Also, the reader should know at least one of the target platforms: IBM AIX or IBM OS/2 Warp.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Heinz Johner is an Advisory Systems Engineer at the International Technical Support Organization, Austin Center. Before joining the ITSO in mid-1996, he worked for IBM Switzerland, and he had the overall responsibility for AIX, DCE, and systems management projects for large customers. He was also involved as a consultant in various other customer projects in the same technical areas.

Patrik Larsson is a Systems Engineer at Rowika AB, an IBM business partner in Sweden. He is also an IBM-certified AIX engineer (all levels). He provides AIX, Catia-integration, and DCE/DFS support (design, implementation, problem resolution) to customers in small and large UNIX environments.

Ingolf Lindberg is an IBM Senior Systems Engineer in Denmark. He has 14 years of experience in distributed systems in the PC environment. He holds a Master of Electrical Engineering degree from the Technical University of Denmark. His areas of expertise include systems connected in Local Area Networks, communication protocols, application development, systems design, and systems management. He has written extensively on OS/2 LAN Server environments.

Miguel Otero is an Advisory System Engineer in Venezuela. He has 14 years of experience in VM problem determination and resolution, REXX application development, and in OVVM and VM implementation. He has worked at IBM for 18 years. His areas of expertise include Office Vision, DB2, REXX, APPC, TCP/IP, DCE, and DFS. He has written extensively on the OpenEdition for VM/ESA DCE implementation and on POSIX implementation in VM/ESA.

Thanks to the following people for their invaluable contributions to this project:

Dave Bachmann and the members of his DCE performance team
IBM Austin

Liz Hughes and the members of her DFS development team
IBM Austin

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Part 1. DCE and DFS Performance Tuning

Chapter 1. Introduction

Performance analysis and problem determination in large, heterogenous and distributed environments is not an easy task. Not only may the number of involved components introduce unknown levels of complexity, but the final goal of such investigations and considerations may not be obvious and clear right from the beginning. This chapter discusses the term *performance* and gives some distinct views of what it can mean.

1.1 What is Performance?

Without looking for an exact, scientific definition in a reference, we all have a certain understanding of what good performance in information technology (IT) environments means:

- Fast
- Short response times, immediate reaction
- Quick start and stop of requested programs and services

We are not going to invent a new definition for performance specifically for distributed environments. There are a number of benchmarks available for evaluating the performance of a CPU or how fast a terminal can display some graphics, and for many other purposes, but it is almost impossible to define methods to measure how good a distributed computing environment performs. Generally speaking, performance is the ratio of a certain work, divided by the time it took to complete. In a computing environment, performance is usually regarded as ″how long does this request need to be serviced″, or ″how many requests can this system handle in a single second″. All these phrases basically refer to the same ratio: work divided by time.

Performance tuning in a distributed environment concentrates mainly on:

- Achieving fast, or at least acceptable, response times for remote requests to be fulfilled
- Circumventing bottleneck situations that may seriously slow down a whole infrastructure in any shared resources, such as servers
- Providing scalability options for the expected growth of the overall system
- Keeping the required investments in an acceptable range

These all involve thorough planning (design), methods for evaluating, and ways to detect and resolve performance-related problems.

Every installation site needs to investigate which functions are the important—which are the ones that should be carried out quickly within an acceptable amount of time? This may not only be for people who are waiting for a response; many services requesting programs also have timing dependencies which may cause problems if a response is not received within a certain period of time.

It is obvious that a distributed application cannot be faster than a local one unless remote, high-performance computing power is involved in a certain distributed transaction.

1.2 Why Do We Want the Best Performance?

It is a well-known habit of the human being that he/she does not like to wait. Waiting for a machine to deliver an expected result is widely considered boring, even if it lasts only seconds, or even parts of a second. People using computers—widely known as *users*, or *end users*—are by no means exceptions to this rule, and we all know that we easily get frustrated when we have to wait over and over for computers to fulfill our requests.

Although this may not sound like a strong technical reason to invest fortunes in order to improve performance, it is in fact one of the most important reasons. Employee motivation and productivity depend on many things, one of which is certainly the infrastructure they are to use. Also, customer services, one of the most important business areas for many companies, is more often than not directly dependent on computer systems and their quick processing capabilities.

Despite the human being—be it an employee in a company or a company's customer—as the main requester for performance, other requirements do well exist. In a manufacturing area, for example, machines involved in a manufacturing process need exact and timely information. An airline reservation system is another example of a distributed system where performance affects us all as customers. A passenger may purchase the last available ticket of a certain flight at almost any place in the world and another wants to know immediately and reliably from a travel agent if that last seat is still available on that flight or not.

Asked for the level of performance people want, ″best possible performance″ would probably be the answer most often given. This can only be achieved with the most powerful computers and networking technology. For obvious reasons, this cannot be the aim of performance tuning.

1.3 What is Appropriate Performance?

There are several reasons why companies cannot achieve—or even want—best performance. The most obvious reason is the lack of adequate resources, or, in other words, the lack of sufficient budget for equipment and labor. This is certainly not out of the ordinary because every organization has its limited financial resources (and even if it had the financial ability, finding the right skills and equipment could yet be another limitation of resources).

Depending on the environmental requirements, the achievement of an appropriate level of performance is the art of either:

1. Providing the best achievable performance results with a given budget by selecting the best components and fitting them together in a proper way, or
2. Providing the best achievable performance results by configuring and customizing existing equipment, middleware, and applications

In both cases, exact knowledge of the technical processes and the performance requirements is a mandatory requisite for the task of performance tuning. It is the intension of this book to serve as a guide for professionals involved in this kind of activity. Performance tuning starts with the design of a distributed application and goes on with its implementation, involves a proper network, system design and component selection, and involves a number of small tuning actions on many of the involved components.

Performance tuning typically is a never-ending task. A system administrator should recognize the tasks of continuously monitoring critical resources and finding and resolving bottlenecks as a constant responsibility. In simple terms, performance tuning is the task of eliminating bottlenecks. It is almost a natural law that every system (including distributed systems) has a bottleneck. Eliminating a bottleneck will immediately bring up the next one, and so on. The vision of having a well-balanced system in a distributed environment is almost a theoretical one, since in practice there are so many changing operating conditions that make such a goal impossible. Conflicting targets exist as well, for example a small investment budget versus high-performance server systems or distributing replica servers versus manageability.

Chapter 5, “Planning for Performance” on page 93, describes some important issues you should bear in mind during the process of system planning for good performance.

Chapter 6, “Step-by-Step DCE Set Up Geared to Performance” on page 123, then explains how the base components should be installed in order to achieve a smoothly performing installation.

Finally, Chapter 7, “After-Installation Performance Tuning” on page 141, and Chapter 4, “Tools and Methods for Performance Evaluation” on page 63, describe what to do after an installation to keep overall system performance at satisfying levels.

Chapter 2. Process Flow and Components Affecting Performance

In a running DCE cell, multiple DCE servers and DCE clients are participating. These servers and clients form the infrastructure of one or more system solutions.

The DCE components are traditionally divided into the DCE core services that are the foundation for any other DCE services and DCE applications. The DCE core services provide a framework that is necessary for DCE applications to make use of the rich set of services that the DCE structure provides. Besides the applications themselves, the DCE core services generate DCE framework-related load on the systems participating together, and they generate additional network traffic. The following sections describes the DCE framework-related load and network traffic, which can have an effect on the overall system performance. The sections describe:

- The DCE services in a distributed environment. DCE server functions and DCE clients are distributed within the core services each providing subfunction within the overall DCE core services.
- The application programmer's view of a DCE framework usage, which describes the subfunctions a DCE server and DCE client will make use of.
- Selected basic DCE core service flows, their impact on network traffic, and which details the traffic imposed by the DCE core services and the network load created during normal operations.
- The performance impact on components, which describes the components involved on the systems. DCE daemons have very few configuration parameters that can be set.
- The performance impact from an end-to-end view, which focuses on typical bottlenecks affecting the overall performance in a DCE cell.
- Selected testing performed, which contains actual test.
- Figures for selected DCE-related scenarios.

To establish a balanced, smoothly performing DCE-based system, multiple performance considerations apply. A DCE-based system is very dependent on the DCE core services. For this reason, DCE core services can make use of replicated and backup services.

This chapter concentrates on the DCE core services and does not cover the DCE Distributed File System (DFS). In Chapter 3, "The Distributed File System" on page 47, we look more closely at the DFS.

A primary purpose of this chapter is to quantify network traffic imposed by DCE core functions. While this may not create any performance problems in a small, LAN-interconnected DCE cell, it can certainly require some attention in medium or large installations with relatively slow WAN links in between the high-speed LANs.

2.1 An Overall View of the Distributed Components

The basic nature of DCE is very dynamic. Whenever DCE applications are active, the DCE core services must be available. The experienced DCE cell administrator usually can quickly identify the bottlenecks a system will have. To remove bottlenecks, the cell administrator makes use of a number of actions that often make use of DCE features. The basic concept is to estimate the traffic and load created by the overall DCE system and add sufficient capacity in places where problems can occur. The other concept is to bring most used DCE core services near to the DCE application clients. This is done by making use of the replication capabilities of the DCE core services.

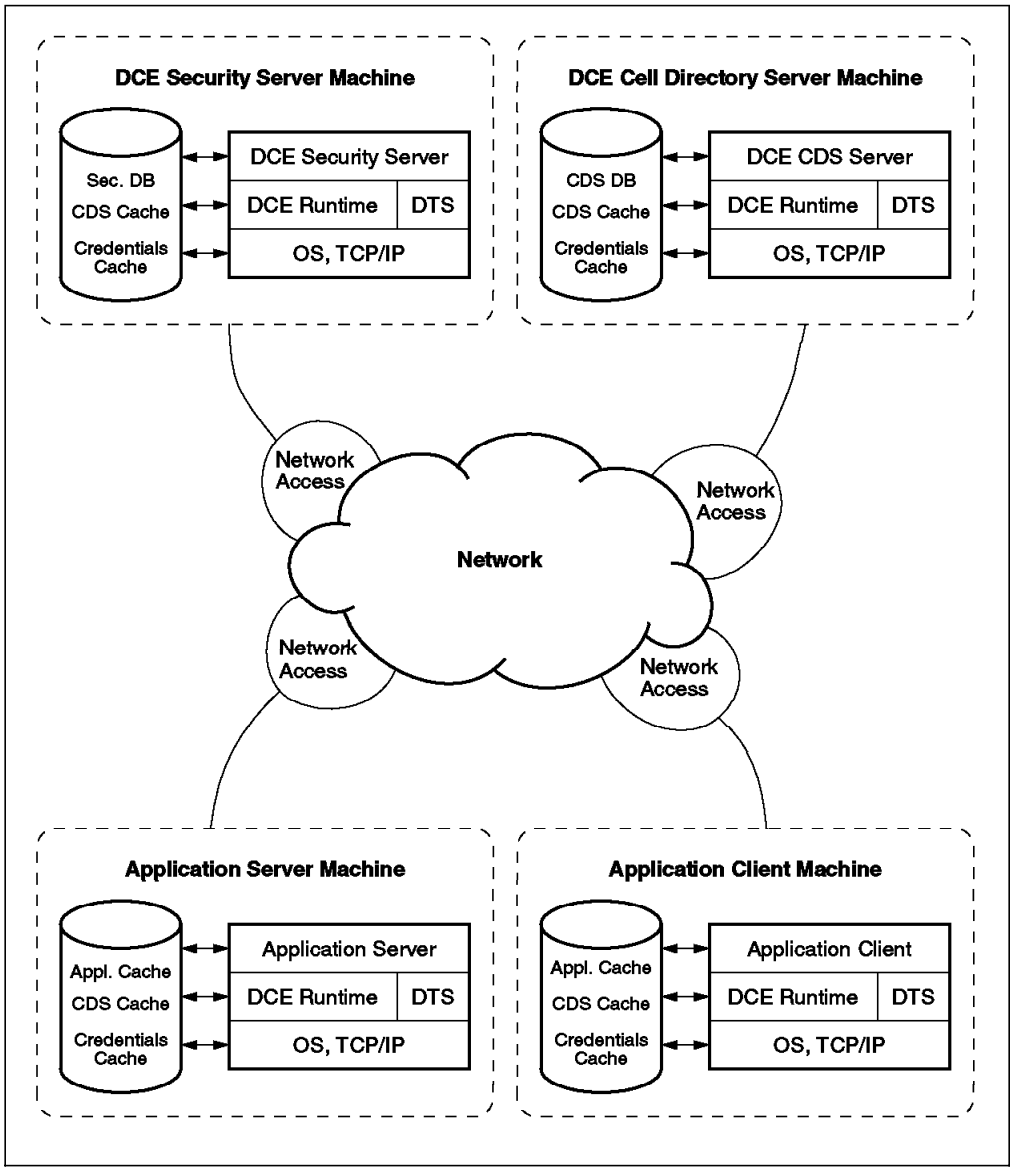


Figure 1. Overall View of a Distributed DCE Client/Server Environment

Figure 1 illustrates the typical functional blocks that make up a minimal DCE cell with a DCE application server (bottom left) and a DCE application client (bottom right). For the sake of simplification, only one replica per server service is shown, and we left out any DTS and DFS services. There are several components involved in DCE-related operations on each machine. For a further

investigation and understanding of performance issues, it is important to mention that each machine has its own connection to the network, which is not necessarily the same type in terms of throughput for all machines. Although the major part of the network can be of a high-speed type, some clients (or even servers) could be attached through relatively slow links. This is depicted with the ″Network Access″ clouds in the figure.

The load and network traffic DCE applications and DCE core services introduce are measurable. Whenever a DCE application client attempts to make use of a DCE application server, the client must perform authentication or make use of an already authenticated identity. The authentication process makes use of the DCE Security Server, which, after positive authentication, provides the DCE client with an Extended Privilege Attribute Certificate (EPAC) and some other pertinent information. The EPAC describes the user and which groups the user is a member of.

The next step typically involves a lookup into the Cell Directory Service (CDS) namespace. This lookup is necessary to find the location of an application server. The DCE application client contacts the DCE application server and presents the EPAC together with some other information. The DCE server performs authorization, if implemented, based on the EPAC and grants or denies access to the requested resources on the server. Repeated DCE remote requests to the application server do not require the same amount of information exchange because authentication information is stored in a cache within the DCE client part of the DCE runtime.

While Figure 1 on page 8 shows the components involved in each machine role for referential use, Figure 2 on page 10 depicts a more practical, large DCE cell with centralized server machines, as can be found in many large-scale installations for development sites or at universities with thousands of client machines and tens of thousands of users.

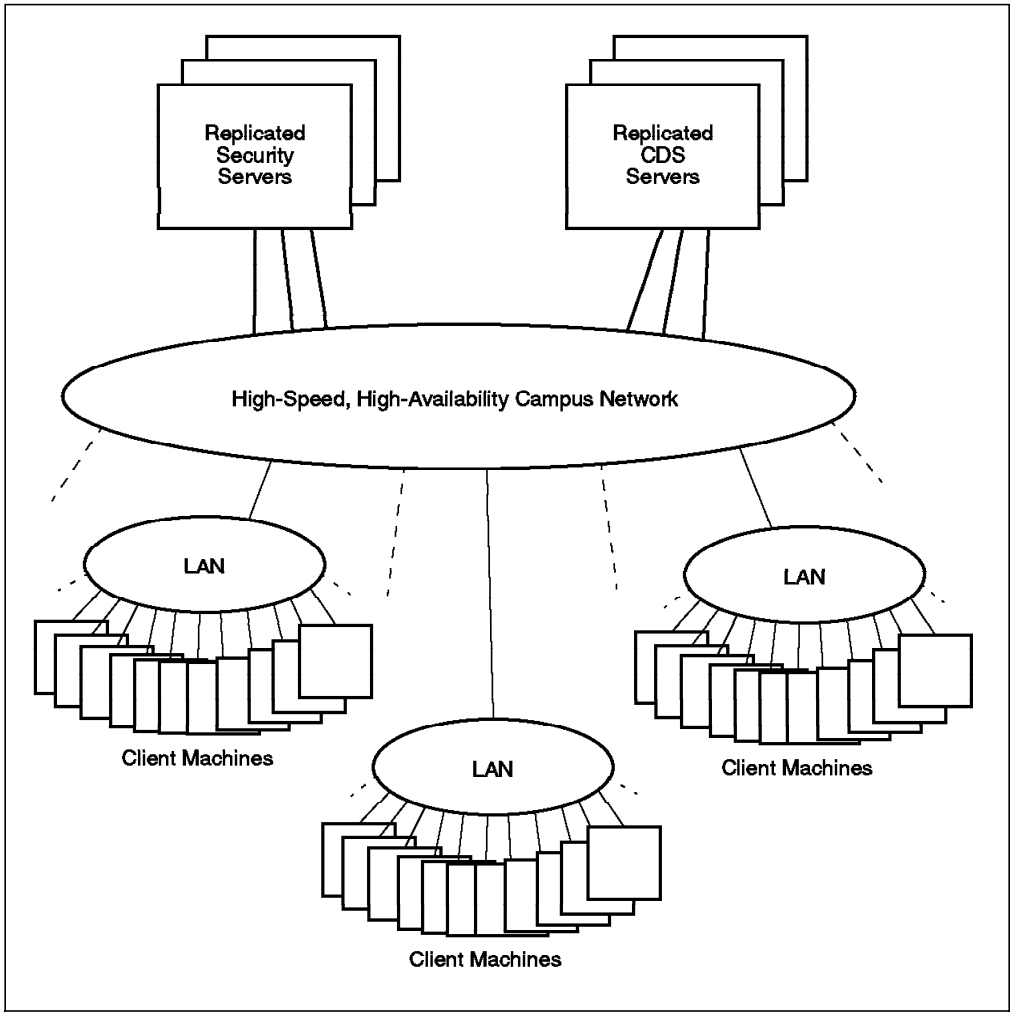


Figure 2. Large DCE Cell With Centralized Servers

Such DCE cells as outlined in Figure 2 typically consist of two to three DCE Security Servers and two to three CDS Servers. The main reason for having multiple, replicated servers is to increase overall availability. Unlike most other client/server applications, such a DCE cell with thousands of clients concentrates its load in a few central servers and network end points. It is the purpose of this chapter to show where potential bottlenecks in such installations could be.

DCE components communicate with each another by using DCE Remote Procedure Calls (RPCs). The following sections explain some general concepts about the DCE RPCs in order to understand why the DCE component imposes a certain amount of network packets and traffic to perform specific actions.

2.2 Introduction to RPC Events

Throughout the rest of this chapter, we see that DCE core services create some network traffic between server and client systems. We need to know about this traffic in order to be able to size the network and the computer's interfaces. At first glance, the amount of traffic seems to be high, even for basic DCE services such as a user log-in. Only if one understands the complexity of DCE and its authorization protocols can the requirement for this traffic can be understood.

We will not attempt to explain DCE network traffic with all its details, but we will explain the basics and the involved components where appropriate so that it is easier to understand why a simple action may lead to more network traffic than one would have anticipated if these details were not known.

2.2.1 Variations in RPC Calls

DCE RPC and DCE Security cooperate together to provide DCE authentication and secure communications. In order to use authenticated RPC, the DCE client must already have established a security environment (i.e. a security login context for the current process environment). The DCE server application registers its name and the type of authentication service it supports. The two types of authentication services DCE provides are secret key authentication, which is based on Kerberos, or no authentication.

The DCE client application makes a call to indicate which authentication service, protection level, and authorization service it wants to use for DCE RPC communications with a given DCE application server. This DCE function call is `rpc_binding_set_auth_info()`. The authentication service can be either secret key authentication or no authentication. The protection level ranges from authentication at the beginning of a DCE RPC session, to authenticating each message or packet to ensure that a packet has not been modified in transfer, to a level where all user data is encrypted. As a general rule, selecting a more secure protection level imposes a higher performance impact. The reason for this is that the security mechanisms involve encrypting and decrypting of data, which takes up CPU time.

The authorization service selected by a DCE client can be either uncertified or certified. In uncertified authorization, the authorization information sent to a DCE application server consists only of the DCE application client name. In certified authorization, the authorization information consists of the Unique Universal Identifiers (UUIDs) of the DCE application authenticated name and the groups the name belongs to. The certified authorization information is in the form of an Extended Privilege Attributes Certificate (EPAC), which is produced by the Privilege Service of the DCE Security Server. In both the certified and uncertified authorization service, the authorization information is sent securely.

The authentication and authorization information about the DCE application client is used by the DCE application server to determine whether the DCE application client should be granted access to the resource it has requested. The DCE application server knows through the EPAC the identity of the DCE application client and what authorization groups the DCE application client belongs to. The DCE application server can therefore compare the DCE application client credentials against information in DCE application-server-controlled Access Control Lists and determine whether the DCE application client should be granted access.

The bottom line is that a single application service called by a single remote procedure call may cause very different types of network and CPU usage depending on the services that are being used along with the call.

The next sections describe some typical events that occur when using DCE core services, events produced either by applications or by the DCE core services themselves.

2.2.2 Common DCE RPC Events

Using DCE RPC, there are common steps a programmer will make use of when implementing DCE server and client applications. Similarly, the DCE core services will make use of the same (or similar) steps to provide the services they represent. The steps are represented in the following table show the simplest steps involved when not making use of authentication. The table shows the events when an applications server is started up and when an application client connects to this server.

Table 1. DCE Common Events for DCE Servers and Clients

Event	DCE Server	DCE Client
Register interface	`rpc_server_register_if()`	
Create binding information	`rpc_server_use_all_protseqs()`	
Register Endpoints	`rpc_ep_register()`	
Advertise server location	`rpc_ns_binding_export()`	
Listen for RPC calls	`rpc_server_listen()`	
Get binding information		`rpc_ns_binding_import_begin()`
Make remote procedure call		execute
Find server system	network	DCE client stub
Find server process	endpoint mapper	endpoint mapper
Prepare input		marshall input
Receive input and dispatch to DCE stub	DCE server stub	
Convert input	unmarchall input	
Execute remote procedure	execute procedure	
Prepare output	marshall output	
Receive output		DCE client stub
Convert output		unmarshall output

Upon start, the DCE application server performs the rpc_server_register_if() call to inform the DCE runtime about the interfaces it offers. The interface specification is created by the IDL compiler and contains information related to the RPC procedures it offers to clients.

The rpc_server_use_all_protseqs() call applies the network interface information into the binding information. The information applied is the protocol that can be used and the network address used by the protocol. This information can be queried with the dcecp rpcentry show command, for example:

```
dcecp> rpcentry show /.:/sample/dce_server
{8d33b761-3f98-11d0-917e-10005a250d1b 1.0
  {ncadg_ip_udp 9.3.1.131}
  {ncacn_ip_tcp 9.3.1.131}}
```

In this example, the exported protocol sequences are ncadg_ip_udp and ncacn_ip_tcp. The network address for the shown protocols is 9.3.1.131.

The rpc_ep_register() call creates an element in the local endpoint map. This will enable a client, by using the binding information together with the port number, to find the remote procedure. The dcecp endpoint show command can be used to list this information, for example:

```
dcecp> endpoint show -interface {8d33b761-3f98-11d0-917e-10005a250d1b 1.0}
{{interface {8d33b761-3f98-11d0-917e-10005a250d1b 1.0}}
 {binding {ncacn_ip_tcp 9.3.1.131 1149}}
 {annotation {dce server version 1.0}}}

{{interface {8d33b761-3f98-11d0-917e-10005a250d1b 1.0}}
 {binding {ncadg_ip_udp 9.3.1.131 1121}}
 {annotation {dce server version 1.0}}}
```

In this example, the application server makes use of socket port number 1149 for the ncacn_ip_tcp and port number 1121 for ncadg_ip_udp. In network traces, those sockets port numbers are the direct link to which RPC server is participating in the network traffic.

The rpc_ns_binding_export() call exports the created binding information to the CDS namespace. This is a one-time operation, by which the Name Service Interface (NSI) places the binding information into a CDS directory entry.

The rpc_server_listen() call places the application server into listen mode. This indicates the server is ready to accept remote procedure calls from clients.

The DCE client application performs an rpc_ns_binding_import_begin() call to obtain binding information the DCE sever application made available. Together with other rpc_ns_binding() functions, the DCE client application is able to find the location of the remote procedure.

The application client then performs the remote procedure call(s), which lead to some kind of application-related data transfer. This data transfer involves marshalling and unmarshalling of input and output data. The application client's runtime part uses the binding information to:

- Connect to the application server's endpoint mapper using a well-known port
- Retrieves the socket port number of the application server's procedure(s)
- Connects to the DCE server procedure(s) using the socket port(s) obtained

The endpoint mapper uses the well-known socket port number 135 for the process of obtaining socket port numbers.

This can, for example, be seen on an OS/2 Warp system running two instances of a DCE client application (a similar output on AIX can be obtained with the `netstat -A` command):

```
C:> netstat -s
-------------------------------------------------------------------------
                          AF_INET Address Family :
SOCK     TYPE     FOREIGN PORT     LOCAL PORT     FOREIGN HOST     STATE
====  =========  =============   =============   =============  =============
231     STREAM           1149            1071       9.3.1.131    ESTABLISHED
215     STREAM           1149            1068       9.3.1.131    ESTABLISHED
...
34       DGRAM              0    loc-srv..135         0.0.0.0    UDP
32      STREAM              0    loc-srv..135         0.0.0.0    LISTEN
...
```

The output of the `netstat` command in the above example has been shortened. It shows that the DCE client application is started two times, having a connection to the remote socket port number 1149. This socket port number (1149) corresponds to the one found in the previously executed dcecp command.

If traces are taken, the endpoint mapper traffic is always in process, whenever a client does not know the socket port number of the DCE server. This type of sequence normally creates four frames on the network, one for each socket port to be identified.

2.3 Process Flows and Imposed Network Traffic

In order to be able to calculate network loads and network interface utilization figures for the involved machines, a system planner needs to know a good estimate of the anticipated traffic imposed by the services running on all the networked machines. More than that, he/she also needs to know the peak hours during normal daily work schedules. We discuss this latter topic in more detail in 2.5, “An End-to-End View of Performance Tuning” on page 39.

The following sections describe, in terms of network traffic, the events that take place during the typical scenarios occurring in a DCE environment. The most common events that take place are described.

The following process flows and events are described:

- The startup of a DCE client machine. DCE client services need to register with the DCE cell and therefore create some network traffic and server load.
- The user login process, which causes some load to be generated on the network and the Security Server. The amount of load generated by the DCE login process is not controllable by administrative action.
- The application server registration, which causes export of binding information into the CDS namespace. The amount of load created by the export action is normally very insignificant because DCE servers only perform the binding export every time on start. DCE application servers are assumed to be long-running application. If this is not the case, significant load may be generated, which affects performance on both the network and in the CDS.

- The DCE application client generated load. The client authentication data (credentials) is cached locally, but the authorization information (EPAC) needs to be transferred to the server when establishing contact, together with the Remote Procedure Call parameters, if authorization is performed in the DCE application.
- The DCE Distributed Time Service (DTS) Server and clerk will generate some traffic to the network. The amount of traffic generated depends on the selected configuration.
- The Security Server replication, which causes load on the Master Security Server in addition to the load created on the network due to replica update data.
- The CDS and directory clearinghouses, which will introduce load on the local clearing houses and traffic on the network.

The sections are divided into an introduction to the scenario and an example of the network traffic generated. The results of the example traffic generated are commented.

Important Notice

In the following sections, as well as in subsequent chapters, we refer to figures and numbers that have been measured in a certain test environment.

As an ongoing effort to improve the quality of its products, IBM regularly releases updates and enhancements to its products, including DCE and DSS. Such updates can also include performance improvements in different areas without special notice or announcement. Also, various factors determine the operation of DCE functions, such as the size of a cell, the number of principals in the registry, or the type of protocols used.

The numbers and figures outlined in this publication are therefore for information only and should not be considered as being representative of or guaranteed for other environments or valid after future product updates. They give you, however, an understanding of what network traffic and CPU load is imposed by the described operations.

2.3.1 Client Machine Startup

When a DCE client machine starts up, or when DCE client services are started on a machine, the DCE client daemons need to be registered with DCE. These are at least the DCE daemon (`dced`) and the CDS advertiser (`cdsadv`). Depending on the client machine's configuration, a DTS clerk (`dtsd`) and a DFS client (`dfsd`, `dfsbind`) may also be started up.

The information from this test is especially useful in environments where client machines are powered off and on regularly, such as Personal Computers, which may be turned off every evening and back on every morning, or they might be power-cycled even several times during a normal working day.

Registering such client daemons involves some related network traffic that we want to quantify. Although the traffic of all the single DCE client daemons may not be as interesting as them all together, we will show them separately.

The following table summarizes the total traffic (inbound and outbound) measured at the network interface of a AIX DCE client machine generated when starting DCE client services. As with all of the following tests in this section, they have been done several times, usually five to twenty times, in order to get good average numbers. Through filtering, only DCE-related traffic between the client machine and the involved DCE server machines has been captured:

Table 2. Network Traffic Created by Starting DCE and DFS Client Daemons

Example Measurement	Average Number of Packets	Average Packet Size (Bytes)	Maximum Traffic (Bytes), see Note
Start of `dced`	81	413	33999
Start of `cdsadv`	132	270	36060
Start of DTS Clerk	281	250	70426
Start of DFS Client	215	256	58414
Total, Full Client	709	274	198899

Note: The figures in the last column of Table 2 and in subsequent tables in this chapter are not average values; they represent the highest observed value from all the tests in a certain test run. Actual values for most practical use can be calculated by multiplying the average numbers of packets and the average packet sizes. As an example, the total average traffic for all DCE and DFS client daemons would be about 190 KB.

Details about the test scenarios can be found in the appendix section under E.4.1, “Start of the DCE Client Daemons” on page 248.

Analysis, Comments and Additional Information About the Results: Some additional information may be helpful to understand the figures in Table 2. Supplementary tests have been done in order to provide a better understanding for a correct interpretation.

- Depending on the specific environmental factors, such as local caching in the client, the figures vary in a certain range.
- Roughly, it can be said that a full DCE client machine, including DTS clerk and DFS client, imposes about 190 KB of network traffic in about 700 packets.
- The figures are about the same on OS/2 Full Clients and AIX DCE clients. Since they use the same basic DCE architecture and protocols, any difference would have been a surprise.
- The start of an OS/2 Slim Client, in terms of network traffic to the CDS Server, is about the same as the start of the CDS advertiser on AIX. It does, however, not involve any traffic to the Security Server.
- It can be observed that the network traffic imposed by the start of the CDS advertiser (cdsadv) varies more than others. After a cold start of an AIX machine, the traffic can be up to four times the values shown in Table 2. This is true for OS/2 Warp clients as well.
- The values do not seem to be dependent on the size of the cell. Measurements on client systems in large DCE cells showed about similar figures.

2.3.2 User Authentication (dce_login) and Re-Authentication (kinit)

When an account, for example, typically a user, performs a DCE login, the result of the login is to perform DCE authentication using Kerberos' third-party authentication protocol. The following diagram shows the general interactions between a client and the Security Server for a user login.

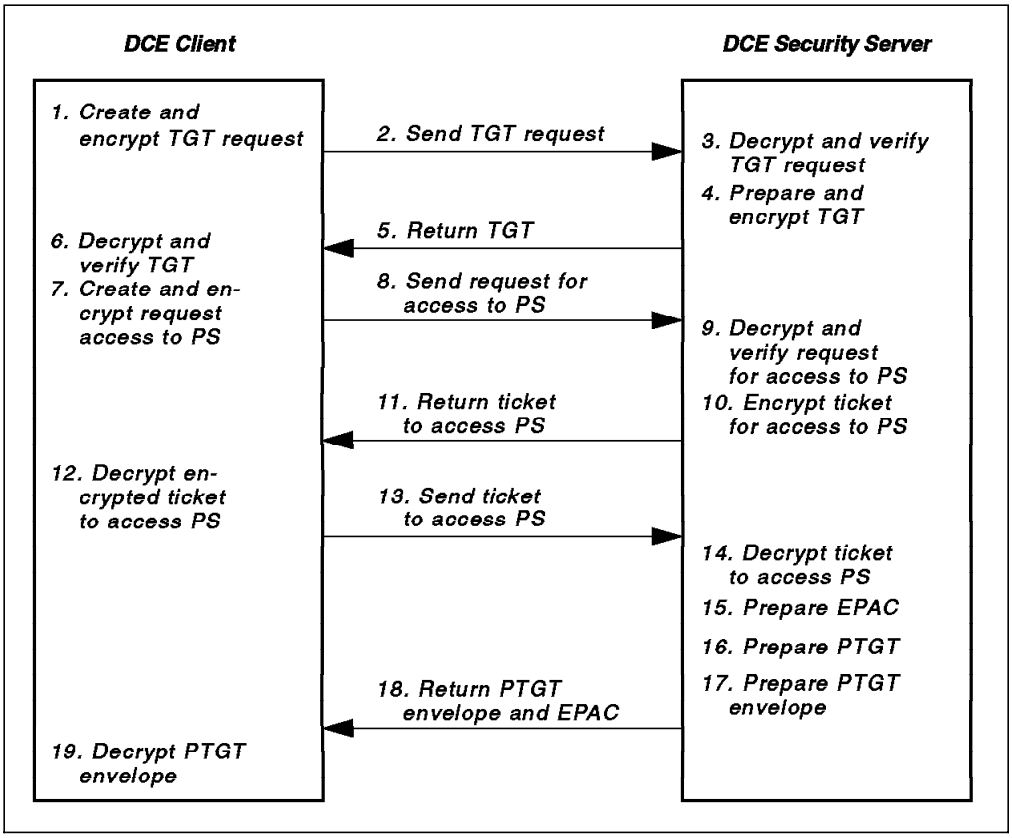


Figure 3. DCE Login (Client Authentication)

The authentication process performs the following steps:

1. The DCE login process prepares a request for a Ticket Granting Ticket (TGT). The TGT is encrypted.
2. The encrypted TGT is sent to the authentication service of the DCE Security Server.
3. The authentication service of the DCE Security Server decrypts the TGT and validates the user using information stored in the Security Server registry.
4. The authentication service of the DCE Security Server prepares a TGT and encrypts it.
5. The encrypted TGT is sent to the DCE client.
6. The DCE login process decrypts and verifies the TGT.
7. The user enters the password. The DCE login process prepares a request to access the Privilege Service (PS) of the DCE Security Server.
8. The request is sent to the Privilege Service of the DCE Security Server.
9. The authentication service decrypts the request to access the Privilege Service and verifies the request.

10. The authentication service prepares a ticket to access the Privilege Service. The ticket is encrypted and packaged in an envelope, which also is encrypted.
11. The encrypted envelope containing the ticket is sent to the DCE client.
12. The DCE login process decrypts the encrypted ticket to access the Privilege Service.
13. The DCE login process sends the encrypted ticket to access the Privilege Service to the Privilege Service of the DCE Security Server.
14. The Privilege Service decrypts the ticket.
15. The Privilege Service prepares an Extended Privilege Attribute Certificate (EPAC) that contains information about the user and a list of groups of which the user is a member.
16. The Privilege Service prepares a Privilege Ticket Granting Ticket (PTGT) containing an encryption key and the EPAC checksum. The PTGT is encrypted.
17. The Privilege Service prepares an envelope containing a encryption key and the PTGT. The envelope is encrypted.
18. The encrypted envelope and the EPAC are sent to the DCE client.
19. The DCE login process decrypts the envelope. The encrypted PTGT and EPAC are stored. The stored PTGT and EPAC will be used by DCE client whenever another DCE server is to be used.

Due to protocol and RPC overhead, there will be more traffic on the network for a login than Figure 3 on page 17 would lead one to believe.

The following table summarizes the results obtained from many sample measurements done in order to establish an estimate of the total traffic during a `dce_login` from an AIX DCE client:

Table 3. Example of DCE Client-Generated Network Traffic during DCE Login

Example Measurement	Average Number of Packets	Average Packet Size (Bytes)	Maximum Traffic (Bytes)
DCE Client Login	65	278	11862

Details about the test scenarios can be found in E.4.2, “User Authentication (dce_login)” on page 251.

Analysis, Comments and Additional Information about the Results: The impact of the network traffic on DCE login performance during peak hours can be considerable. Some additional information and observations:

- The variations in the amount of network traffic among different logins are remarkable, as can be seen in the detailed data in E.4.2, “User Authentication (dce_login)” on page 251. A single login attempt may cause as little as 16, or up to more than 80, packages of network traffic. Also, the protocol sequence chosen changes frequently among different logins.

  This is due to the fact that the DCE login process selects different servers and protocol sequences in a random manner. In order to do this, it queries CDS about all available Security Servers and the protocol sequences they

support and then selects one combination from that list. Due to CDS caching on the client side, the CDS Server is not always involved in this process.

This is the designed way for DCE login to work. The advantage of it is the flexibility; any new Security Server or any other change in the cell configuration will not require an administrator to change any static profiles.

- If network traffic or server load is a concern, there are methods to reduce this traffic by specifying the use of a default Security Server by means of the pe_site file. See 7.2.1, “Security Service, the pe_site File” on page 142, for further explanations on the use of this file.
- As can be seen in the detailed tables in E.4.2, “User Authentication (dce_login)” on page 251, there is more traffic between the client system and the CDS Server than between the client and the Security Server. This is due to the fact that CDS is being queried for binding information to the Security Server(s).
- When the client service chooses TCP as the communication protocol to the Security Server (which is designed to happen randomly), more packets and more overall traffic is involved, as compared to using the User Datagram Protocol (UDP).
- User login from OS/2 clients creates much less network traffic than from AIX clients because it, by default, makes use of the pe_site file (see also 7.2.1, “Security Service, the pe_site File” on page 142). Under normal circumstances, CDS never gets queried during a DCE login from an OS/2 client.
- A reauthentication using the `kinit` command causes about the same amount of network traffic as the login (`dce_login`) does.
- Users belonging to many groups will cause larger EPACs to be transmitted over the network when they log in to DCE, but this additional traffic is usually not much, unless a user belongs to hundreds or even thousands of groups (which should anyway be avoided for other reasons). Tests done with a user who is a member of 100 groups have shown that the number of network packets between the client and the Security Server was increased by about 25 percent and the total traffic (number of bytes) by about 40 percent.
- A login attempt with a wrong password causes about half of the network load of a successful login.

2.3.3 AIX Integrated Login

AIX provides the integrated login facility. With this, users are not required to be administered on every system they are permitted to log on to. Instead, they need only be registered as DCE users in the DCE security registry. On systems configured for integrated login, the `dceunixd` daemon takes care of the DCE login for those users who are to be DCE authenticated (integrated login still allows certain users to be configured and managed locally only).

For a description on how to set up AIX integrated login, please refer to the *DCE for AIX Administration Guide* (available as online document with the DCE for AIX product) or to the ITSO redbook *Administering IBM DCE and DFS Version 2.1 for AIX (and OS/2 Clients),* SG24-4714-00.

As a rule of thumb, the amount of network traffic generated by the AIX integrated login is somewhat higher than a native `dce_login`. We do not provide figures here because they heavily depend on two factors:

- Local caching—If a user has only recently logged in to AIX, a subsequent login may cause less network traffic.
- Group membership—In order to conform to POSIX standards, AIX requires at login time a list of all users belonging to the user's primary group. Several commands in AIX, such as the `id` command (to see its own login ID) also require this list. If a user belongs to a group which has many other members, this may create high volumes of network traffic, not only during the login.

The latter factor must be carefully dealt with. Remember, each DCE user account belongs to at least one group, which is the group `none` by default (`none` is actually a valid group and does not mean, as its name may suggest, no group). Suppose we have a small cell with only one hundred user accounts, but all belonging to group `none`. In such a case, measurements have shown that a single login through AIX integrated login, or by the execution of an AIX command like `id`, causes about 80 KB to be transferred between the client and the DCE Security Server in about 120 packets. This may unnecessarily slow down login time if the client is connected to the server through a slow link.

2.3.4 DTS Time Synchronization

DCE services require the system clocks of all the machines in the cell to be synchronized within certain tolerances, and the DTS was designed for this purpose. However, the use of DTS is not mandatory, any other time synchronizing service can be used instead as long as it does not adjust the system clocks backwards.

The components of the DCE Distributed Time Services are divided into DTS Clerks (clients) and different DTS Server roles. We will not investigate on the different server roles, since they all only create a small amount of network traffic and CPU load. Instead, we only consider clerks and servers.

DTS time synchronization takes place, by default, once a day (it used to happen more often in earlier versions). Server and network load is at a negligible value. Only in the very unlikely event of having a large number of DTS clients requesting service from the same server and at the same time could impose some remarkable network I/O and CPU load on this server. However, synchronization does not happen at precise 24-hour intervals (although the `syncinterval` value is by default set to 1 day = 24 hours). DTS Servers and Clients select a random interval close to the time specified by the `syncinterval` configuration parameter. Actually, DTS synchronization takes place every 20 to 23 hours. By this, network traffic and server load get statistically distributed over the time.

Network traffic introduced between DTS Clients and DTS Servers is low. It is about 20-30 packets with a total of about 5 KB between a client and each DTS Server it uses for a synchronization. Synchronization among Local DTS Servers causes about the same traffic.

Using DTS Time Services, the servers perform binding export into the CDS namespace. This exported binding information is used by other DTS Servers and by the DTS Clerks to locate DTS Servers from which time information can be obtained for time synchronization.

Tuning in respect to DTS is not necessary in most cases. If, however, any network traffic or CPU load is a concern, the following can be considered for performance-related issues:

- DTS Clerks will perform time synchronization requests based on the imported time server binding information. This binding information should be configured to reduce unnecessary traffic on slow lines.
- The DTS traffic imposed to the CDS results in binding import for the DTS Servers specified in the LAN profile. The number of servers the DTS Clerk has to contact is controlled by the `minservers` attribute value.
- If slow network links are involved, DTS Servers should be spread according to the general guidelines for DTS Server location planning. As a general rule, DTS clients should have access to three DTS Server on their local network.

Note on DTS Configuration

If you decide to change the DTS default configuration, bear in mind that `dtsd` does not save its configuration for subsequent starts. Therefore, changes of any DTS parameters, such as the `syncinterval` for example, need to be done each time after `dtsd` has been started.

2.3.5 DCE Application Server Registration

Depending on the authentication requirements, a DCE application server, sometimes also called an RPC server, will perform authentication very similar to the DCE login process. This type of authentication is normally automated by using a keytab file, which contains the principal name and the key (password) required for the authentication.

To export binding information into the CDS namespace, additional considerations apply. Using the authentication and authorization capabilities, the DCE application server must have appropriate access rights to the CDS namespace to be able to add the binding information. These access rights can be set through the ACL by using a principal or by using the login context of an appropriate account.

To gain authorization to the CDS namespace, the application server uses the PTGT, which includes the EPAC (containing user and group membership information). The CDS Server uses this information to perform ACL checking.

With the correct authorization, the DCE server can export its binding information into the CDS namespace. The number of bytes transferred through the network between the DCE application server and the CDS Server depends on the binding information characteristics.

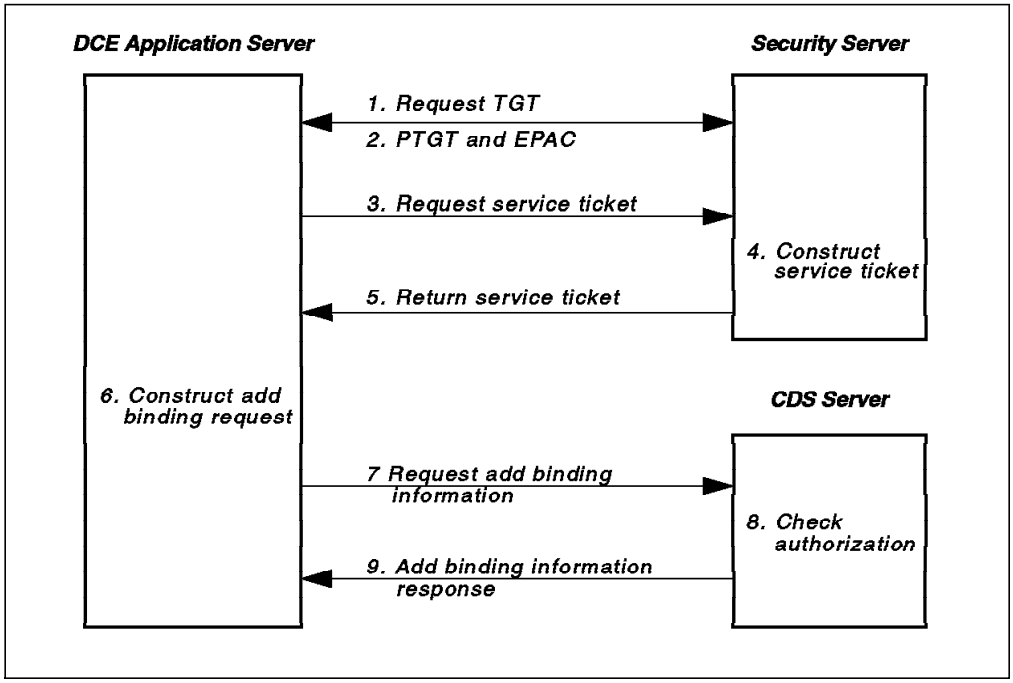


Figure 4. DCE RPC Server Registration into CDS Namespace

The following events occur while the DCE application server adds registration information into the CDS namespace:

1. The DCE application server initiates the authentication by which a TGT is requested.
2. As a result, the application server obtains a PTGT and the EPAC. The EPAC will be used for authorization at a later time.
3. The DCE application server requests an operation to the CDS by creating a service request ticket that is sent to the Security Server.
4. The Security Server constructs the service ticket that includes the EPAC chain information.
5. The service ticket is returned to the application server.
6. The application server constructs the add binding information request and adds the service ticket into the request.
7. The request is sent to the CDS Server.
8. The CDS ACL manager performs authorization checks based on the EPAC chain information available in the request package. If the request is granted, the requested operation is performed, and a response is created.
9. The response is sent back to the DCE application server.

The DCE Server registration into the CDS namespace adds some traffic into the network. In most cases, however, DCE RPC application servers perform the registration only once and stay in operation for a long period of time. Thus, this network traffic usually can be neglected.

The following table shows an example of the amount of traffic generated during DCE RPC server registration. The values may vary depending on the amount of information to be registered in the CDS namespace. The example measurement

traced a DCE application server. All generated traffic from and to the client was considered as part of the DCE application server registration:

Table 4. Sample DCE Application Server Registration Network Traffic

Example Measurement	Average Number of Packets	Average Packet Size (Bytes)	Maximum Traffic (Bytes)
DCE Application Server Registration	66	264	17439

Analysis, Comments and Additional Information About the Results: The following should be considered for performance:

- The binding information stored in the CDS namespace is available as long as the DCE application server is running or until it has been removed from the namespace. DCE clients cache the binding information obtained and assign it an expiration time. When DCE client applications request this information for the first time, the DCE runtime randomly specifies a value between 8 to 12 hours as the default expiration time. When the cached information has expired, it will automatically be updated. To control the automatic cache update traffic for long-running DCE applications, the client application can issue a call to the `rpc_ns_mgmt_handle_set_exp_age()` and set an explicit expiration time.
- DCE application server registration is not taking place very often in most environments. Thus, in general, this kind of network traffic does not require special attention.
- Since this is a write operation to the CDS namespace, any involved CDS replicas will be updated immediately. Depending on the number of these replicas, network load and CPU load on the master CDS Server will increase.

Adding server registration information into the CDS namespace by itself does not normally cause performance problems, except for cases where the CDS Server is a performance bottleneck.

2.3.6 DCE Application Client RPC Call

The authorization process that a DCE application client has to make is divided into two separate steps. The first step is to get a service ticket from the Security Server, and the second step is to gain the necessary authorization from the application server.

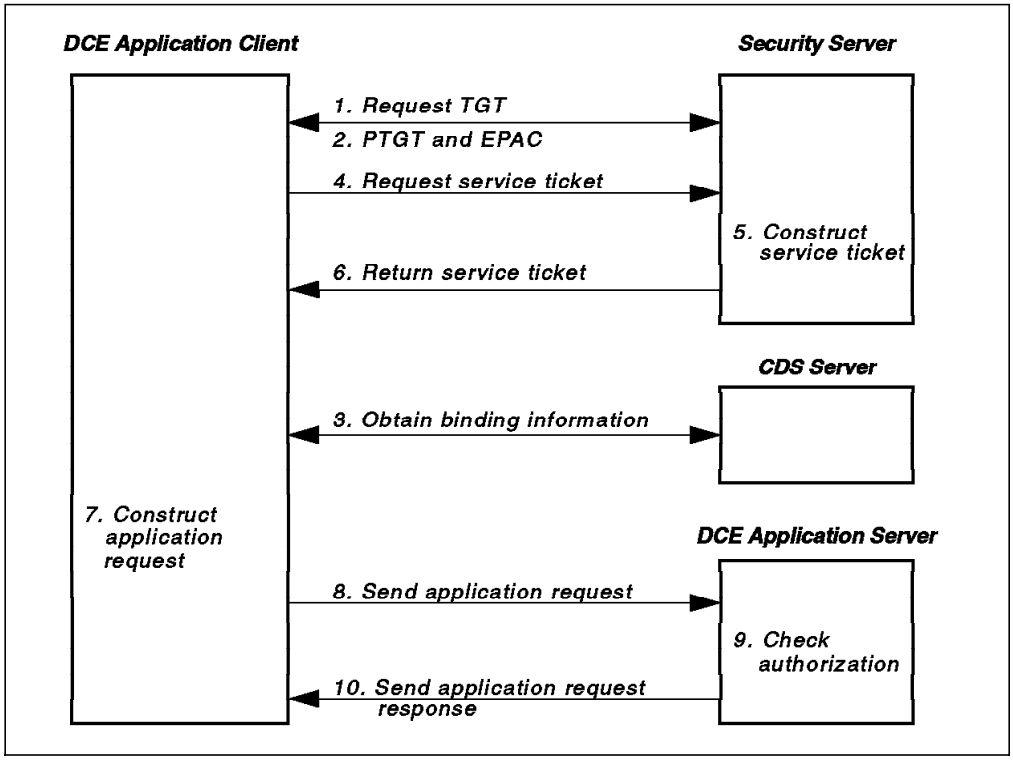


Figure 5. DCE RPC Application Client Authentication Process

The following events will occur for a DCE application client performing a remote procedure call to a DCE application server:

1. The DCE application client initiates the authentication and by which a TGT is obtained from the Security Server.
2. The authentication process continues, and as a result the DCE application client obtains a PTGT and EPAC information. The EPAC will be used for authorization at a later time.
3. The DCE application client performs a binding import from the CDS. The binding information allows the client to locate the DCE Security Server.
4. The DCE application client requests an operation to the DCE application server by creating a service request ticket that is sent to the DCE application server.
5. The Security Server constructs a service ticket that includes the client EPAC chain information.
6. The service ticket is returned to the DCE application client.
7. The DCE application client constructs the remote procedure call request and adds the service ticket into the request.
8. The request is sent to the DCE application server.
9. The DCE application server performs authorization checking based on the EPAC chain information available in the request package. If the request is granted, the requested operation is performed (RPC procedure is called), and a response is created. The DCE RPC application program can choose not to use authorization in the `rpc_binding_set_auth_info()` function call. With the authentication service set to `rpc_c_authn_none`, no authentication will be performed.

10. The response is sent back to the DCE application client.

The DCE runtime library caches data structures from RPC activity and authentication activity in order to improve performance on repeated RPC calls to the same application server. This information is generally kept for up to 10 minutes after the last call before it is released.

In a simple test case, we wanted to know what overhead DCE RPC introduces compared to the amount of raw data sent between an application client and its server. Table 5 is the result of a very simple DCE application transferring a variable number of bytes from the application client to the application server. The amount of traffic generated on the network was measured. Authentication was not used in this example. The test program used for these measurements is listed in Appendix B, "Code Listings of the fst Test Program" on page 231.

The binding information is imported before the Remote Procedure Calls are performed and traced. The protocol selected was `ncacn_ip_tcp` in the example case. The resulting traffic to and from the DCE client is shown in the following table:

Table 5. DCE RPC Protocol Overhead for Simple Data Transfer

Example Measurement	Number of Packets	Average Packet Size (Bytes)	Network Traffic (Bytes)
100 Bytes	21	107	2255
1000 Bytes	26	135	3526
10000 Bytes	29	440	12771
100000 Bytes	109	1001	109097

There is some overhead when using DCE for Remote Procedure Calls (as with any other RPC implementation), which decreases with larger amounts of transferred data.

Analysis, Comments and Additional Information About the Results: The following should be considered for performance:

- The first call to the remote procedure costs some overhead. But repeated calls reduce the overhead considerably.
- The overall overhead is low and compares well with other types of non-DCE RPC.
- Using authenticated versions of the RPC calls generates more traffic, depending on the level of authentication. One reason for the additional traffic on the network is the service ticket containing the EPAC information, which is transferred in addition to application data.
- Additional traffic (and CPU load) will be introduced when using additional package-level protection. The amount and size of the packets transferred will increase.

This is only a single example to show that RPC overhead is not causing much additional network traffic. Additional RPC network characteristics can be found in Chapter 8, "Application Development With Performance in Mind" on page 155.

2.3.7 Security and CDS Server Under Load

In order to characterize the behavior of a Security Server and a CDS Server under heavy load and to get some raw input for system sizing, such servers on different hardware were exposed to heavy load in a test environment. Identical software configuration was used as far as operating system level, DCE code level, registry database, and CDS database contents were concerned.

2.3.7.1 Security Server Under Load

Security Server performance is important mostly for the task of user and application server authentication, since this is its main responsibility. Performance of administration tasks, like adding new accounts or changing passwords, are considered less important in most environments because this kind of work is either done by batch jobs (maybe even over night), or they happen rarely compared to authentication tasks.

The following graph shows the summarized results from the tests. In the test scenario, a number of systems created DCE login requests against a Security Server, more than what the server was able to process. As a result, the CPU utilization of the server was constantly at 100 percent.

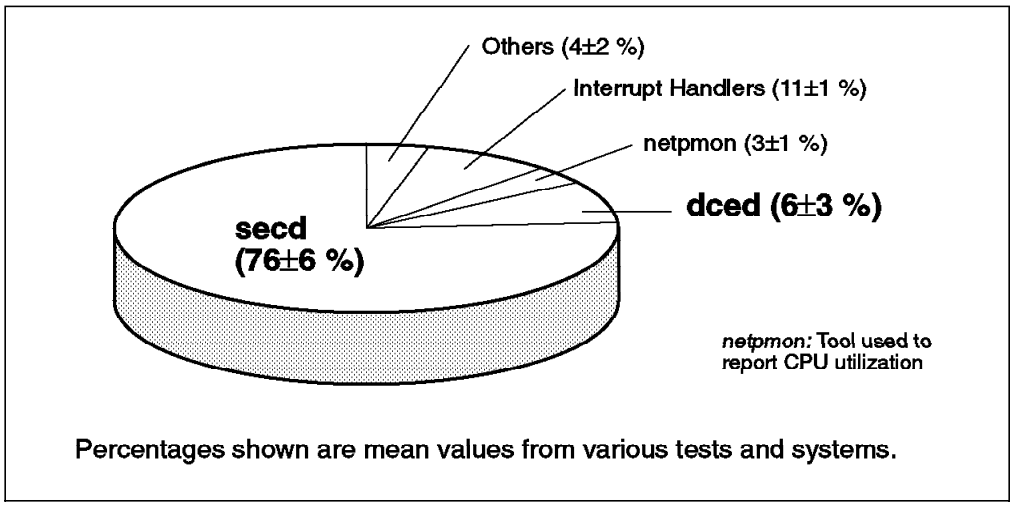


Figure 6. Security Server CPU Utilization Under Heavy Load (AIX)

Details on the tests, including logon performance figures of Security Servers on different hardware, can be found in Appendix E.4.6, “Security Server Under Heavy Load” on page 254.

Although the test scenario was theoretically capable of issuing more than 60 logon requests to the server at the same time, this might not represent a real environment, where hundreds or even thousands of users log on every morning in a certain time frame. For this specific load, the tests may not be suitable to predict server behavior.

The tests did not show much performance improvement when using the only tuning option of secd, the number of listener-threads. It can be assumed, however, that this parameter has more impact in an environment where a large number of requests are issued against a Security Server in a short period of time. The test scenario may have not represented a real environment in this respect.

The tests have also shown that the network I/O was relatively low and should not cause any bottleneck unless other applications or services share the same interface. Disk I/O may require some attention on high-performance servers with relatively slow disk I/O subsystems since every login attempt causes about 10 KB to be written to a disk. In the test scenario (as described in E.4.6, "Security Server Under Heavy Load" on page 254), disk I/O was already high enough to limit server performance. Assuming a server is capable of processing 20 login requests per second, disk I/O would be at about 200 Kb/s, which is not yet critical, but may become an issue to look at more closely.

2.3.7.2 CDS Server Under Load

In contrast to a Security Server, for which a typical test scenario can represent a practical environment to a large extent, it is not possible to define a typical workload for, or even derive performance figures from a CDS Server. While a CDS Server itself may certainly be able to perform a certain number of requests, the variety of different request types and the fact that clients do CDS caching would make such figures almost worthless because we would not know how to apply them to a different practical environment.

For the purpose of characterizing the type of application a CDS Server represents in terms of performance issues, we exposed a CDS Server to a high load created by a number of test scripts running on different machines that did mostly directory and object lists and some create and delete operations. The CPU usage, as shown in the following picture, looks about the same as when the Security Server was tested.

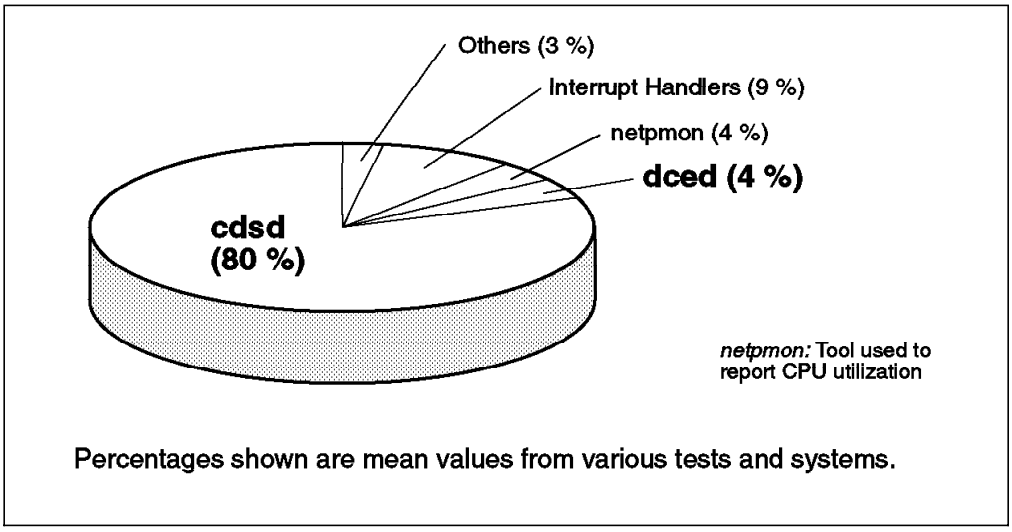


Figure 7. CDS Server CPU Utilization Under Heavy Load (AIX)

A CDS Server, for example the cdsd process, is a CPU-bound process whose performance can almost only be improved with additional CPU performance. Disk I/O might become an issue when updates are made very often. Network I/O is basically dependent on the number of clients sending requests to the CDS Server, but generally is not a concern on a CDS Server.

2.3.8 Security and CDS Server Replication

In this series of tests, we want to quantify some basic replication events that happen during Security Server and clearinghouse replication.

2.3.8.1 Security Server Replication

Updates to the security registry are always directed to the Master Security Server, also called the master registry. Changes are then propagated from the master registry to all Security Server replicas.

Updates to Security Server replicas are performed by update propagation, which is an immediate attempt to apply one change in the master replica to all client (or slave) replicas.

When a master or slave replica receives updates, it applies the updates to its database in memory. A copy of each update is written to a log file together with a sequence number. The servers periodically write their databases in memory to disk. The master replica clears the log file of all updates that have been propagated to slave replicas. Slave replicas clear the local log file after writing changes to disk.

When a master or slave replica restarts, it initializes the database in memory and applies any outstanding updates from the log file. The Master Security Server recreates the propagation queue from the log file. Any outstanding slave updates will then be propagated, if possible.

The Master Security Server and the slave replicas perform authentication.

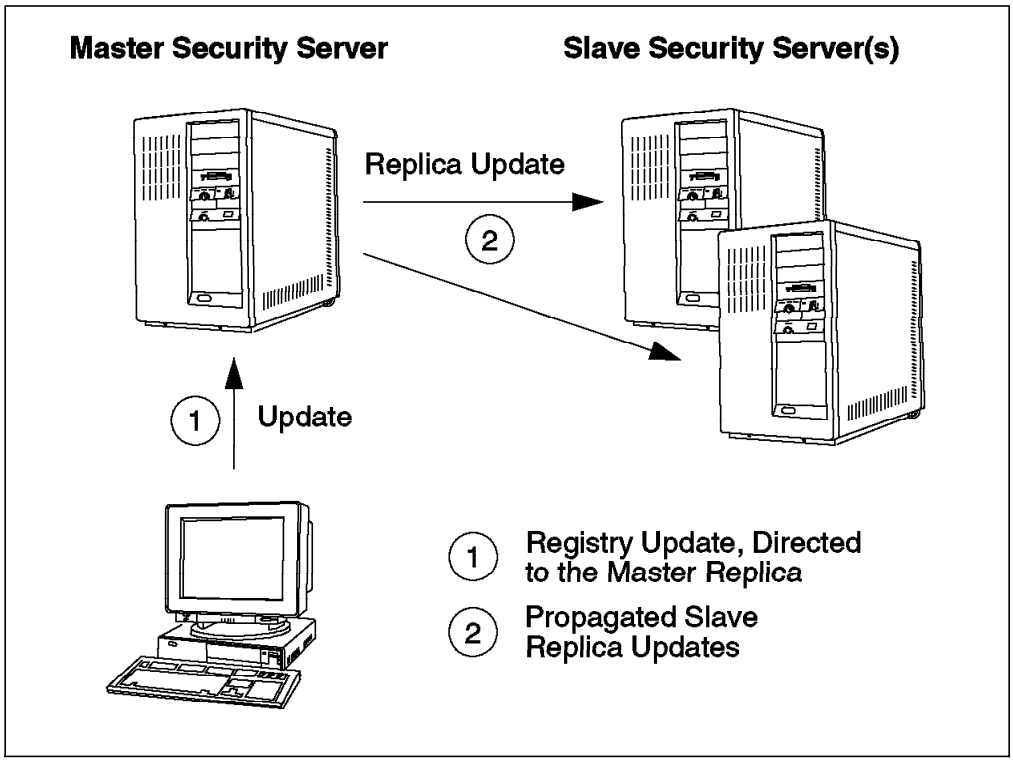


Figure 8. Security Server Replication Propagation

A test was done to observe the traffic involved by a single Security Server replication action. The test case was repeated with a few principal, group and account definitions.

Before the test, the replica were synchronized, for example there were no outstanding updates pending (this can be checked with the dcecp registry dump command, observing the value for updseqqueue). During registry replication, network traffic between the Master Security Server and the slave was traced. The following commands were used to add an account:

```
dcecp> principal create Stst
dcecp> group add none -member Stst
dcecp> organization add none -member Stst
dcecp> account create Stst -group none -org none -mypwd <...> -password <...>
```

The master and slave registry synchronization generated the following traffic on the network

Table 6. Security Server Replication for Account Create and Password Change

Example Measurement	Number of Packets	Average Packet Size (Bytes)	Network Traffic (Bytes)
Adding a New Account	30	205	6135
Changing a User's Password	10	223	2231

The second row lists the measured values for a DCE password change, done by using the appropriate SMIT (Systems Management Interface Tool) on an AIX client, which will also be required to be forwarded to any replica servers.

There was no endpoint activity during the trace because the replica Security Server was restarted just before the trace was initiated.

Analysis, Comments and Additional Information About the Results: The following should be considered for performance- and availability-related issues:

- The Master Security Server must be highly available because it is the only place additions and modifications of registry objects can be performed. In a DCE cell, there is one and only one Master Security Server.
- Replication traffic is not much, but it can sum up if there are many replica servers to be kept up to date by the master.
- If a replica server is not available at the time of a required update, the update process will automatically be retried until successful completion.
- If a replica server is removed permanently (or for a long period of time), it should be deleted from the Master Security Server's replica list to prevent unnecessary update attempts.

  This can either be done by a clean unconfiguration process or by a forced remove with the command dcecp registry delete <replica name> -force.
- The Master Security Server or a slave must be available during DCE login.

At least one Security Server slave replica should be installed for backup and availability reasons.

2.3.8.2 Clearinghouse Replication

Any addition, modification or deletion in the CDS namespace is performed to the CDS Server (or clearinghouse) housing the master replica of the directory being modified. Updates to CDS replica servers are performed either by:

1. An update propagation, which is an immediate attempt to apply one change in the master replica to all replicas of the CDS directory in which the change occurred. Unlike a skulk operation, update propagation does not guarantee that the change is applied to all replicas.
2. A skulk operation, which is a periodic distribution of a collection of updates. The main function of the skulk operation is to ensure that replicas receive changes that might not have reached the replica during an update propagation and to remove outdated information from the CDS namespace. The skulk operation can begin in one of three ways:
   - An administrator can immediately skulk a CDS directory with the command `dcecp> directory synchronize <directory name>`.
   - CDS starts a skulk as a result of namespace management activities that changes the structure of the namespace. Management activities include:
     - Adding or removing a replica
     - Creating or deleting a replica
     - Redesignating replica type
     - Adding or deleting a child cell name in a parent cell
   - The CDS Server initiates the skulk automatically. The automatic skulk is repeated at an interval which is the Background Skulk Time. A CDS Server periodically checks each master replica in its local clearinghouse and initiates a skulk if changes were made since the last successful skulk of that directory

The amount of network traffic generated by the update propagation and skulk operation depends on the CDS namespace changes. To perform an update propagation, the involved CDS Servers must perform authentication.

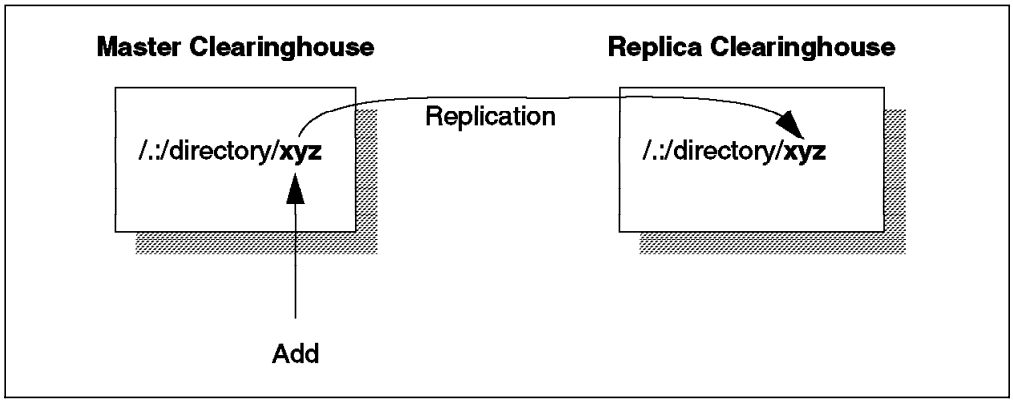


Figure 9. CDS Clearinghouses Replication

The amount of network traffic involved in a CDS replication process very much depends on a series of factors: how much source information needs to be replicated, is it a directory or a leaf object, does it contain many attributes, and so on. In order to have a rough figure of what a simple replication might involve, a test was done while the network traffic was traced between the master and the replica clearinghouse.

The simple test involved the creation of a directory in the CDS namespace within an already-replicated directory. The existing directory was defined with a *high* convergence, such that the replication process took place immediately after creation of the new directory. The command executed to add the secondary directory was:

```
dcecp> directory create /.:/rep_dir/trace_me_test
```

Table 7. Example Clearinghouse Directory Replication Network Traffic

Example Measurement	Number of Packets	Average Packet Size (Bytes)	Network Traffic (Bytes)
Create Directory Replication	8	286	2287

Table 7 shows the traffic between the two DCE CDS Servers.

Analysis, Comments and Additional Information about the Results: The following should be considered for performance-related issues:

- A CDS Server, like a Security Server, should be highly available because DCE server and DCE client applications, including DCE core services, rely on the CDS.
- For each directory entry in the CDS namespace, a master and replica should be considered.
- Each CDS Server maintains at least one clearinghouse. All CDS Servers contain a copy of the CDS namespace root directory (/.:/) and a copy of other replicated directories (by administrator action).
- Replication decisions should be based on where the contents of the CDS directory are being used. As a rule of thumb, place CDS replicas in close proximity to the consumers and CDS masters close to the producers.
- Actual performance monitoring, testing and tuning should be done in a real environment where CDS Servers are to be used since it is almost impossible to predict the actual load from theory.

2.4 The Performance Impact of the Components

After we have done some analysis of the network traffic between DCE services in different computer systems, we will look more closely at the various components involved in such transactions and how they can contribute positively to an overall smooth performance.

Unlike typical local desktop applications, DCE applications and DCE core services require network bandwidth and computing resources in the clients and servers, which introduces more likely spots for possible bottlenecks when scaling DCE into larger environments. The number of remote requests a DCE server, including the DCE core servers, can handle depends on factors like:

- The DCE server resources available
- The network used in terms of the physical network and the local hardware and software components within the DCE servers and DCE clients

- The requirements of each DCE Remote Procedure Call, like the number of DCE logins performed or the application-generated DCE Remote Procedure Call requirements
- The performance requirements at the user level
- The number of calls that can be processed within acceptable response time limits

To accomplish the required level of service, there are in some cases additional configuration options that can be adjusted. The following sections describe the parameters that in some cases can be adjusted or may have an effect on the resulting performance.

In the following section, we discuss the various components introduced in Figure 1 on page 8 and their contribution to the overall performance.

2.4.1 AIX and TCP/IP

No matter what type of platform DCE services are to be implemented on, it is necessary from a performance point of view to have a certain understanding of its underlying components. This section concentrates on the AIX platform. The factors on a platform that have an influence on the performance are:

- The CPU performance

  It is a simple rule of thumb that better CPU performance yields better overall performance results. Most DCE services are mainly CPU bound and gain much with better CPU performance.

- The amount of memory

  Having enough memory is probably the most important performance issue for DCE servers since they rely on having their databases in memory. If these memory pages get swapped out, server performance suffers heavily.

- The disk subsystem performance

  Most DCE services, except DFS File Servers, do not use intensive disk input/output. Thus, disk performance is generally less important for DCE. It can, however, become an issue when a large number of changes are to be done by DCE servers because they write every update transaction to log files on the disk.

- The network throughput

  The network throughput may become a component to look at on heavily loaded servers that may be swamped by many requests in short periods of time.

- The ″others″

  A great threat for any server are additional applications, management tools or audit applications that require a lot of system resources. For example, a powerful network backup tool may add critical load on a network interface.

Using the TCP/IP interfaces available, an application program makes a selection of the socket interface to use. The application uses this interface to perform its applications-related transfers. AIX on the other hand must be able to manage the requests and transfer them to the intended partner of the application.

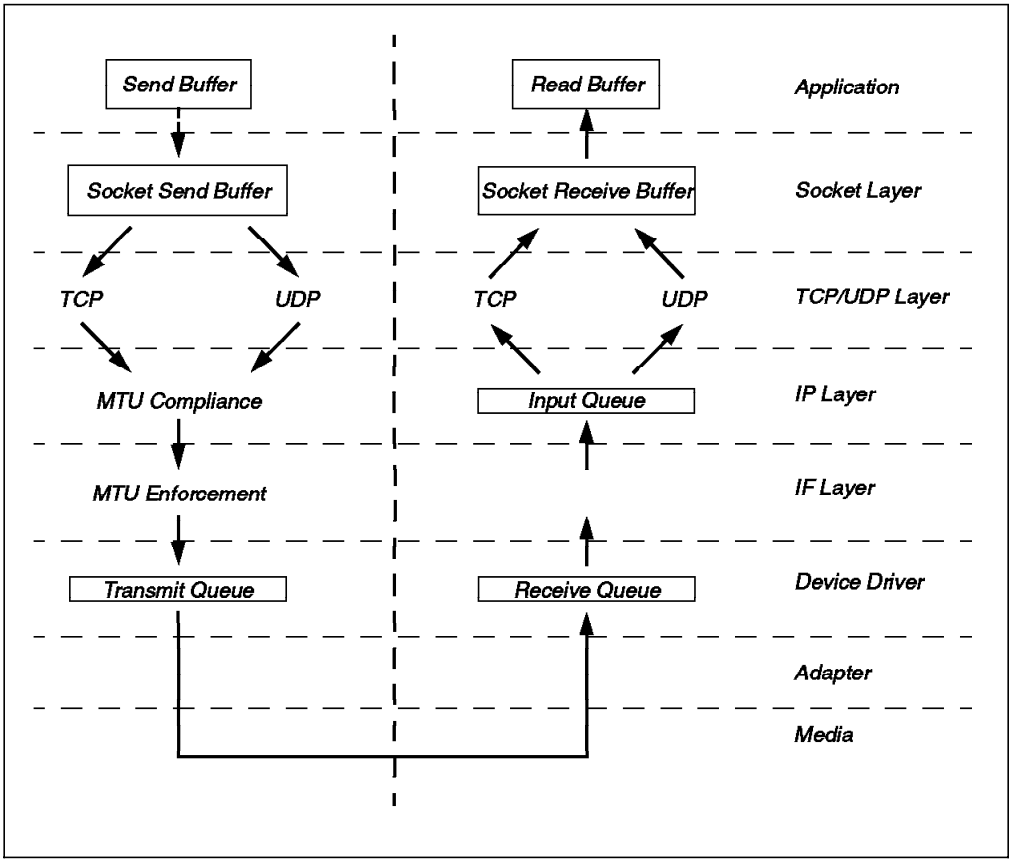


Figure 10. AIX UDP/TCP/IP Data Flow

The simplified application request flow is as follows:

- The application performs a write with a pointer to the transfer buffer, which causes the interface to copy date into a socket send buffer.
- The socket layer transfers the data to either TCP or UDP as specified in the initiation of the interface.
- The packet is packaged into one or more Maximum Transfer Units (MTU) in the IP and IF layers.
- The packets are put into the device driver output queue and transferred through the physical media to the receiving system.
- The packets are placed in the receiving system's receive queue and passed up to the IP layer.
- Fragmented packages are reassembled and placed in the socket receive buffer.
- The application's read request transfers data into the application's read buffer

Client-side tuning for DCE core services on AIX is limited to fulfill the additional memory requirement of the DCE components. There is no absolute lower limit for the memory size in client systems for DCE core services, but you may consider a memory upgrade in cases where low memory is already a concern. Compared to most applications, DCE only adds little additional use of memory. As a rule of thumb, 48 MB of memory or more should be installed for DCE core clients plus some (small) applications. There is no special requirement for CPU, disk and network performance since DCE is not a resource-intensive service.

Obviously, better CPU and network performance increase overall performance for DCE as well as for any other application, but DCE core services do not have special requirements.

Server-side tuning for DCE core services mostly concentrates on the memory requirements and network interface tuning. CPU performance is, as in any case, desirable but usually not the most important issue. The most important fact for DCE Security and CDS Servers is to fulfill the memory requirements since these services are designed to keep their databases in memory all the time. Performance will decrease remarkably when paging (swapping) gets involved due to little memory. For memory planning considerations, see 5.4.1.2, “Security Server Disk and Memory Sizing” on page 106. Network interface tuning will become necessary for moderate or large environments with hundreds and thousands of users or client systems, but will usually not be required for small environments. For a description on how to configure your network interfaces, see 6.1.2, “AIX and TCP/IP” on page 124.

2.4.2 OS/2 Warp MPTS, TCP/IP and NetBIOS

Similar to AIX, DCE core services generally do not need any special tuning on client systems since DCE core services do not significantly increase either CPU usage or network utilization, except for starting up services and for user login. Memory is usually the only system resource that might be considered to be upgraded or sufficiently installed.

Additional application load may, however, require powerful processors and a large amount of memory and high-performance network interfaces. DCE core services will certainly gain performance in such systems as well. If network throughput is a concern due to heavy application load, see 6.1.3, “OS/2 Warp, TCP/IP and NetBIOS” on page 126.

2.4.3 DCE Runtime Services

The DCE runtime services are required on every system that runs DCE, or for every DCE application, respectively. They are responsible for handling requests on behalf of the applications and for forwarding them to the DCE servers on the network. This may include as little as simply forwarding a request or as much as marshalling and encrypting data and automatically finding an appropriate server for the request.

DCE runtime services do not have configurable parameters, but there are a few environment variables that can be used to control certain options. In terms of performance tuning and problem prevention, the important ones are:

- `RPC_UNSUPPORTED_NETIFS` and `RPC_UNSUPPORTED_NETADDRS`

  Prevents DCE from using specific network interfaces when running on a multihomed host (for example, a system with more than one network interface). Example: `RPC_UNSUPPORTET_NETIFS=sl0:en1:et1`.

  `RPC_UNSUPPORTED_NETADDRS` works basically the same way but on an address level rather than on an interface level. Example: `RPC_UNSUPPORTED_NETADDRS=192.136.33.2`

  See also 6.1.2.2, “Disabling Unused Network Interfaces for DCE” on page 125.

- `TRY_PE_SITE`, and `BIND_PE_SITE`

If TRY_PE_SITE is set to 1 (for example, TRY_PE_SITE=1), security services will take binding information for the Security Server(s) from the pe_site file rather than consulting CDS services. The entries in the pe_site file are tried sequentially until success. If non of the entries point to a valid server, CDS is used in the normal way.

BIND_PE_SITE does the same as TRY_PE_SITE, but will cause the security client to not consult CDS if all entries have been tried unsuccessfully.

See also 7.2.1, “Security Service, the pe_site File” on page 142.

2.4.4 The Network

As we have seen in the first part of this chapter, many DCE core services generate a number of relatively small packages between clients and servers using both TCP and UDP protocols. As a consequence, the network should be tuned to quickly forward such small frames from end to end in both directions. For example, it should have a small latency time. As an example, if the network adds only 20 milliseconds to each packet to be transmitted from one end to the other, the user can easily experience more than one second delay when logging on to DCE. A satellite link, as such an example, could slow down DCE core services in a noticeable way, although it might have a high nominal transmission capacity.

Network and network interface tuning for DCE core services is not necessary on client systems because this amount of network traffic does by far not introduce any difficulties for AIX. As a rule of thumb, a LAN network interface on a mid-range RS/6000 with AIX is easily capable of handling 500 network packets per second without any special tuning. With tuning, 1500 packets and more are achievable.

Since most traffic concentrates on the server side of the network, it can be necessary to apply some performance optimization rules to this side. These can be:

- Put DCE Servers on small subnets.
- For performance and availability, put replicated DCE servers on different subnets.
- Remove network-intensive services from the same subnet.
- Optimize router throughput for DCE-type of traffic.

2.4.5 DCE Core Servers

DCE core servers basically only run a single daemon in addition to the DCE runtime services daemons: secd on a Security Server and cdsd on a CDS Server. There is little that can be configured for the performance of these daemons; it is more important to provide an optimized environment for these processes to run in. The following two subsections discuss some of the important aspects.

Some general rules apply for DCE core servers (as for any other servers) to perform optimally:

- Replicated services should run on machines with equal performance, preferably with the same hardware configurations.
- Care should be taken when running additional tools and applications on the same server as these could compete with the DCE core servers for the available resources. CPU-intensive performance monitoring and analysis

tools are good candidates for closer investigations. Also, most fancy-looking screen savers are very CPU-intensive.

- Any system backup activities should be avoided during peak-hours.
- Any batch jobs should be avoided during peak-hours.
- If possible, server systems should be dedicated for their purpose. It is never a good idea to use DCE servers as multiuser systems or as other types of application servers.
- The network segment to which a server is connected should not carry a high base load.

2.4.5.1 Security Server

A Security Server runs the secd daemon. This single, multithreaded process maintains the security registry database and is responsible for all activities of the Security Server. There are no additional processes started and controlled by secd.

The Security Server process, secd, is CPU-bound and requires a certain amount of physical memory to store the registry database. The secd process is designed to keep this database in memory all the time, and its performance will be degraded as soon as parts of this database get paged out to the paging space due to lack of memory. For estimating memory requirements, see 5.4.1.2, “Security Server Disk and Memory Sizing” on page 106.

As long as enough memory is installed to keep the registry, the Security Server performance almost only depends on the CPU performance. Since most of the requests sent to a Security Server are authentication requests—representing read-only accesses to the registry database—only little disk I/O is involved in normal operations. Only when updates, such as adding or changing user accounts, happen very often or during peak hours, does disk subsystem performance become important. Using RAID subsystems for increased availability is not critical for Security Servers even though they might have lower I/O rates as opposed to single disks. To keep disk I/O low, update operations should be avoided during peak hours. As a matter of fact, however, registry updates due to user password changes happen most often during login peak hours. Tests that have been done to measure some performance statistics are described in 2.3.7.1, “Security Server Under Load” on page 26, or in more detail in Appendix E.4.6, “Security Server Under Heavy Load” on page 254.

By default, the Security Server secd process creates a number of listener-threads for processing incoming requests. For environments where a high number of requests to a Security Server in a short period of time can be expected, secd in the AIX implementation has a startup option (the `-t <count>` parameter) that controls the number of these listener-threads between the default of 5 and up to the maximum of 15.

Security Server performance may be degraded during checkpointing. Checkpointing is the task of writing changes of the registry database back to disk (but a change log file is maintained and updated instantaneously with changes). Under normal conditions, secd will checkpoint in regular intervals, and this will take only some few seconds. If checkpointing during peak hours is a concern, it can be forced by stopping and restarting secd or by the following command sequence:

```
dcecp> registry disable
dcecp> registry enable
```

This forces secd to checkpoint the registry database. The `enable` subcommand should follow the `disable` subcommand immediately, because the Security Server will not accept any updates when disabled. This procedure can best be applied shortly before peak hours start.

The network interface on a Security Server is not a bottleneck that requires any tuning in most cases. Only if the number of requests is higher than what secd can process, will the interface buffers possibly fill up. The network interface tuning parameter `thewall` may then be required to be set to higher values on AIX 4.X. See 6.1.2, “AIX and TCP/IP” on page 124, or 6.1.3, “OS/2 Warp, TCP/IP and NetBIOS” on page 126, for further details on TCP/IP tuning.

For considerations on initial Security Server installation and configuration, see 6.3.1, “AIX Security Servers” on page 132.

For performance considerations when replication is used, see section 2.5.3, “Server Replication Policies” on page 42.

2.4.5.2 Cell Directory Server

The Cell Directory Server works very similar to the Security Server, and basically most of the facts mentioned in the previous section, as far as performance issues are concerned, apply to CDS Servers as well. The CDS Server daemon, cdsd, is designed to hold its database in memory for fast operation. If sufficient memory is installed, CDS Server performance mostly depends on the CPU performance. Disk I/O performance may become more important during peak hours due to write operations to the CDS database from starting DCE servers.

Checkpointing on a CDS Server can be forced with the command:

```
dcecp> clearinghouse initiate <clearinghouse name> -checkpoint
```

Network tuning might become necessary when a large number of clients request services from a CDS Server at the same time. In such situations, cdsd might not be able to process all requests fast enough, and network buffers may therefore fill up. CDS uses both UDP and TCP as default protocols. Due to the different characteristics of the two protocols, it might be advantageous in certain environments to favor one over the other. UDP, for example, requires less overhead from cdsd and might therefore be better suitable in heavily loaded environments.

For performance considerations when replication is used, see section 2.5.3, “Server Replication Policies” on page 42.

2.4.6 Application Servers and Clients

In most DCE installations, DCE serves as a platform for applications that make use of the powerful services DCE provides. Users will most likely not see and use DCE directly, but they will use it indirectly through the use of such applications. Application developers get a whole bunch of new features with DCE, such as DCE RPC, security and directory services. Many, or even most, of the new features DCE provides involve resources not local to a single machine. Performance will therefore not only be determined by local resources but also by a complex system of distributed resources in servers, network components, and

the client systems themselves. A thorough application design can help to optimize the performance of such DCE applications. Figure 11 on page 38 depicts these basic difference between a local-only application and a two-tier client/server application.

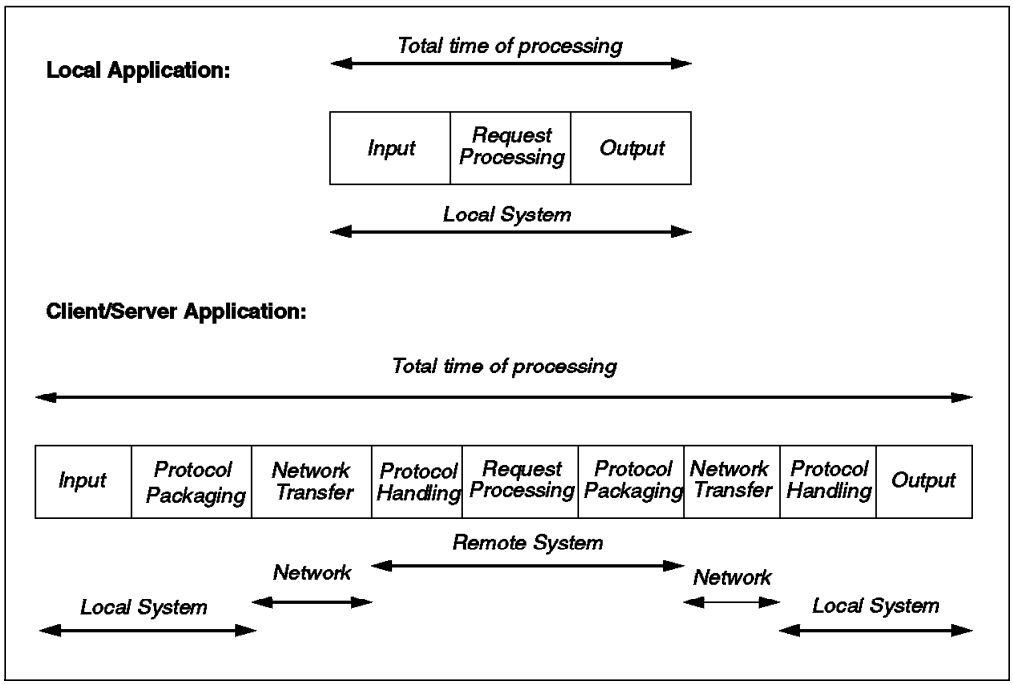


Figure 11. Local versus Client/Server Application

When using distributed DCE applications, the following basic guidelines should be considered for optimal performance:

- Keep servers close to clients

  A network connection always imposes some overhead and additional delays as compared to local functions, especially when a Wide Area Network (WAN) is involved. In general, the closer a server is to a client, the faster the connections are.

- Incorporate methods to find and use the closest server

  Basic and typical use of DCE directory services is to locate an appropriate server. Without special provisions, DCE directory services returns a list of servers that match the client's search criteria. Which server should a DCE client application choose? By using profiles in addition to simple directory and group entries in the CDS namespace, topology information can be incorporated in the default search algorithm so that client application can pick up the nearest server.

- Store semi-static information locally

  As a matter of fact, any kind of distributed application, including shared file systems, gains performance by storing data locally rather than requesting it every time it is required from the server. The disadvantage is to eventually access non-current data. DCE client applications should store as much information as possible for an adequate length of time depending on the frequency of change. The most obvious information a client application should store locally and reuse when required is the binding information to its server(s).

- Keep the amount of transferred data small

  Although the use of complex data structures may ease software development, care should be taken when such data structures need to be passed on to a server in an RPC call. Marshalling may take some time, and the transfer of additional data requires more network bandwidth. As a general rule, only required data should be passed on as parameters in RPC calls.

- Choose the appropriate protocol

  Depending on the specific environment, selecting a certain protocol sequence may also improve performance. As outlined in 8.4, “The Right Protocol Sequence” on page 159, TCP may yield a slightly better performance in a LAN- based environment, but this cannot be considered as a general rule. Tests will need to be done in order to evaluate the difference.

Most of these guidelines are not specific to DCE but to any kind of distributed applications.

For a more detailed description on DCE application development performance considerations, see Chapter 8, “Application Development With Performance in Mind” on page 155.

2.5 An End-to-End View of Performance Tuning

After having a look at the details of single transactions, like a user authentication process, and the single components involved in a distributed computing environment, in the previous parts of this chapter, we can now put the bricks together and look at the whole structure.

Performance tuning generally comes down to three single questions:

- Where is (are) the bottleneck(s)?
- How can it (they) be fixed?
- What unexpected behavior could degrade performance?

It is almost a law of nature that there always is a bottleneck. As soon as it gets fixed, the next one in the line will hinder an application from running faster. At first glance, the third question would seem to belong in the problem determination part of the book; however, it applies equally well as a topic under performance tuning. A system may be operational and appear to be working well, but could still be improved by controlling and preventing certain hidden bottlenecks.

Let’s go back to Figure 1 on page 8 and Figure 2 on page 10. Generally speaking, a distributed environment consists of a couple of server systems, some network connections, and a number of client systems. Computers may be of different types with different operating system platforms, and they have their own network connections with their specific characteristics.

Figure 12 on page 40 shows such an environment, still simplified to a large extent because there could potentially be hundreds of servers and thousands of client systems.

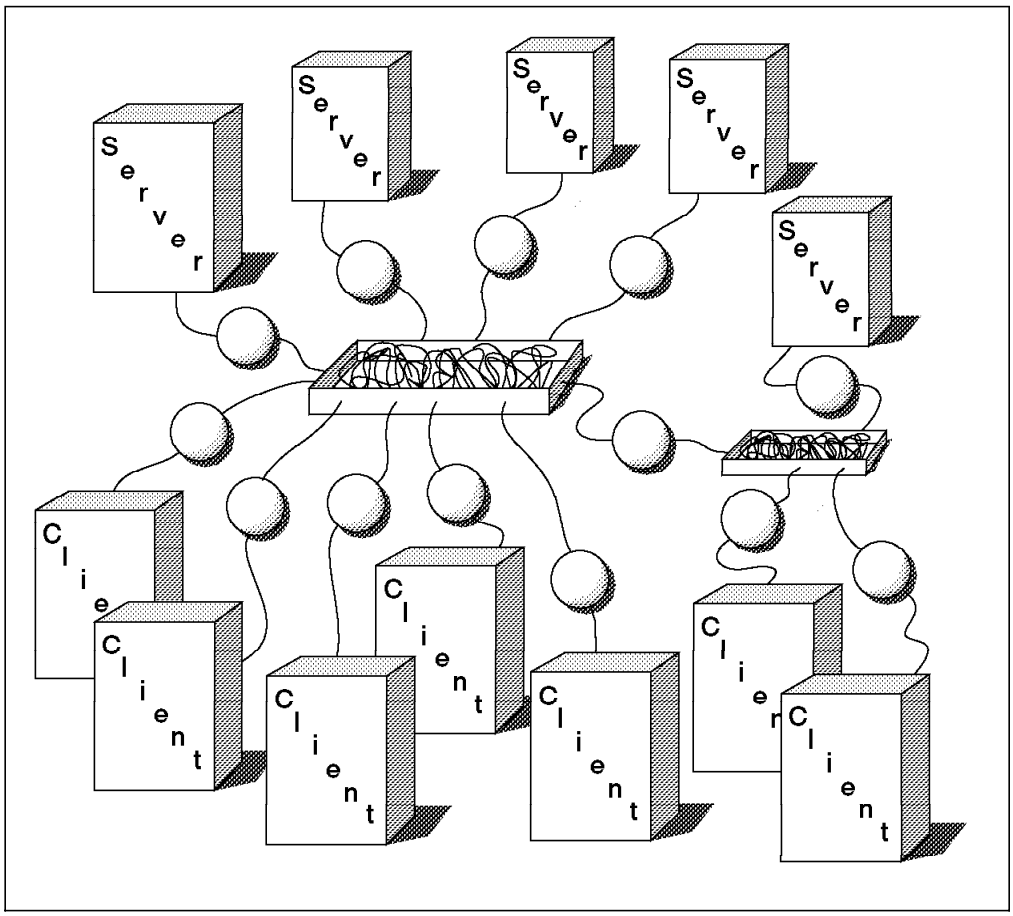


Figure 12. Generalized Distributed Computing Environment

The most important facts to consider in medium- and large-scale installations are the network topology, operational specifics, server replication policies, and server selection policies.

2.5.1 Operational Specifics

Despite the many technical options available to design and implement a DCE-based environment, the way it is used later on has an important impact on its ability to achieve the performance expectations of the DCE core services and applications. The planners and system administrators need to consider:

- Peak business hours

  Peak business hours are usually the most challenging threat for a system. A typical situation could be a large bank with many branches, all opening their services to customers at 8:00 a.m. It can therefore be expected that many, or even most, users log on to their systems between 7:50 and 7:55 every morning. In this short period of time, Security and CDS Servers are not only required for user authentication, but most password changes and application startups are likely to happen in the same time frame. Based on the figures in the earlier section on this chapter, some estimates on system and network load can be done.

- System monitoring and management

  System monitoring and management is very important for capacity planning and problem prevention. The additional load of such tools on a system must be taken into account. Most often, simple information gathering with

standard operating system commands do about the same, but require a lot less system resources compared to most analyzing tools. Report generation, which is usually a batch-type of process, can be off-loaded to other systems.

- Batch processing

  Batch processing is a common way to add or update registry or CDS information on the respective servers. In either case, not only the master replica server will be affected but also slave replica servers will be involved in any update actions. For obvious reasons, batch processing should be avoided on servers during peak hours.

- System backup

  System backup is a regular task important on every system. Modern, high-performing backup products and techniques require a considerable amount of system resources and sometimes even run at higher process priorities than most applications, such as DCE server processes. It is therefore important to use such tools with care and only at times when the servers are free from load.

- Other applications

  Ideally, DCE server systems run dedicated for their purpose. Other applications should be avoided on servers due to availability, security and manageability reasons.

2.5.2 The Network Topology

A proper DCE cell design must take the network topology into account and place servers, replica servers and clients in a suitable way.

If the whole DCE cell runs on a network of a campus-type environment, for example, where there are high-speed, LAN-type interconnections between all involved machines, the placement of servers and clients is of less importance. It is, however, desirable to have servers connected more closely to high-bandwidth backbone segments. Other design principles, like availability, physical location considerations and security, are important, too, and may even take precedence over performance considerations. Performance tuning on the server systems themselves may be more important. The additional network traffic induced by DCE core services is most often negligible.

Slow links in a network always introduce some potential bottlenecks. DCE provides, and uses by itself, a wide range of built-in flexibility options to locate servers and to do arbitrary load balancing among them. The drawback is that DCE client services generate some additional network traffic for look-up and binding construction purposes that are not strictly related to the application logic. On slow links, this might cause some extra, undesirable delays. There are methods to speed up DCE services over slow links, depending on how a slow link is involved. We will discuss the most typical situations.

Dial-Up Connections from Single Clients: Dial-up connections, for example from employees working at home or travelling around, commonly run at transfer rates of 14.4 or 28.8 Kbps. If DCE services are being used extensively over such connections, speeding them up may be desirable. The DCE authentication process (`dce_login`, or `dcelogin`, respectively) can be accelerated by using the static configuration contained in the pe_site file. This is already the default on the OS/2 implementation, but needs to be configured on AIX. See 7.2.1,

"Security Service, the pe_site File" on page 142, for more details. DCE client applications should provide the option of using static binding information for their servers, for example through the use of an environment variable or a configuration file. This way, directory look-up overhead can be avoided.

Remote Multiuser System(s), Remote Workgroup: If a group of users, say 5-50 users, work on one or a few client systems that are remotely connected through a slow network link that uses DCE applications, the same basic tuning options apply as mentioned in the last section (dial-up connection). If the DCE applications are mission-critical, installing a DCE Security and/or CDS Server replica at the user's location can be an option for availability reasons. However, this can introduce a dangerous side effect: Other client systems, not local to this group, learn about the existence of such a server and may start to use it, creating a lot of load on this low-bandwidth network link. It is important to avoid this situation by having other client systems use specific, preferred servers. See 7.2, "Server Preferences" on page 142, for more details.

Interconnected Local Area Networks (LANs): If separate, high-speed LANs having a considerable number of users connected to each of them are interconnected using low-speed links, server replication should be considered to spread the load and to increase availability. To avoid cross-using servers from clients on the other side of the link, methods as described in 2.5.4, "Server Selection Policies for Clients" on page 45, and in 7.2, "Server Preferences" on page 142, should be applied.

2.5.3 Server Replication Policies

Server replication increases availability and performance. Both, however, do not come for free without additional provisions. Performance can only be improved when some kind of load-balancing technique is in place. The simplest and most common way to achieve this is to let the clients randomly select from a list of available servers.

If replication is to be implemented (which is strongly advisable in any production environment), design decisions must be taken into account in order to have the right number of servers placed at the right place. In a fast, campus-type network environment, placing servers is presumably not critical for performance, but it may be for availability. Most often, servers are connected to different high-speed backbone segments for both availability and performance reasons.

If slower network links, like WAN-interconnections, come into play, the question may become more delicate on whether to place a server on either end of this connection. Figure 13 on page 43 shows a typical situation where some client systems, for example in a bank's branch, are connected to the headquarters through a WAN connection.

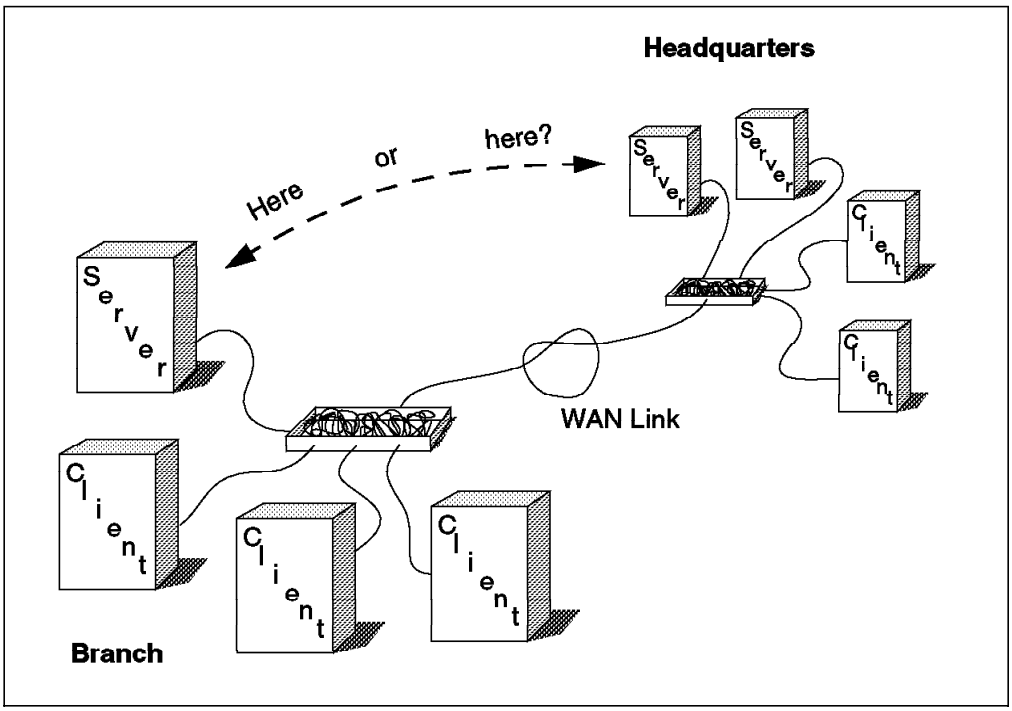


Figure 13. Server Placement

There is no general answer for this question. If the branch needs to be self-sufficient in case of a WAN-link outage, a server will probably be required in the branch unless all application services are designed and implemented for such a fallback situation without a server. But in terms of cost and administration, this is probably the most expensive solution if there is a large number of such WAN-connected branches.

In technical terms, despite the cost and administration overhead, there are good reasons for both configurations depending on various considerations.

Having a server on the client side of the WAN is preferable or obvious when:

- Clients need a high-speed connection to the server.

  This is usually not necessary for DCE core servers (Security and CDS Servers), but may speed up an application when the server in question is a DFS File Server.

- The server does not get or need much data from a central side in order to stay up-to-date.

  If the server in question is a DCE Security Server in a large cell, it will probably receive many updates (user and group administration, password changes, and so forth) every day, which may impose more traffic on the WAN than what local clients would introduce when this server was remote.

- The updates necessary can be handled by the central server(s).

  Decentralized servers require updates to stay current. They must receive updates—depending on the business requirements or the technical implementation—either immediately or during batch runs. These updates must be fed by a central server that must be able to deal with the number and the complexity of distributed data updates.

Having the server(s) centralized on the other hand is advantageous if:

- There is not much traffic between the clients and the server.

  DCE core functions, for example, do not require a high network bandwidth when properly implemented and do not need to be replicated in such an environment in most cases unless availability is a strong issue.

- The complexity of updates jeopardizes a reliable operation.

  Remote database updates are not always easy to handle, especially when a large number of remote servers are to be kept current. If data currency is an issue, centralized servers are much easier to control.

- If there is much update traffic to the server.

  A remote DCE Security or CDS Server (depending on configuration) will instantly receive updates from their master replica.

Security Server replication is capable of handling single updates only, for example, only updates will be distributed to replica servers, not the whole registry database. If a new replica server is being added, however, the whole registry database will be transferred to that new replica in order to make it current. This transfer does not necessarily require the Master Security Server as the source; a new replica server can receive the registry database information from any other replica server. As a raw figure, this copying process generates about 2 KB of network traffic per registry object (for example, 10,000 objects will generate about 20 MB to be transferred).

Replication policies for the CDS namespace need a thorough planning as the complexity of single directory replication is higher than compared to security registry replication. In large-scale DCE cells, certain directories in the CDS namespace may become very large, for example the /.:/hosts directory if there are many client systems in the cell. The entries in this directory, on the other hand, usually do not require often and fast access since they are most often accessed only when client systems boot up. For this reason, this directory could be placed on a less-powerful CDS Server with limited memory since only booting machines are affected.

Luckily enough, replication of DCE core services is easily configurable just in case it should become necessary, or unnecessary, to have an additional replica. As a general rule of thumb, unless the geography (WAN connections) or business-critical applications require replicated services at certain locations, DCE core servers should be replicated onto two or three equally powerful, centrally located servers.

Example Calculations: Some simple calculations, based on figures in the earlier sections of this chapter, may help to understand, or even help to decide on whether to place a server on one or the other side of a WAN connection:

- Assuming a DCE cell with 10,000 registry objects, a daily update ratio of 5 percent, and a branch with 50 users. The updates cause 500 update transactions to a remote DCE Security Server. This is not much more, but is definitely more, than the logon traffic imposed by 50 users in that branch if the Security Server was centrally located.
- Let's further assume (see example above) 2,000 users log on to DCE between 7:45 a.m. and 7:55 a.m. If the password restrictions require the users to change their passwords every month, 5 percent changes can be expected every day, and these changes will most likely happen during the same logon period. Since every update goes to the DCE Master Security

Server and then immediately to the replicas, the replica server in the branch will receive 100 updates via the WAN as compared to only 50 logon requests from local users.

- Assuming the WAN connection is a 64 Kbps link with 20 milliseconds latency time (time to be added to every frame transmitted due to router buffering and so forth.). If there is no DCE core service replica in the branch, a user will experience an additional delay of about 3.5 seconds due to the WAN when performing a DCE login (65 packets x 20 milliseconds + (65 x 278 x 8) bits / 64 Kbps = 3.5 seconds; see 2.3.2, “User Authentication (dce_login) and Re-Authentication (kinit)” on page 17).

2.5.4 Server Selection Policies for Clients

Once servers have been placed close enough to the clients in order to prevent bottlenecks, the clients, on the other hand, must be able to locate the servers that are in close proximity.

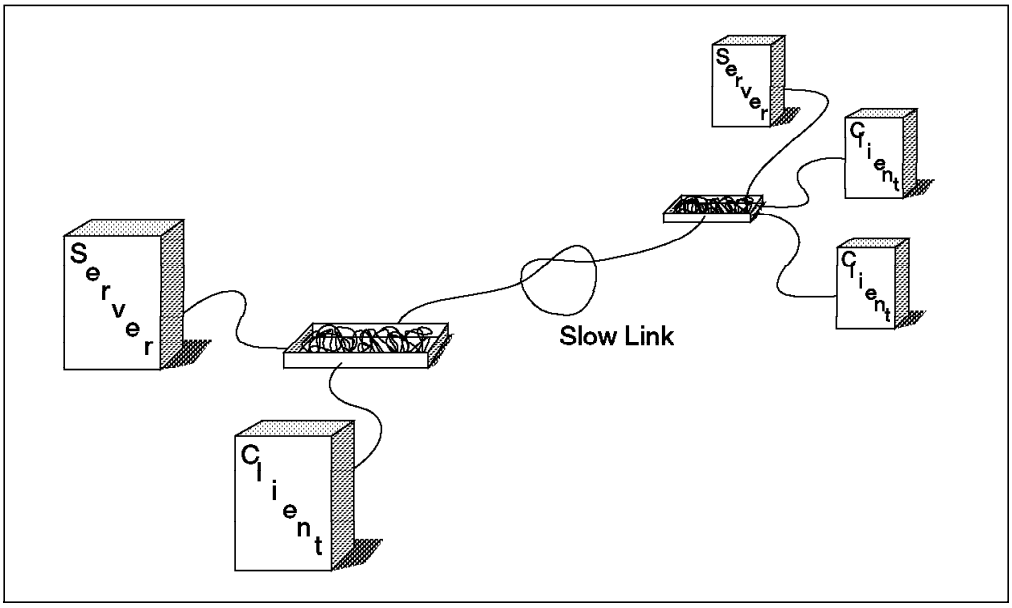


Figure 14. Local and Distant Servers

Figure 14 shows a situation with a client (in the front) and two servers. Assuming both servers could be suitable for the client’s needs, it is obviously desirable that the client binds to the local server. In order to get binding information for a server, a client sends a request to the DCE Directory Service (not shown in the illustration). Unless configured appropriately, CDS returns a list of servers in a random order, and it is up to the client to select one from this list. It the client by default selects the first in the list, there is a 50 percent chance to get connected to the (wrong) server across the slow link instead of the local one. This chance even increases if more servers are added. The same happens to all the other clients, and this can end up in having undesirably high delays due to unexpected heavy utilization of the slow link. In fact, some DCE core services, by default, use exactly this random pattern, unless otherwise configured. The user authentication process (`dce_login`) on AIX is such an example. Note that this discussion is valid for DCE core services as well as for user-written DCE application servers and clients.

There are various ways to control the selection of servers:

- DCE Directory Services provide search policy configuration through the use of profile entries. Profile entries contain search priorities for the server binding lookup process. Clients do not need to be aware of profile entries, they will automatically get back a prioritized list of server bindings. The disadvantage of this solution is the additional administration necessary to define and maintain location-dependent profile entries. If such profiles are not present, random selection will take place.
- A client can analyze the TCP/IP addresses of all server bindings returned by CDS. If a server address appears to be in the local subnet, it can be assumed to be a local server. This technique is used in some DFS services. The disadvantage is that it obviously does not work if the local server is not in the same subnet as the client, or when the local subnet includes a WAN connection.
- Servers can broadcast their presence on the network, and clients can pick up such broadcasts. Since broadcasts are limited to the local subnet, clients only get the local server(s). DCE CDS Servers follow this scheme, and they broadcast their presence in regular intervals. Alternate methods must be used in clients in case no such broadcasts are received by clients.
- A client can use a local configuration, for example a configuration file or an environment variable, with static binding information. User authentication on DSS for OS/2 Warp (`dcelogin`) uses this technique by default. Although this is the fastest solution (because CDS does not get involved at all), it lacks some flexibility. If a specified server is not available, the client fails to bind. Clients must be able to use alternate methods in such cases (`dcelogin` does).

All of these techniques have advantages and disadvantages, and therefore most client implementations actually use a mixture of these techniques. Whereas the use of profiles is probably the most elegant way, it is certainly not the fastest, and it requires profile management. The fastest method is to have static binding information stored in the clients, but it requires this configurations to be updated when servers are added, moved, or removed.

Appendix D, “Setting Preferred Security Server” on page 241 describes a method on how to set up OS/2 Warp clients so that they can use CDS profile entries to update static, local configuration for `dcelogin`.

Chapter 3. The Distributed File System

Since the needs of today's expanding and heterogeneous environments are growing, the need for a sophisticated technology that makes these environments scalable and manageable arises. DCE Distributed File System (DFS) technology provides the ability to access and store data at remote sites. It extends the view of a local system and extends the size limit of the file system to a distributed file system of almost unlimited capacity, located on several remote systems.

In this chapter, we cover the components of DFS, how DFS handles its data flow and its components, and the impact on performance of the resources of our network and its components. More details about configuration considerations can be found in Chapter 6, "Step-by-Step DCE Set Up Geared to Performance" on page 123, and in Chapter 7, "After-Installation Performance Tuning" on page 141.

3.1 Overview and Machine Roles

DFS is an efficient, extensible system that allows you to share your data. Unlike other file systems, DFS uses sophisticated caching algorithms on the client computers which can store large amounts of information on file status as well as data. This means that the client requires less data requests to the file server, thus reducing the network and server load and increasing overall performance. Additionally, the file server keeps track of which clients have cached copies of files, thereby assuring consistency of data.

There are different types of servers in a DFS environment. Each server machine can handle one or more functions or roles. In the next section, we will discuss the function of these machine roles. Note that the machine's roles are not mutually exclusive. Any server machine can assume multiple roles.

3.1.1 The DFS System Control Machine (SCM)

The System Control Machine has the responsibility of controlling various lists of users and groups that can perform administrative functions on a collection of DFS servers, called an *administrative domain* or a DFS domain. The SCM has the master copy of these lists. There can be multiple administrative domains in a DCE cell, but a certain DFS server is only a member of one domain at a time. There must exist one SCM per administrative domain. The purpose of administrative domains is to delegate the managing responsibility of a part of DFS to a separate group of administrators.

The SCM machine is the first machine that must be configured for any new domain. It can be used to distribute the administrative lists for that domain from its /opt/dcelocal/var/dfs directory to any subsequent server machines added to that domain.

There are two important processes that run on an SCM; the `upserver` and the `bosserver` processes. The `upserver` process controls the distribution of the administrative lists to all the other server machines in the domain.

Another process, `upclient`, runs on DFS Fileset Server machines (see 3.1.4, "The Fileset Server Machine" on page 50), and it contacts the `upserver` process on the

SCM for this domain every five minutes, by default, to synchronize its own copy of the administrative lists with the SCM.

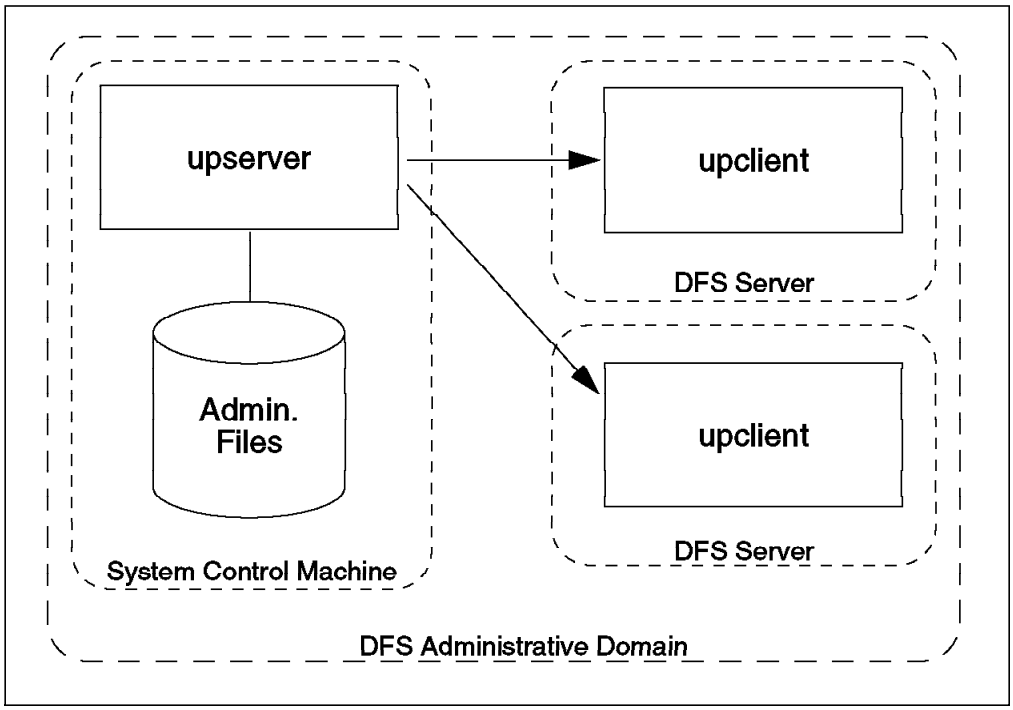


Figure 15. Upserver/Upclient Processes

The bosserver process is the Basic Overseer Server process, and it is not associated with any specific machine role. It runs on every DFS server machine and its primary function is to supervise processes that are listed in its BosConfig configuration file, located in the /opt/dcelocal/var/dfs directory. It restarts any failed processes automatically if they are listed in BosConfig file.

The SCM machine has to be defined as a DCE client, along with all the other DFS Servers, in order to use the required authentication and name services.

3.1.2 The Binary Distribution Machine (BDM)

The Binary Distribution Machine can be used to distribute identical versions of binary files to other machines in the cell. This binary files are used for processes and command suites, and they are stored in /opt/dcelocal/bin and related directories. This binary distribution mechanism may be used for DFS servers to receive the proper binaries that relate to their CPU type and version of operating system.

Binary Distribution Machines use the same upserver/upclient utilities that the System Control Machine uses. Like the SCM, the BDM runs an upserver process, and other DFS servers run the upclient processes.

Each DFS Server keeps a copy of the DFS binaries in a local directory. The binaries are installed on a single BDM that acts as a source for the others. This would distribute the binaries to other machines in the domain that are of the same CPU/OS type.

3.1.3 The Fileset Location Database Machine

This machine maintains and stores the Fileset Location Database (FLDB) that holds information about exported filesets. DFS clients contact the FLDB database to query the actual location of a directory or a file they need to access.

The Fileset Database Machine runs the `flserver` process that maintains the FLDB. The purpose of the FLDB machine is to take a pathname for a file that is located in the DFS filespace and determine the location of the Fileset Server that has that file. Pathname requests come from DFS clients' cache managers on behalf of a user or an application. Thus, the user never has to know the physical location of a file.

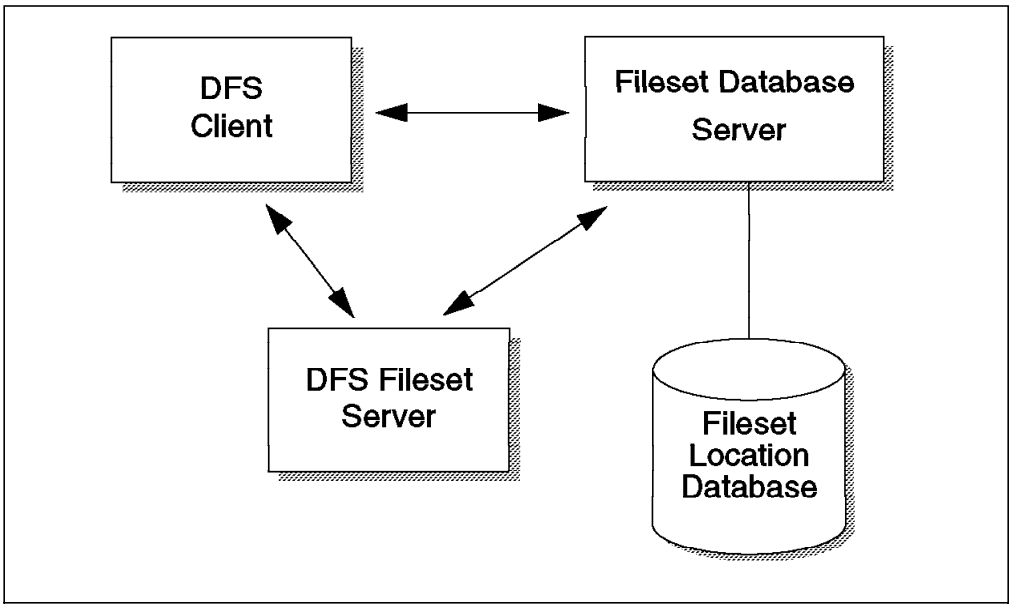


Figure 16. Fileset Server Process

FLDB machines should be replicated for availability reasons. An odd number should be considered because the automatic voting process for the primary FLDB Server works best with an odd number of systems.

The primary database is the one that handles the updates. All the others will be read only copies of the primary. The process to determine which FLDB will be the primary is accomplished by the use of internal *ubik* routines. These ubik utilities are based on the action of a quorum of vote. The ubik coordinator, which runs on the primary FLDB site, periodically sends an RPC to each secondary FLDB site and waits for a response from the coordinator at the secondary FLDB site. This serves as a vote to maintain the current synchronization site for a fixed amount of time.

If the primary FLDB Server fails or gets disconnected for some reason, the ubik algorithm on the FLDBs vote for another machine to take over as a primary FLDB site. The FLDBs update themselves to the latest FLDB information that is available and continue to operate according to the ubik algorithm.

3.1.4 The Fileset Server Machine

A cell may have many Fileset Server machines (commonly referred to as File Servers) configured. One of the benefits of DFS is this type of scalability, the ability to simply add new File Servers to increase storage capacity.

A File Server machine actually stores the data in the filespace on its local disk(s). The administration for this machine, as far as DFS is concerned, is done by an appointed set of administrators that do not necessarily have to be logged on to this file server machine.

The main processes running on a File Server are the file exporter, `fxd`, and the fileset server process `ftserver`. The file exporter runs as part of the kernel. It provides the same services across the network as the operating system provides for a local disk.

3.1.5 Private File Server Machine

The purpose of a Private File Server is to exclusively allow individual users to export and control their data. Therefore, a user on this machine can export file systems and have full control over the filesets that he/she has exported.

A Private File Server machine is very similar to a Fileset Server Machine with the exception that it controls its own administration and is therefore independent from a System Control Machine.

3.1.6 The DFS Client Machine

The DFS client machine allows a user (or multiple users) to access files from the DFS filespace. A DFS client machine can be either a single or a multiuser workstation. It communicates with Fileset Server machines to access files for application programs, provides local data storage, and does actual computation. Note that any other DFS servers can also function as DFS clients, if configured appropriately.

Each DFS client machine runs the following processes:

- `dfsd` — This process initializes the Cache Manager. The Cache Manager communicates with the FLDB and File Servers to obtain the location of files and the data the user has requested. It also can be used to alter aspects of the Cache Manager's cache, such as its location and size.
- `dfsbind` — It runs on both DFS client and Fileset Server machines. This user-level process works as an intermediary between the DFS kernel processes and DCE Security and CDS Services. The `dfsbind` process contacts CDS to resolve DCE pathnames that it receives from the DFS client; if it encounters a junction for the DFS filespace, it returns information about the Fileset Location Database Machines for the cell. The `dfsbind` process is also used to obtain user-authentication information.

3.1.7 The Backup Database Machine

A backup database machine stores the backup data, which contains administrative information used in the DFS backup system, such as the dump schedule for backups and the groups of filesets to be dumped to tape in each backup. The information in the database can also be used to restore data from tape to the file system should it become necessary.

There is one master copy of the Backup Database per cell. As with Fileset Location Database machines, you can have more than one of these machines in order to eliminate a single point of failure. It is advisable to configure three or a larger odd number of Backup Database machines in a cell.

These backup database machines use ubik technology in the same way as the FLDBs. This means that there will be a primary site as well as secondary sites. The ubik routines determine among multiple copies of the database which one is the master. These routines keep communication with each other.

The DFS backup mechanism allows you to do a file system backup while the system is in use. This is achieved by creating an internal replica of the filesets before backup is done. The net effect is that the replicated copy remains stable while users might be changing the original files.

We will not go into any further detail concerning the DFS backup process in this book. As far as tape backup performance is concerned, tape-drive units are the common bottleneck in a backup process.

3.1.8 Tape Coordinator Machine

The Tape Coordinator Machine is used to physically back the filesets up to tape. It communicates with the backup database to receive instructions on which filesets to backup and when this should be done. The Tape Coordinator machine can physically be the same machine as the backup database machine.

In general, the Tape Coordinator machine could have one or several tape units attached to it. A Tape Coordinator Machine runs a process called `butc`. A Tape Coordinator uses an identification number, called tape coordinator ID or TCID. When you issue backup commands, you have to use the TCID, which is a unique number in the cell to identify the tape coordinator. If you have several tapes attached to the DFS Tape Coordinator Machine, you will have one tape coordinator process (and unique TCID) for each tape unit on that machine.

As mentioned under the last section, we will not further detail DFS backup operations in this book.

3.2 DFS Internal Components

DFS is built on the concept of a client/server architecture. The client requests data or information and the server provides it. The communication between the server and the client is handled with DCE Remote Procedure Calls. Besides the machines' roles in DFS as discussed in the last sections, there are function blocks, or components within DFS, that we want to look at in the following sections.

3.2.1 Cache Manager

The Cache Manager (CM) is the major client part of DFS. When the user does a request, the CM looks to see if the requested data is already cached on the local system; if not, it sends the requests on to the File Server for the data, then caches it locally when supplied. Information about the fileset location, obtained from the FLDB Server, is cached as well.

The CM utilizes a sophisticated token-management system that handles separate tokens for read and write access. The CM also utilizes a delayed-write

policy that allows you to propagate the data to the server under certain and controlled conditions.

3.2.2 File Exporter

The File Exporter is the major server part of DFS and runs on DFS Fileset Servers. It handles data access requests for files that the server manages. The server is also responsible for cleanups and recovery after broken connections. It uses the Token Manager to handle synchronization of different clients accessing the same file simultaneously.

3.2.3 DFS Local File System (LFS)

The actual file system for the cell. It manages the storage of files on disk and manages the ACLs. It also handles the ability to replicate, backup, and move filesets without the interruption of the services. In DFS, the use of LFS allows the file system to achieve global appearance and high availability. The use of the LFS in DFS introduces two important enhancements for availability. One is the use of a logging technique for reliable recovery from systems crashes. The other is the use of filesets. They hide the complexity of the physical file system structure and aggregates, which represent a partition of disk storage.

3.2.4 Token Manager

This function runs on every Fileset Server. It synchronizes the access to files by multiple clients. It keeps track of cached data and works together with the Token Manager function in the Cache Manager, running on DFS clients. The Token Manager allows DFS to keep its load low and reduce the network traffic by guaranteeing correct access. Based on the tokens, mandatory and advisory locking for files and records on files is provided.

Figure 17 on page 53 shows the basic relationship between the main DFS components in a classical client/server configuration.

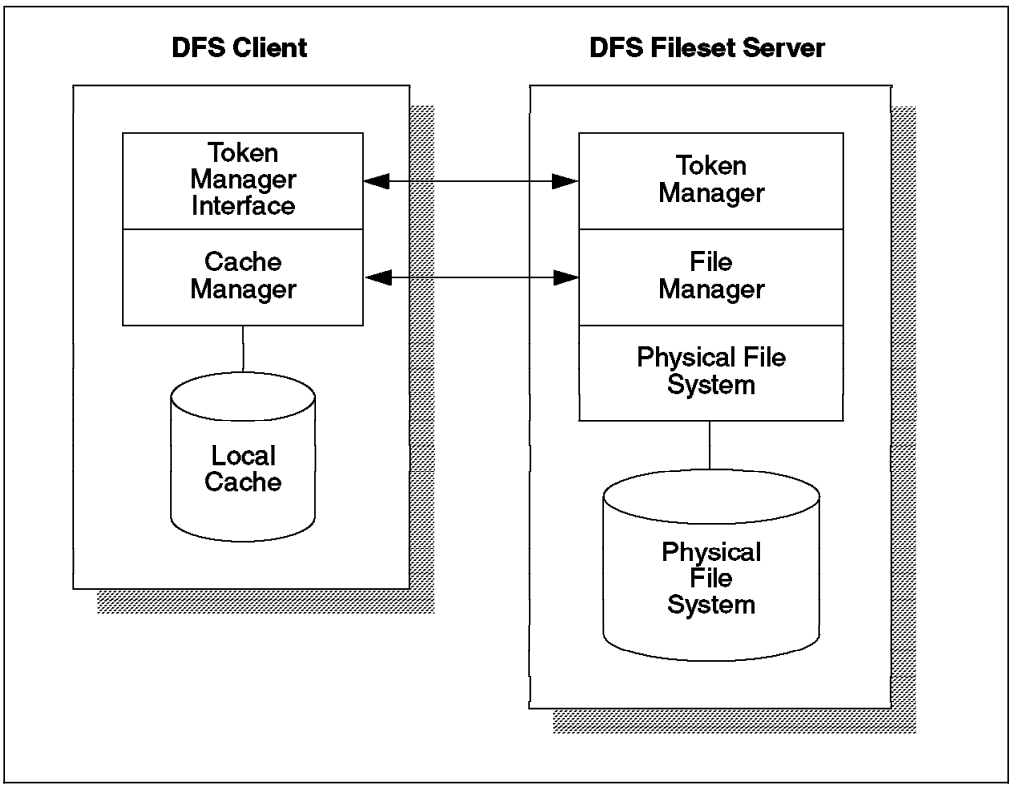


Figure 17. DCE/DFS Client/Server Interaction

In the next sections, we discuss how DFS works and how the components contribute to an overall performance of the file system.

3.3 DFS Flow of Controls

DFS provides the user in the DCE cell with a global view of a set of files and directories that are exported from one or multiple File Servers. The sum of all these exported files and directories looks like a single filespace for the cell.

In DFS, data storage space is organized in three levels: *files* (and directories), *filesets* and *aggregates*. The file is a unit of user data. Files are organized in a hierarchical directory structure. The directories themselves are located on filesets. The filesets reside on aggregates. A fileset is a collection of directories and files managed as a single unit. The aggregate is the unit of disk storage, that can contain one or more filesets. It is also the unit of data exporting. Filesets can vary in size, but they can obviously not be bigger than the housing aggregates.

Figure 18 on page 54 shows the relationship between fIlesets and aggregates on a physical disk space.

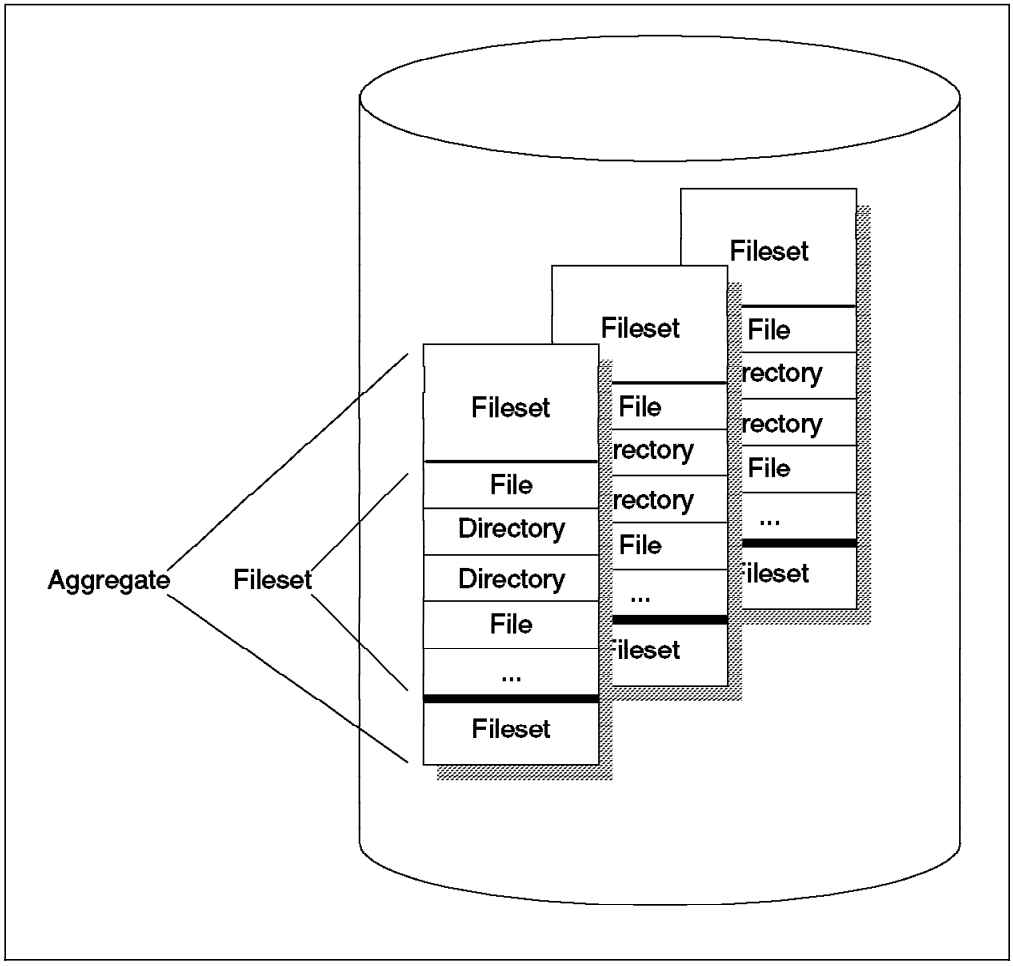


Figure 18. DFS Aggregates and Filesets

When a file is accessed from a DFS client, data is moved between the File Server and the client in fixed-sized segments, called *chunks*, with a length of 8 to 256 kilobytes, depending on configuration. These chunks are held in the cache by the client, and there can potentially be a large number of them. Chunks remain valid in the client's cache until another client makes a modification to the data it represents. This means that, once a chunk is cached on the client, access to its contents will be comparable to local file access, as far as performance is concerned. Cache chunks can even remain valid across client or server reboot.

When an application requests to read a portion from a file for the first time, the first chunk of the file gets transferred from the server. When the last byte of a chunk is written, the whole chunk will be sent back to the server. Files that are smaller than the chunk size occupy a single cache chunk file, but only with the actual length as the file.

When the Cache Manager resolves a path name to locate a file, it begins at the top-level fileset in the cell and accesses a read-only version of the fileset whenever possible. If any fileset in the path name has no read-only version, the Cache Manager accesses only the read/write fileset versions until the filename is resolved.

Transfer of data between clients and servers is handled by background threads in the DFS code. This means that a user process can continue to run while data

is being transferred in the background. There is a pool of such threads, and thus several transfers can be done in parallel.

DFS supports administration and access security by supporting Access Control Lists (ACLs), available with objects in DCE LFS filesets, and by using administrative lists. The communication between DFS clients and servers takes place using an authenticated RPC mechanism. DFS uses the principal's credentials, issued by the DCE Registry Service upon user log-in, to verify the information related to the user, group, machine, and account information against ACLs.

As an excellent trade-off between reliability and performance, DFS uses an algorithm that allows the flow of RPC packets to adapt dynamically to the level of reliability in the network. It uses a slow start technique to avoid overrunning the communication channel.

DFS controls the consistency of files by requiring that clients obtain a *token* for any file operation they perform. When a client wants to access a file, it first checks to see if it already has a valid token for that file. If this is not the case, it will request it from the File Server. Through the token mechanism of DFS, the server is able to check if another client has currently been granted access to the same file, and it will not issue a new token if the requested access endangers file consistency, for example when multiple clients request write access to a file.

Tokens are divided into two main categories: *status* tokens and *data* tokens. Status tokens controls the access to the file's attributes, like ownership, modification time, and so forth. The data tokens represent a certain range within the file or a particular part of it. This makes it possible for several clients to work at the same time with the same file. Revoking the data token from a client will force it to immediately store any modified pieces represented by that token back to the server.

3.4 Client Caching in DFS

The client caching mechanism is one of the key features of the architecture and performance advantages, which makes DFS a very scalable distributed file system. Client caching allows the majority of file read/write operations to occur locally on the client, thereby avoiding network traffic and server resource utilization.

The client cache can be configured to reside in memory or on disk; either choice is useful for reducing server load and network traffic.

The DFS Cache Manager (CM) is responsible for the local caching of file and directory data on DFS clients. Caching means that once any portion of a file is requested and received by a client, a copy of that data is kept in the client machine's cache.

The first time data is read, a user gets so-called cold cache performance; subsequent reads of the same data result in hot cache performance. The hot cache performance compares well with local file system performance.

When an application on a DFS client issues a read request for DFS file data, the minimum amount of data that will be fetched from the DFS server is a chunk. For large, sequential file read accesses, larger chunk sizes generally give better

performance by reducing the overhead associated with the underlying network communications traffic.

DFS also incorporates a read-ahead mechanism. It improves the sequential read performance through its capability to detect a sequential read access. This allows for data to be already available in the cache when an application actually requires it. This happens only if an application starts to read file data in a sequential manner.

Read access to the DFS cache comes in two flavors, *hot cache*, which refers to the situation where data is already present on the client, and *cold cache,* where it must be fetched from the server.

The total size of the client's cache generally determines the chance of a hot cache hit. A small cache size obviously reduces the chance of having a file or a file chunk already in the cache when the file is referenced.

Once data in the cache has been modified, these changes must eventually be propagated to the File Server that is responsible for permanent storage of the file. DFS and NFS implement similar delayed-write policies for forwarding these modifications. Modified DFS file data is propagated to the File Server under several conditions:

- On a periodic basis (every 30 seconds) by one of the DFS client processes
- If the server revokes the client′s write token
- If the DFS client cache becomes full
- Upon a file close or fsync operation

Although the minimum unit of transfer across the network is a DFS chunk for read operations (unless a file is smaller), it is not the case for write operations. In AIX, the minimum amount that will be transferred is a page (4 KB). Therefore, if only a few bytes of data are modified, only 4 KB are sent across the network, rather than a whole chunk.

3.5 Cache Performance Considerations

The amount of local disk space or memory allocated for the Cache Manager to use for its cache affects the speed of file access. A larger cache generally means that the CM has to contact the File Server machine less often, resulting in less traffic on the network. If the cache size is small, it will fill up rapidly, forcing the CM to discard cached copies of data to make room for newly requested data.

The amount of disk space or memory used for caching depends on several factors. The amount of available, free disk space or the amount of memory available on the machine places an physical limit on the cache size. The selection of what type of cache to use on a DFS client can to be done with the following considerations in mind.

3.5.1 Disk versus Memory Cache

The default DFS configuration uses disk cache, but memory cache can optionally be configured instead.

You can consider the following to assess the disk versus memory cache decision in your environment:

- If small amounts of data are frequently reaccessed, the smaller access time of a memory cache would be beneficial to speed up applications.
- If the typical work involves large files that would benefit from caching, a disk cache should be used because it can easily be configured to be large enough (provided sufficient disk space is available).
- Memory cache generally provides better performance either when data is often re-accessed due to its shorter access times compared to a disk cache or when data is typically accessed only once, because any caching would then be useless and disk caching would even slow it down unnecessarily.
- If the system shows any sign of being memory bound, for example if much paging is involved in normal operation, memory cache should not be used for DFS.
- If the disk subsystem does not provide adequate read/write performance, as certain RAID configurations do, or when using diskless clients, memory cache would be more adequate.

If a system runs with disk cache, the size of an adequate memory cache can be estimated issuing the `cm getcachesize` command toward the end of a typical work load period, Divide the number of 1 KB blocks being displayed by 0.9 to determine the memory cache size to hold the same amount of data (about 10 percent of the blocks in the cache are reserved for CM internal record keeping).

For more information on how to set up a DFS cache, or to change a DFS cache after installation, refer to Chapter 6, “Step-by-Step DCE Set Up Geared to Performance” on page 123, and to Chapter 7, “After-Installation Performance Tuning” on page 141.

3.5.2 Cache Location

The standard location for a disk cache (/opt/dcelocal/var/adm/dfs/cache) can be changed to take advantage of better performance or larger space availability on other disk partitions.

In order to benefit from its performance improvements, a disk cache should be located on an area of physical disk space where fast access is possible (high disk I/O rates), just as is required for a system’s paging space:

- In the center, or outer edge area of a physical disk drive, depending on its specific characteristics (normally, center area for small disks: outer area for disks with more than 200 MB).
- In a separate filesystem.
- On less frequently used disks, for example on disks that are not in the rootvg volume group.
- Not in the same physical volume as a paging space or equally distributed with paging spaces among several disks.

- Distributed among several disks, using the *maximum* policy for the Logical Volume containing the cache file system.

3.5.3 Cache Size

The cache size has an influence on how often the Cache Manager contacts the File Server. Increasing the cache size generally results in better performance because less cross-network traffic and server access is necessary.

The default cache size is 10000 KB, which is sufficient for small client environments that make occasional use of DFS. A cache size of 50 to 70 MB may be appropriate for heavy DFS usage on a single-user client system, and 100 to 150 MB may be appropriate for a multi-user system. A cache smaller than twice the chunk size is rounded up. You have to know and consider the type of work load that a DFS client will have when you define the amount of the cache size.

Determining the appropriate DFS cache size for a particular system will need some investigation. You may begin by estimating the sum of:

- The sizes of the set of DFS-resident data files that are read at least once a day
- The amount of DFS-resident data that is generated by the user(s) on the system each day
- The sizes of the DFS-resident programs that are executed more than once a day

The amount of memory that can be used for a memory cache should not exceed 25 percent of the available physical memory. A good rule of thumb is to use 5-10 percent of the installed physical memory, going up for heavy DFS usage, or decreasing it if DFS is only occasionally used.

The cache size is specified in the CacheInfo (CachInfo on OS/2) file, located in the /opt/dcelocal/etc directory, and can be overridden with the `dfsd -blocks <size>` command option. This parameter applies to both disk and memory cache.

3.5.4 Chunk Size

The size of chunks can be modified to take advantage of fast networks or to compensate for slow networks. Also, application-specific requirements can be met with configurable chunk sizes. The number of chunks that the cache can store is given by the combination of total cache size and chunk size.

More information on how to configure chunks can be found in 6.7, “DFS Setup” on page 137, and in Chapter 7, “After-Installation Performance Tuning” on page 141.

3.5.5 Server Preferences

The Cache Manager maintains a rank of Fileset Server and Fileset Location Database Server (FLDB) machines. These ranks determines which server the CM contacts to access a file or to query the FLDB. Every Cache Manager builds this list based on certain criteria, like the network proximity. This rank list can be modified using the `cm setpreferences` command in order to specify certain servers to be contacted first.

See 7.2.4, “DFS Fileset Servers” on page 145, for more details on how to do this.

3.5.6 Unstored Data

If the Cache Manager cannot contact a File Server machine to write data back to it, it keeps the unstored data in the cache. It then continues to attempt to contact the File Server Machine until it can write this data back to it. Since this unstored data makes a certain portion of the cache unusable for other data, it can be released with the `cm resetstores` command, which will discard any modifications done to this data.

3.6 DFS File Server

DFS File Server performance is essential for overall DFS performance, since this can potentially become a bottleneck. The following should be considered for best DFS Server performance:

- Use fast disk subsystems. For obvious reasons, disk I/O is a key factor on DFS File Servers.
- Place aggregates in such a way on the disk(s) that fast I/O can be achieved.
- Tune DFS Server parameters. See 7.4, “AIX DFS Server” on page 148, for a description of tuning parameters.
- Add replica servers with read-only filesets in order to spread load among different machines.

3.7 Replication for Performance

One of the principal uses of replication is that the load balance on the servers yields better overall performance.

Replication is the process of creating one or more read-only copies of the data being accessed by clients. This allows for read-only copies on machines and even multiple sites. Therefore, after a machine failure, clients can access the information from a different source.

Read-only filesets are known as *replicas* and are usually placed on different physical machines in the distributed file system. A replica contains the same information as the read/write fileset, but is limited to read access only. Updates can only be done on the read/write filesets.

There are two types of mount points for filesets that play an important role in the decision on whether the read/write or the read-only fileset is going to be accessed:

- Regular mount points — Any type of fileset can be mounted via this type of mount point.
- Read/write mount points — Only read/write filesets can be mounted and accessed via this type of mount point.

It is important to understand the algorithm that the Cache Manager applies to locate any specific filesets in order to implement correct replication mount point structures.

The Cache Manager running in each DFS client system interprets path names. When it encounters a fileset mount point, it looks up information about the fileset in the FLDB, starting at the root mount point. As long as the Cache Manager finds mount points for which a read-only fileset exists down the path to the requested fileset, it accesses read-only filesets only. If a read-only fileset does not exist, it accesses the read/write fileset instead. Once it encounters a read/write fileset that is not replicated, any underlying mount points will also access the read/write fileset even if a read-only fileset exists. Once it finds an explicit mount point, it only accesses read/write filesets, even if the underlying mount points are regular mount points associated with replicated filesets.

Two types of replication are available with the DCE LFS: *release replication* and *scheduled replication*. Using release replication, also know as manual replication, read-write fileset are copied only on request to the replica server(s). This type of replication is useful either if the fileset seldom changes or if you need to closely monitor the replication process.

With scheduled replication, or automatic replication, you specify replication parameters that dictate how often DFS is to automatically update replicated filesets with new versions of source filesets. This type of replication is useful if you prefer to automate the process and do not need to track exactly when releases are made. An immediate update of the read-only copies of a source fileset can be requested at any time with the `fts update` command.

Both types of replication produce the same results. The source filesets are copied to different server machines.

DFS automatically tracks every fileset location, even when the fileset is moved between aggregates or machines. The location of any fileset is automatically maintained by the Fileset Location Database (FLDB).

Even though any fileset can be replicated, we have to be careful in what we replicate. If the type of access is read-only, replication works good; filesets with frequent, or mostly write accesses, should not be replicated. They would cause a lot of additional network traffic because their replica would have to be updated frequently. Since updates can only be performed on the read/write fileset, the File Server would be even more loaded if it had to constantly update all the read-only filesets.

3.8 Lab Testing and Verification

Some tests have been conducted in order to have a rough input for network layout and utilization planning. The test results are not included in this book because they are difficult to apply to environments other than the specific test installation. This is mostly due to the various ways DFS was being used. The tests included simple file transfers for small and large files, which represents sequential file access. Applications may use DFS with random access and even with file locking.

The basic tests have shown that:

- The overhead in terms of total traffic in relation to user data is higher when small files are transferred. For large files, such as a 1 MB file, the overhead was only about 22 percent, including all network and protocol headers.

- DFS caching is very efficient. After a file from within DFS has been accessed once, there was only little network traffic on subsequent reads, provided the file has not been changed in the meantime.
- There is no relevant difference between AIX and OS/2 DFS clients in regard to the network traffic.
- Accessing files in DFS periodically involves the Security Server and thus does generate some traffic between the client and the Security Server.
- The FLDB Servers must be able to communicate with each other for mutual synchronization. The network layout and configuration must allow this type of communication with an adequate speed.
- The DFS client appears to be communicating with all FLDB Servers in the cell. However, this can be changed through configuration if a certain FLDB Server is to be favored over others, see 7.2.5, “DFS FLDB Servers” on page 146.

Chapter 4. Tools and Methods for Performance Evaluation

Besides architectural and theoretical decisions that have to be made from the overall DCE cell design down to detailed machine configurations, a system administrator or a trouble shooter requires adequate tools that he/she can use in the daily business. Such tools support him/her in gathering and collecting performance-relevant metrics, and they can also supply the necessary information for problem determination and analysis.

There are a number of tools available, either as part of the operating systems, as part of the products, or provided by third parties. Different tools can often be used for the same purpose, and it is up to the system administrators to choose the ones they like most.

This chapter gives an overview over a variety of such tools, from low-level operating system trace and report tools through specialized DCE analysis products.

Most tools serve two purposes: They provide information that can be used in either performance evaluation and tuning, or they can be used in a problem determination and solution process. Although the problem determination and solution process is the subject of the second part of this book, the tools are described here in this first part. References added in the problem determination part will point to these tools.

4.1 DCE Tools

The DCE and DSS products on AIX and OS/2 Warp come with several utilities that can be used for performance evaluation or problem determination. The most important utility, although not specific to the tasks of performance tuning or problem determination, is the `dcecp` command. It is the general purpose command to use for inquiring about configuration details and operational status. Besides `dcecp`, which was introduced with OSF DCE 1.1, the former DCE administration commands, such as `cdscp`, `rpccp`, or `rgy_edit`, can still be used in Versions 2.1 of DCE for AIX and OS/2 Warp. There is, however, no guarantee that these commands will still be supported in future versions of the products.

Since these standard commands are widely known and well documented in either printed or online manuals, we do not cover them in this chapter.

The following sections describe some utilities that are shipped as part of DCE and DFS.

4.1.1 DFS Server Monitoring with SCOUT

The `scout` utility is included as part of the DFS base server fileset on AIX (dce.dfs_server.rte). It can be run from any machine within the cell, including clients, and it monitors and reports DFS Fileset Server statistics at regular, selectable intervals.

Figure 19 on page 64 shows a sample output screen from scout. The command line invocation of scout for this example was:

```
# scout -server dfsfil1 dfsfil2 dfsfil3
```

Three DFS Fileset Servers are monitored. The scout utility reports per server:

Conn The number of currently open connections with principals (users or machines)

Fetch The number of file fetch requests sent to this server since server startup

Store The number of file store requests sent to this server since server startup

Ws The number of clients that have been communicating with this server within the last 15 minutes

Server The Fileset Server name being checked with scout

Disk attn The free disk space per aggregate on this Fileset Server, reported in kilobytes

```
                                    Scout

Conn     Fetch       Store     Ws       Server       Disk attn: > 95% used
----  ----------  ----------  ----  ---------------  ----------------
 294      116705      102240   279      dfsfil1       lfs1  1098723 -
                                                     lfs2  3374278 -
                                                     lfs3  4571320 -
                                                     lfs4  1775639 -
                                                     lfs5   413813 -
                                                     lfs6  4823216 -
                                                     lfs7  3581571 -
1527       45759       76367  1635      dfsfil2      *rom0        0 -
                                                    *rom1        0 -
                                                     lfs1  2860935 -
                                                     lfs2  3370124 -
                                                     lfs3  3084901 -
                                                     lfs4  3800194 -
                                                     lfs5  3478476 -
                                                     lfs6  2055724 -
                                                     lfs7  1233084 -
                                                     lfs8  2734026 -
                                                     lfs9  2795642 -
1430       30075       33660  1615      dfsfil3      *rom1        0 -
                                                     lfs1  2657095 -
                                                     lfs2  1798875 -
                                                     lfs3  2080479 -
                                                     lfs4  2178749 -
                                                     lfs5  1856563 -
                                                     lfs6  1852822 -
                                                     lfs7  1065852 -
                                                     lfs8  1562717 -
                                                     lfs9  3285964 -

Probe 15 results
```

Figure 19. Sample scout Output Screen

On the output screen, scout turns those aggregates to inverse video that have a utilization of 95 percent or more. This is a default threshold value. Other threshold values can be set for the other reported figures, and scout will highlight the values as soon as the thresholds are exceeded. The scout utility can ideally be used to determine load balance between different Fileset Servers and to monitor aggregate utilization.

The scout utility is described in the *DFS Administration Guide and Reference*.

4.1.2 DFS Tracing with DFSTRACE

The dfstrace tracing tool is part of the DFS client code (dce.client.dfs.rte fileset on AIX, included in DFS client for OS/2 Warp), and it allows you to trace DFS internal activities either on the client or on the server side. The operation principle of dfstrace is that it traces and stores information about selected events and later dumps this information into a file on request. There is a default log known as *DFS syslog* where significant events, such as loss of communication to DFS servers and unexpected errors, are logged. When problems are suspected on DFS clients and/or servers, it is a good idea to dump this log to a file and review the information. This is done with the command dfstrace dump -file <filename>. When working with IBM customer support, this is typically the information you will be asked for to collect. You can also check the error messages found in this log with the appendix of the DFS Administration Guide and Reference (softcopy documentation shipped with the product).

As with any trace tools, dfstrace can produce a large amount of output data if additional logs are activated; thus its use should be limited to those events that are in question, and it should only be run when necessary.

To see a list of events that dfstrace can trace on the local machine, depending on its configuration, issue the dfstrace lsset. This gives a small list on a DFS client, but lists many separate event sets on a DFS server.

The following is a short example of what information dfstrace can provide. The example is a DFS File Exporter trace taken on a Fileset Server when a file was copied into the DFS filespace. The commands in this example were:

```
# dfstrace setset -set fx -active
# cp /tmp/new_file /:/data/new_file
# dfstrace dump -set fx -file /tmp/fx.log
# dfstrace setset -set fx -dormant
```

This created the formatted trace file, /tmp/fx.log, with the following contents (some lines have been spilled for better readability):

```
DFS Trace Dump -

   Date: Tue Feb 18 11:07:11 1997

Found 1 logs.

Contents of log cmfx:
time 416.669660, pid 16004: PX GetTime
time 416.669735, pid 16004: PX GetTime returns 0
time 421.077941, pid 16004: PX in LookupRoot Volume 0.1
time 421.078809, pid 16004: PX LookupRoot returning Vnode 1, Unique 1, code 0
time 421.083604, pid 16004: PX in LookupRoot Volume 0.4
time 421.084322, pid 16004: PX LookupRoot returning Vnode 1, Unique 1, code 0
time 421.089474, pid 16004: PX Lookup 1.4.1.1/new_file, flags 0x1
time 421.089653, pid 16004: PX Lookup fid 1.4.1.1 maps to volp 0xa539400, vp
                            0xa3e7208
time 421.089927, pid 16004: PX Lookup returns fid 2ff22a14.1.0.a542600 (tid
                            804398904.512949000), code 2
time 421.097888, pid 16004: PX CreateFile dir cff84e21.4.1.1, name new_file,
                            mask 0x19
time 421.098263, pid 16004: PX Lookup fid 1.4.1.1 maps to volp 0xa539400, vp
```

```
                              0xa3e7208
time 421.101980, pid 16004: PX CreateFile returns fid 1.4.7.36, new token ID
                              855793182.2485, code 0
time 421.113580, pid 17552: PX StoreData fid 1.4.7.36, mask 41, position 0,
                              length 10000
time 421.113896, pid 17552: PX Lookup fid 1.4.7.36 maps to volp 0xa539400, vp
                              0xa3f71f8
time 421.114283, pid 17552: PX piping data for length 10000
time 421.129677, pid 17552: PX end piping data
time 421.130581, pid 17552: PX StoreData returns 0

DFS Trace Dump - Completed
```

A trace on the Cache Manager (event set cm) for the same file operation example would have generated much more trace information.

A detailed description of dfstrace, its use and capabilities can be found in the *DFS Administration Guide and Reference* manuals for AIX and OS/2.

4.1.3 Using UDEBUG

The udebug utility is part of the DFS base server code (fileset dce.dfs_server.rte on AIX). It allows you to display ubik status information relevant to a specific DFS server. The following example shows the various information pertinent to a server, specified on the command line:

```
# udebug -rpcgroup /.:/fs -server dfsfil1 -long
Host 192.168.2.2, his time is -2
Vote: Last yes vote for 192.168.2.2 at -3 (sync site); Last vote started at -3
Local db version is 856087625.6067
I am sync site until 86 (3 servers)
Recovery state 1f
Sync site's db version is 856087625.6067
0 locked pages, 0 of them for write
This server last became sync site at -190570

Server 192.168.2.3: (db 856087625.6067)
    last vote rcvd at -5, last beacon sent at -4, last vote was yes
    dbcurrent=1, up=1 beaconSince=1

Server 192.168.2.4: (db 856087625.6067)
    last vote rcvd at -4, last beacon sent at -4, last vote was yes
    dbcurrent=1, up=1 beaconSince=1
```

The example lists server statistics about an FLDB Server that is the master of a total of three FLDB Servers. Additional information displayed includes the database versions of each server and whether or not this server is or was the master, also called *sync site*.

The udebug utility does not provide performance-relevant information, but it can be used to track the voting process and to determine which server has been elected by the ubik process to be the master.

4.1.4 Monitoring DCE Server Activity Using RPC Statistics

The DCE RPC interface provides functions to retrieve statistics of a DCE RPC server. Each DCE runtime instance keeps track of internal counters that contain the following information:

`rpc_c_stats_calls_in`	The number of remote procedure calls received by the runtime instance
`rpc_c_stats_calls_out`	The number of remote procedure calls initiated by the runtime instance
`rpc_c_stats_pkts_in`	The number of network packets received by the runtime instance
`rpc_c_stats_pkts_out`	The number of packets sent by the runtime instance

The information can be retrieved with the OS/2 Graphical User Interface (see 4.3.1, “Using the OS/2 DCE Graphical User Interface for RPC Statistics” on page 70), by DCE Manager for NetView (see 4.4, “IBM DCE Manager for AIX” on page 77), or by creating a small DCE application.

The information provided can be used to estimate the load on the DCE core servers. The actual numbers retrieved from servers are usually less important than their changes over a certain period of time. Or, they can also be used to compare the utilization of two or more servers.

For example, assuming a DCE Master Security Server and two Replica Security Servers are being used, the load on all servers should be about equal when they are placed symmetrically in a campus-type network (but the Master will have slightly more load due to replication traffic). If this is not the case, there is either a lot of update traffic, which must be performed by the Master Security Server, or if the traffic to the Security Servers are read-only, there could be connectivity problems on the network. This can be caused by the network itself or by the method the clients use to select a Security Server (see 7.2.1, “Security Service, the pe_site File” on page 142).

The values provided by the RPC statistics are counter values. For analysis purposes, a calculation of delta values is useful. This is done by substraction of an earlier value from the current value to find out what the statistics for this period of time are.

The DCE runtime RPC statistics can also be retrieved by using the `rpc_mgmt_inq_stats()` function. By default, the DCE RPC runtime allows any DCE client to call the `rpc_mgmt_inq_stats()` function as long as the DCE client is able to retrieve the binding information. To restrict access to the `rpc_mgmt_inq_stats()` function, the DCE server application must supply an authorization routine.

A DCE application can use the RPC statistics information to determine which server to select. For example, if a DCE server performs nested DCE RPCs (see 8.7, “Using Nested RPCs for Server Load Distribution” on page 164), the RPC server doing the distribution of the DCE RPCs can use the information gathered from the RPC statistics and determine which RPC server should perform the next operation.

4.2 AIX Tools and Utilities

AIX provides many low-level tools that, in many cases, provide fast and simple help for a system administrator to overlook a certain situation. It is very common to use such tools, incorporated in shell scripts, as a base for further system monitoring and performance-data collection.

It is beyond the scope of this book to describe the many options for all of these tools, but they are well documented in the standard documentation for AIX. There are other tools available with AIX or from third parties, including public domain and shareware libraries; we will only mention the most commonly and widely used on the AIX platform here.

ps: The ps command is a very simple yet powerful tool that gives you a status of the currently running processes and their use of resources. It is used mostly as a first command to check whether a certain process is running or not. With the correct options set, ps can provide very detailed information about processes, like:

- Their memory usage
- Their priority
- The time when a process has started
- The parent/child relationship between processes
- The amount of CPU time a process has already consumed

Although ps is more often used for problem determination, it provides valuable information about the resource consumption of a process for performance evaluation and tuning.

vmstat: The vmstat utility provides fast and reliable information about the system's resource usage. It lets an administrator know immediately if a system does excessive paging (indicating either a short-of-memory situation or a potential memory leak within an application) or what the CPU utilization is. Other information provided by vmstat is the process run queue situation and the file input/output rates.

Administrators often run vmstat as an alternative to graphical system activity reporting utilities in order to detect short-of-memory situations or looping processes, causing the CPU to be busy (100 percent) all the time.

The vmstat utility is part of the *Accounting Services* fileset (bos.acct) on AIX 4.x.

iostat: Similar to vmstat, the iostat utility reports resource utilization with a focus on disk input/output. When run in continuous mode, it reports the amount of data read and written from and to every disk in the system, along with some other statistical data about the system's resources, like CPU utilization.

The iostat utility is part of the *Accounting Services* fileset (bos.acct) on AIX 4.x.

netstat: The netstat utility has a variety of options. It can be used to diagnose TCP/IP port level connections, as well as to gather statistical information about a system's network interfaces. When used with the -s, -v, or -m parameters, netstat provides information about the network interface utilization as well as error-counter values that indicate a possible network problem or a shortage of requested system resources. For performance evaluation, network interface utilization is an important factor for servers. The netstat utility is usually also

used as a first aid for verifying correct system settings when suspecting network routing problems.

netpmon: The `netpmon` tool collects relevant information during a period of time and provides a very detailed report about the CPU and interface utilization of the different processes. It can ideally be used during application verification and test to determine whether an application is CPU or I/O bound.

`netpmon` is part of the Performance Analysis and Control Commands fileset (perfagent.tools) on AIX 4.x.

iptrace and ipreport: The `iptrace` is an easy-to-use tracing facility for networks. It can be run with filtering options and is therefore capable of capturing network traffic between two systems only or all network traffic through a specified interface. Data collected by `iptrace` is stored in a file until it is stopped. The `ipreport` utility then interprets this data and generates a readable report by interpreting the captured network traffic.

`iptrace` and `ipreport` are part of the *TCP/IP Server* fileset (bos.net.tcp.server) on AIX 4.x.

Performance Toolbox for AIX (PTX): The Performance Toolbox for AIX (also called PTX, or xmperf) is a separately orderable product and is therefore not included in AIX. It is a very powerful collection of graphically oriented tools that includes most of the functionality of the other tools discussed in this section. It offers a large variety of different, standard and customizable presentation options, like bar graphs, gauges, and pie charts. Many PTX features are especially geared for performance analysis and evaluation for systems and applications.

Application instrumentation, for example, as described in 4.9, “Application Instrumentation” on page 86, makes use of the capabilities of PTX. Almost any performance-relevant data, such as CPU utilization, process resource utilization, input/output rates, and so on, can be monitored and analyzed with PTX.

A major advantage of PTX is its ability to collect and display performance data from multiple hosts at the same time. PTX comprises a manager part and a client part. The manager part of PTX collects data from one or more client parts and displays it in an appropriate representation. This way, several DCE servers, for example, can be monitored through one PTX manager.

4.3 OS/2 Tools and Utilities

OS/2 has tools available for debugging and tracing of events. Tools related to performance are not available in the base operating system, except for OS/2 API tracing.

The Multi Protocol Transport Services (MPTS) provides a number of tools that perform traces on MPTS-related components. These tools are:

- NetBIOS applets, to analyze traces and obtain capacity parameters
- TCP/IP `netstat`, `iptrace` and `ipformat`
- LAN Adapter and protocol system dump program (LAPSDUMP), to perform detailed LAN adapter and protocol snapshot analysis

To perform detailed or none intrusive performance evaluation of one or more components within a system, supplemental tools must be used.

4.3.1 Using the OS/2 DCE Graphical User Interface for RPC Statistics

The OS/2 DCE Graphical User Interface can be used to query the RPC statistics for the DCE core servers (see 4.1.4, “Monitoring DCE Server Activity Using RPC Statistics” on page 67). Other RPC severs have the RPC statistics information available, but to obtain the information, a binding handle must be retrieved. This is prepared for the DCE core services.

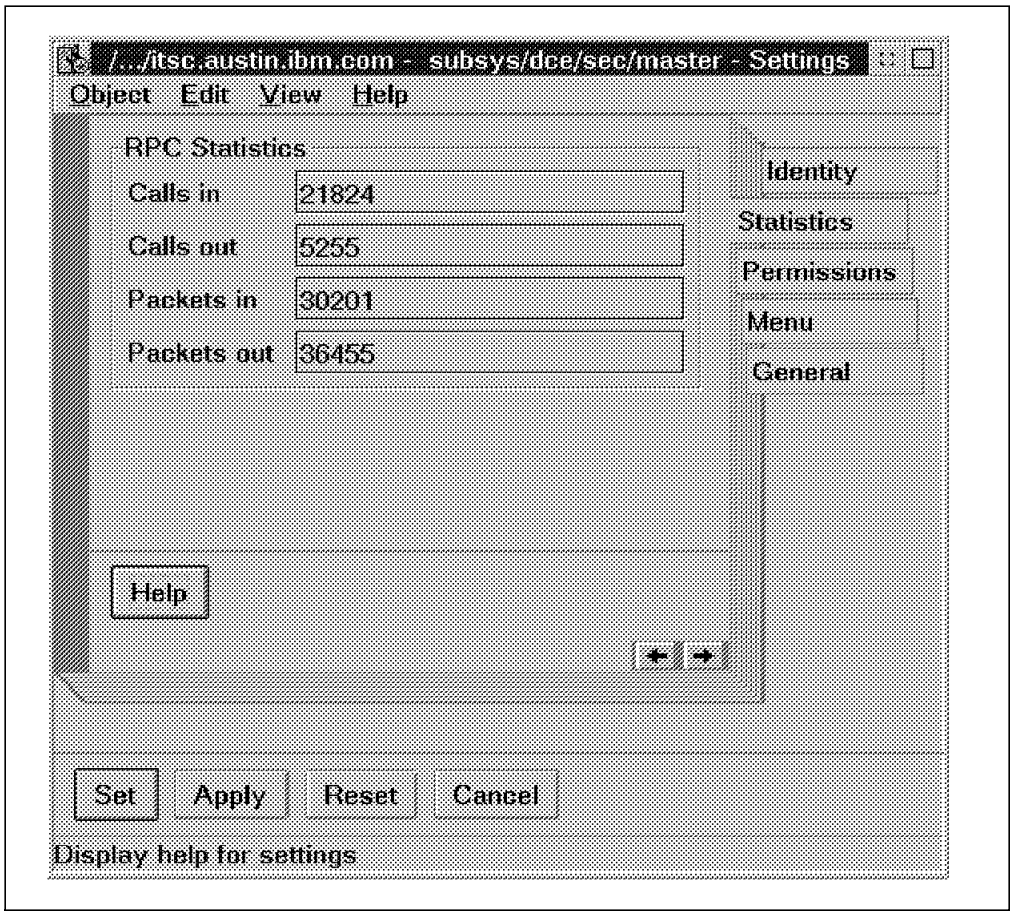


Figure 20. Example of OS/2 DCE GUI RPC Statistics

The information can be used to compare the load of similar or identical DCE RPC servers. If, for example, two replicated DCE RPC server machines are intended to balance the client request load, the administrator would retrieve RPC statistics information in two folders, each representing the RPC server. By refreshing the information from time to time, the skew of the intended balanced load could be investigated.

4.3.2 System Performance Monitor/2

The IBM System Performance Monitor/2 Version 2.0 (SPM/2) is an application that enables access to a collection of hardware and software performance values (performance metrics) in an OS/2 32-bit environment.

SPM/2 is divided into:

- A graphing and logging tool used to perform collection of performance metrics that can be used for report generation and replay for further analysis
- A memory analyzer (THESEUS2) that contains SPM/2 functions that can assist and validate specific problems. THESEUS2 grants access to application memory, OS/2 kernel and OS/2 subsystem memory. THESEUS2 has the capability to format OS/2 data structures into understandable wording
- A remote data collection facility that can be used to collect performance metrics for many different systems at one time
- An application programming interface that allows additional application specific performance metrics to be added

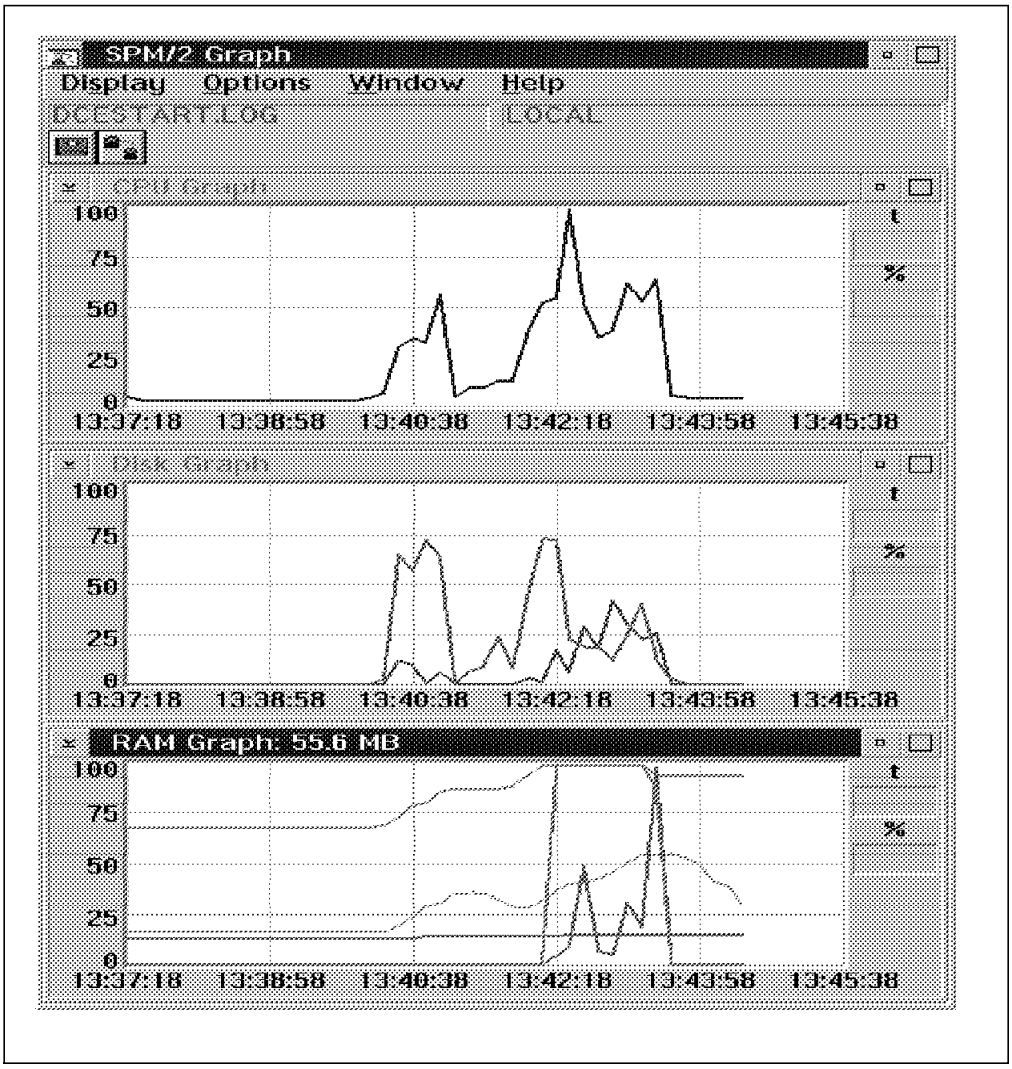


Figure 21. SPM/2 Graphing of DCESTART

The SPM/2 graphing window displays a real-time picture of a reduced number of performance metrics. Although the performance metrics selected are much more extensive, the SPM/2 graphing is limited to the following performance metrics:

- CPU utilization
- Disk activity
- Memory, which is divided into

- Used memory
- Working memory set
- Resident memory
- Number of pages paged in (SWAPPER.DAT)
- Number of pages paged out (SWAPPER.DAT)

The collection of performance metrics can be performed for every process running in the system. This allows for very detailed analysis on a per process and OS/2 thread level. Collecting performance metrics affects the system under test because SPM/2 will require approximately 10 percent of the CPU resources available. Memory and disk activity will also be affected. To get more reliable values, a performance measurement should not collect more than 5 to 10 performance metrics at one time.

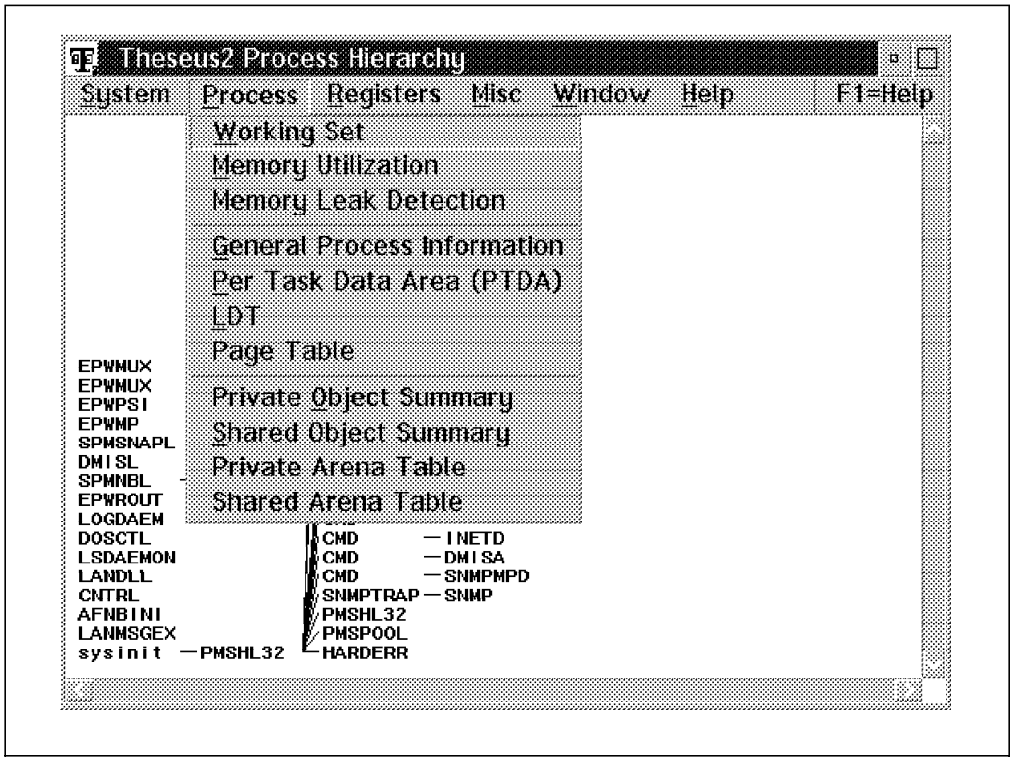


Figure 22. THESEUS2 Preparing to Obtain Working Memory Set

Using THESEUS2, a detailed analysis of memory-related resources can be performed. Features like Memory Working Set and Memory Leak Detection are often used by application programmers.

The other THESEUS2 features are OS/2 kernel related and often used by system programmers for in-depth analysis.

The Internal OS/2 Data Structures are described, for example, in the redbook series *The OS/2 Debugging Handbook*, SG24-4640 through SG24-4643.

4.3.3 TME 10 NetFinity Server Version 4.0

TME 10 NetFinity Server Version 4.0 (NetFinity Server) provides a range of functions to manage individual and groups of systems. NetFinity Server provides the following main functions:

- **Inventory functions**, which are divided into:
  - Hardware configuration and installed software
  - Workstation and user information
  - Retrieving information using Desktop Management Task Force (DMTF) specification of Desktop Management Interface (DMI)
- **Monitoring functions**, which are divided into:
  - Usage monitoring of memory, CPU, disk and instrumentation counters of applications, and selected TCP/IP traffic counters
  - Monitoring critical files and devices
  - Setting threshold conditions for monitored resources and define notification and other actions when a threshold limit is exceeded
- **Remote Workstation Control**, which includes taking control of keyboard and mouse of a remote workstation
- **Software Distribution**, which preforms preparation and distribution of data and software packages
- **License Control**, which tracks the usage of licensed software on NetFinity Server-managed systems
- **Scheduling events**, which activates services at specified times

Using NetFinity Server, the administrator can remotely manage the setup for each managed system.

Figure 23 on page 74 shows a CPU utilization chart of a managed system. From the CPU utilization, the threshold values are added. Whenever a threshold is exceeded, a notification is sent to the Alert Manager.

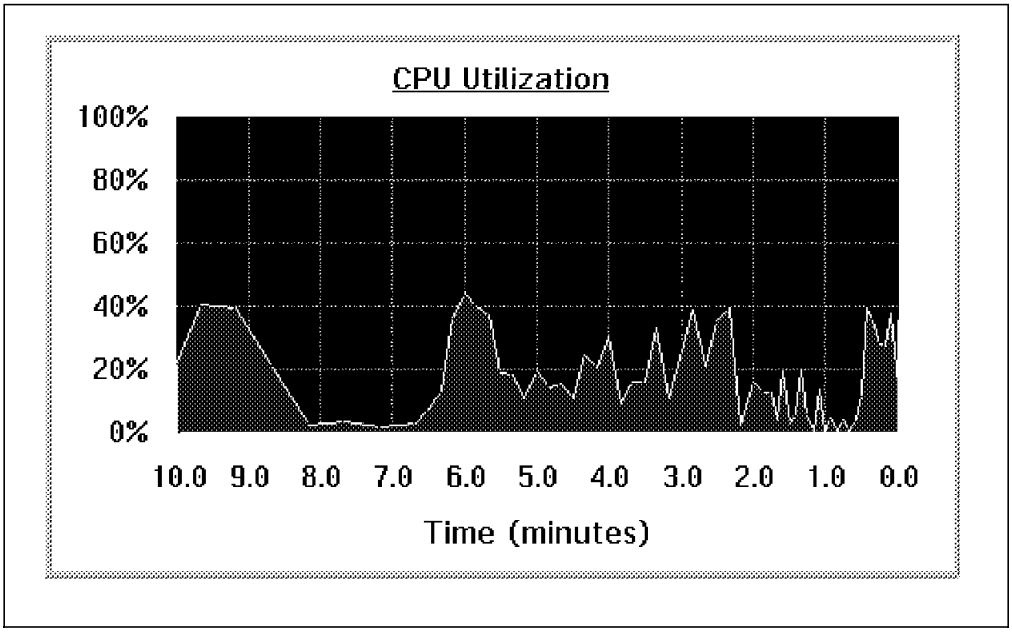


Figure 23. NetFinity Server Monitoring CPU Utilization of a Remote Workstation

Alert actions are configured in the Alert Editor.

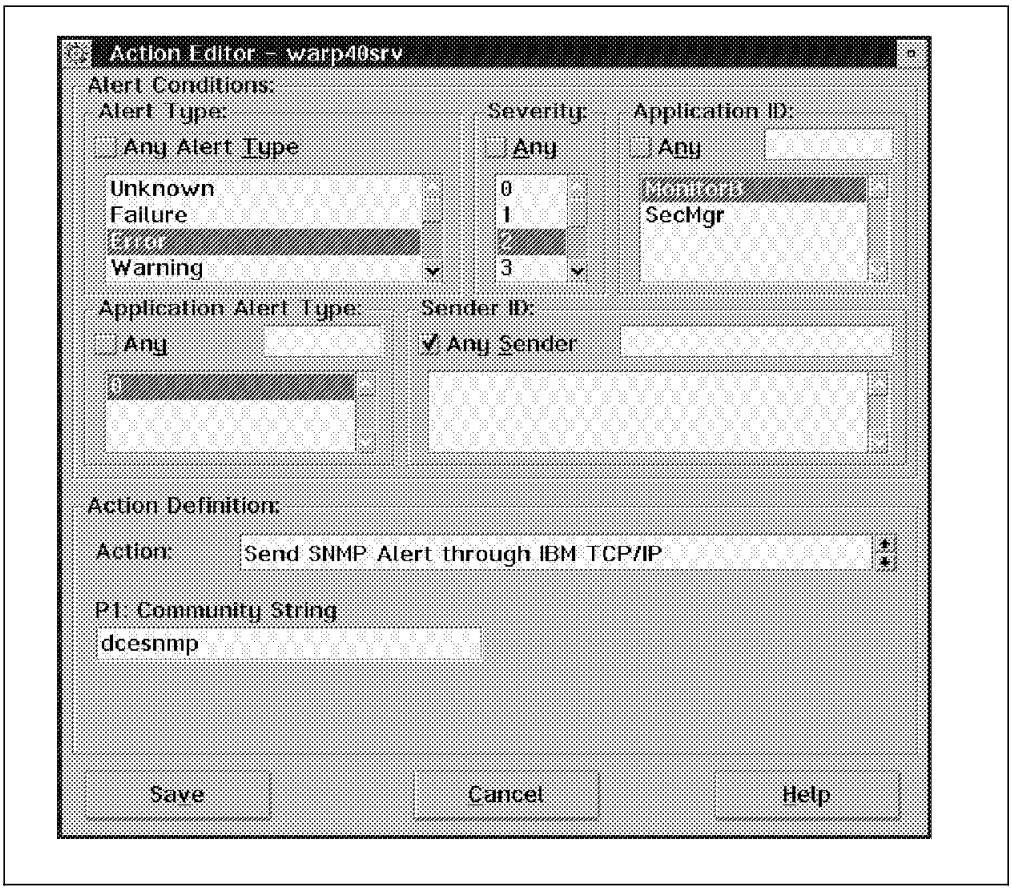


Figure 24. Setting Threshold in NetFinity Server

In the Alert Editor, events can be specified as in the example shown in Figure 24.

NetFinity Server can be used to monitor DCE servers with automatic alert generation when specified thresholds are exceeded.

4.3.4 IBM SystemView Agent and DCE SNMP

The IBM SystemView Agent is the OS/2 framework allowing a Simple Network Management Protocol (SNMP) manager to perform SNMP requests into an OS/2 system. Together with the DCE SNMP agent, it is possible to collect and monitor vital status information together with updated counter values

The IBM SystemView Agent does not by itself display values, but requires a SNMP Manager to retrieve the values.

4.3.5 IBM DatagLANce

IBM DatagLANce is a LAN protocol analyzer product, available for OS/2. It can monitor both, token-ring or Ethernet networks. It allows a network specialist to perform:

- Data capturing, by selecting which traffic to capture (all or by filter)
- Data analysis, by performing protocol decoding together with detailed frame analysis
- Statistical history, by recording statics over selectable time intervals
- Alarms, by creating an alarm log entry or calling a pager when a preset threshold is exceed or by issuing SNMP TRAPs whenever one or more preset thresholds are exceeded
- Traffic statistics, by tracking the traffic each station is generating
- Network glance, by allowing to take a real-time look into LAN frames or take a look into frames while frames are captured
- Traffic generation and playback, by transmitting a single frame into the network under different load conditions and by playing back captured frames into DatagLANce network analyzer functions

IBM DatagLANce captures frames on the LAN and will not interfere with the systems that are being traced.

Although DatagLANce is not specific to DCE, it can be used to monitor network unitization of servers without running an additional tool on the servers themselves.

4.3.6 Monitoring OS/2 DSS File and Print Server

The performance of an OS/2 DSS File and Print Server can be monitored by evaluating the LAN Server component separately from the installed DCE components.

The capacity usage can be determined by using the `net` commands or the graphical user interface. The values for shared resources and file locks can be determined by repeated calls to the `net files` and `net share` commands. The upper limits are set within the IBMLAN.INI control file. Other capacity-related parameters may also need to be adjusted.

The response time of a OS/2 LAN Server can be obtained by using the `net statistics` command. The mean response time is an average of the time the

server needs to process Server Message Block (SMB) requests. For smoothly performing systems, the value is very low (below 1 ms).

Obtaining the mean response time over a long period (weeks or months) will give an indication of the performance of an OS/2 LAN Server. If the mean response time is increasing, there may be a performance problem coming or already in place.

Another method of performance monitoring is to make use of established test scenarios that represent the work load an OS/2 LAN Server will perform. Repeated use of the test scenarios can be used to see a trend in the OS/2 LAN Server performance over time.

4.3.6.1 DSS Tuning Assistant

Adjusting the OS/2 LAN Server capacity and performance parameters can, for example, be made by using the DSS Tuning Assistant. It will make use of input parameters and a number of assumptions. With this in place, the DSS Tuning Assistant suggests changes to the system configuration files. The suggested changes are optimized to OS/2 Warp Server and are not suited for the DSS environment because the DCE components have a requirement for memory that is not included in the DSS Tuning Assistant. Therefore, the amount of recommended memory required by the DCE components must be excluded from the Tuning Assistant calculations. This can be done by specifying a memory size parameter to the DSS Tuning Assistant:

```
wstune /M:x
```

where x is the amount of memory in the system.

For example, in a system having 48 MB of memory with DCE components that require 14 MB of memory, the call to DSS Tuning Assistant would be:

```
wstune /M:34
```

This will, in many cases, keep the disk cache parameter in control. The disk cache will be the only variable parameter for the OS/2 LAN Server. For an advanced OS/2 LAN Server, this parameter is set in the HPFS386.INI control file. For the entry OS/2 LAN Server, the `diskcache` parameter is specified in the `config.sys` file.

Setting the `HPFS386` disk-cache parameter too high can cause performance degradation because excessive paging might occur.

The performance evaluation of the OS/2 DCE components is nearly identical to the methods already described. The OS/2 DSS File and Print Server makes use of the Security Server registry and the CDS namespace.

For example, if the DSS client logon takes considerably more time than usual, there may be a performance problem in the DCE components involved. Similarly, a LAN Server resource alias resolution (`net use alias`) may take a long time. If the time is increased from normal, there may be a performance problem related to the CDS because alias information is stored within the CDS namespace.

4.4 IBM DCE Manager for AIX

The IBM DCE Manager for AIX Version 2.1 is a systems-management tool integrated into NetView for AIX and designed to help the system administrator monitor and manage Distributed Computing Environment (DCE) environments.

Because it is integrated in NetView for AIX, it supports the same look and feel of the user interface used in NetView and provides a graphic representation, called maps and submaps, for machines and services.

DCE Manager is capable of managing multiple cells. It adds a new map with associated submaps to the root submap of NetView for AIX. Like other services of NetView for AIX, DCE Manager provides the ability to auto-discover the DCE cell, an easy and convenient way to create the submaps.

The following example shows the DCE cell submap, representing all services available in the DCE cell.

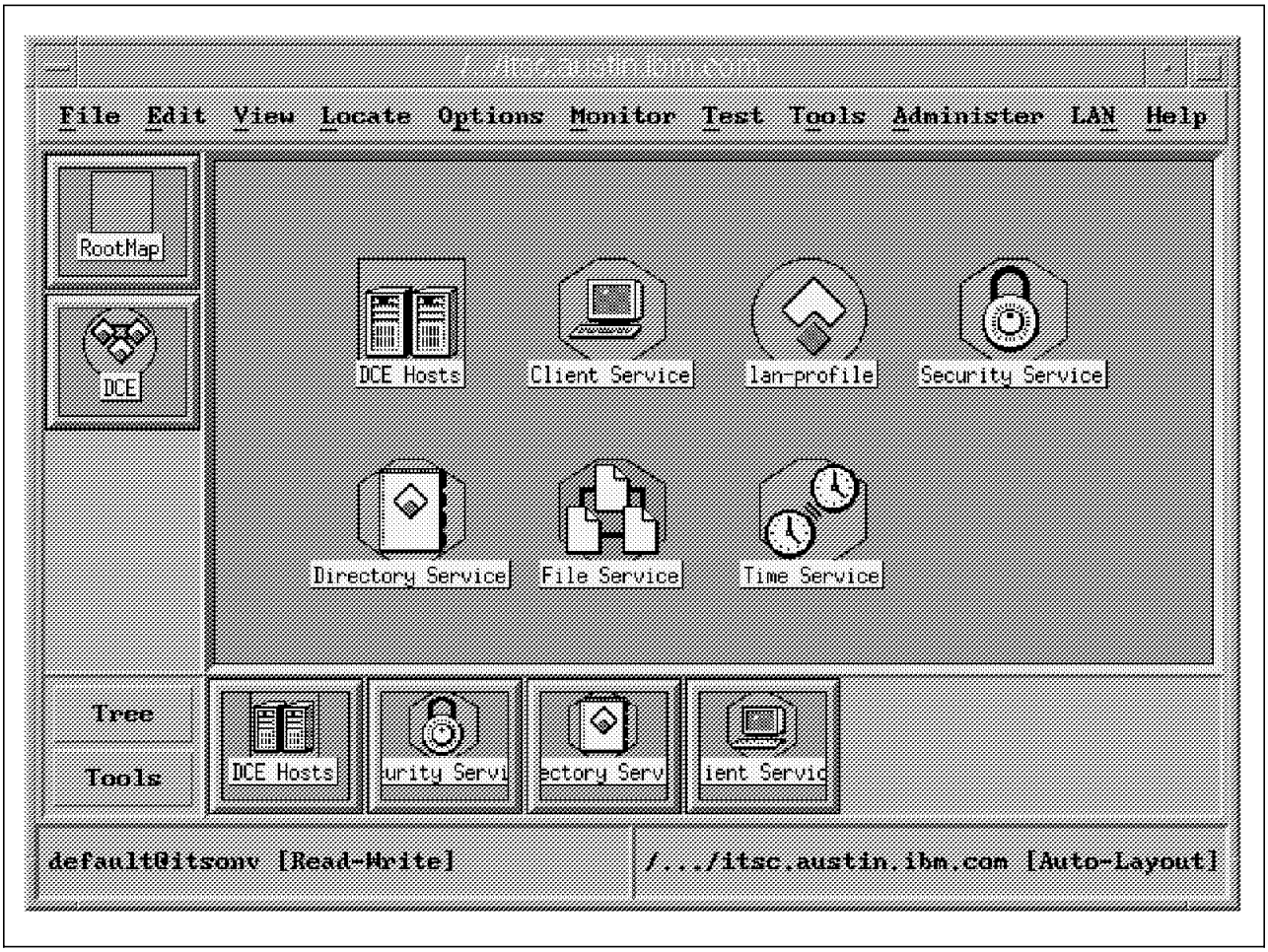


Figure 25. Sample DCE Manager Cell Submap

As with other objects on the various submaps of NetView for AIX, DCE Manager changes the color depending on the status of the objects. This is very helpful to immediately recognize any problems in the cell, especially when servers are being monitored. DCE Manager does a status polling at configurable intervals to update the status (color) of the objects.

DCE Manager supports service-related submaps. This allows you, for example, to monitor all servers of a certain type on a single submap. As an example, the submap in the following Figure 26 on page 78 contains the security server of a cell. In color representation, it is shown in green, indicating healthy operation.

For performance evaluation, especially for tracking the load balancing among servers, DCE Manager displays network and DCE RPC statistics per service on DCE hosts, as the example in the next figure shows. For DFS monitoring, DCE Manager captures and displays several file server statistics.

Since DCE Manager is integrated into NetView for AIX, powerful tools, such as Management Information Base (MIB) browser or network management facilities, support the system administrator in fast problem tracking and determination.

Figure 26. Sample DCE Manager Security Server Submap with Statistics

DCE Manager for AIX Version 2.1 runs on AIX 4.1.3 (or higher) and on NetView for AIX Version 4.1.1 (or higher).

4.5 SMT (IBM Developer Connection CD-ROM)

The SMT (System Monitoring Tool) tool runs on AIX and is available from the IBM Developer Connection CD-ROM. It allows monitoring and analysis of DCE RPC protocol transactions by interpreting network traffic. The SMT tool is distributed into a host workstation SMT application and to one or more remote monitoring agents (RMONs) connected to the network segments on which DCE machines communicate with each another. The documentation that comes with SMT on the CD-ROM also contains the description of how to use an OS/2 RMON agent.

As an alternative to the remote RMON facility, traces taken with AIX's `iptrace` or Sun's `etherfind` facilities can be converted and imported into SMT.

Although SMT is not primarily designed to assist in a performance tuning process, it might help, however, to analyze DCE network traffic in order to optimize applications.

The main SMT window is an X-Windows-based presentation of the DCE RPC traffic that has been captured. Figure 27 on page 80 shows a sample RPC traffic analysis between a client and the Security Server, monitored while a user performed a `dce_login`.

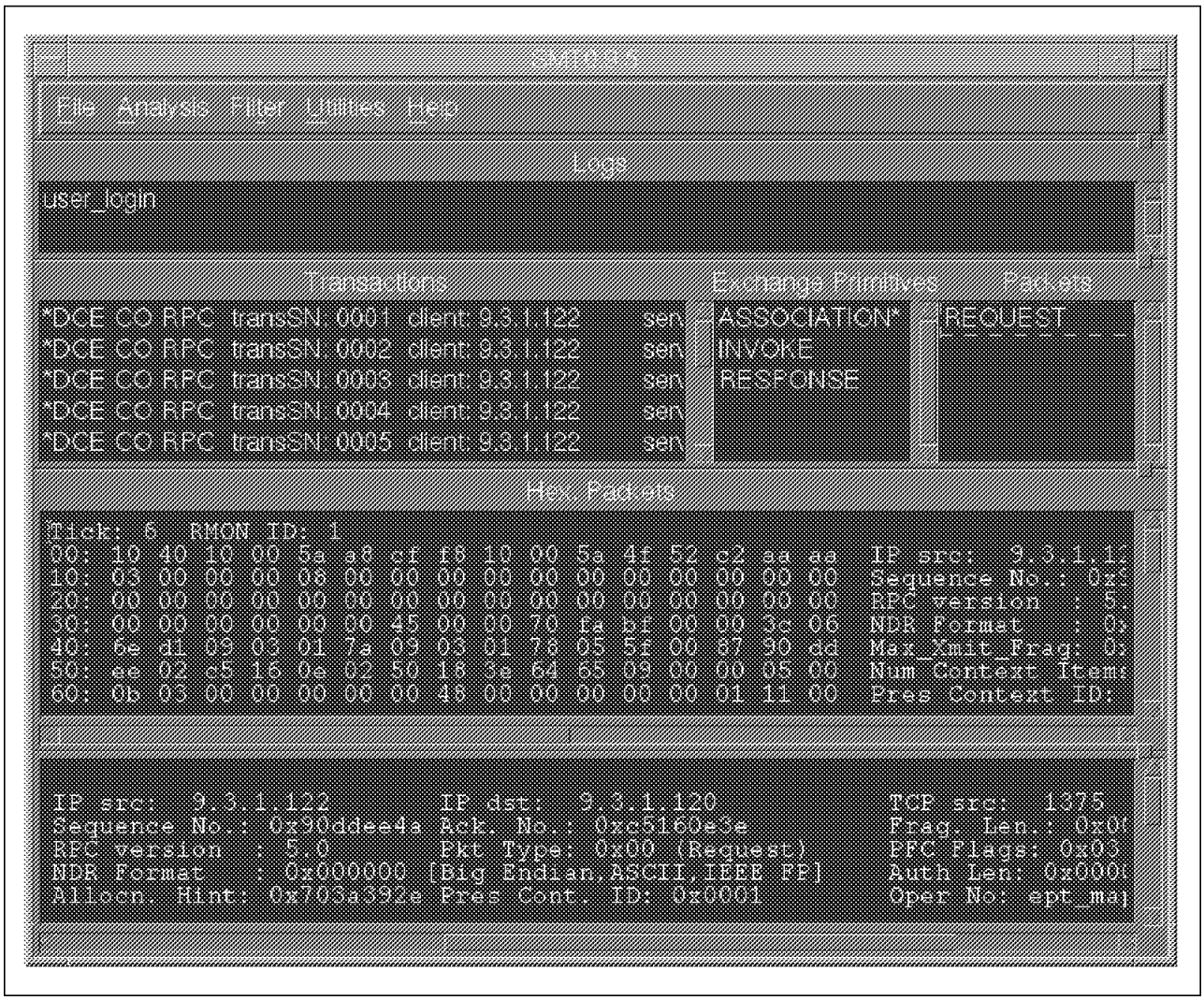


Figure 27. Sample SMT Analysis Window

Through the use of a configuration file, UUID-to-name mappings can be done such that SMT displays readable text instead of long UUID strings. As can be seen in the bottom sub-window of the example in Figure 27, SMT not only displays hex values, but interprets the traffic to a large extent.

SMT is available on the IBM Developer Connection CD-ROM.

4.6 CUBSCOUT (IBM Developer Connection CD-ROM)

The cubscout tool is basically the same as scout (see 4.1.1, “DFS Server Monitoring with SCOUT” on page 63), with two differences:

- It only gives you one snapshot. For example, it does not run and monitor DFS Servers continuously in specified intervals.
- Its output is ASCII text (scout is curses-based); so it can be used and analyzed by shell script commands.

The displayed values are the same as those displayed by scout. The cubscout tool is available on the IBM Developer Connection CD-ROM.

4.7 DCESTAT (IBM Developer Connection CD-ROM)

The dcestat tool runs on AIX and comes with a collection of associated utilities that can be used to poll DCE server status' and gather load data from them. It can be used primarily for supervising DCE server processes, but can also be helpful in collecting performance-related figures.

The basic functionality of dcestat is to do a low-level query to the specified DCE server program using the standard DCE RPC routines rpc_mgmt_inq_stats(), or rpc_mgmt_is_server_listening(). Either a standard DCE acronym, such as secd, or any valid DCE server name, for example /.:/subsys/dce/sec/my_host can be used to specify which server should be checked. A common use of dcestat could be to ″ping″ the DCE servers in a cell at regular intervals to check their availability.

The dcestat tool is available on the IBM Developer Connection CD-ROM.

4.8 DFSSTAT

The dfsstat utility runs on AIX and is a non-supported, use-as-is utility. It will be made available by about mid-year 1997 from the IBM DCE Support Web page at http://service.software.ibm.com/dssdce.

The dfsstat utility is a low-level tool that reports various RPC, DFS and AIX VMM statistics that may be useful in performance analysis, load monitoring, or environment characterization.

Most of the output is very low-level and not needed for normal administration. However, its output can be used for detailed performance analysis. A couple of cases where the output can help are shown.

4.8.1 Usage

Dfsstat uses the following parameters:

dfsstat [-rcs]

r collects DCE RPC data

c collects DFS client data

s collects DFS server data

By default r, c, and s data is collected.

Here is a sample output from dfsstat:

```
# dfsstat

KERNEL RPC
----------
ccalls     scalls     txpkts     rxpkts     retrans    rxdups     oo_pkts
4560       4544       11202      11201      2          0          0
rxfacks    txfacks
741        747
DFS CLIENT
----------
vn_lkups   vn_rdir    vn_gattr   vn_sattr   vn_read    vn_write   vn_map
64787      2461       4278       1009       1419       1280       8
```

```
rd_faults  wr_faults  pr_faults  cachehits  inflight   rd_waits
24         160        0          11         5          22
lookups    fstatus    fdata      readdir    gettokens
86         69         9          215        100
sstatus    sdata      reltokens  revokes
289        245        2052       286
DFS SERVER
----------
lookup      lkuproot    fstatus     sstatus     fdata       sdata
1138 25%    5  0%       78  1%      1037 23%    9  0%       160  3%
readdir     mkdir       rmdir       create      rmfile      rename
215  4%     69  1%      69  1%      376  8%     676 15%     100  2%
link        symlink     fetchacl    storeacl    gettoken    reletoken
100  2%     200  4%     0  0%       0  0%       103  2%     128  2%
sctx        gettime     setparam    bulkkalive  bulkfetchVV totalcalls
11  0%      0  0%       0  0%       0  0%       0  0%       4474
```

4.8.2 Output Description

This section details the various output categories produced by the dfsstat utility.

KERNEL RPC: The KERNEL RPC portion of the output from dfsstat contains these values:

ccalls — The number of RPC client calls made.

scalls — The number of incoming server calls received by the local RPC server. Note that even a client can have scalls since it exports a token revocation server for the DFS File Server to issue token revokes to (DFS RPC calls only).

txpkts — The number of RPC packets transmitted, including retransmits.

rxpkts — The number of RPC packets received, including rxdups and oo_pkts.

retrans — The number of packets retransmitted. This should be compared against txpkts. Ratios less than 1 percent are excellent. Ratios above 5 percent should be investigated. Ratios above 10 percent are undesirable. Look for causes of network packet loss or server overload.

rxdups — The number of duplicate packets received out of the total packets received. Duplicated packets are an indication of network loss of the original sent packet or a loss of an acknowledgment. Investigate possible causes of network loss for rxdups to rxpkts ratios above 10 percent.

oo_pkts — The number of out of order packets received. This can be a sign of network loss or heavy network traffic. UDP does not guarantee sequential delivery of packets. Ratios of oo_pkts to rxpkts above a few percent should probably be investigated.

rx_facks — The number of fragment acknowledgments received. This is associated with DFS data reads and writes that use the RPC pipe mechanism to stream data between the server and client. (For this purpose, you can consider a fragment the same as an RPC packet.)

tx_facks — The number of fragment acknowledgments transmitted. This is associated with DFS data reads and writes that use the RPC pipe mechanism to stream data between the server and client. (For this purpose, you can consider a fragment the same as a RPC packet.)

DFS CLIENT: The following counters are displayed in the DFS CLIENT section of the output:

`vn_lkups` The number of lookup vnode operations performed in DFS. Typically, this relates to system calls that take a file pathname as input. Examples are `open()` and `stat()`.

`vn_rdir` The number of readdir vnode operations performed in DFS.

`vn_gattr` The number of getattr vnode operations performed in DFS. Many system calls that work with files will get file attributes.

`vn_sattr` The number of setattr vnode operations performed in DFS. System calls like `chmod()` and `utimes()` will use this vnode operation.

`vn_read` The number of read vnode operations performed in DFS. This relates to the read system call. Note that mapped file I/O will not use the read system call. AIX uses mapped files for binaries. Also the AIX C compiler and linker makes use of mapped files.

`vn_write` The number of write vnode operations performed in DFS. This relates to the write system call. Note that mapped file stores will not go through the write vnode operation.

`vn_map` The number of map vnode operations. This is related to the `shmat()` and `mmap()` system calls.

`rd_faults` The number of read page faults issued by the AIX Virtual Memory Manager (VMM) to DFS. Note that DFS is integrated with the AIX VMM. DFS creates memory segments for files and then performs data I/O on the segment. This allows DFS to support mapped files and adds a layer of fast “memory” caching above the DFS client cache. When DFS data is not in VM memory, the VMM must fault to a DFS page fault handler that either gets the data from the DFS client cache or retrieves it from a DFS server. One interesting statistic to examine is the ratio of `rd_faults` to `vn_read`, which for some environments can give an indication of how a data working set fits into system memory (RAM). If the ratio of `rd_faults` to `vn_reads` is high, it may indicate that adding system RAM could improve performance. The usage of mapped files must also be taken into account.

`wr_faults` The number of write faults issued by the AIX VMM. This is the VMM calling DFS to give it pages with dirty data so DFS can store it in the DFS cache and possibly to the DFS server as well. The `wr_faults` are driven by `vn_writes`, mapped file stores, and VMM page replacement. If the number of `wr_faults` is significantly greater than the number of `vn_writes`, this may indicate a high amount of page replacement activity due to thrashing. Increasing system memory may result in better performance.

`pr_faults` The number of protection faults issued by the AIX VMM.

`cachehits` The number of `rd_faults` that were serviced by data from the DFS client cache.

`inflight` The number of `rd_faults` where the requested data has already been requested and is currently arriving from a DFS server. Inflight data can result from sequential page faults on the same DFS “chunk”, which is usually several pages in size, or read ahead, which can be triggered by the AIX VMM or the DFS client based on file access patterns.

rd_waits The number of wait loops an rd_fault takes before the requested inflight data has arrived. As data is steaming in from a DFS server, the page fault path will be notified periodically to see if enough data has arrived to satisfy the fault.

lookups The number of file lookup RPCs made to a DFS File Server. This should be compared to the vn_lkups statistic to get an idea of how many lookups are resolved in the DFS client's name lookup cache. In some environments, increasing the name lookup cache with the dfsd -namecachesize option can reduce lookup RPCs. The -stat option should be increased equally with the -namecachesize option. For example: dfsd -namecachesize 2000 -stat 2000. On AIX the default values are based on the amount of system memory, with typical values being around 400 for a 32 MB system. For single user workstations, the defaults are usually sufficient for a modest name cache hit ratio. Multiuser systems may benefit from increased name caches and status caches.

fstatus The number of fetch status RPCs made to a DFS File Server. This call is used to get file attributes. Most DFS RPCs return file attributes with the result. The fetch status RPCs may be made when permissions need to be calculated for a new user accessing a file whose name may already be cached. Increasing the status cache may reduce fetch status RPCs.

fdata The number of fetch data RPCs made to a DFS File Server. Fetch data RPCs are required when requested data is not in the DFS client cache. When data is not in the VMM or in the DFS client cache, then a fetch data RPC must be made to retrieve the data from the DFS File Server.

readdir The number of readdir RPCs made to a DFS File Server. Compare this against vn_rdir. The higher the vn_rdir compared to readdir, the better the cache hit rate.

gettokens The number of gettoken RPCs made to a DFS File Server. Tokens are the internal mechanism DFS uses to maintain cache coherency between clients and servers. Most DFS RPCs return token rights. When there have been data collisions or directory content changes that required revocation of tokens from a client or when a client needs to renew a token that is about to expire, gettoken RPCs are usually required.

sstatus The number of storestatus RPCs made to a DFS File Server. storestatus RPCs are used to store file attributes to the DFS File Server.

sdata The number of storedata RPCs made to a DFS File Server. The storedata RPCs are used to store file data to the DFS File Server.

reltokens The number of internal DFS client-released tokens. This statistic should be ignored.

revokes The number of revoke RPCs the server has sent to the client. When the server has given the right to the file to someone else, or when the ticket has expired, the server revokes the ticket for that file from the client.

DFS SERVER

`lookup` The number of lookup RPCs received by a DFS server.

`lkuproot` The number of lookup root RPCs received by a DFS server. This RPC is made by DFS clients when they first cross a DFS mount point and then periodically there after.

`fstatus` The number of fetch status RPCs

`sstatus` The number of store status RPCs

`fdata` The number of fetch data RPCs

`sdata` The number of store data RPCs

`readdir` The number of read directory RPCs

`mkdir` The number of directory create RPCs

`rmdir` The number of directory remove RPCs

`create` The number of file create RPCs

`rmfile` The number of file remove RPCss

`rename` The number of rename RPCs

`link` The number of hard link RPCs

`symlink` The number of symbolic link RPCs

`fetchacl` The number of fetch ACL RPCs

`storeacl` The number of store ACL RPCs

`gettoken` The number of get token RPCs

`reletoken` The number of release token RPCs

`sctx` The number of set context RPCs received by a DFS server. DFS clients make set context RPCs to set up a “DFS connection” to a file server. A connection represents a DCE principle at a client. Connections may periodically by renewed (normally every four hours) or reactivated when they become stale.

`gettime` The number of get time RPCs received by a DFS server. DFS clients use get time calls as “keep-alives” during periods of idle activity to keep cache coherency state active at DFS File Servers. Normal RPCs act as keep-alives. Idle client systems typically send a keep-alive about every 90 seconds when there are “active” tokens at the client.

`setparam` The number of setparameter RPCs received by a DFS server.

`bulkkalive` The number of bulk keep-alive calls received by a DFS server. Replication servers make this RPC to DFS File Servers that hold the read/write replicas.

`bulkfetchVV` The number of bulk fetch version calls received by a DFS server. Replication servers make this RPC to DFS File Servers that hold the read/write replicas.

`totalcalls` The total number of RPC calls received by a DFS server.

4.8.3 Examples

Note that these examples are based on experience and may not be accurate in all environments. Be flexible with the ratios and always validate the result in your system before starting to make changes.

4.8.3.1 Network Stability

Collect the retrans, txpkts, rxdups, rxpkts, and oo_pkts outputs.

If any of the following ratios are true, you may have a network problem.

```
retrans / txpkts > 0.05
rxdups / rxpkts > 0.1
oo_pkts / rxpkts > 0.03
```

4.8.3.2 System Memory

Collect the rd_faults, vn_read, wr_faults, and vn_writes outputs.

If the number of wr_faults is much larger than the number of vn_writes, an increase of system memory may increase performance.

The ratio of rd_faults to vn_read can give an indication of how a data working set fits into system memory. If the ratio is high, more memory may increase performance. This is a value to check when you are evaluating system performance. You can compare values before and after adding system memory, for example.

4.8.3.3 Cache Hit Rates

Collect the cachehits, rd_faults, lookups, vn_lkups, readdir, and vn_rdir outputs.

cachehits / rd_faults = data cache hit rate

lookups / vn_lkups = hit rate for filenames. If below 80 percent, try to increase the -stat and the -namecache parameters of dfsd.

1 - (readdir / vn_rdir) = hit rate for directory contents

4.9 Application Instrumentation

Application Instrumentation (AI) is available on AIX, and it allows you to trace DCE specific events within application servers and clients. AI makes use of the System Performance Measurement Interface (SPMI) and feeds performance-relevant data through this interface to the agent function of the Performance Toolbox for AIX (PTX). See also “Performance Toolbox for AIX (PTX)” on page 69. The application needs to be recompiled with adequate compiler options in order to function as a Dynamic Data Supplier (DDS) application for PTX. The manager part of PTX is then able to retrieve this data and present it in various ways to the user. Figure 28 on page 87 depicts this data flow between a DDS application and the monitoring PTX manager.

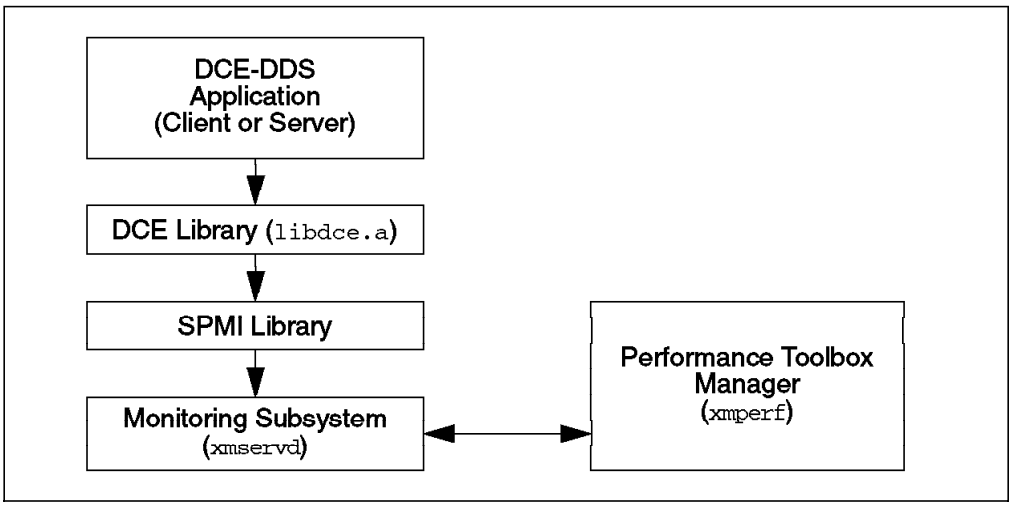


Figure 28. Data Flow for Application Instrumentation

Application Instrumentation allows you to monitor network statistics (TCP and UDP calls), packet counts, marshalling and unmarshalling time, and total elapsed time. DCE server and client applications can be monitored separately.

In order to get AI to work, several steps need to be prepared:

- The application code must perform an initialization call for the monitoring routines. There is an example on how to do this in the sample client and server programs in the /usr/lpp/dce/examples/arithmetic directory. (The DCE example programs are part of the dce.tools.appdev.adt fileset, which must be installed on the system.)
- The application must be compiled for the use of Application Instrumentation.

  This is done with the `-spmi` command line option of the Interface Definition Language (IDL) compiler. Several flags control the level of instrumentation:

  - `c` — Instruments the client portion of the application
  - `s` — Instruments the server portion of the application
  - `e` — Measures the elapsed time of entire RPC calls
  - `m` — Measures the elapsed time marshalling and unmarshalling
  - `q` — Calculates the sum-of-squares time for elapsed times
  - `p` — Measures call counts by protocol

  As an example, an IDL compiler invocation with all flags would be:

  ```
  # idl -spmi csempq my_program.idl
  ```

- By default, the DCE library, `libdce.a`, only contains empty stubs for the additional instrumentation calls. This is to make sure that an application with instrumentation built into it can be run on any system. When instrumentation is to be used, a shared object within this library must be replaced by one that supports monitoring. A description on how this needs to be done can be found in the /usr/lpp/dce/examples/inst/README.AIX file. (These example files are part of the dce.tools.appdev.adt fileset, which must be installed on that system.)
- The monitoring subsystem must be installed. When installed, `xmservd` will automatically be started by `inetd` as soon as the DDS application is started.

Once these prerequisites are in place, the application can be run and its performance data can be monitored using xmperf (the PTX Manager), which does not need to run on the same system because of its remote monitoring facility.

A sample analysis of an application server could look like the following figure.

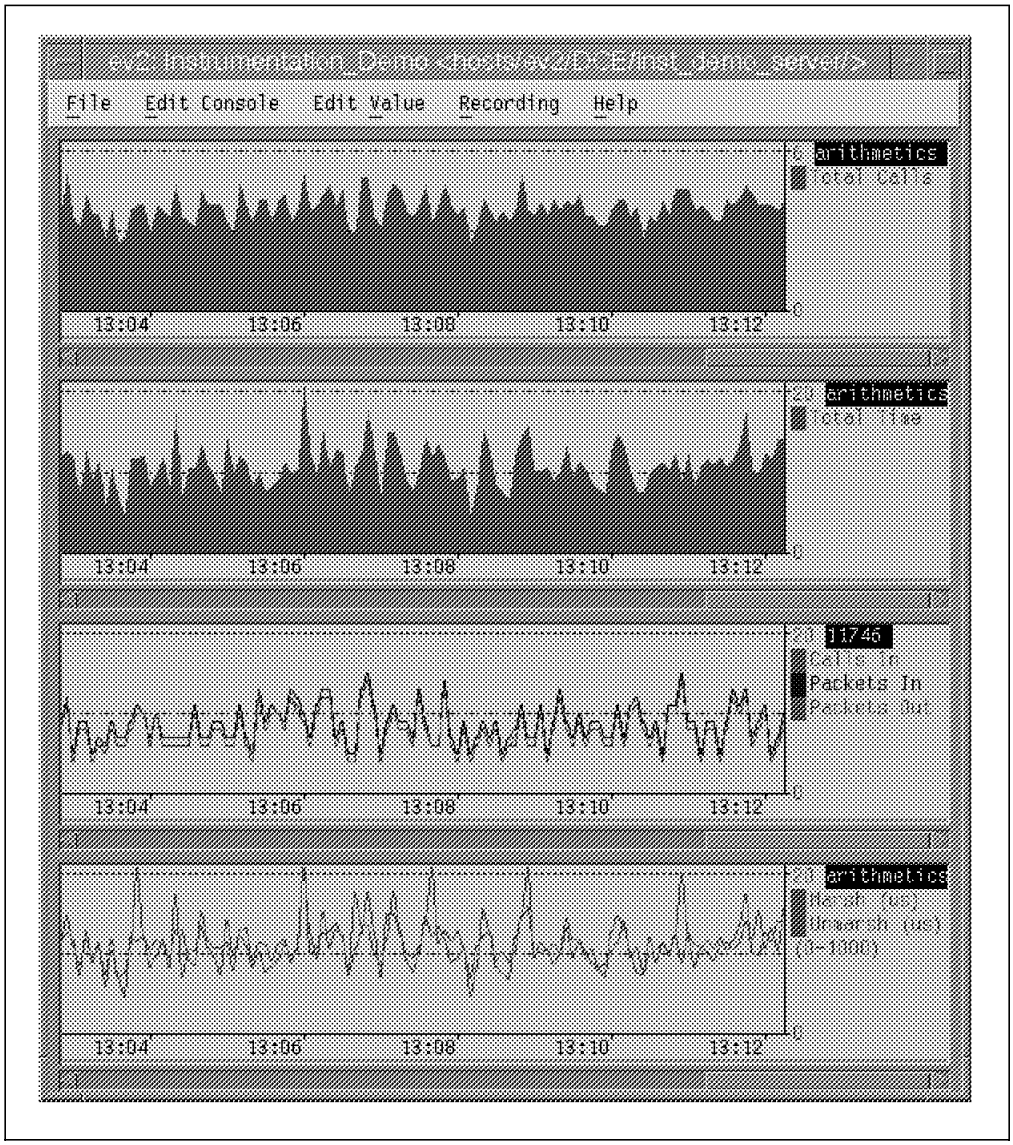


Figure 29. Sample Instrumentation Graphs

There are several ways you can choose to display the data with xmperf. As an example, an instrument graph similar to the one in Figure 29 can be built following these steps:

1. From the xmperf main window, select **Monitor**, and then select **Add New Console**.
2. Type in a name for the console, or accept the default, and select **Proceed**. A new, empty console window appears.
3. In this new window, select **Edit Console**, then select either **Add Local Instrument** or **Add Remote Instrument**, depending on whether your application runs locally or on a remote system. If you have chosen a remote instrument, a list of remote hosts pops up from which you may chose one.

4. A pop-up window then gives you a list of available Dynamic Data Supplier statistics. For DCE Application Instrumentation, click on the **hosts/<hostname>/DCE/...**, and subsequent lists will provide you with the available options.
5. Once you have selected a displayable value, a dialog window lets you choose certain options, like color and description text.
6. For other displayable values, repeat steps 3 through 5.

4.10 Tivoli TME 10: DCE Management Tools from Santix Software GmbH

The Tivoli TME 10-based DCE management products from santix software GmbH are primarily designed to administer and supervise DCE cells with its global resources. The santix DCEmgmt suite consists of four members:

- **DCEmgmt/Cell Manager**—This helps the cell administrator through its graphical user interface to configure, modify or display relevant information for a DCE cell, such as the attributes of dced, the clearinghouses, the registry, replicas within the cell, servers, DTS Servers and clients, CDS objects, and ACLs. It allows remote, profile-based management of host data, server configuration and keytab files.
- **DCEmgmt/Security Manager**—Its primary focus is the management of user accounts, groups and organizations, based on profiles. It extends the user administration functionality of TME 10 into the DCE domain.
- **DCEmgmt/Event Manager**—Monitors server health and server performance statistics, maintains and monitors thresholds, and provides a wide range of trigger features based on various events. Event Manager keeps audit trails and forwards DCE audit and serviceability events to a central console for correlation with other event sources.
- **DCEmgmt/DFS Manager**—Manages any kind of DFS resources on servers and clients, including backup and fileset replication. It provides auto-discovery of DFS resources. Important for performance evaluation, it collects server connection and utilization statistics.

With the exception of the Event Manager, the DCEmgmt applications are primarily geared for management of a DCE/DFS cell and its resources. However, the products provide a variety of monitoring and test capabilities that can be used for performance measurement and evaluation, as well as for problem determination. This is especially true for the Event Manager application. They all help an administrator to quickly gain an overview of the cell structure and its resources.

All members of the DCEmgmt application suite are well integrated in Tivoli's Management Environment 10 (TME 10), a framework for highly scalable, platform-independent, enterprise-wide systems management.

The DCEmgmt management products are available (or will soon be available) on AIX, other major UNIX platforms, and on Windows NT.

4.11 DCE/Sleuth from IntelliSoft Corporation

DCE/Sleuth from IntelliSoft Corporation is a Motif-based network trace and analysis tool that specializes on DCE-related network traffic. It can be used on any host in the network, since it is capable of tracing any-to-any traffic, as long as it is ″visible″ by the network adapter.

DCE/Sleuth provides a series of different reporting windows, including various graphical representations of gathered data. The following two screen shots are examples of a typical use of these windows. The first lists DCE network traffic, including an accurate time stamp, as it was caught on the network. The second example in Figure 31 on page 91 shows an analysis screen for response times (time between a request and the corresponding response) that can be used to exactly determine the performance of a server, or where some bottlenecks may be.

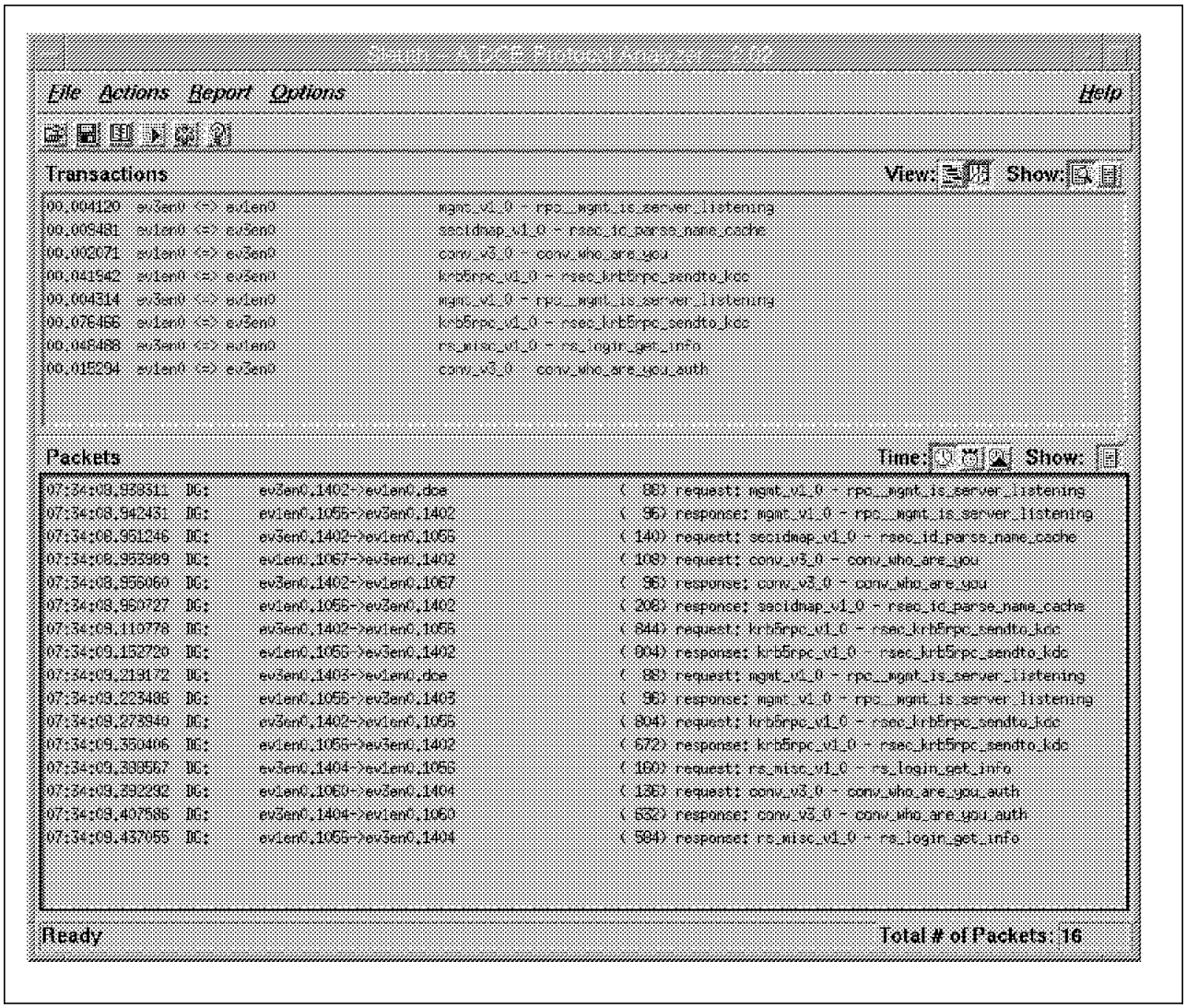


Figure 30. DCE/Sleuth: Sample Standard Trace Window

The sample screen shot in Figure 30 shows a trace taken with DCE/Sleuth while performing a `dce_login` from an AIX client (depending on the printing quality, some details may be difficult to read).

DCE/Sleuth has the ability to import IDL files from your DCE applications. With the knowledge of the interface definitions through these IDL files, DCE/Sleuth can then further analyze every single network package down to all the parameters that have been passed in an RPC call in either direction. DCE/Sleuth also allows you to trace and analyze Encina-related transactions through its ability to recognize Encina traffic. Double-clicking on a transaction opens a window that reports the details about this transaction with various options for the level of detail. DCE/Sleuth already comes with many IDL files for the standard DCE core services and is therefore capable of analyzing those standard DCE RPC calls.

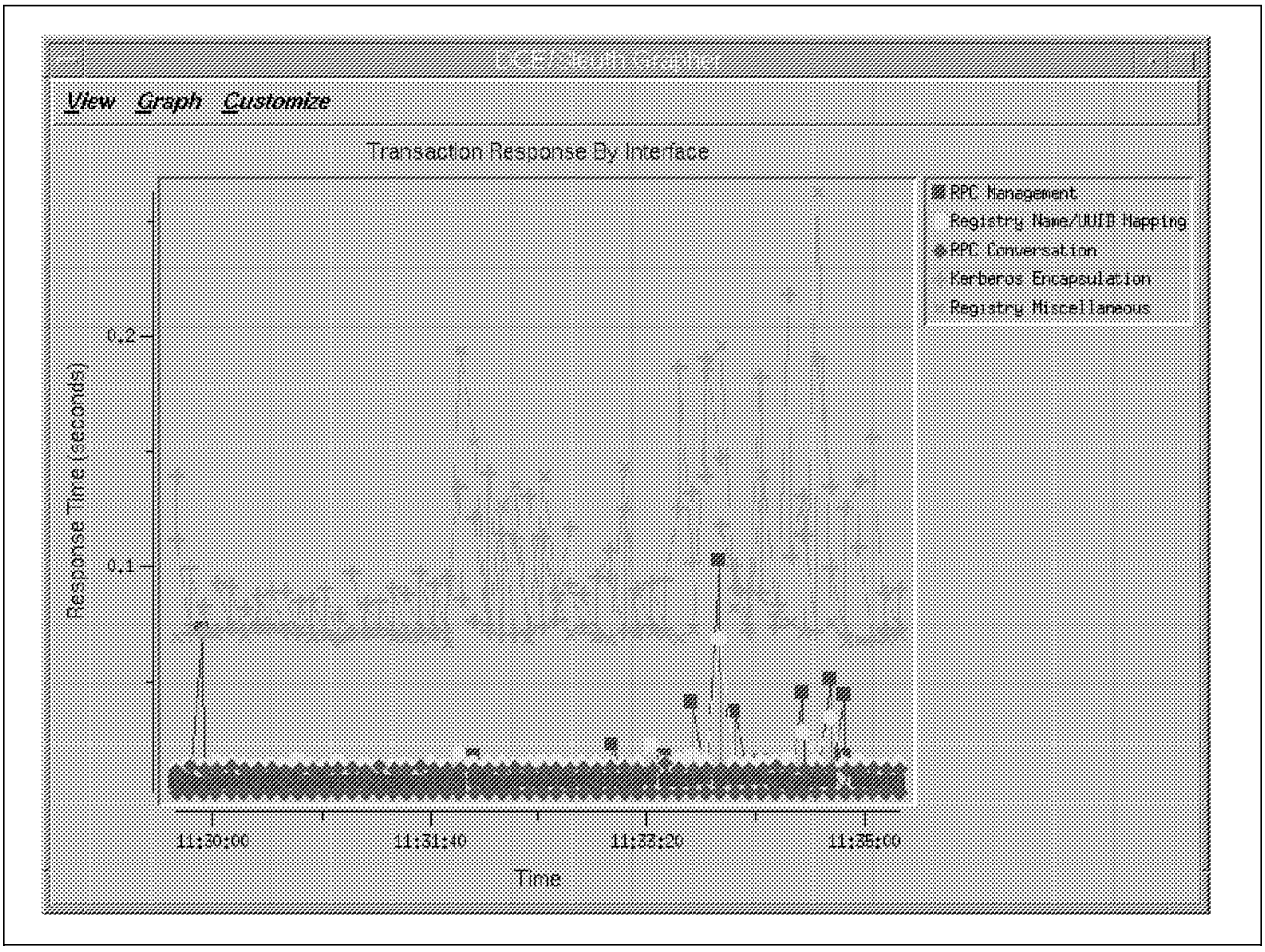


Figure 31. DCE/Sleuth: Sample Response Time Analysis Graph

The response-time graph in the above example shows response times per service on a server (depending on the printing quality, some details might be difficult to read).

DCE/Sleuth supports the application developer during debugging and tuning and provides the system administrator important information necessary for performance measurements, evaluation, tuning, and planning.

DCE/Sleuth runs on AIX and other UNIX platforms. Please note that the above screen shots are examples and may look different from other installations.

Chapter 5. Planning for Performance

The process of performance planning is often very complicated because of the many different factors that are involved. Performance planning will usually not give a correct figure of the parameters that have an influence on the performance, but using a structured approach, it is possible to use a well-balanced estimate of the system performance that can be adjusted depending on the actual load. The performance-planning process for a system implementation includes the following work items that must be in place to ensure control of acceptable performance:

- *Capacity planning* is the work item where the required capacity is estimated, based on the anticipated demand.
- *Performance policy definition* is the work item where the actual service levels will be defined and by which means they can be measured. The actual service levels are often included into documented service-level agreements. Service-level agreements describe, for example, the requested response time, the opening hours, and the method of service-level measurements.
- *Monitor and analyze* is the work item that has to be in place before a system is going into normal operation. The result of this work item is a set of performance monitoring probes and procedures that validate the defined performance policies and gather information on capacity trends.
- *Performance execution and measurement* is the work item that will apply changes into the system based on the observed capacity trends and show how the defined performance policy is being reached. This work item will be an ongoing process.

The next sections describe the DCE-related factors that can have an influence on planning for performance in a DCE distributed environment.

5.1 Gathering Input for Capacity Planning

To perform a well-balanced estimate of the requirements a system will have, it is necessary to collect information of the requirements the new system will introduce. To do this, you have to know what the new system has to do or what it is expected to do.

The process starts by gathering information of the expected result when the system is running. This information, together with the knowledge of the expected load, will give information on the requirements that all the components must be ready to deliver.

For cases where useful requirement parameters are unavailable, a test environment must be established. This test environment must give estimates that can be applied to the real environment.

The parameters that are used for planning in the DCE environment are divided into the following topics and based on the actual function the system will be performing:

- The network traffic, which is bound to the network topology, because the DCE core services will generate network traffic that is not directly related to applications using DCE.

- The memory and disk requirements of the DCE core servers. They house the registry and the CDS databases both in memory and on disk.
- The memory and disk requirements of the DCE client services on the clients, because the DCE core services are performing caching of often-used data on the clients. This requires memory together with disk space to hold these DCE data files.
- The replication of DCE services to place data where it is most wanted will cost CPU, memory and disk capacity both on the sending (master) and receiving (replica) systems.
- The DFS, because of the load it will handle, together with the cache settings on the clients.
- Additional DCE applications themselves in the way they use DCE core services and the local system resources on the systems they run on.
- Other related network traffic because of they way DCE applications may use other network services.

To perform qualified estimates, the network topology, the expected capacity and the requested service-level agreements must be in place.

To perform capacity planning for DCE server components, the following parameters must be estimated or measured. The important parameters to be collected are for each of the DCE components the following:

- Security Server
  - The number and type of registry entries. This determines the size of memory and disk of a Security Server.
  - The usage of the Security Server. This tells how fast a Security Server should be or how many replicas that are required.
- CDS Server
  - The number of directories and CDS namespace object. This is used to estimate the amount of memory and disk size.
  - The usage of the CDS Server. The CDS namespace lookups will be transferred through the network and must be processed by the CDS Server CPU.
- DFS File Server
  - File usage and typical file-access characteristics, and caching efficiency for these access characteristics, determine the necessary performance of a server.
  - The disk size of the DFS Server depends mostly on the total sizes of the filesets (or the aggregates which contain the filesets).
- DTS
  - The displacement of DTS Servers and their courier role. The DTS Servers need to adjust the time in between DTS Servers to provide time services for DTS clients

The basic result of a capacity planning will often be:

1. The requirement for the network

2. The replication requirement because of the network or due to other requirements
3. The disk and memory size of the DCE servers
4. The type of machine to use for the DCE servers

Other planning results can be the DCE client sizing, the integration into existing systems and the administration requirements, which in some cases are much more important than the performance of a system.

5.2 Network Topology

The network traffic imposed by DCE is divided into the core DCE server traffic, the administrative audit setting, and the DCE applications, including the traffic from DFS.

The causes of DCE traffic are as follows:

- DCE authentication (login), which is necessary for obtaining granting tickets from the authentication service and to get the Privilege Attribute Certificate (PAC) from the privilege service of the Security Server
- Security Server registry replication, which will occur when replicas are defined or after updates to the master replica
- CDS lookup requirements, which is used by the CDS clerks to lookup CDS directory and object information
- CDS replication, which will occur as setup by the cell administrator (skulk)
- DTS, which is mainly synchronization messages between the DTS servers and clients
- Other applications using RPC, which is all the traffic generated by the DCE applications servers and clients and the traffic they introduce to the DCE core servers
- DFS, which is mainly parts of files requested by DFS clients
- Audit, which is controlled by administrative action

This traffic has to coexist with already running systems and applications. The typical network traffic that is generated by DCE core components without any special configuration and tuning consists of a relatively high number of small packages of both TCP and UDP protocols. For example, as we have seen in 2.3.1, “Client Machine Startup” on page 15, the startup of the DCE core services including DFS client on a client machine generates about 700 packets of an average size of 270 bytes. The concerns a network administrator may have are different, depending on the type of network. The two main types of networks that can be related to DCE are:

- Networks having sufficient bandwidth to handle the estimated traffic
- Networks having limited bandwidth to handle the estimated traffic

Although a network may have sufficient bandwidth, the network administrator normally makes use of network management tools to identify bandwidth problems together with other problems that may arise in the type of network in use.

5.2.1 Campus Network

A campus LAN can be characterized by having all clients and servers connected to a single high-speed network or to multiple LANs that are interconnected with high-speed bridges and routers. This may include high-speed technology like ATM or FDDI. This type of network supports LAN protocols without restrictions. The transmission speed and throughput capacity on these networks are defined by the type of LAN components used. Normally, the LAN speed and throughput enables this type of LAN to handle any to any network traffic with minor network-related delays.

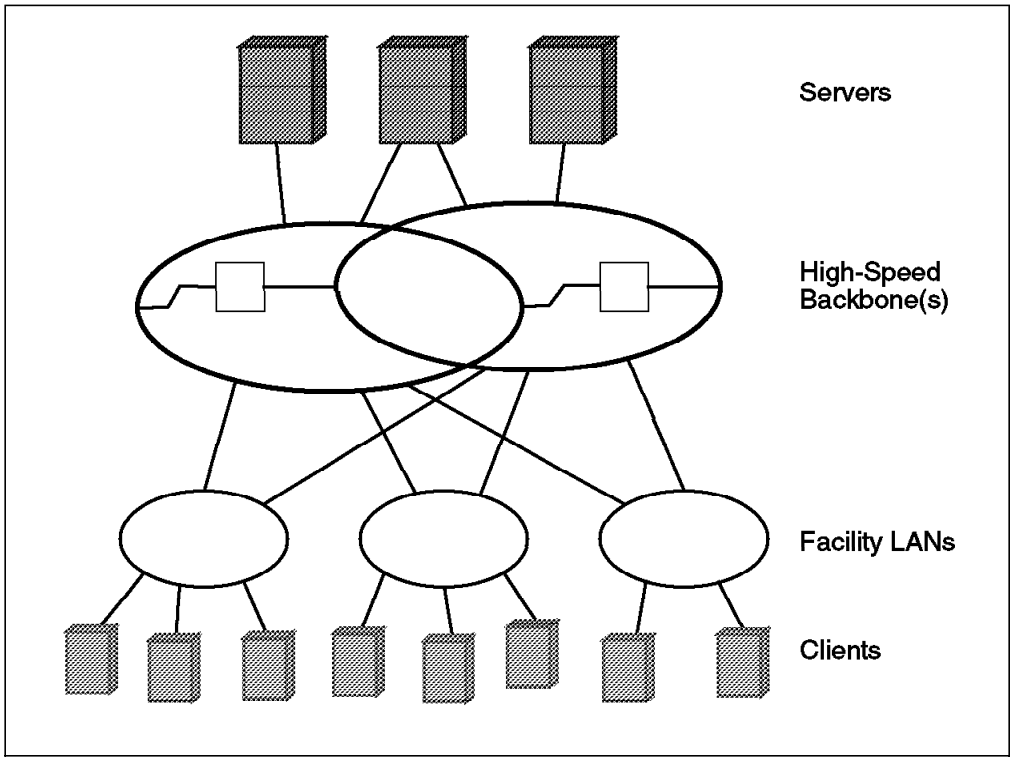


Figure 32. Campus Network Example

For DCE and most other services, this network type causes only few considerations to take into account, such as:

- The location of the DCE servers can be important. Although the network is running with a high throughput rate, the network segment on which the DCE servers a placed may be congested of network traffic not necessarily caused by DCE. A good solution might be to connect servers directly to the backbone networks.
- The connecting components may get congested because DCE-related network traffic must find a route to the DCE core servers.
- Broadcast filtering and broadcast awareness. The DCE core servers advertise their presence on the network. The DCE clients have the possibility to rebroadcast CDS advertisements. This facility is disabled by default, but when enabled, a large number of broadcasts can be generated.
- Network management must include the new DCE components. DCE servers are enabled for SNMP management.

DCE will normally not impose problems related to the network in a campus network due to its high bandwidth, low latency time, and high availability due to redundant routes.

5.2.2 Interconnected LANs

The interconnected LANs scenario can be described as a number of LANs interconnected over non-native LAN connections, such as a dedicated fiber link, allowing LAN network traffic to pass at high data rates. The speed of the connection in between the interconnected LANs is often in the Mbps range. Short propagation delays may occur together with delays during peek periods of network traffic. The link in between the LANs is often multiplexed with other types of services like voice running on it.

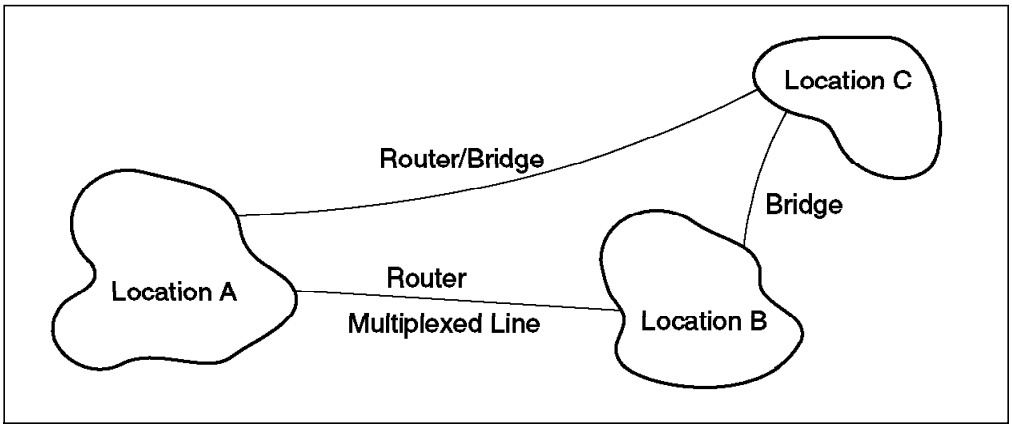


Figure 33. Interconnected LANs

Because of the fast links in between the locations, a minor amount of delay is imposed on the network traffic related to DCE, and DCE on the other hand will not suffer from any short delays.

To reduce the DCE-related network traffic in between the interconnected locations, and to increase the availability of the DCE server services in cases of link outages, DCE server replicas can be placed in each location. Other cases exist where adding a cell for each location may be an alternative.

5.2.3 Distributed WAN Connected LANs

The distributed WAN environment can be described as a number of connected remote locations each having one or more LANs. The typical example of the distributed WAN-connected environment has a central site and a number of remote locations connected to the central site.

The main difference to the campus network and to the interconnected LANs scenarios is that a WAN introduces a relatively slow connection to the network at speeds of, for example, 19.2 or 64 Kbps.

Using DCE in this type of environment requires a number of considerations to be in place.

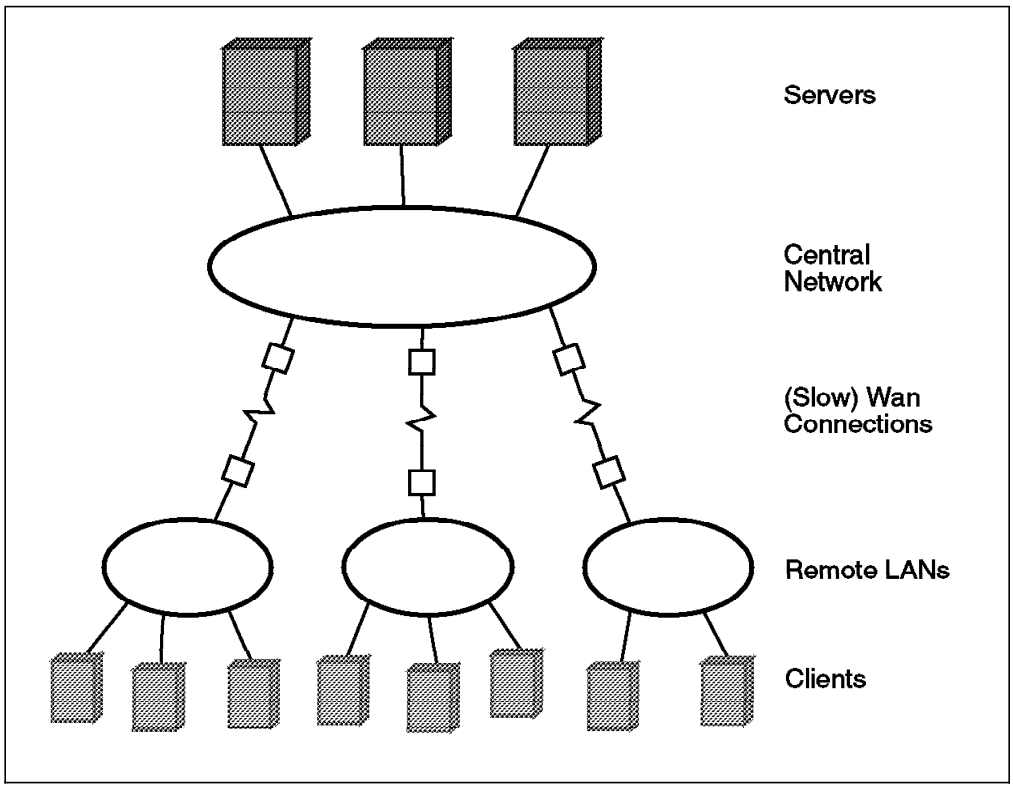


Figure 34. Distributed WAN Environment

The type and amount of network traffic is the key to the planning of the distributed, WAN-interconnected environment. Because of the limited bandwidth available, the planning process focuses on how to optimize the utilization of the remote lines.

With the figures outlined in Chapter 2, “Process Flow and Components Affecting Performance” on page 7, we can do some example calculations.

Assume a remote location has 50 DCE users on machines connected to a such a WAN-connected LAN. They normally perform a DCE login during a 10 minutes period of time before the beginning of their business hours.

The amount of data transferred related to DCE login during this period is:

50 x 12 KB = 600 KB

The 12 KB represents (rounded up) the traffic to a central location having both a DCE Security Server and a CDS Server.

On a slow link with 19200 bps, not including byte headers and trailers, this will, for estimate purposes in the ideal case, take:

(number of bits/transfer rate) = 600 x 1024 x 8 / 19200 seconds = 256 seconds

This 4.5 minutes (256 seconds) is network utilization time due to user logins. Additional delays must be included into this calculation. Because of the long time, there can be time-outs in the DCE login process. This will cause additional network traffic that goes through the remote line.

The result of this example case study is either to reduce the number of DCE logins or to increase the remote link speed.

By increasing the link speed to 64 Kbps, the ideal time to transmit the DCE login traffic will be (the ideal case and for estimate only):

600 x 1024 x 8 / 65536 seconds = 75 seconds

In this case, the number of retries will decrease significantly, but time-outs may still occur if too many users attempt to log in at the same time, or if additional network traffic reduces network bandwidth at the same time.

As another example, we assume that there are 10 Personal Computers, which are switched off after working hours. Every morning, they are powered on and will subsequently start the DCE client services. This causes an additional network utilization time of (see 2.3.1, “Client Machine Startup” on page 15):

10 x 190 KB x 1024 x 8 / 65536 seconds = 238 seconds

Similar considerations exist for other types of DCE related network traffic. Assuming the same remote location as in the last example is running a DCE application accessing a server located at a central site, the DCE client applications will add network traffic related to:

- Finding the DCE application server (CDS)
- Importing bindings (CDS)
- Authentication and authorization (DCE application server)
- DCE application traffic

Assuming the DCE application client is running all day and doing caching of the binding information, there will be network traffic when the DCE application is started along with the DCE application-related network traffic.

The initial network traffic depends on the amount of data included. Assuming the amount of data transferred for one DCE application initiation is 8 KB, transferred in 32 packets, the network transfer time in the ideal case is:

8 KB / 64 Kbps + 32 x 50 ms = 2.6 seconds

The 50 ms is the time a frame will approximately use to be transferred into the remote link hardware/software buffers. The reason for the 50 ms in this example is to include cases where encryption or store-and-forward methods are used within, for example, modem equipment.

The time required does not include the processing time on both the DCE application server and client.

The total response time can be estimated as the remote link delay time plus the time a similar DCE application initiation takes on a LAN.

Using the DCE application on the remote link creates network traffic. The amount of traffic depends not only on the application data but also on the package protection level and the authentication level.

If the total network traffic on the remote link is close to, or exceeds, the maximum capacity, a faster link or the use of replicated DCE servers and addition of local DCE servers must be considered.

If the DCE logins occur only during the start of business hours, a DCE Security Server replica is probably not required for performance reasons. If the use of the Security Server registry is more than for DCE logins, a replicated DCE Security Server can be used to reduce the network traffic over the remote link.

Similarly, using the CDS Server causes some network traffic on the remote link, but because of the cache facilities of the DCE clients, the network traffic is generated only for the initial lookup. In this case, replicating the most used CDS directories can be considered.

5.3 Replication

The DCE Security Server, the CDS Server and DFS FLDB Servers maintain master copies of their respective databases. For CDS, each namespace directory can be replicated separately. For DFS databases, individual DFS filesets or groups of filesets can be replicated. For the DCE Security Server, only the entire registry can be replicated.

Because all DCE services rely on the information managed by the DCE Security Server and parts of the CDS namespace (especially /.:/, /.:/subsys/dce/sec and all /.:/hosts directories), this information must be highly available.

The following sections describe a number of planning considerations for using DCE replicas.

5.3.1 Security Server Replication

A Security Server can replicate the registry for improved performance and availability. Each Security Server replica contains the entire registry. Clients will spread their requests across the Security Server replicas unless configured to use a specific server (see 7.2.1, “Security Service, the pe_site File” on page 142). Replication will reduce the number of requests on each Security Server.

The reason to implement Security Server replicas are the following:

- Availability—to ensure that at least one Security Server registry can be accessed at any time
- Performance—to spread registry lookups onto more than one machine
- Network topology—to reduce network traffic on congested or slow network connections

To make performance and capacity planning for a Security Server, the following factors can have an effect on the Security Server:

- The number of accounts and their usage of the registry
- The number of replicas being used
- The number of addition, deletion and modify operations introduced to the registry

Security replicas can be implemented at any time since it is easy to configure (or remove) a replica. Because of the characteristics of the registry’s use and the credentials caching on the DCE clients, the peak load of a Security Server is often at the start of business hours. Disregarding the network topology with its

possible implications on replication, a minimum of two Security Servers should be used.

If the performance requirements or the usage of the Security Server requires the addition of additional Security Servers, they can easily be added.

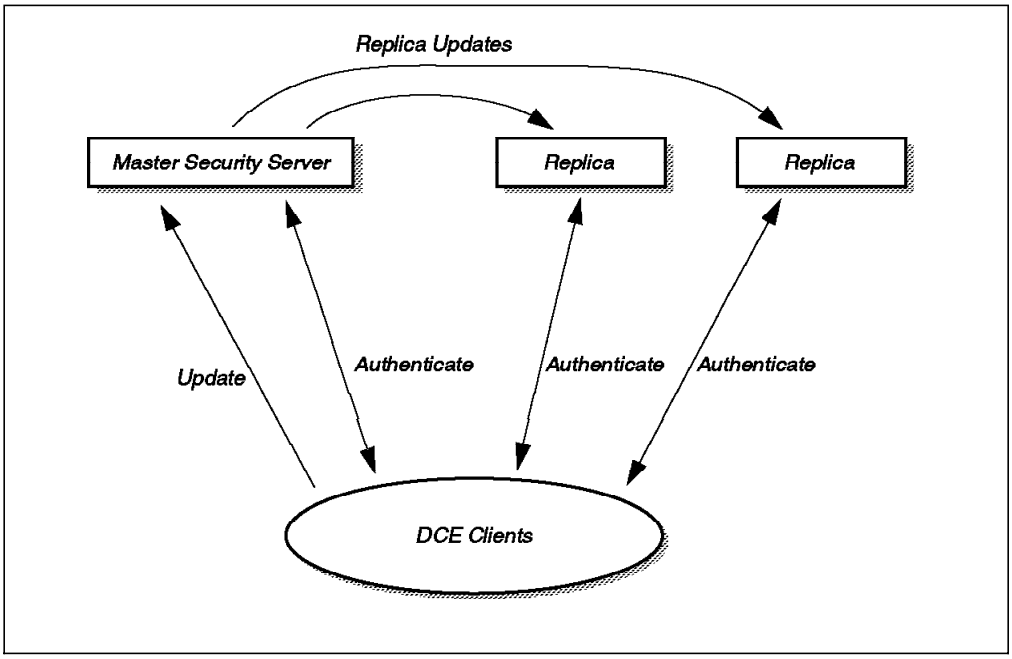


Figure 35. DCE Security Replication Principles

Adding more Security Server replicas into a cell may introduce a bottleneck in respect to the Master Security Server since it is, and always remains, the only machine used for updates of the registry to take place. In addition to handling updates, the Master Security Server is also responsible for propagating any changes to the replica servers. This can be shown with an example.

We assume a DCE cell with 2500 users, having a password policy to require the users to change their passwords once every month. The password change generates an immediate update propagation from the Master Security Server to any replicas. The additional traffic between the Master Security Server and the replica is (according to 2.3.8.1, “Security Server Replication” on page 28) about 10 packets with a total of 2.2 KB. A month has about 20 working days, and we will further assume that most of the user logins (say 2000) will take place within 15 minutes every morning. With these assumptions, an average of 100 password changes will be done every day, most presumably during logon times. This will generate an additional network traffic for replica update of:

100 changes x 2.2 KB/change = 220 KB per replica server

This is probably negligible in most environments. However, if many replica servers are being used in a LAN/WAN environment for high availability, this kind of traffic and the additional load on the Master Security Server can become considerable. Any change to the master registry is propagated to the replicas. Adding a DCE account, as another example, creates about 6 KB of replication traffic between a master and a replica. This can easily sum up to Megabytes when many replicas are involved and user administration is done in masses with

batch processing. This should be considered when slow network links are involved.

Another planning issue for DCE security is the ticket lifetime. Clients contacts a Security Server during login to obtain credentials, which are used to access other DCE servers. The credentials are saved on disk for later use. The lifetime of the credentials is limited, and they have to be refreshed after expiration. Application programs often do this automatically; users have to issue a `kinit` command. The lifetime is determined in the policy set up by the administrator.

The default credential life time can be displayed by the following command:

```
dcecp> registry show
{deftktlife +0-10:00:00.000I-----}
{hidepwd yes}
{maxuid 2147483647}
{mingid 100}
{minorgid 100}
{mintktlife +0-00:05:00.000I-----}
{minuid 100}
{version secd.dce.1.1}
```

The default ticket lifetime in DCE, as shown in the first output line in the above example, is 10 hours, intended to be sufficient for a normal work day. If many users or services usually require longer daily access, this ticket lifetime should be adjusted accordingly, which saves a remarkable amount of reauthentication overhead. Changing the default ticket lifetime attribute to 12 hours can be done with the following command:

```
dcecp> registry modify -change {deftktlife +0-12:00:00:00.00I-----}
```

If Security Servers are being replicated in a LAN/WAN environment (for example, having replica servers at each end of slow links), then it becomes very important to implement server preferences. DCE Security Services by its own is not capable of choosing the nearest server; clients may still connect to remote servers across the slow link. See 7.2.1, “Security Service, the pe_site File” on page 142 for an explanation of how to accomplish this.

5.3.2 CDS Replication

The administrator can create replicas of the CDS namespace and place them in one or more CDS clearinghouses. The administrator normally replicates the directories within the CDS namespace that are most frequently used by the clients. The DCE clients will cache information obtained by queries, thereby reducing additional traffic until the client cache entries expire.

The default lifetime for an entry within the CDS clerk cache in a random value selected by the RPC runtime. The range is by default in between 8 to 12 hours. To handle read operations to the CDS namespace, the CDS clerk performs the following operations:

- The read operation starts normally by looking into the copy of the attribute data the application is requesting.
- If there is no local copy, a local copy is created by reading the requested attribute from a CDS Server namespace.

- If a copy already exists, the expiration age is checked. If it has expired, a local copy is created by reading the requested attribute from a CDS Server namespace.
- If updating is impossible, the expired local copy remains in cache, but the read operation will return an error condition to the requesting application (`rpc_s_name_service_unavailable`).

Because of this characteristics, the number of entries within the CDS namespace and the number of DCE clients performing namespace lookups determine the number of CDS replicas to use.

To complicate the process of planning for performance, the CDS Servers have master and slave replicas for each CDS namespace directory. This is true for all directories except the root (/.:/) namespace directory. The root namespace directory is replicated from the master replica to all slave replicas.

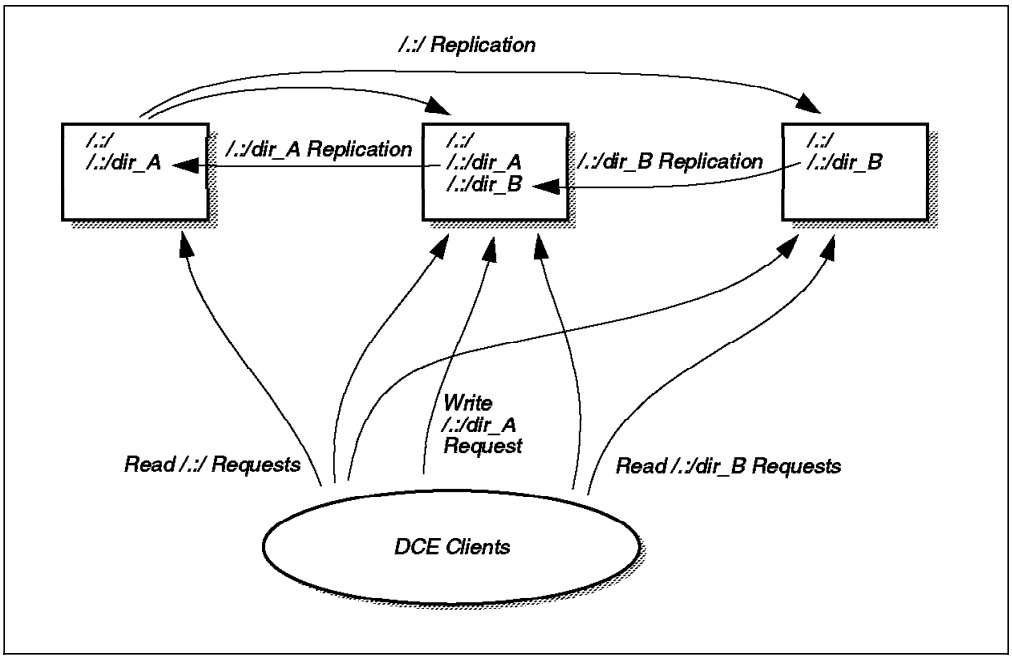


Figure 36. DCE CDS Replication Principles

As Figure 36 shows, read requests can be handled by any replica server that holds a replica of the requested directory. Write requests can only be performed on the master replica.

The reason to implement CDS Server replicas are the following:

- Availability, to ensure the root and the most-used CDS namespace directories can be read at any time
- Performance, to spread CDS namespace lookups on more machines
- Network topology, to reduce network traffic on congested or slow network connections

The use of CDS replicas introduces the same kind of replica traffic as for the example of the password change of a Security Server (see previous section). For the CDS replication, this type of traffic is valid for:

- The CDS root master

- The CDS master of a directory

Because all DCE components rely on the availability of the CDS, at least one replica of the CDS Server should be installed for availability reasons. For a replica CDS Server, the root (/.:/) will be replicated by default. The following minimum CDS namespace directories should be replicated as a minimum:

```
/.:/
/.:/subsys/dce/sec
/.:/hosts
/.:/hosts/*
```

Relying only on the CDS replicas alone, a DCE client machine will be able to start the security client (DCE Security Server is available) and directory clients. A DCE login will also succeed.

DCE application servers will most probably not be able to start with this minimum configuration by relying only on replicas as specified above because they require write access to the namespace.

The /.:/hosts directory and its subdirectories are used to create the machine credentials whenever the DCE daemons are started on a machine.

5.3.3 DFS Server Replication

DFS File Server replication minimizes the effects of a machine or network outage. Filesets that contain frequently used files such as application binaries and on-line documentation can be replicated onto multiple servers within a cell. If a client loses access to its current server, it automatically locates any available replica server. Replicas are updated upon request, or according to a predetermined schedule.

Replicating a fileset means to copy a read-write fileset from a File Server machine and placing a read-only copy, called replica, onto another File Server machine. Both systems, where the master and the replica reside, must be configured as DFS File Server Systems and run the File Exporter processes. DFS provides replication of DCE LFS filesets (replication of a non-LFS fileset is not supported).

For planning purposes, the same basic rules apply as for Security or CDS Servers:

- Replication generally increases performance and improves availability.
- Provisions should be in place for clients to ensure they prefer to access replica servers on fast links, rather than those behind slow links.
- Servers (replica) should be in close proximity to the clients to assure short, fast access.
- Replication introduces additional cost in terms of administration and in terms of machine and network load in order to keep the data on the replica servers current.
- For the last reason, replication of large, frequently changing data should be avoided.

Specifically to DFS, some additional considerations apply:

- Large, static data, such as program code or document archives, is well suited for replication.
- Frequently changing files, such as users' home directories, probably do not benefit from replication since it is almost always accessed by one single user (no performance benefit), and the access is most likely to be a read/write access anyway, which cannot be a replica.
- Because an update of a replica may involve a large transfer of data, the placement of a DFS File Server replica ″behind″ a slow link must be considered carefully.
- Depending on the amount of changes per unit of time made to the DFS Fileset Location Database, DFS FLDB Servers may need to exchange a considerable amount of data to keep the slaves current. This must be taken into account when a slow network link is in between FLDB Servers.
- The path to any read-only fileset in DFS filespace should only contain read-only filesets. Only this way will the Cache Manager actually access the read-only version of a fileset.

A common, often-used standard configuration in a campus-type network is to have three FLDB Servers and, depending on the amount of data, a certain number of File Servers.

For other types of network topologies, it is still a good idea to keep servers centrally located unless performance or availability require them to be close to the clients. If DFS replica servers are to be distributed, replication of the security and CDS Servers should also be thought of; otherwise a remote DFS replica server, or a DCE client, may be unable to authenticate or access CDS namespace if a network link becomes unavailable. Access to DFS would not be possible anyway in such a situation.

5.4 System Sizing for Capacity

The sizing of a system is normally performed on calculations or estimates based on the expected load the system will be required to handle. This is valid for the memory, disk and CPU sizing.

Because of the nature of the Security Server and CDS, which keeps the actual registry or namespace databases in memory, there are special considerations to take into account.

5.4.1 Security and CDS Server Requirements

The Security Server maintains a registry of principals, groups, accounts, and organizations, which is kept in memory and on disk. Adding more definitions into the registry will require more memory and disk space. The DCE design point is to load the entire registry into memory when the Security Server is started for optimal performance. When the required memory to store the registry exceeds the available memory, paging will occur. As long as the active part of the registry is in memory, no significant degradation of response time will be noticeable.

The CDS Server also keeps the directory (namespace) on disk and in memory. The CDS Server makes use of a checkpoint file, which is a file containing the entries kept in memory. This checkpoint file is updated whenever the CDS is

paused or stopped, or, if the CDS Server is not paused or stopped, the checkpoint file is updated every 24 hours. The Security Server writes its database (if there are any modifications to it) to disk every two hours.

Note: The in-memory database used internally by the Security Server will grow if entries are repeatedly created and deleted. This growth is temporary, until the Security Server is restarted. If the server is stopped unexpectedly, such as in a power loss situation, it will replay all activities since the last 2-hour checkpoint had occurred, and it will show the same growth as would have been observed when the updates were first done.

5.4.1.1 Security and CDS Server Data Files Locations

The data files used by DCE on AIX is /opt/dcelocal/var directory, which is a link to /var/dce. The directory location on OS/2 is x:optdcelocalvar, with x: being the drive selected during installation time. The files are located in the following directories using the AIX directory specification relative to /opt/dcelocal/var:

- ./adm/directory/cds - CDS cache and control files
- ./adm/time - DTS files
- ./adm/dfs/cache - DFS cache on disk
- ./audit - Auditing directories and files
- ./dced - Files used by the dced daemon
- ./dfs - DFS Server files
- ./directory/cds (dircds on OS/2) - CDS checkpoint and log files
- ./ems - Event Management files
- ./rpc - RPC-related files
- ./security - Security Server directories and data files. The rgy_data subdirectory contains the registry database
- ./svc - The warning and error logs

The sizes of Security Server and CDS data files will grow with the number of database objects defined in the databases.

5.4.1.2 Security Server Disk and Memory Sizing

The data contained for an account is divided into the unique combination of the principal, group and organization definitions (PGO). Each principal and group has ACL definitions related to it, which will consume disk space. From actual testing on DCE 2.1 and AIX, which indicates the values are very similarly on both platforms, the following figures for disk space requirement have been observed:

Table 8. DCE 2.1 Security Server Registry Objects Disk Space Estimates

DCE Security Server Registry Changed Files	Estimated Size Change in Bytes
Principal	340
Group	320
Organization	300
Account	250
ACL	420
Extended Attributes	150

These values are for planning purposes; actual sizes may differ. Namely the ACL entries are dependent on the actual ACLs being used for a particular account. We will use the higher number (800 bytes) for the following estimates.

The disk space requirement for a user account can, for planning purposes, be estimated to approximately 1 KB because an account contains principal, account and ACL definitions (340 bytes + 250 bytes + 420 bytes = 1010 bytes).

For checkpointing, the actual disk space requirement needs to be doubled. Please read the note at the end of this section for a general comment on checkpointing considerations.

The Security Server keeps a memory image of the registry data, which was found to be about three times the values as when stored on disk. The memory usage of the Security Server depends on three factors:

- The number and types of objects in the registry database
- Changes to the registry temporarily require more memory that will be released later on
- When the registry is replicated, the memory requirements are higher. This is believed to happen because it keeps track of the replication status for its objects.

To summarize, the total disk and memory requirement per account on a Security Server is about:

Table 9. DCE 2.1 Security Server Registry Disk and Memory Estimates

Registry Entry	Registry Disk Space	Memory Estimate
Account	1 KB	approx. 3 KB

Note

The users and group definitions for obtaining the figures in the last examples have been created by rgy_edit, which does not create any CDS entries. Using dcecp for user creation, both registry and CDS directory entries will be created for each individual user. This is a default function of dcecp. Most installations, however, do not require a CDS directory for users. In order to keep CDS free from unnecessary entries, requiring extra memory, disk space, and performance, user accounts should not be created by using dcecp user create. If dcecp is used, use the following sequence, which will not create a user directory in CDS:

```
dcecp> principal create <principal>
dcecp> group add none -member <principal>
dcecp> organization add none -member <principal>
dcecp> account create <principal> -group none ...
```

Alternatively, the user directories can or should be deleted after being created, or the TCL script for adding user accounts can be modified accordingly to not create user directories in CDS. Although the latter is technically the best solution, care must be taken because this TCL script may be replaced with any software update subsequently applied.

5.4.1.3 CDS Server Disk and Memory Sizing

The requirements for the CDS Server are as follows. The objects stored in the CDS namespace can vary in size because of the attributes connected to the object. The following raw estimates of the directory and objects sizes can be used:

Table 10. DCE 2.1 CDS Storage Sizes

CDS Namespace Entry	Estimated Size
CDS Directory	6.9 KB
CDS Object	1.2 KB

The values above have been obtained by adding a large number of directories and objects into a namespace and observing the growth of the CDS checkpoint file. The object defined did not contain any attributes.

Note on CDS Disk Space Requirement on AIX

The disk space requirement for a CDS directory entry on AIX was found to be about 12.9 KB, as opposed to 6.9 KB on OS/2. This is due to a change in the code made to the OS/2 implementation in order to reduce disk space requirements. The same change will be incorporated in DCE for AIX through a program update (PTF).

Standard CDS hosts' entries require the following estimated sizes:

Table 11. CDS Host Entry Storage Sizes

Namespace Entry Type	Number	Estimated Size
/.:/hosts	1 directory	6.9 KB
/.:/hosts/*	4 objects	4.8 KB
Total		11.7 KB

For the OS/2 DSS File and Print Server, additional objects and attributes are added to the CDS namespace. The following CDS entries are applied when using OS/2 DSS File and Print Server realms (see also 6.6, “OS/2 DSS LAN Server Environment” on page 135 for further information on foreign registry and directory synchronization).

To support OS/2 DSS File and Print Server realms, the following CDS namespace directory entries and objects are created by using PCSRVX83 as the OS/2 LAN Server computer name and ITSO1 as the OS/2 LAN Server Domain Name:

```
Directory: /.:/hosts/PCSRVX83
Objects: cds-clerk, cds-server, config, dts-entity, profile, self
Directory: /.:/resources
Directory: /.:/resources/realms
Directory: /.:/resources/realms/ITSO1
Directory: /.:/resources/realms/ITSO1/files
File alias definitions represented by objects
Directory: /.:/resources/realms/ITSO1/printers
Printer alias definitions represented by objects
Directory: /.:/resources/realms/ITSO1/public_apps
Public application definitions represented by objects
Directory: /.:/resources/realms/ITSO1/serial_devices
Serial device definitions represented by objects
Directory: /.:/resources/realms/root
Directory: /.:/resources/realms/root/files
Directory: /.:/resources/realms/root/printers
Directory: /.:/resources/realms/root/public_apps
Directory: /.:/resources/realms/root/serial_devices
Directory: /.:/subsys/dce/fgnreg
Object: lspwsync (The DSS password synchronization server object entry)
Directory: /.:/subsys/dce/pwd_mgmt
Objects: lspwsd (The DSS password strength server object entry)
pwsync (The DCE password syncrhronization server object entry)
Directory: /.:/subsys/dce/sec
Object: PCSRVX83
Directory: /.:/subsys/realms
Directory: /.:/subsys/realms/ITSO1
Object: domain_sync, realm_child_list
Directory: /.:/subsys/realms/ITSO1/servers
Object: PCSRVX83
Directory: /.:/subsys/realms/root
Object: realm_child_list
Directory: /.:/subsys/realms/root/servers
```

Adding the first realm into a DCE cell creates the following CDS directories and namespace objects:

Table 12. Initial OS/2 DSS Realm CDS Storage Sizes

Namespace Entry	Number	Estimated Size
/.:/hosts	1 directory	6.9 KB
/.:/hosts	6 objects	7.2 KB
/.:/resource directories	12 directories	82.8 KB
/.:/resources objects	0 objects	0 KB
/.:/subsys	7 directories	48.3 KB
/.:/subsys	7 objects	8.4 KB
Total		153.6 KB

For each additional OS/2 DSS File and Print Server realm, the following estimates can be used:

Table 13. New OS/2 DSS Realm CDS Storage Sizes

Namespace Entry	Number	Estimated Size
/.:/hosts	1 directory	6.9 KB
/.:/hosts	6 objects	7.2 KB
/.:/resources directories	5 directories	34.5 KB
/.:/resources objects	0 objects	0 KB
/.:/subsys	1 directory	6.9 KB
/.:/subsys	2 objects	2.4 KB
Total		57.9 KB

Adding one additional OS/2 DSS File and Print Server into an existing realm requires the following entries and objects in the CDS namespace:

Table 14. Additional File and Print Server CDS Storage Sizes

Namespace Entry	Number	Estimated Size
/.:/hosts	1 directory	6.9 KB
/.:/hosts	5 objects	6.0 KB
/.:/resources directories	0 directories	0 KB
/.:/resources objects	0 objects	0 KB
/.:/subsys	0 directories	0 KB
/.:/subsys	1 object	1.2 KB
Total		14.1 KB

Adding OS/2 DSS File and Print Server Alias and Public applications definitions make use of additional attributes that are stored with the object. Attributes used by OS/2 DSS File and Print Server have the prefix LS_ (for example,

LS_resource_name has a value for disk drive and path specification in a directory alias). The attributes used depend on the alias type and for each alias type in size, but using 1 KB as an average size value is a good estimate. The same value of 1 KB can be used as an estimate for Public Application attribute values.

Table 15. File and Print Server Alias and Public Application Definition CDS Sizes

Namespace Entry	Number	Estimated Size
`/.:/resources` **objects**	**1 objects**	**1.2 KB**
`/.:/resource` **attribute values**	**1 set of attributes**	**1.0 KB**
Total		**2.2 KB**

The total disk requirements for the CDS namespace using the estimated size values are as follows:

- The checkpoint file size which can be calculated as the size of the data file for the namespace entries
- Free disk space of at least the size of the checkpoint file for temporary use (see note at the end of this section)
- The paging space required to hold the complete CDS namespace if there is not enough memory installed on the system to hold the namespace database
- The CDS log file, containing changes, which only grows as long as changes are applied to CDS, until CDS writes a new checkpoint file

The name of the checkpoint file on AIX contains the word `checkpoint` in the /opt/dcelocal/var/directory/cds directory. On OS/2, the file name of the CDS checkpoint file can be determined by typing the `cdsfiles.map` file in the x:optdcelocalvardircds directory. The file name in question can be determined by finding the name entry that has the ″checkpoint″ string included. This entry points to (->) the name of the checkpoint file.

The memory requirements for a CDS Server on OS/2 or AIX is about three times the size of the disk space requirements for the checkpoint file.

Important Note on Disk Usage for Security and CDS Servers

During checkpointing, the Security and CDS Servers replaces the database files on the disk with a new one. This is done by temporarily creating new files for the databases and then deleting the old ones. For this reason, the filesystem(s) containing the registry or the clearinghouse checkpoint files must be of at least twice the size of the databases because it/they must be able to temporarily store two copies of these databases. If this is not the case, the server processes are not able to checkpoint, which is a severe, unrecoverable error.

5.4.2 OS/2 DSS File and Print Server ACL Manager

The ACL manager of the OS/2 Entry DSS File and Print Server makes use of external files. These files are located in the x:IBMLANACCOUNTS directory:

- db_acl.acl
- db_nam.acl
- db_obj.acl

The files are owned by OS/2 LAN Server and will increase based on the number of ACL entries. The sizes for a realm having approximately 2500 directories and 10000 files are 3.1 MB for db_acl.acl, 2.6 MB for db_nam.acl, and 3.7 MB for db_obj.acl, respectively.

The ACL manager of the OS/2 Advanced DSS File and Print Server will store the ACL entries within the file system in a similarly way as the OS/2 LAN Server Advanced through the HPFS386 file system.

5.4.3 Security Server Disk and Memory Sizing Example

In an example scenario, a DCE Security Server will support 2500 defined users and 50 groups. The system has 100 realms defined, and there are approximately 10 DCE application servers defined.

The 100 realms will create an account for each additional server in the realm. Each realm will register the domain name as an account. For the example, the realms are each an OS/2 LAN Server Domain.

There are three Security Servers and three CDS Servers. Those DCE servers each require an account definition in the DCE Security Server registry.

Additional registry entries exist for the DCE core servers and the support for the OS/2 DSS File and Print Servers. They are, in this example, assumed to be 12 accounts. (The exact number is not important because it is small as compared to the number of user accounts.)

The user machines requires another account registry database entry.

The capacity for the registry database is calculated as follows:

Table 16. DCE Security Server Disk and Memory Sizing Example

Registry	Estimated Disk Size	Estimated Memory Size
2500 Principals	2500 KB	7500 KB
50 Groups	16 KB	50 KB
10 Application Servers	10 KB	30 KB
100 Realm Server Machines	100 KB	300 KB
100 LAN Server Domains	100 KB	300 KB
6 DCE Core Servers	6 KB	18 KB
12 Additional Accounts	12 KB	36 KB
2500 User Machines	2500 KB	7500 KB
Total	5244 KB	15734 KB

The calculation results in the following requirements for the registry.

- The memory requirement for the registry is approximately 16 MB.
- The disk space requirement for the registry is approximately 6 MB.

Other principals may well exist in the above example for other machines and services. The estimated values must be validated over time by observing the actual disk and memory utilization on the DCE Security Server.

5.4.4 CDS Server Disk and Memory Sizing Example

In an example scenario, 200 users are migrated to use an OS/2 DSS File and Print Server. The users already are using a AIX-based DCE server application. The users are all running OS/2 DCE clients.

The OS/2 DSS File and Print Server will support 10 public OS/2 applications, 20 shared printers, and additional 30 shared file directories.

The OS/2 machines will not utilize the Slim Client option. The AIX DCE application server and the DCE core services have approximately (using the `cdsli` command) 20 directories and 50 objects in the CDS namespace.

The memory and disk calculations for the CDS Servers are as follows (using rounded values):

Table 17. DCE CDS Server Disk Capacity Sizing Example

CDS Namespace Entry	Estimated Disk Size	Estimated Memory Size
200 User Machines	2340 KB	7020 KB
1 DSS Initial Realm	154 KB	462 KB
1 DSS Realm Server	58 KB	174 KB
20 Directories	138 KB	414 KB
50 Objects	60 KB	180 KB
10 Public Application Definitions	22 KB	66 KB
20 Printer Alias Definitions	44 KB	132 KB
30 Disk Alias Definitions	66 KB	198 KB
Total	2882 KB	8646 KB

The file space requirement on the CDS Server is approximately 3 MB for the namespace and the memory used by the CDS namespace is approximately 9 MB. Actual values can vary due to different numbers of directories, objects and attributes in the namespace.

5.4.5 CPU Performance

The DCE Security and CDS Servers are designed to have their databases in memory. The reason for this is to make the DCE servers respond as quickly as possible without relying on disk subsystems, which are much slower for random access. Because of this nature of the DCE core servers, the requirement for the memory should be fulfilled in order to allow these servers to respond with best performance to requests.

The limits determined by tests, for example, showed that a DCE Security Server on an RS/6000 model 550 was able to handle approximately five DCE logins per second (details can be found in appendix E.4.6, “Security Server Under Heavy Load” on page 254). This test was performed over a period of 10 minutes, showing the DCE daemon secd (Security Server) using most of the CPU time during this period.

Those five DCE logins per second do not require a high network bandwidth (although the network traffic is determined by the clients, rather than by the server). As a rule of thumb, an RS/6000 is easily capable of handling 500 or more TCP/IP packets per second. Disk I/O was found to be moderate, not causing a bottleneck.

The conclusion of this is that the DCE Security Server is CPU bound while processing DCE logins. But at very high rates (having a more powerful CPU), disk I/O can become considerably higher.

In the same way, the CDS Server is CPU bound, for example, its performance is almost only dependent on the CPU performance, provided enough memory is installed to hold the namespace database.

Other tests have shown that the sizes of the registry database, or the size of the namespace, respectively, have no remarkable impact on the performance of

Security and CDS Servers. It can be assumed that this is due to the binary search algorithm used by these two server implementations.

To perform estimates of the CPU size and capabilities to be used, the use of DCE benchmark testing can help. The test can include the anticipated demand on the systems.

5.4.6 Memory

Because of the in-memory techniques implemented by the DCE Security and CDS Servers, the memory required to support the anticipated configuration must be available. If this is not the case, paging can occur, which may degrade DCE server performance dramatically.

Use the planning sections of this chapter (see 5.4.1, “Security and CDS Server Requirements” on page 105) together with the recommended system sizing of the installation documentation to select adequate memory on the systems running DCE servers. The planning section allows you to estimate the memory usage for the Registry and CDS databases. It does not, however, include memory for the server code or for any other applications running concurrently.

5.4.7 Disk I/O

The DCE servers are designed to keep their data in memory during normal operation. Unless other applications running on the same system heavily require them, the performance of the disk I/O subsystems is not important for DCE servers in most cases. The exceptions are on high-performance servers, for checkpointing or for event management logging and auditing. In those cases the disk I/O performance can become a more important factor on DCE core servers.

Since, by default, the disk I/O system on an SMP (Symmetric Multiprocessors) machine does not scale up the same way as the CPU performance, it easily may become a bottleneck on a heavily loaded SMP machine. It is good system performance planning practice to also improve the disk I/O subsystem when improving the overall CPU performance by adding more processors to it.

5.4.8 Network Interface

The performance of the network interface can have an impact on the overall system performance of a system. However, network load on DCE core servers (Security, CDS or Time Server) that is caused by DCE is usually low and does not require any interface tuning, since these servers are CPU-bound. Only if other applications, imposing a certain network load, run at the same time on the same machines, will interface tuning become desirable or necessary. Or, if the CPU performance is high, for example on an SMP machine, and it is used heavily by a large number of clients, interface tuning techniques may be used to speed up the overall system, or at least to prevent the network interface from becoming a bottleneck.

As a general planning guideline for OS/2 server machines, it is recommended to use good adapter hardware, such as busmaster adapters, and adequately tuned protocol interfaces can speed up DCE server operations.

The limits of the protocols in use can influence the performance of a system. Using, for example, NetBIOS on OS/2 systems, the nature of the NetBIOS introduces upper limits for the number of sessions available.

Using non-native protocols like the IBM AnyNet product family may introduce capacity limits or additional requirements on the systems. For example, using IBM AnyNet TCP/IP over APPC will require the installation of supported SNA services on the systems.

5.5 Distributed Time Services

Distributed Time Services (DTS) components—if used at all—should be planned and configured according to general cell-design guidelines. Briefly, DTS Servers should be located centrally, and DTS Courier Servers should be placed close to the clients. This is obviously only feasible in a certain network topology, having many clients spread over different networks. In a campus-type network, DTS Courier Servers are not likely to be used at all. DTS has a very small impact on overall performance since its components are in a dormant state for most of the time. By default, clients only synchronize with servers once every 24 hours, which causes little network traffic and CPU load. Server synchronization also occurs rarely (unless otherwise configured) and does not impose a remarkable load on either the network or on the machines.

5.6 DFS Planning for Performance

Planning for DFS must take moderate to high data rates and the available network bandwidth into account. Together with availability requirements, this gives the input for server replication and location planning.

5.6.1 Planning for DFS Servers

One aspect to consider is the capacity of work that a File Server has. Depending on this capacity and speed, a DFS File Server will be able to handle more client requests in a given period of time. As far as the planning for replication is concerned, DFS Fileset Servers where fileset replicas reside should be, as a general rule of thumb, close to the clients in terms of network topology in order to speed up file transfer operations. While this is less crucial in a campus-type network (where servers are more likely to be installed centrally), it may be a key planning issue in a network with low-speed WAN connections. The frequency of propagating updates to replica sites and the amount of data that changes between propagation needs to be taken into account as well as the speed of the network between the replica sites and the DFS File Server holding the read/write fileset. The key decision factors for DFS Fileset Servers are:

- What are the availability requirements? High availability requirements have to take network outages into account (or require special provisions to prevent them). For increased availability, all DCE and DFS services need to be replicated. It is not sufficient to replicate a DFS Server only at the client's location since it requires access to Security, CDS and FLDB Servers for proper operation.
- What performance is to be provided to client machines? If high performance file access is a requirement in a WAN network environment, DFS Fileset Servers need to be placed close to the clients.
- What is the expected ratio between client file access and update traffic due to fileset replication? It is certainly not adequate to replicate filesets that are frequently updated but only rarely accessed by clients. The only instance where this may be considered is when clients need high-speed access to the fileset.

- What is the cost of hardware, software and administration? Replication of any services almost always increases investment and operating costs. This, of course, should be in balance with the other requirements.

DFS FLDB Servers are more likely to be kept in a central location, allowing them to communicate with each another with adequate network speed. Each FLDB Server must be able to communicate with each other FLDB Server for synchronization and sync-site evaluation.

DFS Fileset Server machines need to be sized according to the anticipated disk I/O load. The performance of the I/O subsystem is more important than CPU performance. This can be achieved by fast disks, multiple disk I/O adapters and disks, and/or with fast disk subsystems (such as IBM's SAA disk subsystems). The network interface also needs some attention since it must be able to handle the traffic. DFS Fileset Servers should therefore be connected to fast networks, such as FDDI, ATM, or other high-speed networks. The required CPU performance depends on the type and amount of load imposed to those servers, but in general, there is no need for high-performance CPUs, such as SMP machines.

Sizing FLDB Servers is difficult since the load they have to carry is hard to predict. Unless fileset information is updated very often, DFS clients cache data they have received from FLDB Servers and thus there is normally not a high load on these servers. From observations in large environments it can be said that FLDB Servers do not need high-performance CPUs. As with any server, it is important to monitor resource utilization in order to plan ahead of time for necessary upgrades.

For both, Fileset and FLDB Servers, PowerPC-based systems might be better suited than systems with a Power or Power2 architecture due to the type of work load.

On DFS Fileset Servers that are expected to be heavily stressed, it may be desirable to increase the number of file exporter kernel daemons and token-revocation kernel processes to increase their efficiency (see 7.4, "AIX DFS Server" on page 148, and 7.5, "AIX DFS Client" on page 148, for further details on configuration of DFS Servers and clients).

5.6.2 Planning for DFS Clients

Unlike servers, the location of clients is determined by other business-related factors, rather than by performance-planning issues. DFS client planning for performance therefore concentrates on network performance (the network path between clients and servers) and configuration options on the DFS clients themselves.

Configuration on DFS clients for performance is further described in 7.5, "AIX DFS Client" on page 148.

5.7 Planning for SVC, EMS, and Auditing

Cell and system administrators can configure and make use of a number of serviceability, event and auditing functions within DCE.

The DCE Serviceability Interface (SVC) is used by DCE core services and can be used by DCE applications to perform messaging of conditions or events. The messages are routed to a destination controlled by environment settings or by a message route control file.

See also Appendix A, “Using the DCE Debug and Messaging Facilities” on page 225.

5.7.1 DCE EMS Planning

The DCE Event Management Services (EMS) is an interface and system that provides asynchronous event support for DCE applications.

EMS makes use of a concept of EMS event suppliers and EMS event consumers. An EMS event channel is used to perform asynchronous communication in between EMS event suppliers and EMS event consumers.

EMS event suppliers are any DCE core services or DCE applications. EMS event consumers can be any DCE application having an interest in receiving asynchronous events from local or remote DCE processes.

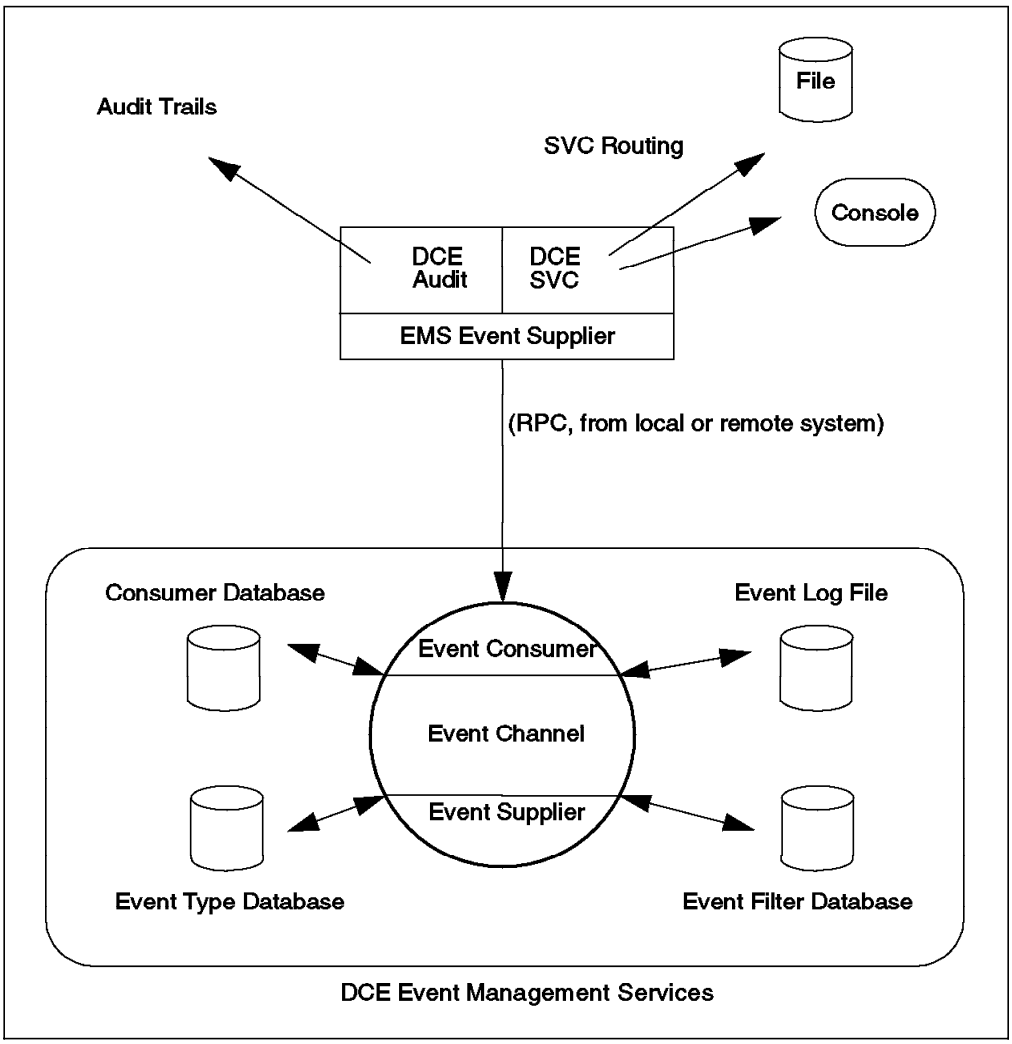


Figure 37. DCE Event Management Service

EMS makes use of a consumer database. This database contains lists of registered consumers. When a connection is broken, an event supplier will make use of the consumer database and try to reestablish the broken communications. The Event Type database contains a list of event type definitions. The information stored is attribute names and which data type is associated with the attribute.

The Event Filter database contains defined event filters. EMS supports a two-stage event filtering by:

1. Supplying a filter at the event supplier. Only events that pass this filter will be transmitted by EMS.
2. Supplying a filter at the event consumer

EMS writes all events to the local EMS Event Log. An Event Log entry is deleted when the event has been transmitted to all consumers that are supposed to receive the Event Log entry.

In order to integrate the DCE Event Management Services (EMS) into the DCE SVC routing, the destination of EMS is used.

In order to estimate the impact on the performance of DCE servers while using EMS, a number of factors can have effects on the resulting performance of an application. Some of these factors are:

1. Staged filtering. By using the filter capacity of the event supplier, the number of data transferred can be minimized.
2. The EMS is functioning on a per-DCE server basis. Activating EMS for more DCE servers in the same system can affect the overall performance of the system.
3. Use network connections with adequate bandwidth. Performing transfer of events can impact network performance.
4. Use the SVC routing table to reduce the types of events transferred to EMS.

To make measurements of the impact of using EMS on a running system, the expected setup of EMS can be tested in a test environment. The numbers generated during the test can be used as estimates for the production environment.

5.7.2 DCE Auditing Planning

DCE auditing contains facilities to perform auditing of events within DCE applications. The auditing events can be logged to an audit log file, called an audit trail.

Filtering is enabled on the basis of name entries, which for example can be principal names, group names and cell names. The condition to write an audit log entry is selected on the outcome of an operation. This outcome can be success, deny, failure or pending.

Due to the possible impact of auditing on the performance of the core services, auditing should only be activated and used when required.

The size of the audit trail file and whether the audit daemon should perform auditing can be controlled by the following environment variables:

- `DCEAUDITON` and `DCEAUDITOFF`, which controls if auditing is performed or not
- `DCEAUDITFILTERON`, which enables filtering
- `DCEAUDITTRAILSIZE`, which sets the audit size threshold

If no file threshold size is specified, the audit trail files will grow uncontrolled.

5.8 Planning for DCE Applications

During application development or in the evaluation phase of a DCE application, the following performance considerations may be of use in evaluating the performance of an application:

- Applications can perform excellently while being tested on a local LAN with sufficient bandwidth available. The performance can be degraded more than expected by using slow, remote links.
- Use tools to evaluate the timing of an application. A network trace can be used to estimate the timeline of the application. If the application is spending most of the time on the network, the application can be sensitive for network congestions or interruptions.

- Validate the time-out behavior of an application due to network slowdown. This will give an indication of the network-related recovery facilities of the application.
- Estimate the capacity that the application will make use of.

For planning and designing of DCE applications that are being developed, see Chapter 8, “Application Development With Performance in Mind” on page 155.

5.9 Planning for Performance Summary

Planning for performance in the DCE environment requires an understanding of the anticipated use of the DCE systems. Using this knowledge, estimates can be established to size the DCE systems for capacity and performance.

The network topology plays a key role in planning for performance. Using a campus-type network or interconnected LANs may normally not cause concerns. The use of DCE from remote networks requires a number of considerations. The displacement of replicas can be used to overcome some drawbacks introduced by remote networks.

Using DCE, the network must be reliable; otherwise delays, congestions or time-outs may occur.

If performance is a critical factor, the size of a cell and the number of DCE replicas can be increased, but there will always be only one master that is responsible for updates for any type of replicated service.

If the anticipated use of the DCE system is unavailable, then the use of worst case estimates can be considered. They can create an estimate of the capacity a DCE system may be able to handle.

Planning for performance requires a knowledge of the expected growth of the system. If the system will be growing and the planning for capacity and performance indicated bottlenecks in the system, the use of additional cells may be considered.

If the use of DCE is a part of a project where the anticipated use of the DCE servers is undetermined, perform rough estimates and make additional resources available in the project budget. The additional resources can be used to add more processing power, memory or other needed equipment for DCE servers and DCE clients. If the project team does not know the methods to use, IBM provides services for benchmark testing.

Make use of tools available to monitor the performance and capacity of the DCE systems (see Chapter 4, “Tools and Methods for Performance Evaluation” on page 63). With the tools, the performance and capacity trends can be established and planning and execution of changes can be performed before problems arise.

Chapter 6. Step-by-Step DCE Set Up Geared to Performance

The installation of a DCE system requires some prerequisites to be in place. For example, DCE is bound to the TCP/IP socket interface. This implies the TCP/IP environment must be properly working before installation of DCE is attempted.

This chapter describes the detailed network and system considerations that are related to the installation of DCE together with other configuration parameter recommendations. The chapter contains separate sections for the DCE core services, both for servers and clients, for the OS/2 DSS LAN Server environment, and for DFS.

6.1 Preparing the Environment

The DCE product documentation specifies recommended minimum configuration requirements that must be considered as basis for a successful installation of the required DCE components.

The main environmental task, before the actual installation, is to prepare sufficient disk space allocation for all the paging datasets for DCE static and dynamic files. This must be carried out on both AIX and OS/2 platforms.

For the network interfaces to be used, the capacity and usage of the network interfaces should be checked.

6.1.1 The Network

DCE makes frequent use of the `gethostbyname()` routine, which resolves a host name into an IP address. This IP address is then used on the interface layer to establish a connection to or a session with the remote host(s). This can either be configured by static or dynamic means.

The static method makes use of the `hosts` file, which is located locally on each host. This method introduces additional work whenever changes have to be applied. On AIX, the *hosts* file is located in the /etc, whereas on OS/2 it is placed in the directory specified by the `ETC` environment variable. A more frequently used method for host name resolution is to use a Domain Name Service (DNS).

AIX Version 4 allows even a combination of the local hosts file, together with either DNS or NIS (Network Information System).

Special cases exist, however, where it is desirable to use the static `hosts` file. This occurs, for example, when the DNS server can be accessed only through a slow line or when the performance benefits of a static, local `hosts` file are to be used.

The dynamic means are represented by the Dynamic Host Configuration Protocol (DHCP), which allows the host name and address to be set dynamically on a requesting client. This method introduces a problem with DCE. The DCE registry relies on a static relationship between host names and IP addresses. If DHCP is used, it must be configured in a way that results in the same host name and IP address on every reboot for DCE machines that are registered in the CDS Host. Failing to do this will introduce unpredictable DCE error conditions.

6.1.2 AIX and TCP/IP

When tuning a system, all applications and network traffic must be taken into account. With all applications in mind, evaluate your system continuously and adjust if necessary.

Generally, DCE services do not require special tuning in order to work well. Without tuning, almost all DCE services can run optimally on most systems. Only if DCE shares common resources with other applications, or on heavily loaded servers, should network interface tuning be considered.

The following descriptions are valid for AIX 4.1 and up. A description for general network interface tuning on earlier AIX 3.2 systems can be found in the on-line InfoExplorer manuals.

6.1.2.1 General TCP/IP Tuning for DCE

AIX Version 4 dynamically adapts many network-related parameters depending on the actual load on a specific network interface. For this reason, there are only a few options to be mentioned.

As an overall limit to memory resources that can be reserved to network I/O buffering by the operating system, the network option `thewall` can be increased to make more memory available to the dynamic process of allocating buffers for network transfers. This parameter can be set with the `no` command, for example:

```
# no -o thewall = 16384
```

This command sets the memory limit for network resources to 16384 MB. This is an upper limit and does not, by default, consume this amount of memory.

As a matter of general tuning for UDP traffic, you should increase the number of send buffers that can be queued up for a network device. To see if there are transmit queue overruns, run the `netstat -i` command, and check if the `Oerrs` (out errors) column for the interface contains a non-zero value. The buffers do not occupy any memory until used; therefore you can set them to the maximum value of 150. The current setting can be checked with the `lsattr -El <interface>` command. The device needs to be disabled before you can set the new value with the `chdev` command, or you can change the value in the database by only using the `-P` flag of the `chdev` command, in which case it will take effect only after the next reboot. The following example changes this value for a Token-Ring adapter on host ev3:

```
# ifconfig tr0 detach
# chdev -l tok0 -a xmt_que_size=150
# ifconfig tr0 ev3 up
```

There are two communication parameters that should be increased for security and CDS servers that are expected to be heavily loaded. They are `sb_max` and `udp_recvspace`. To make sure these parameters are never the bottleneck, set them to high values. The `sb_max` parameter value should be over 1000000 and `udp_recvspace` over 900000. The value for `sb_max` is just a limit and does not cause use or allocation of any memory. The `udp_recvspace` parameter, on the other hand, defines an actual buffer size for incoming UDP frames and the above recommended value may be reduced if the system is short of memory or the expected load is not too high. All actual values should be a multiple of 4096. The values can be set with the `no` command. Add the commands to the /etc/rc.net file so they will be executed each time you reboot the system.

Example lines in /etc/rc.net:

```
no -o sb_max = 1024000
no -o udp_recvspace = 921600
```

Example of a manual change:

```
# no -a
                    ... = ...
                sb_max = 65536
                    ... = ...
         udp_recvspace = 41600
                    ... = ...
# no -o sb_max=1003520 -o udp_recvspace=901120
# no -a
                    ... = ...
                sb_max = 1003520
                    ... = ...
         udp_recvspace = 901120
                    ... = ...
```

As is can be seen in Chapter 2, “Process Flow and Components Affecting Performance” on page 7, DCE core services often create network packets in the range of 500 through 800 bytes in length. If remote networks are involved, which is true in almost any moderate or large environment, the default TCP MSS (Maximum Segment Size) should be increased, if the network components permit. This value can also be set using the `tcp_mssdflt` parameter with the `no` command. AIX uses a conservative default of 512 bytes based on a minimum requirement for any IP routers. If the network components permit, higher values generally yield better network throughput since the chance for packet fragmentation decreases. The following example sets the MSS default to 1024 bytes:

```
# no -o tcp_mssdflt = 1024
```

Note that this changes the MSS default across the entire system. It can also be altered for single target networks by using the `route` command, or for specific network interfaces by using the `ifconfig` command. Be aware that these changes, if not configured properly together with other components, may have a negative impact on the overall performance, rather than the expected improvement.

DFS automatically adjusts all the communication parameters it needs to be changed; therefore you never have to worry about setting communication parameters on a DFS server or client. Note that you may need to adjust things because of other programs or daemons running on the DFS server or client, but never because of DFS itself.

6.1.2.2 Disabling Unused Network Interfaces for DCE

DCE, as a default, uses and advertises all available network interfaces; therefore you have to make sure that all configured interfaces are correct, including any routing. Any network interface which is not supposed to be used by DCE should be specifically disabled for DCE, preferably before DCE is installed and configured. Slower interfaces, especially SLIP or X.25, should be disabled for DCE if not explicitly required. This is done by including them in the `RPC_UNSUPPORTED_NETIFS` environment variable. The variable should be set in the /etc/environment file.

Tip

If you include a multihomed server in your cell, it is a very good idea to put all unused interfaces in this variable, even if they are not configured. This way you protect the cell against the possibility of someone configures the interface later on without taking DCE into consideration.

The following is an example line in /etc/environment file of a machine that has an Ethernet, a token-ring and a SLIP interface configured. Only the token-ring interface should be used by DCE:

```
RPC_UNSUPPORTED_NETIFS=en0:et0:sl0
```

Alternatively, the variable RPC_UNSUPPORTED_NETADDRS can be used to disable a certain address, rather than an interface, from being used by DCE. Example:

```
RPC_UNSUPPORTED_NETADDRS=192.168.32.32:192.168.45.1
```

Disabling network interfaces for DCE should be done on any multihomed machine that runs DCE servers. Other than for DCE, this variable does not have any effect for the use of the interfaces.

6.1.3 OS/2 Warp, TCP/IP and NetBIOS

Similar to AIX or any other operating system platform, network operation parameters on OS/2 Warp may be tuned in order to improve network throughput and decrease CPU load for network-related protocol processing. Unlike AIX, DCE on OS/2 Warp supports NetBIOS sockets. This is covered in the following sections.

6.1.3.1 General Tuning Options

On OS/2, the default configuration of the LAN adapter and protocols is performed through the MPTS configuration program. The configuration of the TCP/IP part is performed by the tcpcfg command, which is represented as an icon in the TCP/IP folder. The internal workings of the OS/2 TCP/IP interfaces are very similar to AIX. The allocation of the mbufs is performed dynamically, similar to AIX 3.2.

The MPTS version provided by OS/2 Directory and Security Server (DSS) makes use of two types of mbufs. They are the small mbufs (256 bytes long) and the large mbufs (4096 bytes long). Although the mbufs are allocated dynamically, they can be preallocated by parameter settings in the CONFIG.SYS:

```
RUN=d:MPTNBINCNTRL.EXE /SM x /LM y
```

where /SM and /LM are the keywords for small and large mbufs. The x and y are the specified number of mbufs. The specified number of small mbufs is rounded up to the nearest multiple of 128. The specified number of large mbufs is rounded up to the nearest multiple of 2.

Table 18. MBUF Allocation Numbers on OS/2 Warp

MBUFs	Minimum Number	Default Number	Maximum Number
small	512	512	4096
large	64	144	180

Using the maximum settings of mbufs, the storage requirements will be approximately 1744 KB, whereas the default settings requires 704 KB.

MPTS provides a method whereby an application can improve performance by indicating to the transport that pointer and parameter validation is not required. By default, this validation is disabled in MPTS. Therefore, applications must validate pointers and parameters before passing to MPTS; otherwise problems can occur.

To enable the pointer and parameter verification by MPTS, the following run command in CONFIG.SYS can be modified to contain:

```
RUN=x:mptnbincntrl.exe /V
```

where /V is the keyword enabling pointer and parameter verification.

Selecting the type of LAN adapters can also have an impact performance. Normally, the decision is to select high-performance LAN adapters, preferably busmaster adapters on machines providing server functions and cost-effective adapters on client machines. The selection of busmaster adapters on servers is advantageous because busmaster adapters perform efficient transfer on the LAN while reducing the load on the host CPU, as compared with non-busmaster adapters.

6.1.3.2 OS/2 MPTS and NetBIOS

The OS/2 DSS use of the NetBIOS socket interface adds many names into the local NetBIOS name table. Together with the session-oriented structure of the NetBIOS socket interface, a high amount of NetBIOS resources must be allocated to the DCE part of OS/2 DSS.

The number of NetBIOS resources free to other NetBIOS applications, including the DSS OS/2 LAN Server and Requester components, will be reduced.

Because of the fixed maximum numbers of the NetBIOS configuration parameters for sessions, commands and names, the consumption of NetBIOS resources must be calculated carefully.

The default settings, set by the OS/2 DSS tuning assistant, provides sufficient NetBIOS resources for smaller environments. Large environments must consider the use of a combination of NetBIOS and TCP/IP because of the fixed limits of NetBIOS resources.

6.1.3.3 TCP/IP Tuning for DCE

When running on token-ring networks, the other parameter having a positive effect is the MTU size. To optimize the LAN adapter and protocol support, the segmentation of frames should be avoided. This can be done by keeping the frames intact when they are transported from the protocol driver to the adapter driver.

Using an IBM Token-Ring shared RAM family adapter would cause the following settings to be modified in the adapter settings

```
XMITBUFSIZE=1536
XMITBUFS=1
RECVBUFSIZE = 512
RECVBUFS=20
```

The reason for the changes are as follows:

1. The transmit buffer size is used to hold the maximum TCP/IP frame that can be created using MTU default size of 1500. The frame contains the LAN header and the MTU, which equals to 1536 bytes.
2. The shared RAM on the adapter is a limited resource. The number of transmit buffers will be limited to enable more receive buffers.

The adapter settings are configured in the PROTOCOL.INI file either by using an editor or by using the MPTS configuration program (Figure 38).

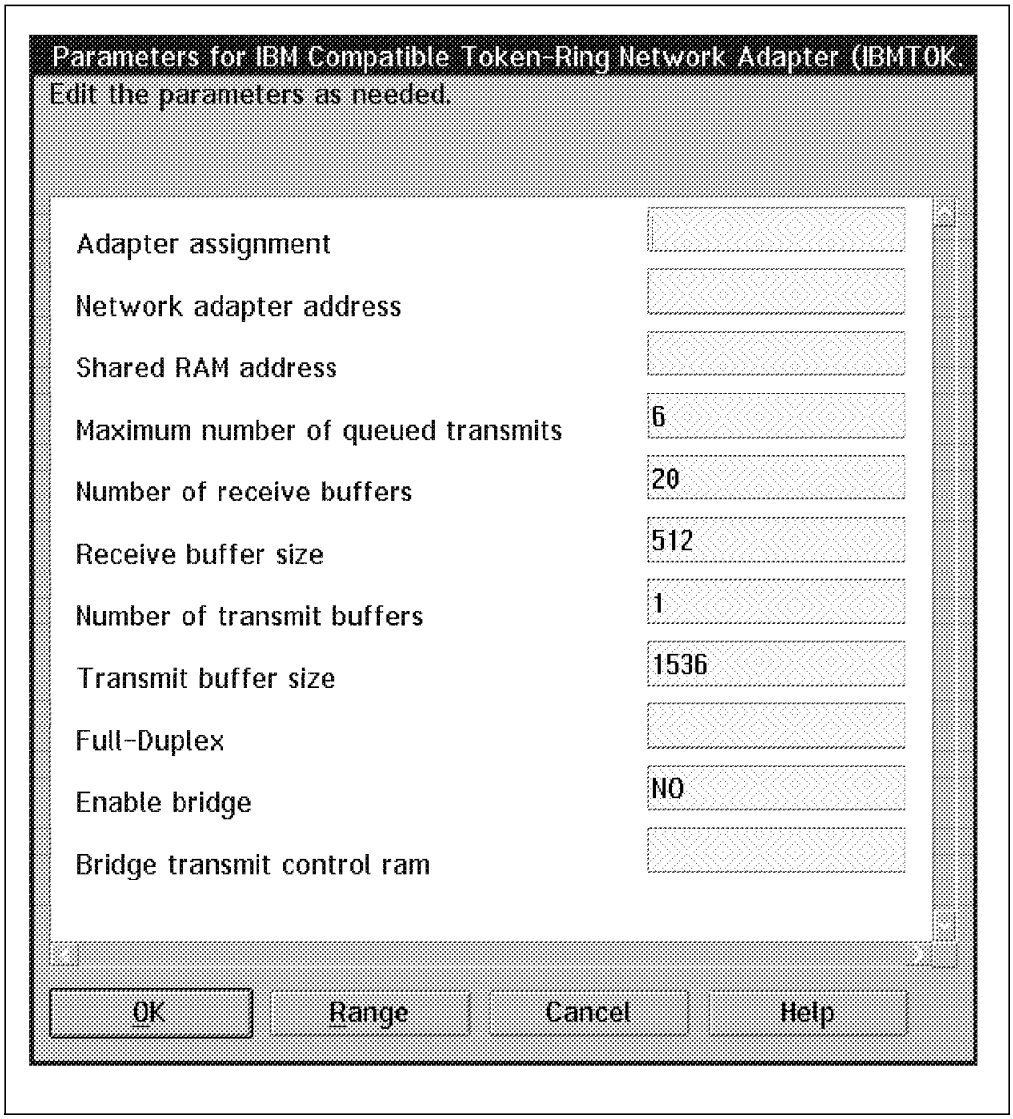


Figure 38. MPTS Configuration for IBM Token-Ring Shared RAM Adapter

On high-performance LAN adapters, the options available for adapter parameter settings of the adapter driver software is much more extensive. In those cases, it is possible to extend the number of transmission and receive buffers.

For example, the IBM LANStreamer family of adapters provide additional memory for receive and transmit buffers. The parameters that in some cases are adjusted from the defaults are the:

- MaxTransmits and MaxTxFrameSize, which are optimized to use up to a maximum of 128 KB of memory
- SizWorkBuf and MinRcvBuffs, which are optimized to a value less than 1 MB. The 1 MB limit is for all LANStreamer adapters installed in a machine.

6.1.3.4 Disabling Unused Network Interfaces for DCE

In the OS/2 Warp environment, DCE will by default make use of all the network interfaces available. If interfaces have to be excluded, the OS/2 environment variable RPC_UNSUPPORTED_NETIFS should be set in the CONFIG.SYS file. Example:

```
SET RPC_UNSUPPORTED_NETIFS=sl0
```

If any local machine IP address has to be excluded for DCE network traffic, the

```
SET RPC_UNSUPPORTED_NETADDRS=192.168.32.32:192.168.33.32
```

can be used. This excludes the specified address(es) active on the local machine from being used by DCE.

6.1.3.5 NetBIOS Sockets

In order to use NetBIOS sockets, the USEMAXDATAGRAMS parameter in the NetBIOS protocol section must be set to YES in the PROTOCOL.INI control file. The default setting of USEMAXDATAGRAMS is NO.

The minimum configuration of the NetBIOS sockets interface for DSS are as follows:

```
NetBIOS sessions = 50
NetBIOS commands = 80
NetBIOS names = 50
```

These NetBIOS resources must be available after all other consumers of NetBIOS resources have started; otherwise the DSS installation will fail. Another consumers of NetBIOS resources is, for example, the OS/2 LAN File and Print Server.

Configuration is performed in the Socket/Multi-Protocol Transport services configuration program. It is started using either the MPTS program from a command line or by using the MPTS Adapter and Protocol Services icon.

The configuration of the NetBIOS sockets parameters can be done using the graphical configuration interface (Figure 39 on page 130).

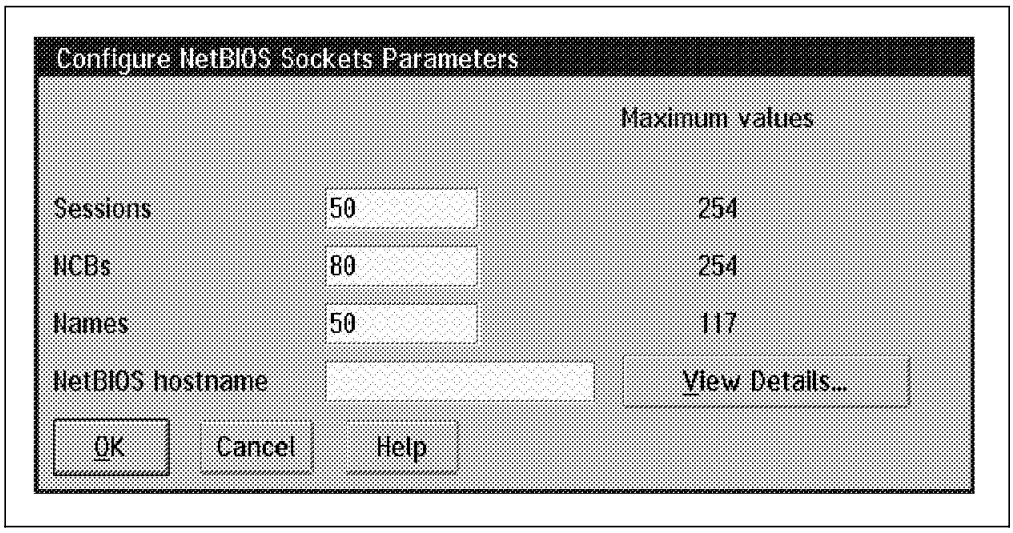


Figure 39. Configuration of NetBIOS Sockets Parameters

Although NetBIOS names are allowed to be 16 bytes in length, the NetBIOS sockets interface will only allow 11 characters. This is due to the fact that internally the NetBIOS sockets interface uses another naming structure where both the name and the socket number are used.

Socket numbers for NetBIOS sockets can be obtained from the `netstat -s` command. The `AF_NB` section lists the NetBIOS sockets sessions.

6.2 AIX DCE Clients

DCE client services are optimized in terms of performance for typical application usage. There are no configuration options available during installation that have a specific impact on performance. On DFS clients, however, there are options to tune with a certain influence on client performance, see 6.7.2, “AIX DFS Clients” on page 139, for further details.

Since DCE relies on the operating system and other underlying components, like the network, DCE will also gain from any performance tuning being done on those components. DCE client services are mainly CPU bound, except for program loading, which is disk-I/O bound. DCE client services do not require any unusually high amount of disk storage or memory.

As a problem prevention matter, it is a good idea to create a separate file system for /var/dce. This makes sure DCE does not interfere with system space needed for other parts of the system, and vice versa. Since DCE stores important runtime data in this storage area, a large print job could, for example, hinder DCE by filling up the /var file system.

Example:

```
# mklv -y dcelv rootvg 2
dcelv
# crfs -v jfs -d /dev/dcelv -m /var/dce -A yes
Based on the parameters chosen, the new /var/dce JFS file system
is limited to a maximum size of 134217728 (512 byte blocks)

New File System size is 16384
# mount /var/dce
```

This example creates a file system of 8 MB and mounts it over /var/dce. Placing this filesystem in a volume group other than rootvg may improve access performance in most cases. It is important to note that this file system has to be created and mounted prior to installing DCE client code; otherwise DCE installation will already have allocated some files which would be unaccessible during subsequent DCE configuration.

A size of 4-8 MB for /var/dce would be sufficient for normal clients machines, like end user desktop stations or small multiuser systems. Since DCE stores various log files under /var/dce, the space usage heavily depends on the size of the log files, which should be monitored over time.

All credentials received from the security service are stored in files under /var/dce/security/creds. These files are never removed by DCE (unless you use the kdestroy command, or after a reboot when the DCE client processes are being started from /etc/inittab) and can thus fill up the file system if you do not take action. The credential files are relatively small, and the file system is therefore more likely to be filled up by its i-nodes, rather than the actual file system space. DCE supplies the rmxcred utility, which should be used periodically to remove old credential files. A good idea is to run rmxcred as a cron job, typically once every day. Note that you must be root to execute rmxcred. If you like, you can create a separate file system for the credentials. See also 11.1.10, “AIX Filesystem /var/dce I-Node Limit” on page 211.

Example: Add the following line to root’s cron jobs to automatically remove credentials every night that have expired for more than eight hours.

```
0 4 * * * /usr/bin/rmxcred -h 8 1>/dev/null 2>/dev/null
```

Info

Even though the /.:/lan-profile sounds like it has something to do with profiling, it has nothing to do with performance tuning. Its only purpose is to contain routing information used by DTS. There is nothing you need to do with this.

For after-installation tuning tips, take a look at Chapter 7, “After-Installation Performance Tuning” on page 141.

6.3 AIX DCE Servers

This section describes considerations for installation and configuration of the DCE core servers only; for network tuning, see the previous sections in this chapter.

As with the clients, there are no tunable options during a normal configuration of a DCE server that affects performance. Since DCE servers usually need to be highly available, some provisions on the platform may be good to be recalled:

- DCE servers should be run on dedicated machines that may even be physically protected and have limited (and controlled) administrator access.
- If heavy load is expected, network tuning may be considered (see 6.1, “Preparing the Environment” on page 123).

- The performance of DCE core servers almost only depends on the CPU performance of the system. Disk I/O and network performance are less important.
- Sufficient memory should be installed in Security and CDS Server machines (see 5.4.1, “Security and CDS Server Requirements” on page 105).
- Additional high-availability features, such as RAID disks, may be considered for DCE core servers.

Since DCE servers carry database-like services in which they store vital information for the whole cell, special attention should be called to storage space and storage media. We will look at the core servers next, and for the DFS Servers, please refer to 6.7.1, “AIX DFS File Servers” on page 138.

6.3.1 AIX Security Servers

The Security Server stores its database, sometimes called the *registry database*, in several files beneath /var/dce/security/rgy_data. This registry files are vital for the Security Server, and there should never be a ″disk full″ condition in the filesystem where these files reside. Since the standard /var filesystem is used by many processes and applications, such as mail and print spooler, /var can easily run out of free space, which would cause the Security Server to abort due to a non-recoverable error.

It is therefore recommended to use a separate filesystem for /var/dce (see also 6.2, “AIX DCE Clients” on page 130), or even for /var/dce/security, in order to be independent of other, sometimes ill-behaving processes. The size that needs to be available for the registry database depends on the number of objects that it is supposed to store. Section 5.4.1, “Security and CDS Server Requirements” on page 105 gives you guidelines for estimating the size of the registry database.

The only tuning parameter for the Security Server process (secd) is the number of listener-threads. This is explained in the after-installation chapter in 7.3.1, “Security Server” on page 146.

6.3.2 AIX CDS Servers

There are no options available in respect to performance when a CDS Server is installed and configured. A CDS Server runs a single process (cdsd) that has no performance-relevant options.

As with the Security Server (and DCE clients), a CDS Server stores vital data in the /var/dce file tree. Without special provisions, this represents a certain risk, as, for example, a large print job could fill up the whole free space in the filesystem. It is therefore recommended to create a separate filesystem and mount it over /var/dce, or on a CDS Server directly over /var/dce/directory/cds, where cdsd stores the CDS database. The recommended size for the CDS database can be estimated using the guidelines outlined in 5.4.1, “Security and CDS Server Requirements” on page 105.

6.3.3 AIX DTS Servers

AIX DTS Servers do not have configurable options during the installation that are relevant to performance. Performance in respect to DTS is of less importance and comes to the question on how can DTS be configured such that it uses as little resources as possible. The DTS Server and client processes (dtsd) are in a dormant state most of the time and therefore do not require resources other than virtual memory during these periods.

Although DTS Servers (and clients) have many configuration options, only the time intervals of their mutual synchronization may have an influence on the system and network usage. Synchronization, by default, takes place once every 24 hours, which does not cause a remarkable load and does not need to be changed in most cases.

In large, distributed environment with many interconnected networks, a proper distribution of DTS Courier Servers helps to reduce network traffic from DTS clients to central DTS Servers.

6.4 OS/2 Warp DCE Clients

The OS/2 DCE client exists in two configuration options: the *Full Client* option and the *Slim Client* option. The difference in the options is basically the full client option supports DCE Servers locally, whereas the slim client option does not.

The DSS installation program will install all necessary program code and data files for both configurations, and the difference is a matter of configuration thereafter using the graphical DCE configuration utility. Prior to install, make sure to meet the requirements explained in 6.1.3, “OS/2 Warp, TCP/IP and NetBIOS” on page 126.

6.4.1 Full Client Option

The Full Client option of OS/2 DCE 2.1 can consist of the following DCE daemons:

- DCE daemon (dced) providing:
  - Endpoint Mapper Service
  - Security Validation Services
  - System Management Services, which includes Host Data Management, Server Control and Key Table Management
- CDS advertiser
- CDS clerk (cdsclerk), which is started by the CDS advertiser for each logged-in principal that performs CDS name lookup
- DTS client daemon

During the configuration process, the Full Client option will create entries in the CDS /.:/hosts/<hostname> directory.

The installation process of the full DCE client is started by the installation program on the DSS CD-ROM. To install the DCE client, select **DSS for DCE** applications. In the DCE Installation selections panel, the options that are intended to be installed are selected. In the following example (Figure 40 on

page 134), the DCE Base Services, Administration GUI and DCE Books are selected.

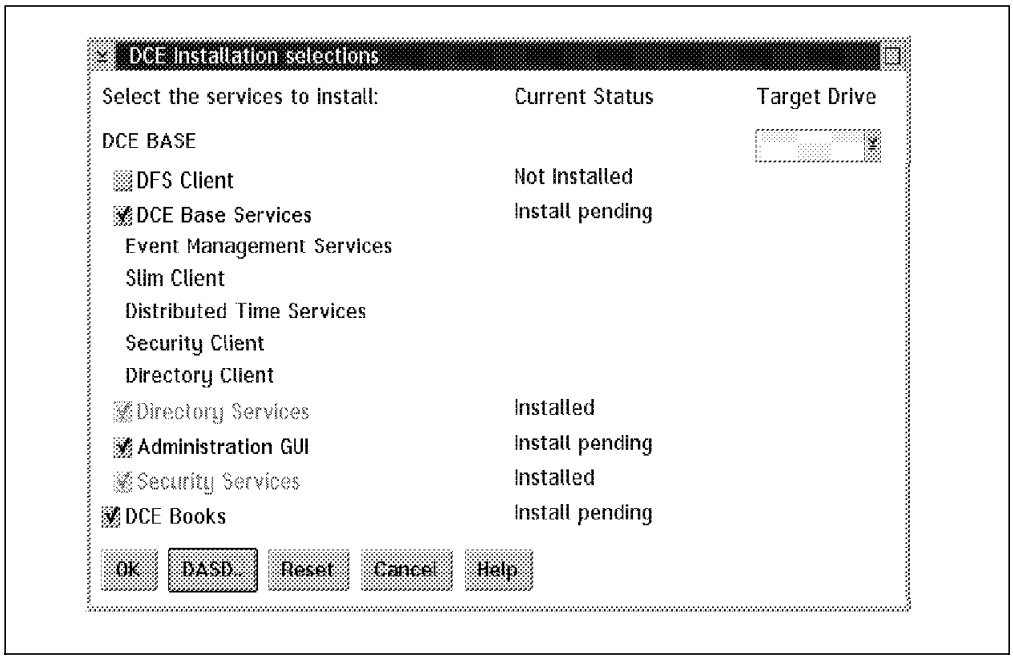


Figure 40. DCECL02.DCE Installation Selections

After DSS has been installed, a reboot is required to activate the changes.

After reboot, the DCE components must be configured. This can be done by selecting either the DCE configuration icon or by starting the DCE configuration program from an OS/2 command prompt and calling dcecfgg, which is the name of the graphical DCE configuration program. In the configuration process, the TCP/IP host names or TCP/IP addresses the of the Security and CDS Servers need to be entered. For an OS/2-based system having the DCE core services placed on NetBIOS, the NetBIOS names of those systems need to be configured.

After the configuration has been performed, the DCE components are ready to be used immediately. There are no configuration options pertinent to performance.

6.4.2 Slim Client Option

The Slim Client option of OS/2 DCE 2.1 consists of a single instance of the cdsclerk, which takes over the role of CDS advertiser in managing the CDS client cache. The Slim Client is limited to the following DCE functions:

- RPCs (authenticated and unauthenticated)
- DCE login
- CDS name lookups

The Slim Client does not register itself within the CDS /.:/hosts directory and does not obtain machine credentials during startup.

The overall memory requirements are reduced considerably compared to at Full Client configuration.

The configuration of a Slim Client does not have any performance related parameters. The configuration is straight-forward. The addresses or names of the DCE core servers and, optionally, DFS parameters must be specified.

6.5 OS/2 Warp DCE Servers

The installation of the DCE Security and/or CDS Server on OS/2 Warp does not impose any performance-related configuration parameters. Installing the systems on selected hardware is straight-forward. The performance of these servers is only dependent on the hardware (CPU performance), the network and the disk I/O performance, although the latter two are of less importance. If heavy load is expected, general OS/2 system and network tuning may be considered as explained in 6.1.3, “OS/2 Warp, TCP/IP and NetBIOS” on page 126.

The installation of DCE Security Server will perform a replication of the registry database when the DCE Security Server is installed as a replica (slave) Security Server. Depending of the network connections and the amount of data to be replicated, the installation process can take some time. The installation program seems to be stopped, but there is network activity as long as the Master Security Server is available.

The installation and configuration of the CDS Server will perform a replication if the server is configured as a replica. The initial replication of the CDS root namespace will consume some time.

6.6 OS/2 DSS LAN Server Environment

Before the introduction of the OS/2 DSS, the IBM OS/2 LAN Server was using an internal structure for registry and directory data. This information was kept in the OS/2 LAN Server NET.ACC file and in the Domain Controller Database (DCDB).

The NET.ACC file contains the user ID and group definitions (accounts) of an OS/2 LAN Server domain. Additional information for Access Control Lists (ACL) for OS/2 LAN Server resources are stored in the NET.ACC file. For the OS/2 LAN Server advanced, utilizing the HPFS386 disk driver, the file directory ACLs are stored within the file system for performance reasons.

Replication of the NET.ACC user ID and group definitions is performed from an OS/2 LAN Server Domain Controller (Primary Server) to any other defined, additional server within the OS/2 LAN Server domain. Replication of the defined ACLs does not occur. ACLs are bound to the OS/2 LAN Server owning the shared resources.

The DCDB is located on the OS/2 LAN Server Domain Controller and contains information on defined, shared applications, printers, file directories and async resources (alias definitions). These resources can be assigned to users during logon (logon assignments).

Replication of the DCDB can be configured from the OS/2 LAN Server Domain Controller to an OS/2 LAN Server Backup Domain Controller. This replication will copy the DCDB files to the backup domain controller.

When an OS/2 LAN Server domain is migrated to DSS, the OS/2 LAN Server Domain Controller becomes a DSS Domain Controller. The OS/2 LAN Server domain becomes a Resource Domain.

The DSS Domain Controller no longer keeps the master copy of the account information stored in the NET.ACC and the resource definitions stored in the DCDB. These definitions are stored in the DCE Security Server and CDS.

To allow the coexistence of previous version of OS/2 LAN Servers and LAN Clients, there are two functions available to bridge information stored in DCE format into information stored in OS/2 LAN Server format.

These functions are placed on the DSS Domain Controller and have the following functions:

- **Registry synchronization** ensures that all updates to the DSS registry database are propagated (transferred) to the NET.ACC file on the Domain Controller. The Registry synchronization is controlled by the sync_required parameter setting in the IBMLAN.INI control file. In addition to synchronizing the NET.ACC file, the DSS registry-synchronization process is also a DCE cell registry replica.
- **Directory synchronization** ensures that related changes to the DSS directory database are transferred to the DCDB. The directory synchronization is performed by the DIRSYNC server service.

The functions of registry and directory synchronization is no longer required when all of the OS/2 LAN Servers and LAN Clients in the DSS Resource Domain have been migrated to DSS Servers and clients.

Note on Realm Administrator Account

To install a DSS OS/2 File and Print Server into an existing AIX cell and thus create a realm, the installation program uses an upper-case cell administrator account. The installation program allows input of a mixed-case cell administrator account, which normally will be the cell administrator account. The realm being installed requires an upper-case administrator. This ID is created during installation.

6.6.1 Adjusting Disk Cache Size Before Installation

The OS/2 LAN Server makes use of a disk cache, which is either the HPFS386 disk cache for an advanced OS/2 LAN Server or the diskcache specified in CONFIG.SYS, for the entry OS/2 LAN Server. The size of the disk cache is optimized for the OS/2 LAN Server environment and does not take into account the requirements of the DSS DCE components. The default settings of the disk cache for an advanced OS/2 LAN Server (Warp Server) are specified in the table below.

Table 19. LAN Server 5.0 Advanced Default HPFS386 Cache Sizes

Available Memory	HPFS386 Cachesize Value
7 MB to 8 MB	838 KB
8 MB to 9 MB	506 KB
9 MB to 10 MB	1222 KB
11 MB to 12 MB	1652 KB
15 MB to 16 MB	2630 KB
19 MB to 20 MB	3748 KB
23 MB to 24 MB	4992 KB
27 MB to 28 MB	7812 KB
31 MB to 32 MB	7812 KB
32 MB to 96 MB	(Available Memory) -24 MB
more than 96 MB	(Available Memory) * 0.75

Taking the requirements for the DCE components into account, the default cache size does allocate considerable memory for the HPFS386 cache. The value of the cachesize parameter must either be decreased (adjusted down) or additional memory must be installed.

If an adjustment is not performed, paging may occur. The result of paging is performance degradation.

6.6.2 Adjusting MPTS During Installation

During installation of the DSS File and Print Server, another version of the MPTS may be installed, replacing the existing code. The LAN adapter drivers installed will be replaced with the ones supplied by the version of MPTS that is installed.

The protocol sections of MPTS can be adjusted during installation. Especially the NetBIOS parameter settings may be adjusted if the DCE components are intended to use the NetBIOS socket support. In this case, the NetBIOS resources allocated to the OS/2 LAN Server components will be changed. If the resulting NetBIOS resources are not sufficient, additional NetBIOS resources can be configured by using the NetBIOS over TCP/IP facilities of the OS/2 LAN Server component (TCPBUEI support) or by adding an additional adapter resources (NetBUEI). Using other NetBIOS protocol drivers for the OS/2 LAN Server component, such as the IBM AnyNet product family, may also be considered.

6.7 DFS Setup

Before the installation of the DFS Servers and clients, planning of their function and their location should be done. With correct planning, the DFS Servers and clients can provide excellent performance. The following sections describe important provisions for the setup of DFS Servers and clients.

6.7.1 AIX DFS File Servers

The normal installation procedure for DFS Servers in AIX is simple and straight-forward, and there are no performance-related configuration options during the installation using the provided SMIT panels. It is, however, wise to take some basic operating system performance guidelines into account when configuring disk storage areas for DFS services.

Aggregates: You can choose between two kinds of aggregates: LFS (Logical File System) and Non-LFS (AIX JFS or AIX CD-ROM file system). If you use LFS, you will get the following additional abilities:

- Multiple filesets in one aggregate
- Use of extended ACLs
- Replication of filesets
- Fileset headers are used, which makes it more likely that fileset problems can be fixed using the `fts syncfldb` and `fts syncserv` commands, and the `salvage` program can be used to repair aggregates.
- Possibility to enable and disable honoring of `setuid`, `setgid` and special device files

Aggregates on a DFS File Server are most likely storage areas on the disk(s) with heavy access. The tuning of the disk subsystem, or the use of high-performance technologies, such as IBM's SSA disks, is therefore very important for the overall DFS File Server performance. Some guidelines for disk tuning are:

- Spread the data among different disks (striping), either by using RAID technology or by setting the **RANGE** attribute of the housing logical volume to *maximum*.
- Connect multiple disks through multiple adapters to the system; for example, use no more than two disks per SCSI adapter.
- Position the logical volume containing an aggregate in the *middle* area of the disk(s) or in the *edge*, whichever is faster (depending on the disks used).
- Run the `reorgvg` command against existing logical volumes to reorganize them on the disks according to their attributes.
- Place logical volumes containing aggregates on disks other than those that contain the paging space.
- Place logical volumes containing aggregates on disk(s) which are not in the rootvg volume group.

When you create aggregates, you can choose between different `block-` and `fragsizes`. If your average file size is over 16 KB, set both `blocksize` and `fragsize` to 16 KB; otherwise use the default ones, which are 8 KB for `blocksize` and 1 KB for `fragsize`. The way the file server is implemented makes these values optimal. You should not use larger sizes because of this (some DFS implementations do not even support larger sizes).

6.7.2 AIX DFS Clients

Before you install and configure DFS on a client, it is a good idea to create a separate file system for the DFS cache if you use a disk cache. By doing this, an incorrectly defined cache will not fill up /var/dce, and other DCE programs will not use space necessary for the cache.

The default mounting point for this file system is /var/dce/adm/dfs/cache. The necessary size of a disk cache depends on the usage. The default value of 10 MB is enough for most users, for example for single user systems. Remember that the file system must be 15 percent larger than the cache due to internal DFS data, which is perfectly met with a separate filesystem of 12 MB in size.

If you use a memory cache, make sure the host is not memory bound. Do not use more than 25 percent of real memory, and usually 5-10 MB (or 5-10 percent of total physical memory) for a memory cache is sufficient. See also 3.5.3, “Cache Size” on page 58.

Example of creating of a file system (12 MB) for a 10 MB DFS disk cache:

```
# mklv -y dfscachelv rootvg 3
dfscachelv
# crfs -v jfs -d /dev/dfscachelv -m /var/dce/adm/dfs/cache -A yes
Based on the parameters chosen, the new /var/dce JFS file system
is limited to a maximum size of 134217728 (512 byte blocks)

New File System size is 24576
# mount /var/dce/adm/dfs/cache
```

If you can, put the disk cache on a disk other than one that contains paging space. Also make sure the file system is not fragmented and, if possible, is placed at an optimal position on the disk (the placement depends of what kind of disk you have). To put it in a volume group other than rootvg is even better because it will get its own JFS log.

If you already know that the client will be used by many users at the same time, you can increase the number of threads the `dfsbind` process can use. Also, increasing the `namecache` for DFS will improve performance for users accessing many files during short periods of time. See 7.5, “AIX DFS Client” on page 148, which describes the available tuning options after the initial installation.

For more information on how to set and tune DFS Client parameters after the client code has been installed and configured, refer to 7.5, “AIX DFS Client” on page 148.

6.7.3 OS/2 DFS Client

The OS/2 DFS client configuration requires the setting of the DFS client cache size. The cache can either be in memory or placed on disk.

The performance of file accesses can be optimal with the correct setting of the cache size. Selecting a DFS memory cache provides the best performance in cases where enough memory is available. The memory usage is approximately identical to the size of the memory cache. The default setting of 1 MB is for most cases sufficient, but with memory available, an increased cache size can improve performance. A trade-off exists with the default OS/2 disk cache. This

disk cache also consumes memory. Adjustments of the OS/2 disk cache size and the DFS cache size must be considered.

Figure 41 shows an example of the OS/2 Warp DCE Configuration Catalog window where the location and the size of either memory or disk cache can be configured.

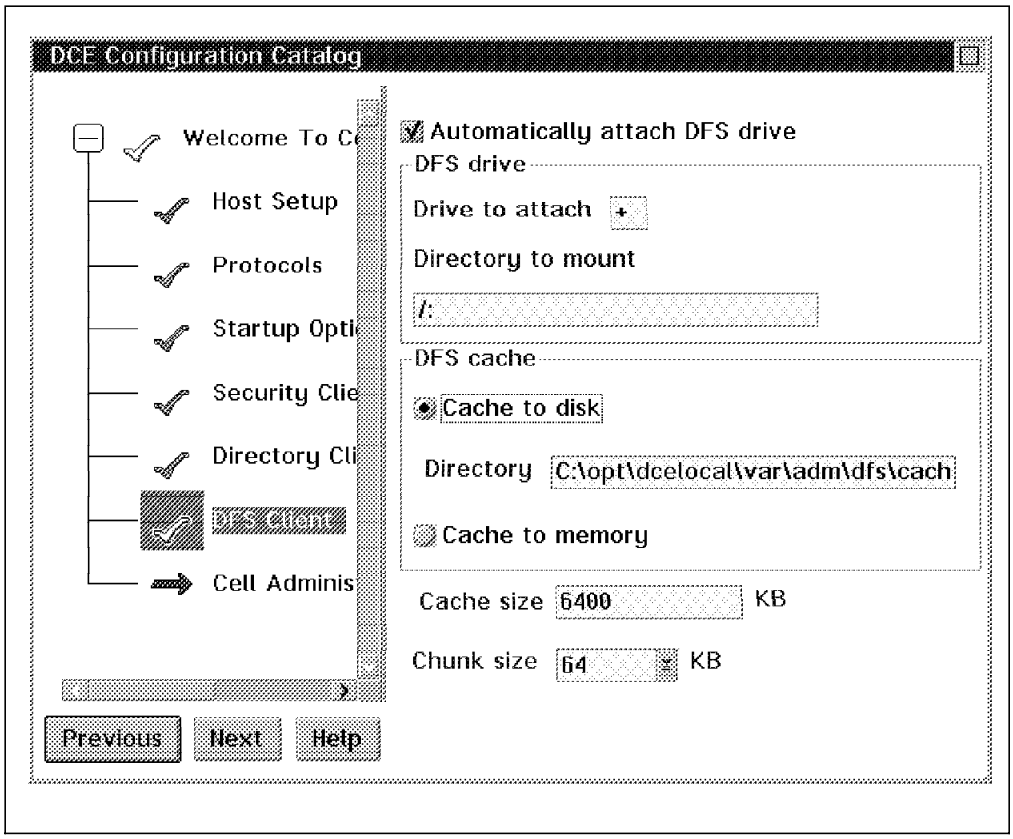


Figure 41. OS/2 DFS Client Installation Parameters

The total amount of memory consumed for the OS/2 DFS client can be estimated to be the size of the DFS client plus the memory DFS cache.

Chapter 7. After-Installation Performance Tuning

The process of keeping a system within the performance expectations and requirements is an ongoing task. The dynamics within a running system requires the administrator to look for bottlenecks and revalidate the assumptions used during the planning phase.

The performance tuning to be applied depends on the system usage. Planning and installation of the system created the intended infrastructure. The daily use of the system and the system modifications caused by the day-to-day operation can cause introduction of unintended performance bottlenecks.

Also, rather than trying to install and configure every single component in a DCE cell in advance based on estimates that might have been wrong or uncertain, it is a good idea to closely watch system performance parameters in an early phase of an installation, or in a designated pilot, and base further performance tuning activities on the results from these real-life measurements.

This chapter describes methods and recommendations that can be used to change performance-relevant system settings in a DCE/DSS environment after all the components have been installed and configured and after the cell has been brought to life.

7.1 General System Monitoring and Tuning

From what we have seen in the last chapters, DCE performance tuning is much more system and network tuning than only DCE tuning. In fact, DCE has few tunable options, but relies on a smoothly performing infrastructure.

This section summarizes some general system and network tuning activities that should be considered regularly, especially if the DCE system environment is frequently changing or rapidly growing, for example by continuously adding client systems or by the introduction of DCE-based applications. The following resources should be monitored regularly, at least on server machines:

Memory usage	If memory becomes short, paging (swapping) may occur, which slows down server performance.
Paging (swapping)	If paging (called swapping on OS/2) occurs, it is usually an indication that the machine does not have enough memory. Paging should be avoided on servers by adding sufficient memory.
Disk I/O rates	If the disk subsystem cannot handle the requested data transfers fast enough, the CPU may be hindered by other tasks waiting for I/O requests to complete.
Network I/O rates	A certain combination of CPU and network adapter is capable of handling a certain rate of network I/O transfers, which can usually be improved by tuning.
CPU utilization	A CPU should be capable of handling all incoming requests. If the CPU utilization is too high, requests may be dropped, causing client services to fail.

Interface status	If a server has any kind of resource problems, or if there is a network problem, network requests may be dropped. The various counters available through the `netstat` command can be used to indicate such a situation.
Network status	The network must be monitored to ensure it is in a healthy state. Communication dropouts can hinder any service, not just DCE.

7.2 Server Preferences

One of the key points with DCE is that the whereabouts of the DCE and DFS services are not necessary information for the clients. This is usually a good thing, but sometimes we can gain performance by specifically choosing the servers we want to use.

Typical situations where this might be necessary are discussed in 2.5.2, “The Network Topology” on page 41, and in 2.5.3, “Server Replication Policies” on page 42.

The preferences for most, but not all, of the DCE and DFS services can be set. Since this is mostly basic DCE functionality, their implementation on AIX and OS/2 Warp are identical or similar. We will therefore describe them together and point out the differences.

7.2.1 Security Service, the pe_site File

By default, the security client queries the CDS in order to get binding information for a Security Server. The CDS responds with a binding vector containing a list of all Security Servers in the cell. The client will then try the first one in that list. If the client, after a time-out period, has not made contact, it will try another one, and so on. This will automatically balance the load on the Security Servers.

An exception to this is when a DCE client starts up, because the DCE client services are required to authenticate with Security Services before they can access CDS. For this reason, the security client keeps a list of binding information for the Security Server(s) in a file, called the pe_site file, located in /opt/dcelocal/etc/security, which is linked on AIX to /etc/dce/security. On OS/2 Warp, this is x:optdcelocaletcsecurity (x being the drive selected during installation). This file contains a list of all Security Servers available in the cell at the time the ps_site file was created, each having two entries for either the TCP and the UDP binding. The following example shows the contents of a pe_site file with three Security Server entries:

```
/.../my_cell 00616b74-98d4-1eae-b63d-02708c2d0132@ncacn_ip_tcp:192.168.32.1[]
/.../my_cell 00616b74-98d4-1eae-b63d-02708c2d0132@ncadg_ip_udp:192.168.32.1[]
/.../my_cell 00616b74-98d4-1eae-b63d-02708c2d0132@ncadg_ip_udp:192.168.33.1[]
/.../my_cell 00616b74-98d4-1eae-b63d-02708c2d0132@ncacn_ip_tcp:192.168.33.1[]
/.../my_cell 00616b74-98d4-1eae-b63d-02708c2d0132@ncacn_ip_tcp:192.168.34.1[]
/.../my_cell 00616b74-98d4-1eae-b63d-02708c2d0132@ncadg_ip_udp:192.168.34.1[]
```

The pe_site file contains an implicit order. For example, when it is used by the security client, the first entry is always tried first. If it fails, the second entry is chosen, and so forth. To select the order of preference, we can rearrange the order of servers listed in the file. (Note that there are two lines for each server).

Rather than using CDS for obtaining binding information, which always results in a random selection of any Security Server in the cell, the security client can be directed to get the bindings from the pe_site file. To tell the client to use the pe_site file, you have to set one of two environment variables. If you set TRY_PE_SITE=1, the pe_site will be used first, and if no servers in the file are available (the file does not have to contain all servers), the CDS will be queried for servers. If you set BIND_PE_SITE=1, then only the pe_site will be used, resulting in an error if none of the servers in the pe_site file is available. If none of these variables are set, or if they are set to a value other than 1, CDS will be used to obtain binding information.

By setting either environment variable and ensuring the correct order of the Security Servers in the pe_site file, server preferences can be implemented for Security Services. As a side effect, it also allows you to specify the protocol sequence (TCP or UDP) being used between the security client and the first server in the file.

Differences in the AIX and OS/2 Implementation

With DCE 2.1 for AIX, the pe_site file is created during configuration of the DCE client services. It is the up to the administrator to update this file as new Security Server replicas are added to the cell or when servers need to be listed in a specific order in order to implement server preferences. The pe_site file can be re-created using the chpesite command, but when a specific order of the servers listed in the file needs to be in place, manual intervention or roll-your-own methods need to be applied.

If you decide to make use of the TRY_PE_SITE (or BIND_PE_SITE) environment variable, you will best set them in the /etc/environment file. For example, to activate the use of the pe_site file, you could run the following commands:

```
# echo "TRY_PE_SITE=1" >>/etc/environment
```

A reboot is necessary for the machine to re-read /etc/environment.

Note

The chpesite command always puts the Master Security Server at the top in the pe_site file (the first two lines). This must not be changed on either the Master or any replica Security Servers. The order of the Security Servers as they appear in the pe_site file should only be changed on DCE clients machines.

This is specific to DCE 2.1 on AIX, but it might change in future releases (or even updates).

On OS/2, the pe_site is automatically re-created every time the DCE core service are started, which typically occurs when the systems are rebooted. The Security Servers in the pe_site file are listed in a random order. Any manual changes to this file will be lost after a restart of DCE. You can, however, take advantage of this feature by adding the desired information to the specific machine profiles in CDS, from where the information for updating the pe_site file is taken. See Appendix D, “Setting Preferred Security Server” on page 241, for a detailed description on how to achieve this.

Another important difference to DCE on AIX is that the TRY_PE_SITE environment variable on OS/2 is set in config.sys by default. This entry is added during installation of DCE. Only if you do not want it to be used, will you have to edit the config.sys file and remove the corresponding line.

OS/2 can make use of the BIND_PE_SITE environment setting the same way as AIX. It is not included into the config.sys file by default.

7.2.2 Cell Directory Service

The CDS Servers will periodically, every 10 minutes, send out a broadcast informing the CDS clerks about their existence. Clerks receiving this broadcast, for example clerks on the same network, will add the information to their caches.

You cannot set preference for a particular CDS Server on a client machine. However, if you make your own application, you can use the site_bind_update() API to select a CDS Server.

If the client can not receive a broadcast from a server (they may be separated by a router for example), you can manually define a server in the CDS clerk's cache with the dcecp (or cdscp) or command, as in the following examples:

```
dcecp> cdscache create ev1 -binding ncacn_ip_tcp:9.3.1.68
```

or

```
# cdscp define cached server ev1 tower ncacn_ip_tcp:9.3.1.68
```

Note that the cdscp command has a set preferred server option, but this only tells cdscp what server to use, not the CDS clerk.

7.2.3 Time Service

The DTS client queries the CDS for DTS Servers to use for time synchronization. By default, this occurs once every 20-24 hours. DTS makes use of CDS profiles, a means of setting preferences in a search list. Normally there is no need to set preferences for this service since its network traffic is very low and therefore negligible.

DTS synchronizes system clocks, which is required (within limits) by DCE services. DCE core services require that system clocks are within five minutes, DFS Servers require synchronization within ten seconds. DTS uses special techniques to adjust clocks both forwards and backwards without abrupt time changes. This must be ensured by any other technique used in place of DTS.

Another simple approach using a specific server for DTS is to use the dtsdate command every time before DCE gets started on a system, for example, on an OS/2 client which is usually powered off after working hours. This command should not be used while DCE is running because it does an abrupt time change. Example:

```
# dtsdate ev1 1
```

This example synchronizes the local system's clock with the clock of ev1. The dtsdate talks with the DTS Time Server on the remote host. It will fail if the remote host is not a DTS Time Server.

Tip

You may be inclined to use standard UNIX tools like the `setclock` command. Be very careful if you do this because this immediately sets the time to the precise time of the server. This could cause a change back in time which could confuse DCE and could cause problems. The DTS service handles this problem by adjusting the time closer and closer to the server time, and not simply setting the time to the server time.

DTS system resource usage may not be negligible on Slim Client machines. Therefore many administrators choose to only run DTS on servers and use `dtsdate` or other methods, such as NTP, on clients. Contrary to what it says in some DCE documentation, the use of DTS is not required (though time synchronization is very important within the cell).

7.2.4 DFS Fileset Servers

When a DFS client needs to access a file from a DFS Fileset Server, its Cache Manager (CM) queries the FLDB (Fileset Location Database) about the location of that file or any read-only replicas. The CM caches this information and uses it for subsequent accesses to the same fileset. It also associates a *rank* with each File Server, for example it prioritizes the list of Fileset Servers based on criteria such as the IP address (servers on the same subnet get higher priorities).

The priorities, or preferences, are set with a number-ranking system. The numbers go from 1 through 65534, with 1 being the highest preference.

The default ranking is as follows:

Table 20. Default DFS Ranking

Criteria	Default Rank
Local Machine is a DFS Server	5000
Server is in the Same Subnet	20000
Server is in the Same Network	30000
Server is in a Different Network	40000

These settings can be changed in order to adapt to a certain configuration and to set explicit server preferences.

To set the preferences, you use the `cm setpreferences` command. To list them, use `cm getpreferences`, as in the following example:

```
# cm getpreferences
ev2.itsc.austin.ibm.com                         20004
ev4.itsc.austin.ibm.com                         20011
# cm setpreferences -server ev4 19000
# cm getpreferences
ev2.itsc.austin.ibm.com                         20004
ev4.itsc.austin.ibm.com                         19003
```

In this example, ev2 and ev4 are two Fileset Servers in the same subnet (as they have a default rank of 200XX). The example then increases the priority for ev4 by specifying a lower rank (19000).

To minimize the chances that multiple machines have the same rank, the Cache Manager adds a random number in the range of 0 to 15 to each rank. This is done to the default settings and when you set the preferences manually.

The settings are only valid until next reboot. Thus, you should set preferences in one of the startup scripts.

7.2.5 DFS FLDB Servers

Setting the preferences for a FLDB Server works the same way as for DFS Fileset Servers. Include the `-fldb` option on the `cm getpreferences` and `cm setpreferences` commands to specify that the command applies to FLDB Servers instead of Fileset Servers.

Example:

```
# cm setpreferences -fldb -server ev4 19000
```

Setting preferences (ranks) for FLDB Servers can be important, or at least beneficial, when FLDB Servers are replicated in a WAN environment. If correct preferences are not used, a DFS client may access FLDB Servers across a WAN connection, although there might be an FLDB Server in close proximity to the client.

7.3 AIX DCE Core Servers

Once DCE services have been installed, configured and are in operation, it may turn out that some tuning becomes necessary. This happens most often in cells where the growth is hard to predict, or when the growth is slow which does not justify installing high-performance servers in the first place.

The following sections describe the options that are applicable to DCE Security and CDS Servers for the purpose of tuning after they became productive.

As a summary from previous chapters, the most important tuning factor for DCE core servers is the performance of the systems—namely their CPU—they are running on. They are explained in 5.4, “System Sizing for Capacity” on page 105, and in 6.3, “AIX DCE Servers” on page 131, and thus we will not review them here.

7.3.1 Security Server

The Security Server runs the secd process that comprises the whole Security Server role. The secd process relies on a reliable, smoothly performing platform in terms of memory, CPU performance, network and disk I/O performance.

7.3.1.1 Listener-Threads

The only tunable option secd provides is the number of *listener-threads* that are created for processing client requests. The default of five listener-threads can be increased up to a maximum of 15 by means of a command line option of secd. This can be achieved by editing the file /etc/rc.dce and adding this option to the command line of secd at the very end of this file. The following example shows the last lines of /etc/rc.dce, where the number of listener-threads have been changed to 10 with the `-t 10` parameter (highlighted).

```
# The mkdce and mkdfs commands uncomment the appropriate daemons below
# (adding appropriate options) as the machine is configured.

daemonrunning $DCELOCAL/bin/dced -b
daemonrunning $DCELOCAL/bin/secd -t 10
#daemonrunning $DCELOCAL/bin/auditd
daemonrunning $DCELOCAL/bin/sec_clientd
daemonrunning $DCELOCAL/bin/cdsadv
#daemonrunning $DCELOCAL/bin/cdsd
#daemonrunning $DCELOCAL/bin/gdad
daemonrunning $DCELOCAL/bin/dtsd -c
#daemonrunning $DCELOCAL/bin/bosserver
#daemonrunning $DCELOCAL/bin/dfsbind
#daemonrunning $DCELOCAL/bin/fxd
#daemonrunning $DCELOCAL/bin/dfsd

}
```

In many cases, however, it is not necessary to increase this number. A higher number enables the Security Server to process more client requests at a given time, but they are not serviced faster, as the CPU performance cannot generally be increased by adding threads. Increasing the number of listener-threads is therefore adequate in environments where a large number of clients request services from the Security Server at about the same time. The Security Server needs to be restarted for this change to become activated.

7.3.1.2 Filesystems

If it turns out during normal operation (or after a problem) that the filesystem for the registry database becomes too small, it can easily be enlarged thanks to AIX's Logical Volume Manager and the dynamic file system on top of it. The Security Server stores the registry database in /opt/dcelocal/var/security/rgy_data, which is a link to /var/dce/security/rgy_data. It is important that there is at least as much free space in this filesystem as the registry requires for temporary storage of two copies of the registry (see also 5.4.1, "Security and CDS Server Requirements" on page 105). If it turns out that the filesystem housing this directory should be separated from the underlying one, this can be done with standard AIX commands while the Security Server is stopped. The contents of any directory that is to be moved to the new filesystem must be saved and restored after the new filesystem has been mounted.

7.3.1.3 System Environment

It is important to monitor the usage of the system resources on Security Servers in order to proactively maintain high availability with an acceptable performance.

Replica servers help to keep the impact on the running cell minimal while doing maintenance or upgrades on a Security Server. Maintenance on the Master Security Server, however, prevents any updates to the registry, including password changes. If this is unacceptable, or if the down-time of the Master Security Server may not be predictable, the role of the Master can temporarily be moved over to a replica server, allowing further updates to the registry while the Master is unavailable.

7.3.2 CDS Server

The CDS Server process, cdsd, has no command line parameters or other options pertinent to performance. Tuning a running CDS Server therefore concentrates on the system environment, namely memory, CPU performance, and disk and network I/O bandwidth. Adjusting filesystem sizes may become necessary as a namespace grows over time. Having a separate filesystem for the CDS database, or creating one after the cell has become productive, is a good idea to increase availability and to achieve a better level of control.

As with Security Services (see last section), replication is the key to keep the impact on the running cell as minimal as possible during maintenance or upgrades on a CDS Server.

Monitoring the use of the system resources, primarily filesystem and disk free space, is a very important and an ongoing task in daily systems administration.

7.4 AIX DFS Server

If a DFS File Server is waiting a lot for disk I/O, an increase in the following two parameters to the fxd daemon (the file exporter daemon) may increase performance.

-mainprocs This specifies the number of main kernel processes that process data and token requests from DFS clients. The default number is four.

-tokenprocs This specifies the number of kernel processes that accept token revocation requests from DFS clients. These requests occur when a client reconnects with a file server after losing contact and wants to reestablish the tokens it had. The default number of processes is two.

Good starting values for tuning are 10 for -mainprocs and 4 for -tokenprocs. Check the system during heavy load. If it is still waiting a lot for I/O, try increasing them step by step until the wait is minimal. Note that many other things can cause I/O wait; so you will probably not eliminate I/O wait.

These two parameters can be set in /etc/rc.dfs (which is a link to rc.dce) at the end, where the commands to be started are listed.

Example line in /etc/rc.dfs (line is spilled due to length):

```
...
daemonrunning $DCELOCAL/bin/fxd -mainprocs 10 - tokenprocs 4 -admingroup \
subsys/dce/dfs-admin
...
```

7.5 AIX DFS Client

After installing and running a DFS client using default parameters, it might become necessary to change some of its parameters in order to improve performance. Bear in mind, this is normally not required for most type of DFS client systems or applications. Only if your DFS client system is a heavily used multiuser system, using DFS for a major part of its file input/output, might you consider performance tuning. Also, DFS client performance tuning is more effective if most of the file accesses are of a certain type, for example large files

with sequential access. If this is not the case, for example the DFS client accesses small and large files, random and sequential, it is more difficult to find the correct tuning options, or some might even decrease the performance.

There are some options to tune a DFS client. We will discuss them in the following sections.

7.5.1 DFS Client Cache

Make sure the size of the cache is not too small because this is usually the most important tuning factor for a DFS client. Let a user do his/her ordinary work for a while, and check the usage of the cache with the `cm getcachesize` command. If the usage is more than 70-80 percent, the user would probably gain from a larger cache. There is no downside to having a large cache (with the exception of many cold cache hits, as explained in the following section); so if there is enough free disk space available, an increased DFS cache can only benefit from it. You can also use the `dfsstat` command to check how a system uses the cache (see 4.8, “DFSSTAT” on page 81).

Memory Cache: The most common reason for using a memory cache is when you do not want a cache at all. For example, in cases of a large number of cold cache reads, an extra DFS disk cache may actually slow DFS operations down. This would be a good case for a small memory cache.

Watch out for the system becoming memory bound. See Chapter 4, “Tools and Methods for Performance Evaluation” on page 63 for tools to use. If that happens, you can either shrink the memory cache size or change to a disk cache.

The size of the memory cache can be changed by changing the last field in the /etc/dce/CacheInfo file. The number given is in kilobytes. You must restart the DFS client, which means in fact to reboot the machine for the change to become active.

To change from a disk cache to a memory cache, add the flag `-memcache` to the `dfsd` command found in the last several lines of /etc/rc.dfs. Note that the size of a memory cache is still defined in the /etc/dce/CacheInfo file along with the path information for a disk cache. However, the path information is ignored when a memory cache is selected.

Disk Cache: To change the size of the disk cache, two things need to be done. First, if you have a special filesystem for the cache, change the size of the file system (remember that the filesystem has to be 15 percent bigger than the cache itself).

Example of increasing the cache file system by another 4 MB (equals to 8192 512-byte blocks):

```
# chfs -a size=+8192 /var/dce/adm/dfs/cache
```

Secondly, change the last field in the /etc/dce/CacheInfo file with an editor. The number given is in kilobytes.

To make the change active, either reboot the machine in order to restart the DFS client, or set the new size with the `cm setcachesize` command.

Use the dfsstat utility, see 4.8, “DFSSTAT” on page 81, to check the usage of the cache. If there is mostly cold cache reads, a change to a memory cache could increase performance.

To change from a memory cache to a disk cache, remove the flag -memcache from the dfsd command in /etc/rc.dfs. Be sure you have created the cache file system before you restart DFS and that there is enough free space for the disk cache. See also 6.7.2, “AIX DFS Clients” on page 139.

7.5.2 Chunk Size

The DFS client reads data from the server a certain amount of bytes at a time. These fixed-length portions during a transfer are called *chunks*, and their size is called the *chunk size*. The chunk size can be configured to values between 8 KB and 256 KB. The default size depends on whether a memory or a disk cache is used. It is optimal in most cases and does not need to be changed.

Memory Cache: The default chunk size for a memory cache is 8 KB. The higher the size, the higher the performance when large files are sequentially accessed, as long as your network can handle the load. In a fast network, like Fiber Distributed Data Interface (FDDI) or an Asynchronous Transfer Mode (ATM), the largest size, which is 256 KB, can be used. With a normal Ethernet, you probably never want to exceed 64 KB. If the chunk size gets too large, it may cause queue overruns for the interface.

Disk Cache: The default chunk size is 64 KB for a disk cache. If you have a lot of hot cache reads of large files (>256 KB), you will gain performance by having a chunk size of 128 KB since the data in the cache will be contiguous on the disk. Chunks do not necessarily lie next to each other on the physical disk. Normally, you should not use any values other than 64 KB and 128 KB.

The chunk size can be set with the -chunksize parameter to the dfsd command at the end of /etc/rc.dfs.

One case when a smaller chunk size can improve performance is when more than one client does byte locking within a file (this technique is common in the PC world). If the areas of locking are in different chunks, the clients will work against the DFS cache, and there will be no network traffic at all (for reads). Depending on the size of the byte-locking area, you may want to decrease the chunk size to get this effect.

Every chunk is stored in a file (the files will be overwritten by other chunks when necessary). The number of these files are the number of blocks (the size of the cache in kilobytes), divided by 8. You can test the usage by running the du command on the cache directory. If you are not using 85 percent of the cache, increase the number of files. You do that by using the -files option to the dfsd command (started from the /etc/rc.dfs script).

7.5.3 Name Cache

When a DFS client accesses a file, it needs to ask an FLDB Server for the fileset ID of the file and for the physical location of that fileset. To decrease network traffic, this information is kept locally in a memory cache, called the *name cache*. The next time the client references the same file, no network traffic is necessary since the information is in the cache. This also reduces the load on the FLDB Server(s).

If a client accesses a large amount (>500) of files in a short period of time, the name cache should be increased to keep network traffic down and therefore increase performance. You can check the usage of the cache with the `dfsstat` utility (see 4.8, “DFSSTAT” on page 81). The hit rate should be over 80 percent. The size of the name cache can be increased with the `-namecache` parameter of `dfsd`. As an example, the line towards the end of /etc/rc.dfs could then look like:

```
...
daemonrunning $DCELOCAL/bin/dfsd -namecache 500 -stat 500
...
```

When you set the `-namecache` parameter, you should also set the `-stat` parameter to the same value as the `-namecache` parameter. The example above sets both values to 500.

7.5.4 Background Processes

If your DFS client uses DFS heavily and potentially accesses many files from many DFS servers, an increase in the number of main processes and token-handling processes may further improve its performance. An increased number of main processes (the default is two) improves efficiency by performing prefetching and background writing of saved data. The token-handling processes, of which two are running by default, handle the token revocation RPC requests from Fileset Servers. If the DFS client accesses many different DFS Fileset Servers, an increased number of token-handling processes can be beneficial.

The number of these processes can be changed with the `-mainprocs` and `-tokenprocs` flags of the `dfsd` process, started off from /etc/rc.dfs. The command line can be found at the end of /etc/rc.dfs, and it could look like:

```
...
daemonrunning $DCELOCAL/bin/dfsd -mainprocs 4 -tokenprocs 4
...
```

7.5.5 Server Preferences

Depending on your network, you may get better performance by setting preferences to the DFS Fileset and FLDB Servers. See sections 7.2.4, “DFS Fileset Servers” on page 145, and 7.2.5, “DFS FLDB Servers” on page 146 for details.

7.5.6 Miscellaneous Information

Some additional information may help to understand some aspects of the operation of a DFS client.

- The minimum unit of transfer across the network when doing a read is a chunk size (default 64 KB). When doing a write, the unit can be as small as a single memory page, which is 4 KB. Thus unnecessary traffic does not occur when writing small files.
- For security reasons, the default does not allow you to run programs with the setuid bit set from DFS, nor can device files be created in the DFS file space. This can be changed, per fileset, by the `cm setsetuid` and `cm setdevok` commands.
- The priority table for the data in the cache is updated once an hour. Data is never discarded simply because of age. This means that as long as there is enough space for the files in the cache, the files will stay there forever. They

also survive a reboot. Validation of the files occurs when an user accesses them.

- Although it is not normally necessary, it can be useful to flush data from the cache to force a read from the server (during testing for example). This can be done by either the `cm flush` (for a specific file or directory) command or by the `cm flushfileset` command (for the whole fileset holding the specified file or directory).

7.6 OS/2 DFS Client

The OS/2 DFS client has the same type of configuration parameters as the AIX DFS client. The difference in the OS/2 environment is the OS/2 support of DOS and Windows applications. The DOS and Windows applications do not use the DFS-allocated drive differently than a normal disk drive. Some DOS- or Windows-based applications may have problems with the size of the drive (2 GB DOS drive limit) or with the long filenames. File locking is supported by the DOS file-lock interrupt.

Although most of the configuration options available on AIX (as discussed in the previous sections) are available for the DFS client on OS/2 as well, the specific implementations differ. While all DFS startup parameters can be specified in either the /etc/rc.dfs or the /etc/dce/CacheInfo files on AIX, OS/2 does not have an editable startup file and has another file format for the CachInfo file (notice that the filename is *CachInfo* on OS/2, and *CacheInfo* on AIX). DCE and DFS client processes on OS/2 are started through the `dcestart.exe` command on OS/2, which, unfortunately, does not support additional command line options for the processes. Although not recommended, it is possible to start the DFS processes separately with the required command line options, which is a way to specify any of these options as required.

We discuss the OS/2 implementation in the following sections.

7.6.1 DFS Client Cache

The DFS client cache type (disk or memory), location on the disk, and its size, can be configured through the Graphical User Interface from the desktop (or by running the `dcecfgg` command). This changes the settings in the CachInfo file. Alternatively, the cache size can be altered dynamically with the `cm setcachesize` command, as in the following example:

```
cm setcachesize -size 9000
cm: New cache size set: 9000
```

Using this runtime feature, the effect on a cache size change can be tested dynamically without changing the startup values.

Be aware that a disk cache actually uses about 15 percent more disk space than the specified size of the cache.

7.6.2 Chunk Size

The chunk size on an OS/2 DFS client can be changed using the graphical DCE/DFS configuration tool, `dcecfgg`. Alternatively, the CachInfo file can be edited, or the `-chunksize` command line flag of the `dfsd` process can be used.

7.6.3 Name Cache

The size of the name cache, which stores information about file and fileset location, can be set with the `-namecachesize` command line flag of the `dfsd` command.

7.6.4 Background Processes

The number of main and token-handling processes can be set using the `-mainprocs` and `-tokenprocs` command line flags of the `dfsd` command.

7.6.5 Server Preferences

Preferences for Fileset and FLDB Servers can be set using the `cm setpreferences` command after the DFS client has been started. See sections 7.2.4, “DFS Fileset Servers” on page 145, and 7.2.5, “DFS FLDB Servers” on page 146 for details.

7.7 Spreading Replicas

For a general discussion about replicas see, 3.7, “Replication for Performance” on page 59, and 5.3, “Replication” on page 100.

If a server is highly utilized, you should analyze the system and look for any bottlenecks. Use the tools described in Chapter 4, “Tools and Methods for Performance Evaluation” on page 63. If you find that the DCE processes either are causing a bound condition, or they suffer from the high usage of the server, you can either upgrade the machine, move some load (not related to DCE, if any) to another machine, or create replicas of the DCE services.

Chapter 8. Application Development With Performance in Mind

Application development using DCE as the underlying middleware layer is bound to the requirements the application developer is solving. With the wealth of DCE APIs available, it is sometimes not easy to predict the performance of the code generated. Although an application may be running with expected response time on a fast network, the response times on slow links may be surprisingly high. This should be a major concern whenever testing is done. Some of the reasons for long response times are often some unexpected, unwanted or even uncontrolled additional traffic generated by the created RPC applications.

The following sections describe some useful techniques, tricks and tips that can be applied when designing and developing DCE applications.

8.1 Binding Considerations

Binding is the process by which an RPC client establishes a relationship with a server that supports an interface, object or some other resources in which the client is interested. The binding information comprises:

- A protocol sequence that identifies the RPC and the underlaying transport protocols
- An RPC protocol version identifier
- A transfer syntax identifier
- A server host network address
- An endpoint of a server instance on the host
- An object UUID that can be used for selection among more servers and/or manager routines
- An interface UUID that identifies the interface to which the called routine belongs
- An interface version number that defines compatibility between interface versions
- An operation number that identifies a specific operation within the interface

Figure 42 on page 156 depicts the terminology used for RPC bindings.

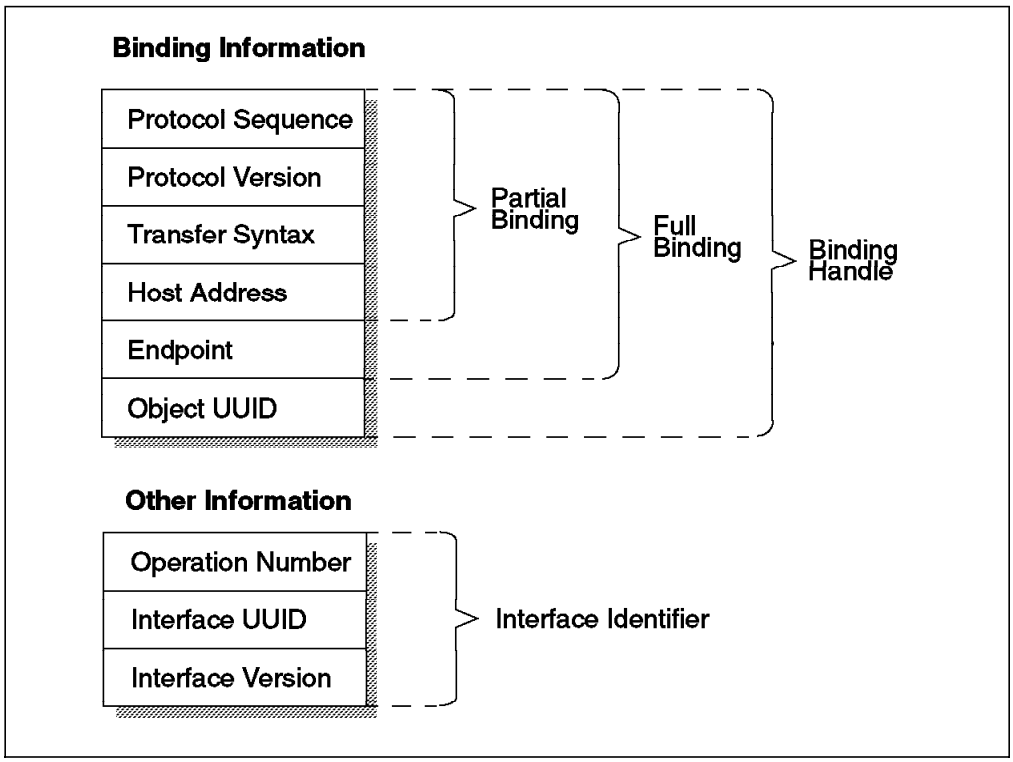


Figure 42. RPC Binding Information

The process of connecting RPC clients to RPC servers normally requires server names and endpoint mapping services. Only having a minimum of required information, an RPC client can then look up the required, additional information by using CDS services and endpoint mapper services. However, neither of these services are strictly required because well-known bindings and endpoints can be used in the form of so-called string bindings.

Using string bindings reduces some load on the network and on the DCE core servers, and it simplifies application logic, but it is generally not recommended to be used because it is a static binding and therefore against the fundamental concept of the DCE dynamic look-up services. CDS is a better place to keep the binding information because it dynamically adapts to a changing environment.

Only if your DCE application is performance critical and runs over slow network links, may you consider using explicit string binding. A typical situation where string binding may improve performance remarkably is when client applications only run for short periods of time, but are used very often. Examples of such applications are DCE-secured tools, like remote shells or other DCE-based management tools, that are frequently used like other operating system commands.

The normal operation of RPC servers and clients begin with the RPC server application exporting its binding information into the CDS. This binding information can then be retrieved by the RPC clients.

The RPC server uses the DCE RPC APIs to:

- Assign types to objects
- Register at least one interface
- Specify which protocol sequences to use

- Obtain a list of binding handles
- Register endpoints
- Export binding information to the CDS namespace
- Listen for RPC client calls
- Do cleanup before termination, including unregistering endpoints

An RPC client application needs to obtain binding information for the RPC server application it wants to communicate with. This is usually accomplished using CDS services, but any other method could be used instead. There is a wide range of different methods for getting binding handles: from static coding (hard coded binding handles, not using CDS at all) up to automatic methods provided by DCE RPC functionality. Variations in between partly involve CDS for constructing a complete binding handle.

If CDS was involved for constructing binding handles, it is recommended to keep (cache) the obtained binding handles in the application client and reuse them every time a connection is to be established with the RPC server, rather than to request them every time from a CDS Server. This reduces network load, network-imposed delays and CDS Server load.

Using profiles and groups within the CDS namespace structure can also have an impact on performance. Although the administration part will improve, the number of RPC client fetches may increase.

8.2 IDL Considerations

With the Interface Definition Language (IDL), the data types to be transferred is defined in an IDL file. Parameters are specified to define the data types, which will have different representations on different platforms. Examples are the string representation, which is EBCDIC on some platforms and ASCII on other platforms. The process of encoding data into the correct form is called marshalling. Marshalling and unmarshalling is done by the RPC stub code.

The time used for marshalling and unmarshalling data can be reduced by the following rules:

- Use basic data types or simple structures whenever appropriate.
- Avoid using pointers. If pointers must be used, try to use reference pointers instead of full pointers.
- Use variable-length-conformant arrays instead of fixed-size arrays.
- Avoid using complex structures as input or output parameters since the use of simple structures instead will result in faster marshalling.

The amount of data transferred must also be considered. Although it is good programming practice to hide data or make generic structures, the actual completeness of the data contents should be validated in the design and programming stages of a project. It is recommended to limit the amount of data that is being transferred to the necessary minimum.

8.3 Using Threads

The DCE threads package is based on the pthreads interface specified by POSIX in 1003.4a, draft version 4. Threads within a process share the resources of the starting process, including files, memory, and signal handlers. Each thread has its own thread identifier, scheduling policy, and priority.

Threads are created by the `pthread_create()` routine. There are thread attribute objects and synchronization objects available to manage and control threads. Each thread has its own stack and associated data structures that will increase the overall memory consumption of a multithreaded application.

Threads execute concurrently. Within a multithreaded application, there are multiple states of execution present at any time. Multithreaded applications offer the following advantages:

- Performance—Threads normally improve the performance of an application when designed and used properly. Multiple threads are useful in a multiprocessor system where threads can run concurrently on separate processors. Multiple threads can improve application performance on single-processor systems by scheduling threads while other threads are waiting on slow operations like input and output.
- Shared Resources—Using threads is a convenient way of structuring an application, while sharing the same address space, open files and other shared resources within the same process.

Some of the drawbacks of multithreaded applications are:

- Complexity—The level of expertise required for designing, coding, debugging, and maintaining multithreaded applications can be higher than for single ordinary applications.
- Reentrancy—The library or function routines have to be reentrant. Otherwise, a locking method must be implemented.
- Priority Inversion—Which is the case when a high-priority thread is blocked on (not granted access to) resources owned by a lower-priority thread.
- Race Conditions—Which is the case whenever shared variables unintentionally are modified by two or more threads at the same time.
- Deadlocks—Which happens when threads are blocked from execution. This happens for example when one thread owns a resource the other thread is requesting, but the first thread is waiting on a resource owned by the second thread.

As a good programming practice, frequent creation and destruction of threads, just for the sake of doing a certain job, should be avoided because it costs some unnecessary overhead. Instead, thread-synchronizing methods should be utilized together with long-living threads to keep them ready to process other requests. This assures optimal use of threads and provides for best performance.

8.4 The Right Protocol Sequence

The protocols of the TCP/IP socket interfaces are divided into the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Either protocol can be used by DCE. The characteristics of the protocols are described in the following sections.

8.4.1 TCP Transport Protocol

The Transmission Control Protocol (TCP) is a transport protocol that provides a reliable mechanism for delivering packets between hosts on an Internet network. TCP takes a stream of data and breaks it into datagrams. These datagrams are sent individually and reassembled at the receiving (destination) host. If any datagrams are lost or corrupted during transmission, TCP will detect this and request a retransmission of the corrupted or missing datagrams. The received data stream is a reliable copy of the transmitted data stream.

TCP is a connection-oriented protocol. It is used to communicate between pairs of applications. After a connection is established, it exists until it is closed.

Due to its connection and flow-control overhead, TCP tends to create slightly more network traffic over reliable networks.

8.4.2 UDP Transport Protocol

The User Datagram Protocol (UDP) is a transport layer datagram protocol that sends and receives whole packets across the network. UDP does not guarantee the delivery of data. UDP provides checksums for both packet header and data portions of a datagram. Applications that require reliable delivery of data should either use TCP or incorporate their own mechanisms for reliable data transfer when using UDP.

UDP is, by its nature, faster than TCP, but the cost of ensuring reliable data transfer and flow control has to be added to protocol drivers within the applications.

8.4.3 Comparison Between TCP and UDP

DCE RPC supports both TCP and UDP protocol sequences. In fact, most DCE core services make frequent use of both protocol sequences, the choice of which to use made by pseudo-random selection algorithms. For example, the `dce_login` command on AIX randomly uses TCP or UDP for its communication with the DCE Security Server.

It is not obvious which protocol sequence generally yields the better results in terms of performance because there are many factors to consider:

- The network may be forwarding packets very fast or slow, reversing packet orders, discarding or corrupting packets, or including bottle necks through slow links.
- The protocol handlers in the systems may be more or less efficient.
- In situations of overload, the behavior of network adapters, device drivers and protocol drivers may be difficult to predict.
- The amount of available buffer space for each protocol sequence in the network nodes is another determining factor for performance and behavior.

The following comparison test (Figure 43 on page 160) was carried out in the IBM lab to evaluate each protocol's characteristics. The test has been done between two OS/2 Warp machines (Warp Version 3, DCE for OS/2 Version 2.1, 90 MHz Pentium, 32 MB RAM, 256 KB L2 cache), using IBM Token-Ring 16/4 Adapters at 16 Mbps.

The tests scenario simply transmitted data blocks of different lengths using DCE RPC pipes either over UDP or TCP. No security was used.

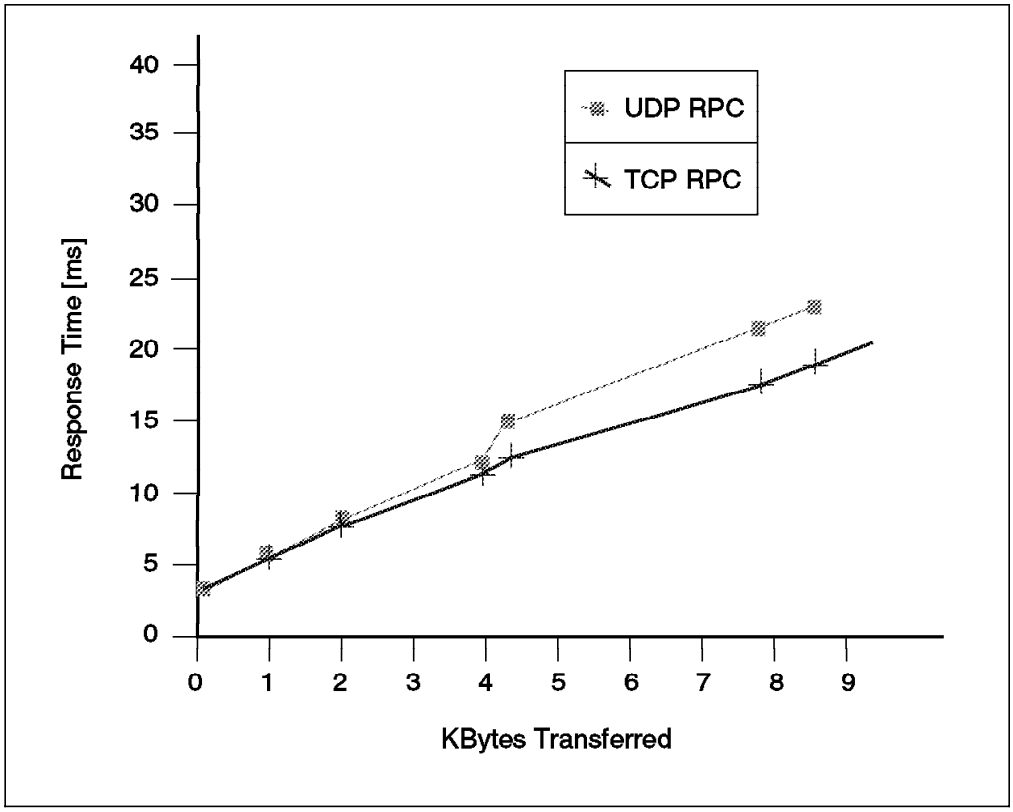


Figure 43. Relative Response Time, UDP Compared with TCP

Normally the response time increases linearly with the data length. Whenever the protocol has to break the data into more packets, there will be an increase in the response time due to the packet overhead. Comparing UDP with TCP, this overhead is more visible for the UDP protocol transferring packets over 4 KB.

The UDP compared with TCP is very alike for small data transfer sizes, but over 4 KB, the TCP protocol has a slightly faster transfer times. The difference remains about the same for larger data transfers (not shown in Figure 43).

Note

Figure 43 should not be interpreted as a guideline for application programmers stating things like, ″use TCP because it is faster than UDP″. The figure can be interpreted as TCP and UDP being about equal in base RPC performance. The tests have been done on a token-ring LAN, which ensures a very reliable and fast transport. The behavior may change if other, especially slower, networks are involved. As has been mentioned, TCP typically involves more packet transfers, which generally has a negative impact on slow lines. Finally, TCP often requires more resources in protocol handlers than UDP and therefore might introduce a higher load on central servers.

8.5 Use of Protection Levels

When a client establishes an authenticated RPC, it can specify the level of protection it requests to use for its communications with the RCP server. The protection level selected specifies how much of the client/server messages are encrypted. Generally speaking, more encryption will reduce overall performance due to more CPU and network load.

The selection of the protection level is entirely a responsibility of the RPC client as long as the RPC server supports the requested level. Authenticated RPC supports the following protection levels:

- `rcp_c_protect_level_default`—uses the default protection level for the specified authentication service.
- `rcp_c_protect_level_none`—establishes no protection level.
- `rpc_c_protect_level_connect`—performs protection only when the client establishes a relationship with the server. There is no encryption on data transferred.
- `rpc_c_protect_level_call`—performs protection only at the beginning of each RPC, when the RCP server receives the RPC request. This level attaches a verifier to each RPC client call and RPC server response. This protection level does not apply to connection-based protocols (TCP).
- `rpc_c_protect_level_pkt`—ensures that all data received is from the expected client. This level attaches a verifier to each message.
- `rpc_c_protect_level_pkt_integ`—ensures and verifies that none of the data transferred between the RPC client and RPC server has been modified. This level generates a cryptographic checksum of each message to verify that none of the data transferred between the RPC client and RPC server has been altered.
- `rpc_c_protect_level_cdmf_priv`—preforms protection as specified by all of the previous levels and also encrypts each Remote Procedure Call argument value. This level encrypts all user data in each call, using the Common Data Masking Facility (CDMF), which is a 40-bit implementation of the Data Encryption Standard (DES).
- `rpc_c_protect_level_pkt_privacy`—performs the same as `rpc_c_protect_level_cdmf_priv`, except using the 52 bit implementation of DES. DES is not generally available outside the USA and Canada.

The performance impact using the different levels of protection is divided into the time to transfer the data and the performance cost on the CPUs performing the encryption and decryption.

The response time impact using different levels of protection have been measured in the same OS/2 Warp environment as described in the last section (8.4.3, “Comparison Between TCP and UDP” on page 159). The next graph in Figure 44 shows the response times when using different levels of protection in the RPCs.

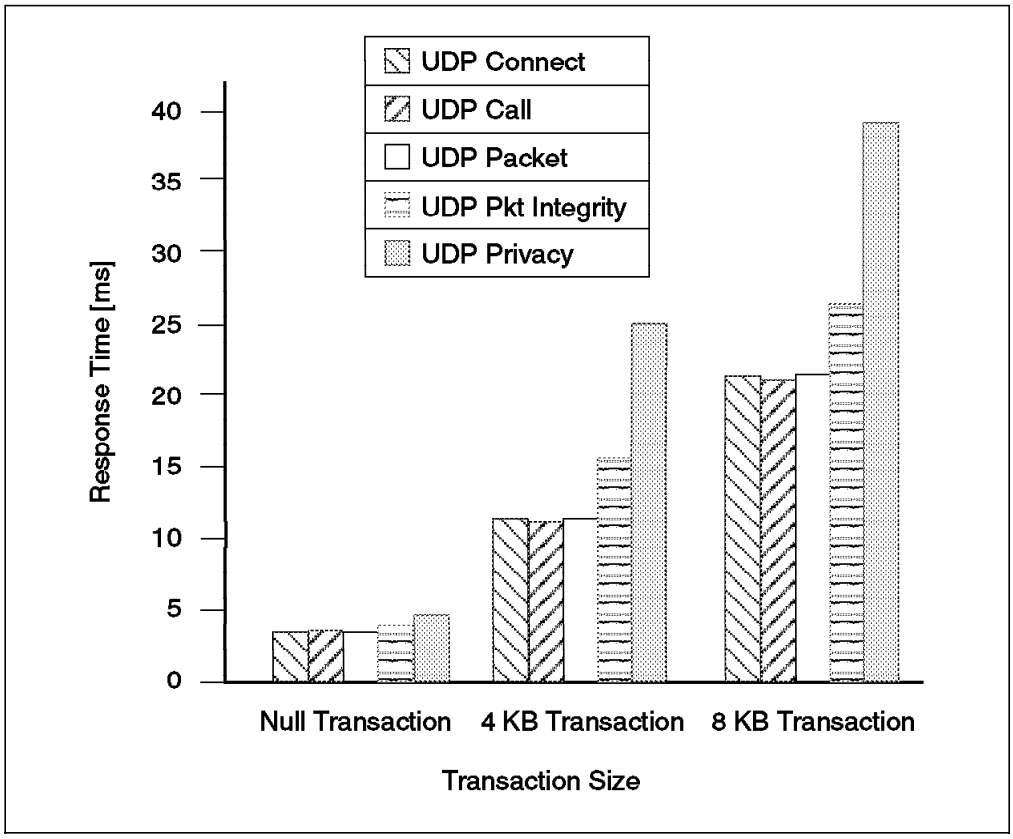


Figure 44. Response Time with Different Levels of Security

The protection levels of none, connect and packet have very similar response times. Using packet integrity increases response time remarkably, and using packet privacy increases it even more significantly. The tests were carried out with the aim of comparing the different security levels. The absolute response times shown are of less importance and may vary depending on many factors.

Similar tests on AIX Version 4.1 and on DCE for AIX Version 2.1 have shown comparable figures. The additional overhead imposed by higher security levels appears to be more noticeable on AIX. The packet privacy level of security results in response times of approximately 50 - 300 percent as compared to the other levels, depending on data transfer size.

Although the test results shown used UDP only, additional testing with TCP as the protocol sequence have revealed no relevant differences.

8.6 Pipes and Arrays in RPC

Pipes are a facility to efficiently transfer large quantities of data by overlapping the transfer and processing of data. Input data is transferred in chunks to the RPC partner. A pipe is declared in the type definition of the IDL file, and the data type is used as parameters in the operations of the interface. Pipes are especially suited for large data transfers.

Arrays, on the other hand, are either fixed-size or of a conformant/varying type, where the size of the array is determined at run time. All of the data in the array is transferred in a single call.

Actual performance lab testing has been done on variable-length arrays compared to DCE pipes. The throughput rates in bytes per second from the testing are as indicated in the next graph. The test environment, using OS/2 Warp machines, was the same one described in 8.4.3, “Comparison Between TCP and UDP” on page 159.

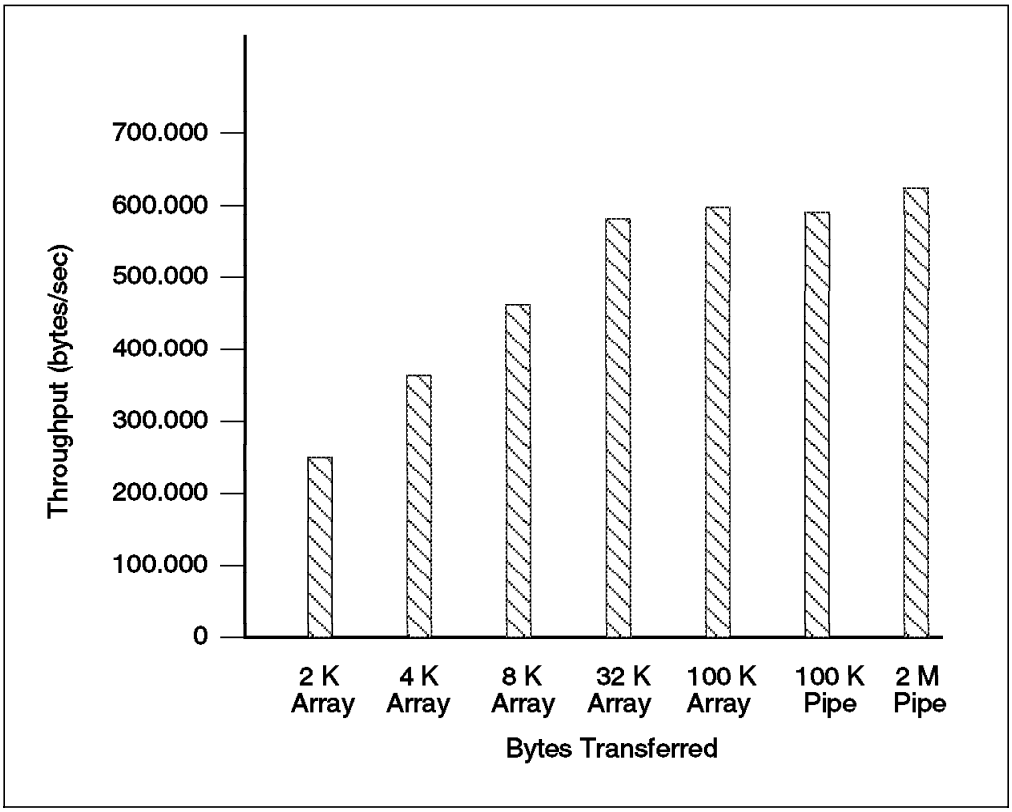


Figure 45. Transfer Rate, Pipes and Arrays

The transfer rate reaches its maximum at the 100 KB data transfer size. This peek is retained for larger data transfers. The 100 KB variable length array case performs slightly better than the 100 KB pipe case.

Pipes have a performance advantage for cases where data does not have to be immediately available. The data can be sent to the receiver via a callback mechanism when it becomes available at the sending side. Using the array method, the application has to wait until data is available. This asynchronous behavior is the primary advantage of using pipes.

8.7 Using Nested RPCs for Server Load Distribution

The server section of a remote procedure can call other remote procedures. This way, the call to the second remote procedure is nested within the first call. Extending the concept of nested RPCs, it is possible to distribute the application work load among multiple DCE servers.

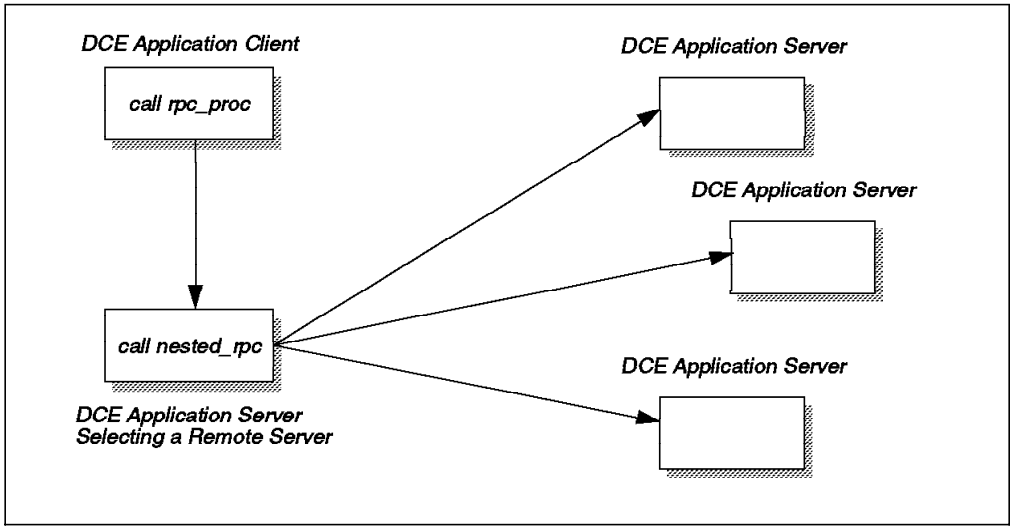


Figure 46. Nested Remote Procedure Call Example

With a proper selection procedure and if the application design permits, the work load can be distributed onto multiple servers without the DCE client application to know this.

8.8 Applications Using DFS

While a DFS client is active, it keeps a local cache of files that has recently been referenced or updated. DFS clients and DFS Servers keep track of the status of all cached files by means of a token-passing mechanism. The DFS clients hold tokens that they have cached for read or update purposes.

Because the DFS client performs caching of files, read and write operations into the same location within a file is very efficient. Reading large files (larger than local DFS cache) randomly will in some cases cause some delays.

8.8.1 General File Input/Output

The Distributed File System (DFS) is designed as a high-performance, networked file system, separating it from others through its special emphasis on scalability, security and performance.

DFS is designed to deal with all kinds of file access, with special focus and optimization on those types of file access that are statistically the most common, like:

- Sequential file access
- Relatively short file life time
- Short to medium-size files
- The majority of accesses are whole-file transfers

Bearing this in mind in its implementation, an application can expect excellent performance from DFS files being accessed. As with any other shared file system, random access of small blocks within large files and frequent change in read and write operations may decrease performance, as compared to the local file system. These operations need to access the File Server each time and cannot be improved by any caching algorithm. Application developers should be aware of this. If applications require this type of access, like database systems, for example, special DFS tuning may be adequate. See Chapter 3, “The Distributed File System” on page 47, or the DFS sections in the other chapters for related information.

8.8.2 File Locking

The DFS clients on AIX and OS/2 Warp support file locking. When a file range lock request is performed, the local DFS client forwards the request to DFS Servers in order to get a lock token. When the file range lock request is granted, the local DFS client manages related lock requests and enforces other local read and write access restrictions based on the restrictions the granted file lock range implies.

File range locks are enforced locally on the DFS client. Other DFS clients using the same shared file must perform file range lock request to ensure the consistency of the shared file, because read and write restrictions are not enforced.

Similarly, DOS and Windows applications running on OS/2 Warp using file lock DOS interrupt are supported.

Using the file range lock interface, the application can control the access to critical sections of the file. Because file range lock is granted by DFS Servers, some additional traffic and delay can occur when using one or more shared files.

Since DFS is a distributed, networked file system, most file operations cause network and server load. Through caching, simple read and write operations do not necessarily cause immediate involvement of remote services.

Locking, as well as file open, close, and so on, usually require the Fileset Server to be involved because it eventually needs to ensure this operations do not interfere with other client’s operations on the same file(s). If performance is a concern, application programmers should use those operations only when necessary since they have an impact on more than just the local file system. For example, files may be kept open for reading all the time when an application frequently reads data, rather than using many open-read-close sequences. This may have, of course, an influence on other applications if they request to access the same file at the same time.

8.8.3 Abnormal Termination

Because DFS clients cache file data and hold DFS tokens, delays or time-outs may occur when DFS clients are not able to return modified file data or DFS tokens. The following problems, which are not application dependent, are possible:

- Other DFS clients may experience delays obtaining access to files because a DFS client did not return file data or DFS tokens. The request cannot be granted until the DFS Server either retrieves the missing file data or token or times out and discards the tokens held by the DFS client in trouble.

- Data may be lost because data was not stored back on DFS Servers.

8.8.4 OS/2 File and Path Case Sensitivity

When the OS/2 DFS client is performing file operations, there can be problems in obtaining the correct file. Directories can contain entries that have different case versions. This causes some problems because OS/2 will evaluate file names without the respect of case.

OS/2 will generally preserve the case correctness, but because OS/2 can have integrated DOS and Windows compatibility sessions, the case correctness is not guaranteed.

Applications need to be adjusted to the case sensitivity.

8.9 AIX Application Instrumentation

Using the instrumentation capabilities, it is possible to obtain statistics such as interface calls, marshalling and unmarshalling times. To enable the instrumentation of DCE RPC code, special parameter settings must be used while performing the IDL compilation together with at set of compile and link options.

Application instrumentation is explained in more details in 4.9, “Application Instrumentation” on page 86.

Part 2. Problem Determination

Chapter 9. Problem Prevention

Problem determination and resolution are essential tasks when running mission-critical applications, since problems will occur sporadically. But before a problem occurs, there must have been a certain condition that made a single component, or the whole system, fail. In rare circumstances, such situations cannot be predicted, nor prevented, as happens with hardware failures or software bugs. In many cases, however, problems can be prevented. This chapter explains how many problems in a DCE/DSS environment can be avoided, where indications of potential problems can be found, and what system administrators should concentrate on when monitoring running systems.

9.1 Housekeeping for DCE and DFS

Housekeeping is the task of performing preventive actions that lower the chance and the risk of component failures. Housekeeping is most important on servers, especially DCE/DFS servers and application servers, since they represent a central resource and a failure may affect a number of users. Housekeeping on client systems is nevertheless important even though fewer users may suffer from a failure.

The following sections list the most important tasks for system administrators, including system resources and parameters that should be monitored regularly as a minimum for ensuring reliable system operation.

9.1.1 System Documentation

Having a detailed and up-to-date set of documentation describing the status of the system(s) is the foundation for a reliable operation.

There are several ways to document the history and the current configuration and status of a system. The fact that it is done at all is more important than which tool or text editor is used for it. The system documentation should include:

- Conventions, such as naming conventions, that had been applied when the systems were installed and configured and those to be applied for new additions and changes.
- A brief description of the network layout (subnetting) connecting important servers and replicas.
- The hardware configuration of the systems. For DCE, network interfaces and their addresses are helpful in many situations.
- A list of the software components installed on the systems, their purpose (if not obvious), and how and when they are being used.
- A history of changes to the hardware and software configuration.
- A description of important procedures, namely emergency procedures.
- Any non-standard setup, such as a special disk partitioning.
- A list and description of any special tools (for example shell scripts) developed for and running in this environment.

More than that, the system documentation should, over time, also comprise a collection of performance data so that proper system capacity planning can take place. This ability to plan ahead is only one benefit derived from a well-documented system. Whenever new support staff is involved, no matter whether as trainees or as emergency supporters, they will need to be able to look up the status and configuration of the system(s) in a fast and reliable manner.

9.1.2 Filesystems Used by DCE

Running out of free space in a fixed-sized filesystem, such as that provided by AIX, is one of the most common problems on server and client machines. DCE core servers are especially vulnerable because they store vital data in the /var/dce file tree, which is by default in the /var filesystem. The same filesystem is usually also used as a temporary storage area by many other applications and tools. Mail and print spooling, for example, which can be issued by any user on a system, can fill up the /var filesystem easily. On AIX, the /var/dce file tree is referenced by DCE indirectly through links from the /opt/dcelocal/var directory. Table 21 lists the most commonly used directories and their links on AIX and OS/2, where x: in the last column represents the drive on OS/2 as it was selected during product installation.

Table 21. Directories and Links Used by DCE for Working Data

DCE Logical Directory	Physical Location on AIX	Drive and Directory on OS/2
/opt	/opt	x:\opt
/opt/dcelocal	/usr/lpp/dce	x:\opt\dcelocal
/opt/dcelocal/var	/var/dce	x:\opt\dcelocal\var

As mentioned at several places in this book, separate filesystems should be used on AIX and mounted over the respective directories underneath /var in order to decouple DCE from other services, thus minimizing the risk of an out-of-space situation. Section 6.3, “AIX DCE Servers” on page 131, gives you some hints for creating these filesystems.

No matter whether separate filesystems are being used for DCE or not, it is very important to continuously monitor the free space available for DCE in the filesystem(s) it uses to store its data, since an out-of-space situation is a severe error from which DCE cannot recover without intervention. Table 22 on page 171 lists the relevant directories that DCE and DFS use to store vital runtime information. The table lists filenames for AIX, where [opt] stands for /opt/dcelocal/var. For OS/2, the slashes (″/″) in the filenames need to be replaced by backslashes (″″).

Table 22. Important Directories Used by DCE and DFS

Directory	Used by, for	Free Space Requirement
[opt]/security/rgy_data	Security Server for registry database	At least the size of registry database
[opt]/directory/cds (OS/2: [opt]\dir\cds)	CDS Server for CDS database	At least the size of the CDS database
[opt]/dfs	DFS Servers	Admin lists, FLDB database
[opt]/adm/dfs/cache (or as spcified in the CacheInfo file)	DFS client cache	15 percent of disk cache size
[opt]/security/creds	Security client for credential cache files	Enough to hold the credential files (see text)

The right-most column in the table lists the minimum free space requirement for the filesystems in which these directories are located. Because both the Security and the CDS Servers temporarily store two copies of their databases during checkpointing, at least the size of their databases must be available as free disk space in those filesystems. A good estimate of the actual sizes of these databases is the total of the sizes of all files in these directories. The credential files usually do not require much space, unless for testing or during periods of frequent logins to DCE. The credential files for expired credentials should be removed periodically with the `rmxcred` command (see also 6.2, "AIX DCE Clients" on page 130). On OS/2, the credential files are removed automatically after every system restart, but you can or should run the `rmxcred` command as well on machines that are not restarted often. On AIX, expired credential files are only removed automatically when a system is rebooted and DCE client processes are started from within /etc/inittab.

Besides the most obvious directories listed in Table 22, DCE uses other temporary space beneath /opt/dcelocal/var (linked to /var/dce on AIX). It is therefore important that there is always some free space available in this directory tree.

Monitoring filesystem utilization is one of the most important aspects of systems management. Critical filesystems, such as /var and /tmp, should be checked on critical server systems at least once every ten minutes.

9.1.3 Data Backup

DCE servers provide online database functionality. Backing up online databases requires some special provisions because:

- The database contents may change during the process of backup, which is basically a file copy operation. If a file is changed while it is being copied, or if some files are altered after others have already been copied, the result could be useless because of inconsistency.
- The state of the database may be unknown at the time the backup (copy) is done. Since DCE services work with in-memory databases, the files on disk representing these databases are not likely to be current.

The basic rationale about a backup is to have at least a second and valid copy of the data. Most company policies require such backups to reside on certain

media, such as tapes. Replication, as an alternative, also provides copies of original data, but on another system. If replicated systems are physically apart from each other (to be independent in case of environmental damage, such as a local fire), they represent the best online backup method for DCE services, provided all information is properly replicated.

Replication, however, does not address the following two issues:

- If a database becomes corrupted for any reason, the replica databases are likely to be corrupted as well.
- Replication represents only a single, current state of a database. There is no history of backups available. For these (and yet other) reasons, it is still advisable to regularly backup DCE database data to tape or to any other adequate media or location.

Note on Security

Any media that stores DCE database data can be misused to gather confidential information about the cell, users and passwords. Therefore, backup containers must be locked up and be treated with the same security precautions as the systems themselves.

Each DCE service needs some special considerations for data backup. It is generally not sufficient to use standard system backup utilities, such as `mksysb` or `tar` on AIX, for data backup. The online databases need to be dumped to disk in a controlled manner before a backup can take place. The DCE `dcecp cell backup` command (not available on OS/2 Warp) can be used to back up the Security and CDS databases. DFS also provides specific tools for data backup from the DFS file space because standard backup tools cannot preserve the ACLs within DFS. The IBM ADSTAR Storage Management (ADSM) product also supports data backup from DFS with ACL preservation.

The *DCE Administration Guides* for AIX and OS/2 Warp and the *DFS Administration Guide and Reference for AIX* contain detailed information about precautions and procedures concerning backing up the various DCE services databases.

9.1.4 Hardware

Depending on the requirements for availability, the hardware being used for servers (and important clients) may need some special attention as well. This not only refers to the type and model of the machine but also to the supporting environment, like power-source and network connections. In fact, faulty network connections cause more troubles than generally assumed, especially when they only fail to work in certain circumstances.

Disk subsystems may also require some special attention. Since most DCE services do not require high-speed disk I/O, disk striping (RAID) with a high level of availability is an option to consider for DCE servers.

Due to the different internal construction, choosing industry strength server models, rather than, for example, desktop models of machines, may further increase the availability. Industrial-strength servers usually have larger dimensions, allowing better cooling of the components, having larger power supplies, and providing better upgradability than small machines.

9.1.5 Systems Management Infrastructure

Overall systems management needs to take at least the following into account:

- System monitoring
  - CPU utilization
  - Memory utilization
  - Network interface utilization
  - Filesystem utilization with threshold actions
  - Check for running processes
  - Error log analysis
  - Report generation for trend analysis
- Release planning
  - Hardware upgrades
  - Software upgrades or migrations
- Basic systems administration (operating system)
  - User/group management
  - Administrator accounts
  - System time and date (if not using DTS)
- Systems documentation
  - System status (configuration)
  - Procedures

There are products available, such as the *Tivoli TME 10* systems management product series, which can do an excellent job in monitoring systems.

9.2 DCE/DFS Process Checklist

One of the first actions to take when suspecting a problem might be to check if all necessary processes are running on the systems. This section lists the DCE processes that are required on various types of participating server and client machines in a cell.

Use the ps command on AIX, or the pstat command on OS/2, to find out if the required processes run on a particular machine. The matrix in Table 23 on page 174 gives you a list of processes and associates them with the different machine roles.

Table 23. Processes on Various DCE Machine Roles

Remarks / Process	dced	cdsadv	cdsclerk 1)	dtsd -s	dtsd -c 2)	secd	cdsd	bosserver	flserver	ftserver	fxd	dfsbind 3)	dfsd	upserver	upclient 4)
Security Client	X														
CDS Client	x	X	X												
OS/2 Slim Client			X												
DTS Client	x	x	x		X										
Security Server	x	x	x			X									
CDS Server	x	x	x				X								
DTS Server	x	x	x	X											
DFS SCM	x	x	x					X						X	
DFS FLDB Server	x	x	x					X	X					X	(x)
DFS Fileset Server	x	x	x					X		X	X				(x)
DFS Client	x	x	x									X	X		

Legend and remarks:

X Primary process(es) for this machine role.
x Prerequisite process for this machine role, but exceptions may exist.
(x) May be running in certain environments, but not required.
1) A `cdsclerk` process may not be running all the time. Zero, one or more `cdsclerk` processes may be running at a given time. On OS/2 slim clients, only one instance of `cdsclerk` is running.
2) The DTS client process (`dtsd -c`) may be running on any machine other than a DTS server, if configured accordingly.
3) On OS/2, `dfsbind` is a DLL, rather than a separate process.
4) An upclient runs only if configured to receive updates.

The matrix in Table 23 does not include the different machine roles and processes for DFS backup, since these processes depend on the state of the backup process.

A systems management tool should supervise these processes and alert an administrator if any is missing.

9.3 Log Files

DCE maintains various log files that can be used for problem determination as well as for problem prevention. Not every event that is being logged necessarily leads to a problem, but it might be an indication of a potential problem situation. It is therefore good system administration practice to regularly inspect these files, at least on important server machines.

The DCE messaging and logging facility is customizable, allowing for individual adaption according to specific situation requirements. Appendix A, “Using the DCE Debug and Messaging Facilities” on page 225, explains this in more details.

The following two sections list the default log files that can be found (without customization) on AIX and OS/2. We will concentrate on the files and their

locations, without going into details about their contents. Many logged messages are easy to understand, while others, unfortunately, do not explain a certain situation clearly enough to understand what happened. Additional investigations (or assumptions) may then be necessary. In any case, the log entries are important to report when further support is requested by software support specialists.

9.3.1 DCE Log Files on AIX

Besides the operating system logs, such as the error log of AIX, DCE maintains a series of log files located in /opt/dcelocal/var/svc (linked to /var/dce/svc). They are:

- /opt/dcelocal/var/svc/warning.log

  This log contains information about warning conditions found in the DCE operation. They usually do not cause any DCE component to terminate prematurely. Example:

```
1996-12-07-23:12:05.468-06:00I----- dced WARNING dhd general main.c 1007 0x2003054c Caught
signal 1. Exiting.
1996-12-07-23:21:58.035-06:00I----- dtsd WARNING dts config logevent_v_ultrix.c300
0x200275f4 Too few servers (2), need 3 servers
1996-12-10-17:25:23.690-06:00I----- dtsd WARNING dts configlogevent_v_ultrix.c300
0x200275f4 Too few servers (2), need 3 servers
1996-12-11-15:02:14.965-06:00I----- dtsd WARNING dts config logevent_v_ultrix.c300
0x200275f4 Too few servers (2), need 3 servers
```

- /opt/dcelocal/var/svc/error.log

  Errors encountered in the DCE operation are registered here, as in the following example lines:

```
1996-12-04-15:53:44.368-06:00I---- cdsd(13154) ERROR cds server db_unix.c 551 0x2002a428
Unable to open file /opt/dcelocal/var/directory/cds/itsc.austin.ibm.com#ev2_ch.version:
status= 2.
(get_dirs):  Error enumerating directories under /.:
(get_objs):  Error enumerating directories under /.:/hosts/ev2
(get_dirs):  Error enumerating directories under /ev2
(get_dirs):  Error enumerating directories under /ev2
(get_dirs):  Error enumerating directories under /help
(get_dirs):  Error enumerating directories under
(get_dirs):  Error enumerating directories under /./users
```

- /opt/dcelocal/var/svc/fatal.log

  The fatal conditions found during the operation of the various DCE processes are logged here. Example:

```
1996-12-04-15:53:44.572-06:00I----- cdsd(13154) FATAL cds server dns_service_ncl.c 576
0x2002a428 Error trying to load a clearinghouse from disk into memory: status = 282108932.
1996-12-04-16:48:22.999-06:00I----- cdsclerk(22476) FATAL cds general clerk_listener.c
623 0x20018950 Routine pthread_mutex_lock failed : status= -1. syntax error
```

- /opt/dcelocal/var/svc/routing

  This is called the DCE serviceability routing file. It is consulted by the DCE services by default, unless otherwise instructed by means of command-line options, or if no environment variable (SVC_COMP_DBG) was set. It contains the instructions for DCE services on what and where error messages are to be logged. See Appendix A, “Using the DCE Debug and Messaging Facilities” on page 225, for more details.

- Specific process log files

  Although /opt/dcelocal/var/svc is the general directory for log files starting with OSF DCE Version 1.1 (AIX and OS/2 DCE V2.1), most processes still support their own log files. They are:

– /opt/dcelocal/var/dced/dced.log

  This file is maintained by the dced process, and it holds data related the dced operation. In the event of any error of the dced it will record the error msg and other related information regarding the operation of the process.

– /opt/dcelocal/var/security/adm/secd/secd.log

  This is the log file of the Security Server daemon secd.

– /opt/dcelocal/var/directory/cds/adm/cdsd/cdsd.log

  The CDS Server daemon cdsd maintains this file at its log file.

– /opt/dcelocal/var/adm/time/dtsd.log

  The log file for DTS. DTS servers, for example, report in this file when they cannot find enough other DTS servers to synchronize with.

– /opt/dcelocal/var/adm/directory/cds/cdsadv/cdsadv.log

  This log file is used by the CDS advertiser (cdsadv) to log conditions specific to its operation.

9.3.2 DCE Log Files on OS/2

OS/2 and LAN/Warp Server provide the syslog utility and the FFST/2 (First Failure Support Technology) facility that show, in a formatted window, dumps, system log, messages, console log, and more. These utilities allow you to look at and analyze the various operating system logs.

DCE on OS/2 Warp maintains the same log files as DCE on AIX, except the process specific files, which cannot be found on OS/2. The log files on OS/2 are located in the following directories:

- x:optdcelocalvarsvcerror.log

  This is the log file for DCE non-fatal error messages that occurred.

- x:optdcelocalvarsvcfatal.log

  Contains DCE fatal error messages.

- x:optdcelocalvarsvcwarning.log

  The log of any warning error messages. They can indicate a situation that may lead to a problem.

An additional file in the same directory as the log files, x:optdcelocalvarsvcrouting contains the configuration for the logging facility in DCE (see also Appendix A, “Using the DCE Debug and Messaging Facilities” on page 225).

Some error conditions are also logged in the x:ibmlanlogsNET.ERR log. This log file must be viewed with the net error command.

Beside the DCE specific log files, some others are to mention:

- x:optdcelocaletccfgdce.log

  The DCE configuration log.

- x:dssinstlogsdssinstMKRSP.LOG

  Contains a progress log, including errors, about the DSS installation process.

- x:os2installdsscfg.log

  The DSS configuration log.

Chapter 10. Problem Determination

Problems occur sporadically and often unpredictably, and we use automated procedures and human interventions to determine and eliminate problems within hardware, operating systems, middleware, and applications.

We also know that the use of distributed computing, with its growing complexity of distributed systems, has introduced new levels of problems. Also many of these problems have a more severe impact on the business than they had before, when computers and applications were not tightly coupled together.

That is why a correct way to approach a problem is so important. In the next sections we describe the common flow of a problem-determination process after briefly describing the process of troubleshooting a problem.

10.1 The Troubleshooting Process

The process of finding the cause of a problem sometimes looks difficult. But with a little bit of order in your process, it could be easy. First of all, you have to set the areas where you think the problem is. Then, after you start to search for the cause, you must first understand very well the nature and the type of the problem.

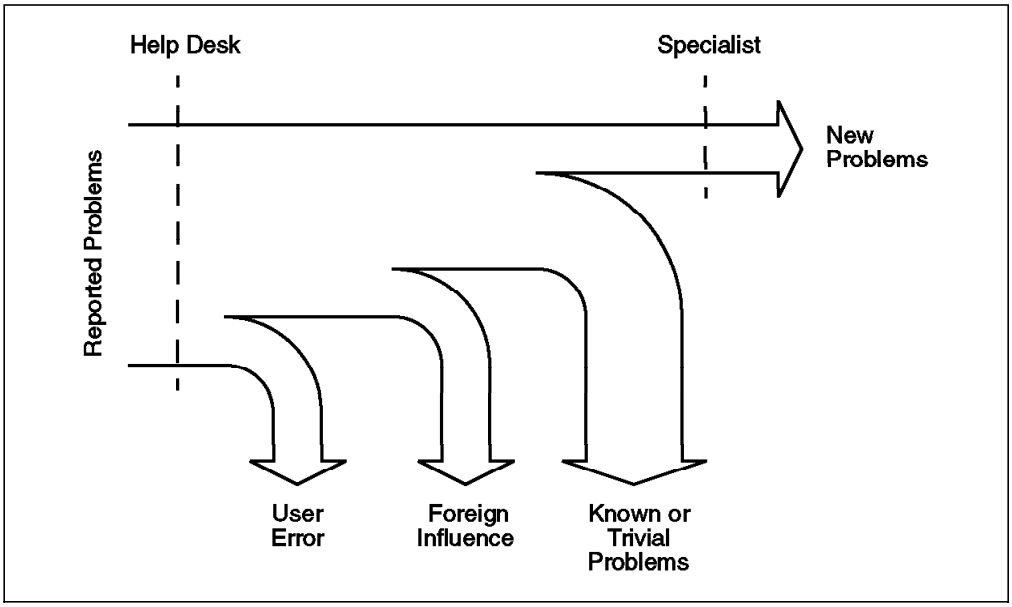


Figure 47. Problem Variety

Figure 47 depicts that only a few problems reported by users end up on the specialist's desk. A help desk might be able to filter out all user errors, foreign influences, such as temporary network outages, and resolve trivial problems, like expired passwords or out-of-space situations, before a problem is handed over to a specialist. If, however, the helpdesk is not able to determine the class of problem, a specialist might get involved in an early state and talk to the user in order to get the necessary information.

The entire process of problem pursuit is the job of a detective. You must not disregard any clue that could lead to the source of the problem. But this can

only be achieved as long as you are completely sure of what the problem really is. This is a mayor first step in the troubleshooting process because we often only see the symptoms of a problem, not the real problem itself. As soon as you know what the real problem is, you can then start to figure out why, how, or/and when the problem appears. In many cases, this second step is very easy.

To initiate your hunt for the problem, you must know first where the problem is detected and what its symptoms are. From a troubleshooter's point of view, there are two main areas where a problem can appear:

Problem in a well-known area — When a problem occurs in a well-known area, your experience and common sense will guide you through the problem-determination process.

Problems in unfamiliar areas — When a problem seems to be in an area where your knowledge is not deep enough, or you have no idea what is going on, then you have to use some techniques to approach the problem.

Once you have detected the area of the problem, then you have to start to find the possible causes of the problem. First, you have to determine who or what is causing the problem. In this process you can easily fall into the most common trap of troubleshooting, falling into a loop. If this happens, it often consumes most of the time of the problem-determination process. You must mark some points of reference in your pad of problem analysis. These points will help you to avoid falling into a loop. You must get out of the loop as soon as you feel that you already performed this or that test without any results, or with the same results.

Another common error is to maneuver oneself into a dead end. A dead end is the moment when you feel that the test or analysis guides you nowhere or to an area that is not involved in the process. This is worst when you fall in a loop of dead ends. Try to avoid this type of condition by recapping all the steps that you are taking. There is always another path to go in the process of a problem determination. The key is to never give up. Do not throw away any possibility. Quite often, the most trivial references lead you to the identification of a problem.

One of the most useful ways to approach a problem situation is to make a list of symptoms and reactions that the system shows to be considered as a problem. With a list of possible causes, you can see how much progress you made. You always have to have the three most important questions in mind: *how*, *when*, and *where* a problem occurs. Prepare a list of initial answers to these questions, and they will lead you to the next set of questions, which will arise from the original ones.

When the initial list is completely checked, make another one with the questions that come from the first stage of your problem determination. When you feel that you have already done this or you lose the track of the problem, stop and recap. Look where you are in the problem-determination path, and try to figure out if you are heading in the correct direction or if you are looping between the causes.

After covering the main aspects of the process of problem determination, we will give you some guidelines to help in the process of problem determination. First, consider the way the problem comes to you.

10.1.1 The Way a Problem Comes to You

An experienced troubleshooter knows that there is a certain bandwidth of problems that cannot easily be categorized into a severity level. If a user, for example, reports inability to login to DCE, it can be as simple as using the wrong password, or as serious as an inoperative DCE cell.

The first step in a troubleshooting process is to carefully listen to the people who report a problem and to associate the right level of confidence in that report. As an example, a new user, not knowing the right terminology to describe a problem, might tell you over the telephone that he/she cannot get into the system. While the actual problem might be a broken keyboard or display, you might be thinking that the Security Server went down.

First, try to understand exactly what is going on. Try to capture the most information that you can. Any information, as trivial as it could look, could give you the hint for the determination of the source of the problem.

The way a problem is reported to you should trigger the next actions. Use your common sense to figure out if the problem is reported correctly or if it is misleading. Users sometimes have a tendency to pre-analyze a problem and then report their findings, rather than the symptoms. This can be helpful indeed if the user is knowledgeable enough, but it can be very misleading otherwise. It is very important for you as a troubleshooter to determine the real symptoms of a problem reported to you, rather than to rely on facts that might be wrong and do not describe the real situation. This is not only true when novice users report problems but also when an experienced system administrator might overlook an important fact. You should therefore always build up your own mind of the problem by verifying those key issues that you base your further decisions on.

If you are looking at a problem that is reported by an expert user that you can trust, you can try to understand what is causing the problem and use the expertise of the user to help you detect the cause. Look for the step in the process in where the problem appears. Use the knowledge of the user in the system or the application to isolate the area where the problem could be located.

In the case of a new, unexperienced user reporting a problem of the system or the application, you should first try to understand if it is really a problem, a confusion, a misunderstanding of the system or application, or a genuine problem. In this case you have to use your experience in trying to obtain the most information from the user. Never undervalue the information that any user could give you, but rely on symptoms, not on assumptions and findings.

Before you carry on with the pursuit of the problem that has just been reported to you, you should go through some important questions to ask either yourself (as a checklist) or the other people involved.

10.1.2 Important Questions to Ask

There are some important questions that a troubleshooter should answer before going into details:

- Is it really a problem or can it be a confusion?
- Did it ever run correctly before?
- When did the problem appear the first time?

- How did the problem become visible?
- Is more than one user affected?
- Can the problem be re-created?
- What has been changed recently?
- Is the foundation (OS, TCP/IP, Network) healthy?

These are the most important questions that you must have on your check list when you approach a problem. The first steps in the problem determination process are the basic ones that will let you know the exact environment and all the participants in the problem.

You should check if the problem affects only one user or whether more users are involved. Look for other users' input. If multiple users seem to be affected, do they all report the same observations? You need the most information that you can obtain; so use other users as sources of complementary information.

After you check the initial condition of the problem, you should review the technical environment of the problem. Even if you think that you know the environment in which the user is having a problem, re-check your knowledge of it. Maybe something changed in the meantime that you are not aware of. Sometimes the changes in the hardware or software of the systems affect the behavior of the system in a way that is not obvious or predictable when doing the change.

After you made yourself familiar with the environment, you should check as a next step that the foundations of the systems are in a good shape. Check the communications, network, the operating systems, servers, everything that could be affecting the problem in one or another way. Look for any hardware error that could be involved in the situation or that can be causing the error.

Avoid assumptions, unless you can be sure they are correct. It is a common error while hunting for a problem to assume that, for example, the hardware or the communication subsystem works fine. Unless you have a certain confidence about the possible source of a problem, you should assume any component to be a potential source.

Check if the problem has ever occurred before or if it is a new one. Also, find out if this function ever worked before or if it is a newly introduced functionality that causes the problem. If it worked before, look for any changes that must have been applied to the system recently or what system parameters might have changed since.

10.1.3 Isolating the Source of the Problem

You must be sure that you are on the right track to the cause of the problem. If in doubt, this can be verified by trying to re-create the problem, if that is possible. If not, for example due the impact that the recreation may cause to the system or the complexity of doing so, you must be sure that the suspected cause of the problem can be isolated or at least tested in such way that you can confirm that you are facing the real problem.

When you face the situation where you are not able to probe whatever your discoveries are the real cause of your problem, you should develop a method for cross-checking the condition that is believed to cause the problem. You could

do this by using other functions or processes that make use of the module, process, or part of system that you think is failing.

You have to investigate—if you do not already know—whether the failing component maintains a log file. If this is the case, you will certainly check it in order to find any reported errors in it. The log files maintained by the components often show the regular activity, as well as errors that occurred.

Also you have to check for any other automatic or manual process that could be involved and affecting your problem. Check to see if the application or task that is in trouble depends on other application data, termination, or any other type of dependency that could exist and could be causing the problem.

If you succeed in this phase, you can be sure that you will find the source of the problem, and correct it. If not, you have to reconsider the approach, do it again, and try to find either the point in where you got off the path or find other ways to check your approach.

10.2 Testing the Components

In the process of the problem determination you need to check the status of the components of the system. The following sections list some steps that will help you in checking the condition of the components in a DCE cell.

10.2.1 Checking Network Connectivity

Check to make sure the machine in question has a working network connection because DCE relies on and requires a working connectivity. A problem with the network can be suspected when a user experiences long response times or time-outs, or when any remote service seems to fail. A network problem does not necessarily need to be at the machine that experiences trouble; a DCE server could become disconnected, which may cause clients to fail or to run very slowly due to connection time-outs.

Unless specifically configured for server preferences, a DCE client should generally be able to connect to all DCE servers. The servers need to be able to communicate with each other for replication and—since they also incorporate DCE client services—to all the other DCE core servers. DFS FLDB Servers must be able to communicate with all other FLDB Servers.

Try to ping the server machines if you suspect a network problem. This will test the basic network connectivity. Watch for unexpectedly long response times and for lost packets.

Depending on the type of environment that your cell could have, you might need to review the layout of the network. A skilled network analyst always starts with a complete understanding of the current network environment. Generally, this approach implies documentation of the network topology, applications, and protocols. If the network configuration is unknown, or if you are in doubt, use the `ping -R <hostname>`, and/or the `traceroute` commands to verify the network paths.

The `ping -R <hostname>` enables the IP record route feature, which causes every router that handles the packet to add its IP address to a list in the IP options field. Example:

```
# ping -R -c 5 ev3
PING ev3.itsc.austin.ibm.com: (9.3.1.122): 56 data bytes
64 bytes from 9.3.1.122: icmp_seq=0 ttl=255 time=3 ms
RR:     ev3.itsc.austin.ibm.com (9.3.1.122)
        ev2.itsc.austin.ibm.com (9.3.1.120)
64 bytes from 9.3.1.122: icmp_seq=1 ttl=255 time=3 ms   (same route)
64 bytes from 9.3.1.122: icmp_seq=2 ttl=255 time=3 ms   (same route)
64 bytes from 9.3.1.122: icmp_seq=3 ttl=255 time=3 ms   (same route)
64 bytes from 9.3.1.122: icmp_seq=4 ttl=255 time=3 ms   (same route)

----ev3.itsc.austin.ibm.com PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 3/3/3 ms
```

Use the ping command to verify if all servers can be contacted from the machine in question if you suspect a connectivity problem. It does, however, only test the network's correct routing capabilities and some basic TCP/IP code in the involved machines, but not any higher services.

As an alternative, you can run the dcecp cell ping command, as in this example:

```
dcecp> cell ping -replica
DCE servers available
```

The -replica option causes the dcecp cell ping command to only check the master and replica of the Security, CDS and DTS Servers. Otherwise, all machines in the cell are being pinged. If a server is unavailable, its name will be listed in the output of the command.

Additional tests, as described in the following sections, are required to check proper operation of any DCE services.

10.2.2 Checking DCE Processes

If you experience a situation where a certain service seems to be not working at all, you need to check if all the necessary processes and files are running on servers and clients. See 9.2, "DCE/DFS Process Checklist" on page 173, for a list of DCE processes that should run on each machine type.

You can check the processes running on a machine by using the ps command on AIX or the pstat command on OS/2. Here is an example of an AIX Security Server:

```
# ps -ef | grep dce
root  7022     1  12   Dec 07      -  2:10 /opt/dcelocal/bin/secd
root  9814     1   0   Dec 07      -  0:48 /opt/dcelocal/bin/dced -b
root 10018     1   0   Dec 07      -  0:05 /opt/dcelocal/bin/dtsd -s
root 10404     1   0   Dec 07      -  0:05 /opt/dcelocal/bin/cdsadv
root 15114 12396   2 15:54:09  pts/3  0:00 grep dce
```

The following example shows the status on an OS/2 Warp machine running both Security and CDS Servers:

```
[C:] pstat | find "DCE" | find ".EXE"
 01BD     0016      18     D:\OPT\DCELOCAL\BIN\CDSD.EXE   01   0200    FE294248    Block
 01BA     0016      17     D:\OPT\DCELOCAL\BIN\SECD.EXE   01   0200    FE013D7C    Block
 01B7     0016      16     D:\OPT\DCELOCAL\BIN\DCED.EXE   01   0200    FDFBE868    Block
 01BC     01B7      16     D:\OPT\DCELOCAL\BIN\CDSADV.EXE   01   0200    FE0ACE8C    Block
 0277     01BC      16     D:\OPT\DCELOCAL\BIN\CDSCLERK.EXE   01   0200    FDF583B8    Block
 D:\OPT\DCELOCAL\BIN\CDSD.EXE   01BD    18    C:\OPT\DCELOCAL\LIB\OS2ACPP.DLL
 D:\OPT\DCELOCAL\BIN\SECD.EXE   01BA    17    C:\OPT\DCELOCAL\LIB\OS2ACPP.DLL*
 D:\OPT\DCELOCAL\BIN\DCED.EXE   01B7    16    C:\OPT\DCELOCAL\LIB\OS2ACPP.DLL*
```

```
D:\OPT\DCELOCAL\BIN\CDSADV.EXE   01BC    16    C:\OS2\DLL\DOSCALL1.DLL*
D:\OPT\DCELOCAL\BIN\CDSCLERK.EXE   0277    16    C:\OPT\DCELOCAL\LIB\OS2ACPP.DLL*
```

As both examples above show, dced as the DCE host daemon runs on both machines. It is a prerequisite for all other DCE processes, except on the OS/2 slim client, where dced is not present.

After checking the basic network connectivity and the presence of the necessary processes, you can check the proper function of the DCE core servers.

10.2.3 Checking Your DCE Identity

Many of the tests that are described throughout this chapter require you to be logged on to DCE, either as any user or as a cell administrator, usually as the user *cell_admin*. You should be aware of the fact that the *root* user on AIX does not automatically have more permissions in DCE or DFS. Only if you are root and do not log in to DCE do you inherit the machine's principal and credentials, which has limited permissions.

Do the following checks to make sure you have the right credentials:

- If in doubt at all, you can take the safe route: A new login to DCE will provide you with new credentials for the principal you specify when performing the login.
- Check the expiration of your credentials. Tickets (credentials) in DCE have a lifetime which naturally expires (typically after 10 hours). If certain functions do not seem to work in the DCE/DFS environment, one of the first things you must check is if the tickets are still valid and check the ID for which you have credentials. A quick way of doing this is by issuing the `klist` command, which lists the cached DCE tickets. The output of the `klist` with valid tickets looks like:

```
# klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../itsc.austin.ibm.com/cell_admin
        Cell:      52b2d832-4e20-11d0-aae1-10005a4f4629 /.../itsc.austin.ibm.com
        Principal: 00000064-4e20-21d0-aa00-10005a4f4629 cell_admin
        Group:     0000000c-4e20-21d0-aa01-10005a4f4629 none
        Local Groups:
                0000000c-4e20-21d0-aa01-10005a4f4629 none
                00000064-4e20-21d0-9701-10005a4f4629 acct-admin
                00000065-4e20-21d0-9701-10005a4f4629 subsys/dce/sec-admin
                00000066-4e20-21d0-9701-10005a4f4629 subsys/dce/cds-admin
                00000067-4e20-21d0-9701-10005a4f4629 subsys/dce/dfs-admin
                00000068-4e20-21d0-9701-10005a4f4629 subsys/dce/dts-admin
                00000069-4e20-21d0-9701-10005a4f4629 subsys/dce/audit-admin
                0000006e-4ee4-21d0-9101-10005a4f4629 subsys/dce/dced-admin
                0000006f-4ee4-21d0-9101-10005a4f4629 ems-admin

Identity Info Expires: 97/04/15:02:26:45
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_63524036
Default principal: cell_admin@itsc.austin.ibm.com
Server: krbtgt/itsc.austin.ibm.com@itsc.austin.ibm.com
        valid 97/04/14:16:26:45 to 97/04/15:02:26:45
Server: dce-rgy@itsc.austin.ibm.com
        valid 97/04/14:16:26:45 to 97/04/15:02:26:45
Server: dce-ptgt@itsc.austin.ibm.com
        valid 97/04/14:16:27:30 to 97/04/14:18:27:30
Client: dce-ptgt@itsc.austin.ibm.com    Server:krbtgt/itsc.austin.ibm.com@itsc.austin.ib
m.com
```

```
        valid 97/04/14:16:27:30 to 97/04/14:18:27:30
Client: dce-ptgt@itsc.austin.ibm.com    Server: hosts/ev4/dfs-server@itsc.austin.ibm.com
        valid 97/04/14:16:27:30 to 97/04/14:18:27:30
Client: dce-ptgt@itsc.austin.ibm.com    Server: dce-rgy@itsc.austin.ibm.com
        valid 97/04/14:16:27:31 to 97/04/14:18:27:30
```

Don't get confused about the warning in the second line of the output. This is normal and does not indicate any problem.

If your credentials have expired when you do a `klist`, you would see the following, and it takes a long time to complete this output:

```
# klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../itsc.austin.ibm.com/cell_admin
        Cell:      52b2d832-4e20-11d0-aae1-10005a4f4629 /.../itsc.austin.ibm.com
        Principal: 00000064-4e20-21d0-aa00-10005a4f4629 cell_admin
        Group:     0000000c-4e20-21d0-aa01-10005a4f4629 <group name unknown>
        Local Groups:
                0000000c-4e20-21d0-aa01-10005a4f4629 <group name unknown>
                00000064-4e20-21d0-9701-10005a4f4629 <group name unknown>
                00000065-4e20-21d0-9701-10005a4f4629 <group name unknown>
                00000066-4e20-21d0-9701-10005a4f4629 <group name unknown>
                00000067-4e20-21d0-9701-10005a4f4629 <group name unknown>
                00000068-4e20-21d0-9701-10005a4f4629 <group name unknown>
                00000069-4e20-21d0-9701-10005a4f4629 <group name unknown>
                0000006e-4ee4-21d0-9101-10005a4f4629 <group name unknown>
                0000006f-4ee4-21d0-9101-10005a4f4629 <group name unknown>

Identity Info Expires: 97/04/15:02:26:45
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_63524036
Default principal: cell_admin@itsc.austin.ibm.com
```

If you are not logged in to DCE at all, you will get this output:

```
# klist
No DCE identity available: No currently established network identity for this co
ntext exists (dce / sec)

Kerberos Ticket Information:
klist: No credentials cache file found (dce / krb) (ticket cache /tmp/krb5cc_212
79)
```

You must then login to DCE, or if you are logged in already, renew your ticket. Renewing the ticket can be achieved with the `kinit` command.

- If you are logged in as the root user on AIX, or if you are on an OS/2 machine when you issue the `klist` command, you will always get an output similar to the one shown above. This is because the root user, or any user on OS/2, automatically inherits the machine's credentials. Make sure you are the DCE principal you want to be by checking the line with your principal name. Look at the following two examples and watch the difference:

```
# klist
DCE Identity Information:
        Global Principal: /.../itsc.austin.ibm.com/cell_admin
        Cell:      52b2d832-4e20-11d0-aae1-10005a4f4629 /.../itsc.austin.ibm.com
        Principal: 00000064-4e20-21d0-aa00-10005a4f4629 cell_admin
        Group:     0000000c-4e20-21d0-aa01-10005a4f4629 none
...

# klist
DCE Identity Information:
        Global Principal: /.../itsc.austin.ibm.com/hosts/ev2/self
        Cell:      52b2d832-4e20-11d0-aae1-10005a4f4629 /.../itsc.austin.ibm.com
        Principal: 00000069-4e21-21d0-9700-10005a4f4629 hosts/ev2/self
        Group:     0000000c-4e20-21d0-aa01-10005a4f4629 none
...
```

The first example shows an output when `cell_admin` is logged in to DCE, whereas the second shows you the `self` principal of the machine you are working on.

- Are you logged into the correct cell? Use the `klist` command to find out what cell you are logged in to and what user ID you have logged in with. Check the cell name. If you are in a foreign cell, check if you have foreign cell identification.

10.2.4 Checking the Security Services

The most common and simple, but effective way to check the proper function of the security services (Security Server) is to perform a login to DCE on a client machine, for example by using the command line commands `dce_login` on AIX, or `dcelogin` on OS/2, respectively. If it succeeds immediately, or within a few seconds, you can be assured that your client system has access to at least one Security Server that works fine. If it takes significantly longer than usual in your environment, say more than ten seconds, but still succeeds, you might have one of the following problems:

- A slow or congested network adds some additional response time. This can be either a normal, temporary or static situation due to the network layout and utilization, or it might indicate a problem. Use the `ping` command to further investigate connectivity or response-time problems.
- The Security Server(s) might be temporarily overloaded.
- Not all of the Security Server replicas are working or are accessible by the client system. Depending on its configuration, a client tries all replica servers in a specified or random order. If a Security Server is unavailable, the client attempts to connect to the next one after a time-out period, which causes a login to take more time. Section 7.2.1, “Security Service, the pe_site File” on page 142, explains how clients select their Security Servers.
- If it takes a very long time (up to minutes), it usually indicates a situation where a Security Server is not connectable through the network, or a Security Server machine is down, since the time-outs for such situations are longer. Check the status of the Security Servers and/or the network to find the cause.

To find out if a specific Security Server replica is working, set the `TRY_PE_SITE` environment variable and point to that server in the pe_site file before performing the login to DCE.

Note

The login test as described does not include the testing of the clients’ DCE components. It is perfectly possible to login to DCE without having DCE client services running on a client. The login test therefore aims for the Security Servers, not for the clients.

If you want to know about the status of the security client services on your client machine, first check if the dced process is running (except on OS/2 slim clients). You can also, or in addition, issue the following dcecp commands on the client machine:

```
dcecp> secval status
1
```

A return value of 0 (zero) means that the security client service is not working; 1 (as in the example) indicates a positive test result. Other error messages may be displayed as well, all indicating a problem with the security client.

Alternatively, if you want to verify your current credentials through the security client services, issue a cdscp secval ping command:

```
dcecp> secval ping
1
```

A return value of 1 indicates that your credentials are valid, and the security client service are working. If you get a 0 (zero) or an error message, you either do not have valid DCE credentials, or you logged into DCE on a machine where the security client service is inoperative, typically because dced or cdsadv is not running.

10.2.5 Checking User Accounts

The following examples give you some tips on looking at specific information about user accounts in case you suspect troubles there.

- To list the names of all accounts in the registry database, use dcecp account catalog command. If you want to see only the names without the cell name, use the -simplename option.

  ```
  dcecp> account catalog
  /.../itsc.austin.ibm.com/nobody
  /.../itsc.austin.ibm.com/root
  /.../itsc.austin.ibm.com/daemon
  /.../itsc.austin.ibm.com/uucp
  /.../itsc.austin.ibm.com/bin
  /.../itsc.austin.ibm.com/dce-ptgt
  /.../itsc.austin.ibm.com/dce-rgy
  /.../itsc.austin.ibm.com/krbtgt/itsc.austin.ibm.com
  /.../itsc.austin.ibm.com/cell_admin
  /.../itsc.austin.ibm.com/hosts/ev1/self
  /.../itsc.austin.ibm.com/hosts/ev2/self
  /.../itsc.austin.ibm.com/hosts/ev2/cds-server
  /.../itsc.austin.ibm.com/hosts/ev4/self
  /.../itsc.austin.ibm.com/hosts/ev3/self
  /.../itsc.austin.ibm.com/hosts/ev2/dfs-server
  /.../itsc.austin.ibm.com/hosts/ev4/dfs-server
  /.../itsc.austin.ibm.com/hosts/itsonv/self
  /.../itsc.austin.ibm.com/hosts/mozart/self
  /.../itsc.austin.ibm.com/hosts/bizet.austin.ibm.com/self
  /.../itsc.austin.ibm.com/warp4Osrv.snmpagent
  /.../itsc.austin.ibm.com/h_johner
  /.../itsc.austin.ibm.com/jensen
  ```

- To check the attributes of a specific account, use the dcecp account show command.

  ```
  dcecp> account show h_johner
  {acctvalid yes}
  {client yes}
  {created /.../itsc.austin.ibm.com/cell_admin 1997-04-11-11:28:01.000-05:00I-----}
  {description {Heinz Johner}}
  {dupkey no}
  {expdate none}
  {forwardabletkt yes}
  {goodsince 1997-04-11-11:27:00.000-05:00I-----}
  ```

```
{group none}
{home /.../itsc.austin.ibm.com/fs/home/h_johner}
{lastchange /.../itsc.austin.ibm.com/cell_admin
1997-04-11-11:28:01.000-05:00I-----}
{organization none}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell /bin/ksh}
{stdtgtauth yes}
```

Some important information you might want to check in case of suspected problems are highlighted in bold type.

- A common situation with users is expired credentials. The easiest way to check if the credentials have expired is to issue a `klist` command. If the output appears right away and the command finishes within seconds, the credentials are still valid. When they have expired, the `klist` output stops for long periods of time when displaying the group membership lines, and all groups will be marked with ″group name unknown″. See also 10.2.3, “Checking Your DCE Identity” on page 185.

10.2.6 Checking the CDS

The important things that you should do when you suspect a problem in the CDS area is to check the health of the CDS Servers and the state of the replica servers. This can best be done by accessing all the data in the servers. The following examples can be used to find this type of information.

- The quickest, most thorough check for CDS is to list all directories and objects in the CDS namespace. Depending on the size of the cell, it might take some time to list all entries (the following output has been shortened):

```
# cdsli -world
o       /.:/AcmeBank_admin
o       /.:/arithmetic_ev2
o       /.:/cell-profile
o       /.:/execs
o       /.:/fs
o       /.:/hjo_bank
o       /.:/lan-profile
o       /.:/sec
o       /.:/sec-v1
d       /.:/DE-Light
o       /.:/DE-Light/gate1
d       /.:/DE-Light/examples
...
d       /.:/hosts
d       /.:/hosts/ev1
o       /.:/hosts/ev1/cds-clerk
o       /.:/hosts/ev1/config
o       /.:/hosts/ev1/dts-entity
o       /.:/hosts/ev1/profile
o       /.:/hosts/ev1/self
...
d       /.:/subsys
d       /.:/subsys/dce
d       /.:/subsys/dce/dfs
```

```
o       /.:/subsys/dce/dfs/bak
d       /.:/subsys/dce/sec
o       /.:/subsys/dce/sec/ev2
o       /.:/subsys/dce/sec/ev4
d       /.:/users
```

By creating this list, the whole namespace is searched, which indicates indirectly that CDS services work. If the command stops for a remarkable time before continuing listing the objects and directories, it might indicate a situation where a CDS Server is not available. Watch the directory where it stops and check the CDS replica server that stores this directory.

- To get a quick overview of the clearinghouse configuration, issue the dcecp clearinghouse catalog and dcecp clearinghouse show commands:

```
dcecp> clearinghouse catalog
/.../itsc.austin.ibm.com/ev2_ch
/.../itsc.austin.ibm.com/ev4_ch

dcecp> clearinghouse show /.../itsc.austin.ibm.com/ev2_ch
{CDS_CTS 1996-12-04-22:00:50.735174100/10-00-5a-a8-cf-f8}
{CDS_UTS 1996-12-04-22:04:12.367027100/10-00-5a-a8-cf-f8}
{CDS_ObjectUUID daeab426-4e21-11d0-a06d-10005aa8cff8}
{CDS_AllUpTo 1997-04-13-18:05:02.596452100/10-00-5a-a8-cf-f8}
{CDS_DirectoryVersion 3.0}
{CDS_CHName /.../itsc.austin.ibm.com/ev2_ch}
{CDS_CHLastAddress
 {Tower {ncacn_ip_tcp 9.3.1.120}}
 {Tower {ncadg_ip_udp 9.3.1.120}}}
{CDS_CHState on}
{CDS_CHDirectories
 {{Dir_UUID dbc5568a-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com}}
 {{Dir_UUID f26dffea-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/subsys}}
 {{Dir_UUID f2e43e80-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/subsys/dce}}
 {{Dir_UUID f3366818-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/subsys/dce/sec}}
 {{Dir_UUID f3df1b84-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/subsys/dce/dfs}}
 {{Dir_UUID f459b894-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/hosts}}
 {{Dir_UUID f4cc3a68-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/hosts/ev2}}
 {{Dir_UUID f52c8210-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/users}}
 {{Dir_UUID e3b81eac-4e23-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/hosts/ev4}}
 {{Dir_UUID e7e371fc-4e23-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/hosts/ev3}}
 {{Dir_UUID 82db694c-4e26-11d0-a621-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/hosts/ev1}}
 {{Dir_UUID ed1f9386-4e31-11d0-a621-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/hosts/itsonv}}
 {{Dir_UUID 647c89d6-9039-11d0-a755-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/users/sec_client}}
 {{Dir_UUID 9dda9cda-9fcb-11d0-a55c-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/users/gwy-prin}}
 {{Dir_UUID 2021edba-9fcc-11d0-a55c-10005aa8cff8}
```

```
  {Dir_Name /.../itsc.austin.ibm.com/users/gwy-admin}}
 {{Dir_UUID 214b49c6-b010-11d0-82b0-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/subsys/DEC}}
 {{Dir_UUID 21ce5104-b010-11d0-82b0-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/subsys/DEC/pc}}}
{CDS_ReplicaVersion 3.0}
{CDS_NSCellname /.../itsc.austin.ibm.com}

dcecp> clearinghouse show /.../itsc.austin.ibm.com/ev4_ch
{CDS_CTS 1997-04-11-17:21:25.967080100/10-00-5a-a8-cf-f8}
{CDS_UTS 1997-04-11-21:46:40.827740100/02-60-8c-2f-06-53}
{CDS_ObjectUUID 0979d9ac-b290-11d0-9f8d-02608c2f0653}
{CDS_AllUpTo 1997-04-14-14:46:17.809587100/02-60-8c-2f-06-53}
{CDS_DirectoryVersion 3.0}
{CDS_CHName /.../itsc.austin.ibm.com/ev4_ch}
{CDS_CHLastAddress
 {Tower {ncacn_ip_tcp 9.3.1.123}}
 {Tower {ncadg_ip_udp 9.3.1.123}}}
{CDS_CHState on}
{CDS_CHDirectories
 {{Dir_UUID dbc5568a-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com}}
 {{Dir_UUID f3366818-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/subsys/dce/sec}}
 {{Dir_UUID f459b894-4e21-11d0-a06d-10005aa8cff8}
  {Dir_Name /.../itsc.austin.ibm.com/hosts}}}
{CDS_ReplicaVersion 3.0}
{CDS_NSCellname /.../itsc.austin.ibm.com}
```

The first command above lists all the CDS Servers in the cell. In this example, we have two CDS Servers, *ev2* and *ev4*. The second and third command examples list the characteristics of the CDS Servers. A look at the second CDS Server (ev4) shows that is has three directories replicated, the CDS root directory (/.../itsc.austin.ibm.com, or /.:), the hosts directory (/.:/hosts), and the directory /.:/subsys/dce/sec.

- You can also use the `cdscp show cell` command to display information about the CDS Servers. It gives you a condensed output, including the IP addresses of the master and replica CDS Servers:

```
cdscp> show cell
                        SHOW
                        CELL   /.../itsc.austin.ibm.com
                          AT   1997-04-14-10:18:46
            Namespace Uuid = dbc5568a-4e21-11d0-a06d-10005aa8cff8
          Clearinghouse Uuid = daeab426-4e21-11d0-a06d-10005aa8cff8
          Clearinghouse Name = /.../itsc.austin.ibm.com/ev2_ch
                Replica Type =   Master
                       Tower = ncacn_ip_tcp:9.3.1.120[]
                       Tower = ncadg_ip_udp:9.3.1.120[]

            Namespace Uuid = dbc5568a-4e21-11d0-a06d-10005aa8cff8
          Clearinghouse Uuid = 0979d9ac-b290-11d0-9f8d-02608c2f0653
          Clearinghouse Name = /.../itsc.austin.ibm.com/ev4_ch
                Replica Type =  Readonly
                       Tower = ncacn_ip_tcp:9.3.1.123[]
                       Tower = ncadg_ip_udp:9.3.1.123[]
```

- You can display the attributes and the statistics of the CDS clerk on a machine by using the `cdscp show clerk` command:

```
cdscp> show clerk
                          SHOW
                         CLERK
                            AT   1997-04-14-10:20:58
                 Creation Time = 1997-04-10-14:22:55.780
      Authentication Failures = 0
              Read Operations = 3433
                   Cache Hits = 2360
               Cache Bypasses = 303
             Write Operations = 23
    Miscellaneous Operations = 89
```

When you receive this output, the CDS clerk is running, and it is responding to requests.

- Using the cdscp show server command, you can see the values of the attributes associated with the server running on the local system (the command must be run on a CDS Server):

```
cdscp> show server
                          SHOW
                        SERVER
                            AT   1997-04-14-10:22:23
                 Creation Time = 1997-04-14-09:20:08.125
              Future Skew Time = 0
               Read Operations = 469
              Write Operations = 47
              Skulks Initiated = 6
              Skulks Completed = 6
   Times Lookup Paths Broken = 0
              Crucial Replicas = 0
         Child Update Failures = 0
             Security Failures = 0
          Known Clearinghouses = /.../itsc.austin.ibm.com/ev2_ch
```

Once you receive this answer, you can be sure that the server is running and responds to requests.

- If you suspect a problem when accessing a directory in CDS, use the dcecp directory show command to check access manually:

```
dcecp> directory show /.:/hosts
{RPC_ClassVersion {01 00}}
{CDS_CTS 1996-12-04-22:01:33.398643100/10-00-5a-a8-cf-f8}
{CDS_UTS 1997-04-14-15:28:07.597365100/10-00-5a-a8-cf-f8}
{CDS_ObjectUUID f459b894-4e21-11d0-a06d-10005aa8cff8}
{CDS_Replicas
 {{CH_UUID daeab426-4e21-11d0-a06d-10005aa8cff8}
  {CH_Name /.../itsc.austin.ibm.com/ev2_ch}
  {Replica_Type Master}
  {Tower {ncacn_ip_tcp 9.3.1.120}}
  {Tower {ncadg_ip_udp 9.3.1.120}}}
 {{CH_UUID 0979d9ac-b290-11d0-9f8d-02608c2f0653}
  {CH_Name /.../itsc.austin.ibm.com/ev4_ch}
  {Replica_Type ReadOnly}
  {Tower {ncacn_ip_tcp 9.3.1.123}}
  {Tower {ncadg_ip_udp 9.3.1.123}}}}
{CDS_AllUpTo 1997-04-14-14:55:43.126749100/10-00-5a-a8-cf-f8}
{CDS_Convergence medium}
{CDS_ParentPointer
 {{Parent_UUID dbc5568a-4e21-11d0-a06d-10005aa8cff8}
  {Timeout
```

```
    {expiration 1997-04-15-09:55:46.374}
    {extension +1-00:00:00.000I0.000}}
   {myname /.../itsc.austin.ibm.com/hosts}}}
{CDS_DirectoryVersion 3.0}
{CDS_ReplicaState on}
{CDS_ReplicaType ReadOnly}
{CDS_LastSkulk 1997-04-14-14:55:43.126749100/10-00-5a-a8-cf-f8}
{CDS_LastUpdate 0}
{CDS_Epoch 2b0e5b94-b4d7-11d0-ab32-10005aa8cff8}
{CDS_ReplicaVersion 3.0}
```

This output tells you, among other information, where the replicas of this directory are. As well as with the `directory show`, you can also list the attribute information of a soft link by using the `dcecp link show` command.

As with most dcecp commands, you could have used the native command as well, which would have been `cdscp show directory /.:/hosts` for the above example. The output differs in its representation, but the information is basically the same.

- With the `dcecp object show` (or the analogous `cdscp show object`) command, you can display information about an object, and as a side benefit you can be sure that the object could be addressed and retrieved.

```
dcecp> object show /.:/hosts/ev1/self
{RPC_ClassVersion {01 00}}
{CDS_CTS 1996-12-04-22:34:11.157378100/10-00-5a-a8-cf-f8}
{CDS_UTS 1996-12-04-22:34:26.927437100/10-00-5a-a8-cf-f8}
{CDS_Towers ncadg_ip_udp 9.3.1.68 135}
```

- The `dcecp cdscache show` operation displays address information about a clearinghouse and servers stored in the local cache of the machine. By getting an answer from this command, you can assume that the cache on the local machine is working.

```
dcecp> cdscache show -clearinghouse /.:/ev4_ch
{CH_Name /.../itsc.austin.ibm.com/ev4_ch}
{Created 1997-04-11-12:31:37.600}
{Others 25}
{Reads 381}
{Tower {ncacn_ip_tcp 9.3.1.123}}
{Tower {ncadg_ip_udp 9.3.1.123}}
{Writes 0}
```

You can see the protocol sequence(s) and IP address(es) of the server that maintains the cached clearinghouse. Also, from the statistics you can see how many times read and write operations were performed by this clerk on the cached clearinghouse.

10.2.7 Checking Access Permissions

Many resources in DCE (directories and objects) have Access Control Lists (ACLs) associated with them. This is not only the case within DFS where files and directories have ACLs to protect them from unauthorized access. CDS directories and objects may be protected with ACLs, too. In normal operation, a DCE administrator does not have to care about these ACLs, but in rare situations, for example when a configuration step aborts due to an error, the ACLs might not be correctly set.

Error messages pointing at such problems usually include a glue in the text, saying, for example, that you do not have enough access permissions to do a

specific operation. ACLs can be viewed and edited with the acl_edit command. The following example shows the ACLs associated with the /.:/hosts/ev1 directory:

```
# acl_edit /.:/hosts/ev1
sec_acl_edit> l

# SEC_ACL for /.:/hosts/ev1:
# Default cell = /.../itsc.austin.ibm.com
unauthenticated:r--t---
user:cell_admin:rwdtcia
user:hosts/ev2/cds-server:rwdtcia
user:hosts/ev1/self:rwdtcia
group:subsys/dce/cds-admin:rwdtcia
group:subsys/dce/cds-server:rwdtcia
any_other:r--t---
```

You should be careful with ACLs since they protect resources. Thus, before you edit ACLs, you should check your identity (credentials) and think about the ACL's current access rights; they might have been set correctly, but you tried with a principal that is denied access on purpose.

To find out what the necessary ACLs should be, it is a good idea to look them up in a working reference cell, for example in a test cell. If you still suspect problems in ACLs, open the rights carefully for the purpose needed.

10.2.8 Checking the DTS

When you suspect a problem in the Distributed Time Services (DTS), you should check initially if the DTS daemon (dtsd) is active on the machines involved in the problem. Both the DTS Servers and the clients need to be checked. The clock synchronization can also be checked, within limits and in terms of accuracy, manually without DCE and DTS involvement.

The dcecp dts show command returns some valuable information about the status of DTS on the local machine:

```
dcecp> dts show -all
{tolerance +0-00:05:00.000I-----}
{tdf -0-05:00:00.000I-----}
{maxinaccuracy +0-00:00:00.100I-----}
{minservers 2}
{queryattempts 3}
{localtimeout +0-00:00:05.000I-----}
{globaltimeout +0-00:00:15.000I-----}
{syncinterval +1-00:00:00.000I-----}
{type clerk}
{clockadjrate 10000000 nsec/sec}
{maxdriftrate 50000 nsec/sec}
{clockresolution 10000000 nsec}
{version V1.0.1}
{timerep V1.0.0}
{autotdfchange no}
{nexttdfchange 1997-10-26-01:00:00.000-06:00I0.000}
{status enabled}
{localservers
 {name /.../itsc.austin.ibm.com/hosts/ev2/self}
 {timelastpolled 1997-04-14-04:41:02.603-05:00I-----}
 {lastobstime 1997-04-14-04:40:58.844-05:00I-----}
```

```
 {lastobsskew +0-00:00:03.760I-----}
 {inlastsync TRUE}
 {transport RPC}}
{localservers
 {name /.../itsc.austin.ibm.com/hosts/ev1/self}
 {timelastpolled 1997-04-14-04:41:02.603-05:00I-----}
 {lastobstime 1997-04-14-04:41:06.841-05:00I-----}
 {lastobsskew +0-00:00:04.237I-----}
 {inlastsync TRUE}
 {transport RPC}}
{creationtime 1997-04-10-13:38:26.262-05:00I-----}
{nointersections 0}
{toofewservers 0}
{providertimeouts 0}
{badprotocols 0}
{badtimerep 0}
{abrupts 0}
{syserrors 0}
{syncs 5}
{enables 1}
{disables 0}
{nomemories 0}
{badlocalservers 0}
```

The `-all` flag in the command causes the output to contain information about the configuration and current counters. As you can see, you can check the configuration attributes of the DTS client (or server, if the command was run on a DTS Server machine). The example shows a client that has only two DTS Servers to access. Accordingly, the `minservers` value was set to 2 (two), and it actually lists two DTS Servers it used for time synchronization. You should see the local servers defined in your cell in the clients' output, as in the example above.

You also can check how many DTS Servers are registered in your cell. To do this, you could use the `dcecp dts catalog` command.

```
dcecp> dts catalog
/.../itsc.austin.ibm.com/hosts/ev1/dts-entity
/.../itsc.austin.ibm.com/hosts/ev2/dts-entity
```

This command only shows static configuration information and does not list currently running servers, but lets you know what they are. You can then test the DTS daemons on remote machines by using the `dcecp clock show` command. You can see and check the actual clock on the remote systems, as well as on the local machine, and you can see the time difference between them.

```
dcecp> clock show /.../itsc.austin.ibm.com/hosts/ev1/dts-entity
1997-04-14-12:33:40.534-05:00I-----
dcecp> clock show /.../itsc.austin.ibm.com/hosts/ev2/dts-entity
1997-04-14-12:31:53.427-05:00I-----
dcecp> clock show
1997-04-14-12:34:01.543-05:00I-----
```

When having problems with DTS, the easiest way to recover is to restart DTS on the machine(s) where the problem is assumed.

10.2.9 Checking the DFS Servers

The processes, according to 9.2, “DCE/DFS Process Checklist” on page 173, that support the DFS operation must be all up and running in order to process client requests. In addition to using operating system commands to check the existence of the processes, you can use other commands to check DFS Servers, as listed below:

- Check if the DFS File Servers are running by issuing the following command on any DFS client machine that have access to the DFS services:

  ```
  # cm statservers -all
  ```

  The usual answer is:

  ```
  All servers are running.
  ```

  If you receive a message telling you that the server is down, that means that the File Server machine(s) listed in the message did not respond to this call and the DFS Cache Manager marks them as being down. These machines may be down, experiencing a temporary network outage, DCE/DFS failure or a hardware failure.

  If you get:

  ```
  cm: Function not implemented.
  ```

  or

  ```
  cm: The system call does not exist on this system.
  ```

  this means that there are no DFS client processes running on the local machine.

- List the cells and the FLDB Servers:

  ```
  # cm lscellinfo
  Cell itsc.austin.ibm.com on host ev4.itsc.austin.ibm.com.
  ```

  You will receive one line of information about every cell that this client has accessed and all host names of the FLDB Servers within these cells. The example shows just one cell with only one FLDB Server.

- Check the status of the DFS processes on DFS Servers:

  ```
  # bos status -s /.:/hosts/ev4
  Instance flserver, currently running normally.
  Instance ftserver, currently running normally.
  Instance upserver, currently running normally.
  ```

  If it cannot contact the DFS server processes on the DFS Server, the output could look like:

  ```
  bos: failed to contact host's bosserver (communications failure (dce / rpc))
  ```

- DFS could be having a problem due to a full aggregate or a fileset quota limit. Although all processes seem to work fine, writing to DFS would still not be possible. Users usually get error messages indicating such a ″no space left″ condition. In such a case, free space has to be made available in DFS, either by deleting files, by rising the fileset quota, or by enlarging the aggregate.

- There are a number of other commands that can used to check DFS health and that can give you a glue of the problem source. Some of the commands you can use for checking proper operation or if error messages are returned with some valuable information are:

  – `fts lsfldb`

- fts lsheader
- fts lsft
- fts lsaggr
- fts lsreplicas

An administrator should use these commands regularly to get familiar with them and to recognize any deviation in case of a problem.

10.2.10 Checking the DFS Clients

You need to check the status of your DFS client when you suspect a problem with it. These are the most common steps to follow for checking the status of your DFS client, no matter whether on an AIX DFS client and an OS/2 DFS client.

Check that the DFS Client is Configured and Started: On AIX, use the df command to check if the DFS client is running:

```
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           24576      8120   67%     1648    27% /
/dev/hd2         1785856     20680   99%    28172    13% /usr
...
DFS             18000000  18000000    0%        0     0% /...
...
```

Make sure the line with DFS as the filesystem as shown in the example appears in the output of the df command. If it does not, either DFS client is not configured or not started. In the latter case, issue the /etc/rc.dfs command as root to start DFS.

On OS/2, you should see the remote drive icon depicting the DFS storage area in the Drives folder that can be found in the OS/2 System folder on the desktop. If this drive icon does not show up, your machine is probably not configured as a DFS client, or DCE/DFS is not currently running.

If you are still having a problem accessing your data, verify that the file system containing your DFS cache is not full (or, on OS/2, the disk containing the DFS cache is not full):

```
# df
Filesystem   512-blocks     Free %Used  Iused %Iused Mounted on
/dev/hd4          16384     2904   83%   1315    33% /
/dev/hd2         901120    15312   99%  19266    18% /usr
/dev/hd9var        8192     6256   24%    142    14% /var
/dev/hd3          24576    21472   13%    152     4% /tmp
/dev/hd1          40960     4960   88%    828    17% /home
...
/dev/dcelv        24576    18520   25%     96     3% /var/dce
/dev/cachelv      24576     4976   80%    643    16% /var/dce/adm/dfs/cache
...
DFS            18000000 18000000    0%      0     0% /...
...
```

In this example, a separate filesystem is used for the DFS cache. You can easily see how much disk space is currently used by the DFS cache, and also by all the other components of your filesystem.

Check Access to the DFS Filespace: If the DFS filespace is not accessible, you need to review the following points of checking. If any error messages are displayed, read them carefully because they might contain an indication of the problem.

- Check if other DFS clients have similar problems, which would indicate a problem with a DFS Server. See 10.2.9, “Checking the DFS Servers” on page 196.
- Are you logged in to DCE as the right user, or have your credentials expired? See 10.2.3, “Checking Your DCE Identity” on page 185, for more details.
- Check if you have a permission problem. If you get an error message indicating insufficient permissions when you want to access DFS, then you should try with another user identity that has sufficient rights. Otherwise, you may want to change the permissions in DFS. AIX DFS, for example, is very restrictive with permissions by default, and you must change the permissions of the DFS root directory before even `cell_admin` is allowed to access it.
- Verify the functions of the DCE core services, the Security and CDS Services. Try to login to DCE:

  ```
  # dce_login <user>
  ```

  Proper function of the DCE core services is a prerequisite for DFS.
- If you have just (re)started your DFS File Server(s), you might be too early to access DFS. After starting up, DFS may be unavailable for up to 10 minutes for re-synchronizing tokens with clients. This is called the TSR (Token State Recovery) mode.
- Check if the FLDB Server(s) is (are) available and if you can list the entries:

  ```
  # fts lsfldb
  ```
- Check if the File Servers are alive and if they hold valid filesets:

  ```
  # fts lsheader -server <server>
  ```
- Check if the system clocks are within about a two minute interval across your clients and the DFS Servers in the cell. You can use any command you like, such as `telnet`, to remotely log in to other systems to verify the other systems’ clocks. DFS does not require high accuracy; thus, manual verification is good enough.

Check Access to Files within DFS: If DFS seems to be running okay, but certain files within DFS cannot be accessed, try the following checks:

- Check where the files reside. Files are physically distributed in DFS, even among different servers, according to system administrator definition. The `cm whereis` command lets you know on which DFS Server a certain file or directory is physically stored:

  ```
  # cm whereis /:/home/java
  The file '/:/home/java' resides in the cell 'itsc.austin.ibm.com',
  in fileset 'home.ft', on host ev4.itsc.austin.ibm.com.
  ```

  This might give you a hint about where to look further if you suspect a problem on the server side.
- Are you logged in to DCE as the right user and are your credentials still valid? See 10.2.3, “Checking Your DCE Identity” on page 185, for more details.

- To check if you have the necessary permissions to access the file or directory, enter the command:

  ```
  # acl_edit <filename_or_directoryname> -l
  ```

  If you have the proper access rights to list the ACLs, then the permissions for the owner, group and others will be shown, along with any individual principal and group permissions. To find out if you are in one of the groups with access permission, use the `klist` command. The output will show you your principal name and the groups that you are a member of. You can compare this list against the ACLs of the object to determine what access rights you actually have.

  Alternatively, you can use the `test_access` (can be abbreviated to `t`) subcommand within `acl_edit` to test your effective permissions:

  ```
  # acl_edit /:/home/java
  sec_acl_edit> test_access w
  Access: GRANTED
  sec_acl_edit> t r
  Access: GRANTED
  ```

- If you want to make sure you have the most current version of the file, you can force a reload of the cache with the `cm flush` command. This should normally not be necessary since DFS takes care of file updates:

  ```
  # cm flush <filename>
  ```

  This command flushes the specified file from your DFS cache.

- If you are accessing a file from a replicated fileset, you might not have the latest version, if the replica was just updated with a new version of a fileset. Use the `cm checkfilesets` command to force DFS to check and load the newest version of the fileset information:

  ```
  # cm checkfilesets
  ```

Diagnosing Write Problems in DFS: When you have troubles writing to files in DFS, try the following.

- Check the fileset quota by running the `fts lsquota` command:

  ```
  # fts lsquota /:/home
  Fileset Name          Quota    Used  % Used   Aggregate
  home.ft                9000    6665    74%    40% = 10439/25800 (LFS)
  ```

  This shows the name of the fileset, the quota in kilobytes, and the percentage used on the fileset and the aggregate. Obviously, these percentages should be below 100 percent.

- Are you logged in to DCE as the right user and are your credentials still valid? See 10.2.3, “Checking Your DCE Identity” on page 185, for more details. If your credentials have expired, you could see an error message like this:

  ```
  dfs: ticket has expired, running unauthenticated.
  ```

- Check for the correct file permissions in DFS. When creating a new file in a directory, or for certain other operations, you will also need to have the correct permissions in the directory where you want to do your changes. For example, creating a new file requires at least the *write* and *insert* permission on the directory containing the file.

- Is the aggregate full? Use the `fts aggrinfo` command to check how much free space is left on the aggregate:

```
# fts aggrinfo -server ev2
LFS aggregate root.dfs (/dev/root.dfs): 3222 K free out of total 3680 (408
reserved)
LFS aggregate data.dfs (/dev/data.dfs): 10525 K free out of total 11056
(1224 reserved)
```

If it appears to be full, you need to free up some space, or you will have to increase its size.

10.2.11 Check DCE Core Server Replication Status

If configured for replication, both the Security and the CDS Servers maintain a set of information related to replication. The information is used by the master component of either service to keep track of the replicas.

The Master Security Server uses a sequence number to keep track of each replica update. To display the current replication information of a security replica, use the dcecp registry show command:

```
dcecp> registry show -master
{name /.../itsc.austin.ibm.com/subsys/dce/sec/warp40srv}
{type slave}
{propstatus update}
{lastupdtime 1996-11-29-10:31:55.000-06:00I-----}
{lastupdseqsent 0.13143}
{numupdtogo 0}
{lastcommstatus 0}

{name /.../itsc.austin.ibm.com/subsys/dce/sec/master}
{type master}
```

And the same on the Master Security Server:

```
dcecp> registry show -replica
{name /.../itsc.austin.ibm.com/subsys/dce/sec/master}
{type master}
{cell /.../itsc.austin.ibm.com}
{uuid f87b86d0-31db-11d0-8821-10005a4f4629}
{status enabled}
{lastupdtime 1996-11-29-10:31:55.000-06:00I-----}
{lastupdseq 0.13143}
{addresses
 {ncacn_ip_tcp 9.3.1.68}
 {ncadg_ip_udp 9.3.1.68}}
{masteraddrs
 {ncacn_ip_tcp 9.3.1.68}
 {ncadg_ip_udp 9.3.1.68}}
{masterseqnum 0.100}
{masteruuid f87b86d0-31db-11d0-8821-10005a4f4629}
{supportedversions
 secd.dce.1.0.2
 secd.dce.1.1}
{updseqqueue {0.13143 0.13143}}
```

As the update sequence queue numbers (lastupdseqsent and updseqqueue) indicate, the master and slave have performed replication successfully and are on the same level of the registry database. A difference in the update sequence can temporarily happen, but should not stay for long. A large difference in the update sequence numbers, maybe along with a lastupdttime that is far in the past, indicates a problem with the replica updates.

The CDS replication mechanism also keeps track of the replication status. Every replicated directory should be checked if a problem in replication propagation is suspected. The following example shows the CDS root directory (/.:) with the pertinent information highlighted:

```
dcecp> directory show /.:
{RPC_ClassVersion {01 00}}
{CDS_CTS 1996-12-04-22:00:52.177135100/10-00-5a-a8-cf-f8}
{CDS_UTS 1996-12-27-18:18:28.020632100/10-00-5a-a8-cf-f8}
{CDS_ObjectUUID dbc5568a-4e21-11d0-a06d-10005aa8cff8}
{CDS_Replicas
 {{CH_UUID daeab426-4e21-11d0-a06d-10005aa8cff8}
  {CH_Name /.../itsc.austin.ibm.com/ev2_ch}
  {Replica_Type Master}
  {Tower {ncacn_ip_tcp 9.3.1.120}}
  {Tower {ncadg_ip_udp 9.3.1.120}}}
 {{CH_UUID 9a0f8c4a-d6bd-11d0-a922-02608c2f0653}
  {CH_Name /.../itsc.austin.ibm.com/ev4_ch}
  {Replica_Type ReadOnly}
  {Tower {ncacn_ip_tcp 9.3.1.123}}
  {Tower {ncadg_ip_udp 9.3.1.123}}}}
{CDS_AllUpTo 1996-12-27-18:18:22.205339100/10-00-5a-a8-cf-f8}
{CDS_Convergence high}
{CDS_InCHName new_dir}
{CDS_DirectoryVersion 3.0}
{CDS_ReplicaState on}
{CDS_ReplicaType Master}
{CDS_LastSkulk 1996-12-27-18:18:25.130181100/10-00-5a-a8-cf-f8}
{CDS_LastUpdate 1996-12-27-18:18:30.589666100/10-00-5a-a8-cf-f8}
{CDS_Epoch c7bfaefc-d6ba-11d0-a4e2-10005aa8cff8}
{CDS_ReplicaVersion 3.0}
```

As can be seen, the master and replica version of these directories are well in sync. The CDS_AllUpTo value shows the date and time when all replicas were successfully updated the last time. In addition, the CDS_LastSkulk and the CDS_LastUpdate values show the date and time of the last skulk and update operations. If the date and time of the CDS_AllUpTo attribute is far in the past compared to the CDS_LastSkulk and CDS_LastUpdate values, a CDS replica server may have been incompletely removed from the cell, or it may have a severe problem. If your CDS namespace contains entries to a clearinghouse that does not exist anymore, see 11.1.9, “Reference to a Non-Existing Clearinghouse in CDS” on page 209.

10.3 Using DCE Debug and Trace Options

Most DCE and DFS components support their own debug and trace options. If you experience problems with DCE, it is always a good idea to look at the various log files (see 9.3, “Log Files” on page 174). The amount and type of information that is being logged is configurable through the serviceability features of DCE.

More information about the serviceability features of DCE can be found in 5.7, “Planning for SVC, EMS, and Auditing” on page 118, and in Appendix A, “Using the DCE Debug and Messaging Facilities” on page 225.

10.4 DCE and DFS Error Messages

It would be beyond the scope of this book to list the error messages issued by DCE and DFS components. There is a separate book available, the *Error Messages Reference*, which is shipped with the products on AIX and on OS/2 in softcopy form. The *DFS Administration Guide and Reference* (*DFS Client Guide and Reference* on OS/2), also shipped with the product in softcopy form, contains a list of DFS error messages in its Appendix.

Chapter 11. Problem Resolution

Performing problem resolution often requires the knowledge of what the problem is. Some problems are common and happen every now and then. The experienced cell administrator normally performs standard checks to make sure the problem reported is not a common or well-known problem.

In some rare cases, the problem reported to the cell administrator may be a serious one that requires the use of backup data. Otherwise, the component in trouble will not be able to resume service. This requires a proper backup of the various DCE runtime data, just as with any other online data.

Some problems and their resolutions are platform specific. The OS/2 DCE and the OS/2 DSS File and Print Server is an example of a platform that in some cases has some unique, platform-related problems.

This chapter lists and describes a number of known problems and their resolutions that can be included into the normal problem-resolution checklists within an organization.

11.1 Common Problems and Their Resolution

Some problems tend to appear every now and then, especially for administrators new to DCE. Here we describe some common problems and how to deal with them. We also talk a bit about more serious problems later in this chapter.

Generally, people tend to reboot too often when they encounter a situation that is new to them and that they do not know how to deal with it. Most often, stopping and restarting one or more daemons is sufficient, rather than rebooting a machine. For example, if there is a problem with a DFS File Server, restarting the `dfsbind` daemon on that server machine often solves the problem.

In some cases, when you are experiencing DCE problems on a client that cannot be solved by restarting DCE on that client, the easiest solution may be to unconfigure and reconfigure DCE and DFS on that client. This is easy to do and only takes a few minutes of time, as opposed to the time required for a complete problem analysis. Note that this is a way to quickly solve a current problem, not necessarily a way to eliminate one. If you are experiencing the same problem over and over, you should of course investigate and eliminate the cause.

11.1.1 Time Skew Too Great

The hardware clocks in computers are based on oscillators that normally are very accurate, but they can easily drift away several seconds a day. After a machine has been powered off, or just rebooted, the clock can also be affected. Sometimes they can even be reset, which will set the date back to January 1970. Finally, the clock settings on new machines are not known.

Since DCE relies on tickets (and a certain time frame in which they are valid) for security, it is very important that all systems in a DCE cell have synchronized clocks.

DCE services have a limit on how large the time skew can be compared to other services. For example, to configure a security client, the client machine's clock

must be within five minutes of the Security Server's clock. For the DFS FLDB to synchronize between other instances of the DFS FLDB, the time skew must be less than 10 seconds.

Therefore, it is not uncommon to have problems if the time has drifted. When you configure DCE on a machine, make it a habit to always synchronize the clock with the Security Server first (with the AIX `setclock` command or the `time` command on OS/2 for example) before you run `mkdce` (or the corresponding SMIT panels) on AIX and `dcecfgg` on OS/2 for configuring DCE.

You must also make sure the machine keeps the time synchronized during normal operation. See also 5.5, "Distributed Time Services" on page 116, and 7.2.3, "Time Service" on page 144, for ways to do this.

Tip

You may want to run the AIX `setclock` command against a DCE server in /etc/rc.net on client machines. This way, the clock is always synchronized after a reboot and before DCE is started. This helps to prevent time scew problems in environments where client systems may be powered off for long periods of time.

Since the system's time is an important issue for DCE, special attention must be payed to systems without a battery-powered hardware clock, such as the nodes of an IBM Scalable POWERparallel System (SP2). An SP2 node gets its system time every time it is booted from the control workstation. This may cause difficulties if the control workstation's time differs from the DCE cell time. Therefore, a control workstation's clock should be synchronized with the cell time, or the nodes should incorporate additional synchronization with the cell time before DCE is being started.

Info

The AIX time zone environment variable, `TZ`, is only used for viewing purposes. The internal clock and DCE operations always uses UTC (Universal Time Coordinated). Therefore the setting of `TZ` does not affect DCE.

For OS/2, the time zone environment variable (`SET TZ=<timezone>`) is used to set the time used by DCE. OS/2's internal clock does not run UTC time, but a local time. For example, if the time zone of an AIX Security and CDS Server is CST (Central Standard Time), then the time difference from UTC is +6 hours. Setting the time zone variable on OS/2 to UTC+6, the time used by DCE is the local time of the OS/2 machine plus six hours, which corresponds with the UTC time set on the AIX servers. DCE supports automatic daylight saving time, but OS/2 does not. Although DCE on OS/2 could switch automatically by specifying the proper time zone variable, it is not recommended to do so, since OS/2 does not change its clock automatically at the same time. This could result in inconsistent time stamps that can cause DCE functions to fail. It is recommended to change the time zone environment variable `TZ` in config.sys at the time as when the system clock is adjusted for daylight saving time.

The following example shows what can happen when system clocks are not properly synchronized. The system's clock in this example is moved forward by about 10 minutes:

```
[C:] time
Current time is: 9:18:19:97
Enter the new time: 9:28

[C:\] dcelogin cell_admin
Enter Password:
Sorry.
Password validation failure. - Clock skew to great (dce / krb)

[C:\] time
Current time is: 9:28:26:28
Enter the new time: 9:19

[C:\] dcelogin cell_admin
Enter Password:
DCE LOGIN SUCCESSFUL
```

If the time skew is too great, DCE client processes also would not be able to start.

11.1.2 Expired Credentials

A DCE ticket has a limited lifetime. When the ticket expires, you have to reauthenticate in order to access DCE and DFS data. Normally, this is done by the user with the `kinit` command. Note that if you use integrated login on AIX, the standard AIX xlock and CDE-screenlock do not do this for you when you enter your password after the screen has been locked.

All credentials you receive from the security service are stored as files on your machine (in /opt/dcelocal/var/security/creds). These files are not automatically removed on AIX (unless you use the `kdestroy` command, or you restart AIX and start DCE client services from /etc/inittab) and can thus fill up your file system over time. Normally on AIX, you run the `rmxcred` utility as a cron job to remove expired files, say once per day.

A problem that may occur is that a logged-in user is away from his/her machine so long that his/her ticket has expired and the files have been removed by `rmxcred` (during a weekend for example). This makes it impossible for the user to reauthenticate with `kinit`. He/she has to do a `dce_login` again. In some environments, this can mean that the user has to log out and log in again. To make sure this does not happen, you should tell `rmxcred` to remove only those files that have been expired longer than the amount of time you allow the users to be inactive (thus saving some expired credentials files).

For example, to allow users to be able to reauthenticate four days after their tickets have expired, you would run the command:

```
# rmxcred -d 4
```

This removes the credential files only for those tickets that have expired longer than four days ago.

On OS/2, the credential files are automatically removed whenever the OS/2 machine is restarted, but `rmxcred` can be run on OS/2 as well, if you do not

intend to restart the machine on a regular basis. This is probably the case with the OS/2 DSS File and Print Servers, on which the number of credentials can grow. Because the File and Print Servers are usually not rebooted often, rmxcred should be used to remove expired credential files.

11.1.3 Server(s) Not Available at Client Boot Time

If for some reason one or more DCE servers are unavailable, the DCE clients will have problems starting all DCE daemons. The error behavior of the DCE client depends of the missing DCE server component.

11.1.3.1 Security Service Not Available

If after five minutes, the DCE client has not found a Security Server, it will end the DCE/DFS starting sequence. Only the dced daemon will be started. When the security service becomes available, the security client part of dced is automatically activated. You need to start all other DCE services manually.

11.1.3.2 Cell Directory Service Not Available

The CDS root clerk will start and become activated without access to CDS. CDS cannot be accessed until a CDS Server becomes available. Note that the CDS cache on a client survives reboots.

11.1.3.3 DTS Server Not Available

The DTS client starts even though no DTS servers are available. It will start to synchronize time as soon as DTS Servers become available.

11.1.3.4 DFS Server Not Available

The DFS client (dfsbind and dfsd processes) will start even if the FLDB (flserver) and the file exporter (fxd) is not running. Normally, DFS will be accessible when DFS services become available. Note that DFS can be in Token State Recovery (TSR) mode for quite some time if DFS is large. If you are experiencing problems even after the TSR period, you can try the following on the AIX:

```
# dfs.clean dfsbind
# dce.clean core
# rc.dce core
# rc.dfs dfsbind
```

For the OS/2 DFS client, the following command can be used:

```
[C:] cm statservers
```

This will force the DFS client to check the server instead of waiting the default interval for a poll (-pollinterval parameter of the fxd command on the DFS File Server machine).

If the problem continues to exist on OS/2, a dcestop and dcestart can resolve the problem.

11.1.4 Host Name Change

To change the host name of a DCE machine, there are a number of considerations. The simplest way for DCE clients is to make use of the DCE configuration utility. Performing an unconfiguration and then a new configuration after the name change is a clean way, and it automatically takes care of the entries in the CDS namespace.

For DCE core servers, the operation is more complicated and should be avoided. Replica servers can be unconfigured and newly configured after the host name has been changed. A general method for master servers is the (temporary) use of replication. By replicating the information stored on a master server, it can be unconfigured after it has been replicated and after the (temporary) replica has been configured to be the new master. After the unconfiguration, the DCE server can be reconfigured with the new host name, and the role (master/replica) can be adjusted to the preferred roles.

For DCE application servers, the host name change can be performed by a stop of the DCE application server, reconfiguration and restart with the new host name.

11.1.5 IP Address Change

Changing the IP address of a system configured in a DCE cell requires some special provisions based on the system's role. On DCE and DFS client machines, the IP address can be changed while DCE and DFS services are stopped. After a restart of the DCE client services (restarting DFS client services on AIX requires a reboot), DCE/DFS will start with the new address and does not require any additional configuration.

For DCE servers, an IP address change involves many steps on the server and client machines and it would be beyond the scope of this book to describe these administrative tasks in detail. As a simple rule for planning purposes, IP address changes on DCE/DFS servers should be avoided whenever possible.

A number of methods and procedures are described in the redbook *Administering DCE and DFS 2.1 for AIX (and OS/2 Clients*), SG24-4714. This redbook also contains scripts that can be used for the IP address change of DCE machines.

11.1.6 Multihomed Servers

DCE (except DFS) automatically uses all available network interfaces that are configured on a machine.

Therefore, in order to configure an additional interface, you must be sure that the routing is correct and the new interface is available to and from all clients. If this is not the case, time-outs will occur when DCE clients try to access the incorrect interface, and this can cause severe slow-down problems for the whole cell.

In section 6.1.2.2, "Disabling Unused Network Interfaces for DCE" on page 125, you can find more discussions about multihomed AIX machines and how to avoid problems associated with them, and 6.1.3.4, "Disabling Unused Network Interfaces for DCE" on page 129, explains the same for OS/2. As a general rule, all slow interfaces should be disabled for DCE, unless they are really needed.

As an alternative method, specific host routes could be added to the client's configuration such that they do not address the invalid interface directly, but through another, valid interface of that server. For example, assume a server has the following three interfaces defined, of which only the Token-Ring interface is going to be used for clients:

```
Token-Ring, IP Address 9.3.1.68, subnet mask 255.255.255.0
Ethernet, IP Address 192.168.1.1, subnet mask 255.255.255.0
X.25, IP Address 192.1.20.3, subnet mask 255.255.255.0
```

Then, in the client's configuration, the following two routes should be added to force the use of the Token-Ring interface:

```
route add host 192.168.1.1 9.3.1.68 1
route add host 192.1.20.3 9.3.1.68 1
```

This latter method can be used when a server needs to support slow interfaces for DCE. A better solution could be to have a router in between the server and the slow network, which would avoid the situation where the server advertises slow network interfaces.

11.1.7 DCE Configuration Fails because of an ″Already Exists″ Error

There may be errors during configuration of DCE clients and servers that causes configuration to fail and where the failed components are marked PARTIAL. If the error message indicates that an object already exists or that it is unable to write to an object that contains its own host name, there is a chance that a previous installation was not removed properly.

In fact, when unconfiguring DCE from a machine, administrators sometimes choose the *local only unconfiguration* method because it is faster and does not need cell_admin's password.

The disadvantage of choosing *local only unconfiguration* is that the DCE databases still carry information about this client machine which may cause subsequent configuration to fail.

A solution is to unconfigure the failed components, but with the *full unconfiguration* method. This does in many cases clean up any left-overs from previous installation for the same machine.

See also 11.1.9, “Reference to a Non-Existing Clearinghouse in CDS” on page 209 for the special case when the CDS database contains invalid data.

11.1.8 CDS Cache Contains Invalid Data

When a CDS clerk gets a broadcast from a CDS Server, announcing its presence, the clerk inserts the information in its cache. This information is never removed. If the clerk tries to access one of the servers it has in its cache, and it cannot, the clerk will remove the 0k tag and will not use that clearinghouse again, unless it receives a new broadcast from that CDS Server.

As an example, the following is a small excerpt of a dcecp cdscache dump output:

```
...
/.../itsc.austin.ibm.com/ev2_ch
    Clearinghouse UID: 394ab6ee-369c-11d0-a72f-10005aa8cff8
  Clearinghouse address:
        ncacn_ip_tcp:9.3.1.120[]
        ncadg_ip_udp:9.3.1.120[]
  Clearinghouse management name: <none>
  OnLan  AddressUsed  0k
...
/.../itsc.austin.ibm.com/ev4_ch
    Clearinghouse UID: 588e6d1a-3ce2-11d0-bfe8-02608c2f0653
```

```
  Clearinghouse address:
        ncacn_ip_tcp:9.3.1.123[]
        ncadg_ip_udp:9.3.1.123[]
  Clearinghouse management name: <none>
  OnLan  AddressUsed
...
```

Note that only the ev2_ch clearinghouse is tagged 0k.

Erasing the CDS Cache: If you have problems creating a clearinghouse with the same name as an earlier, now deleted clearinghouse, you can try to clean out the CDS cache on the machine you are configuring and maybe on all servers as well. You do this by stopping CDS, removing the CDS cache files, and restarting CDS.

Note

If you have configured a client into a cell with no CDS Servers on the local LAN, or if you have added information about a CDS Server using the `cdscp define cached server` (or the equivalent `dcecp` command), do not delete the cds_cache.wan (AIX) or the cdscache.wan (OS/2) files from the directories mentioned below. Be careful when using wildcards as in the examples below to make sure these *.wan files will not be deleted.

For example, on an AIX client:

```
# dce.clean cds
# rm /var/dce/adm/directory/cds/cds_cache.[0v]*
# rc.dce cds
```

On an OS/2 DCE client, the following commands can be used:

```
[C:] dcestop cds_client
[C:\] erase \opt\dcelocal\var\adm\dir\cds\cdscache.*
[C:\] dcestart cds_client
```

In the last example for OS/2, if you want to preserve the cdscache.wan file, erase all files with cdscache.* as their file names, *except* cdscache.wan.

If you still have problems with an inconsistent CDS namespace, study the following section.

11.1.9 Reference to a Non-Existing Clearinghouse in CDS

In certain circumstances, for example if you remove a secondary CDS Server with the local option only, or if a running CDS Server configuration fails or is interrupted, the Master CDS Server may have a reference to a non-existing replica server. This can result in a situation where it is impossible to reconfigure a CDS replica server, but it is also impossible on the other hand to unconfigure this apparently existing replica server. Other symptoms of such a situation may be when the `cdsli -world` command hangs for several minutes before it completes or when you get error messages from CDS like `Unable to communicate to any CDS server`. The word `any` in this error message is misleading and might lead to the assumption that it was unable to communicate with none of the CDS Servers. Actually, it means that *at least one* of the CDS Servers was inaccessible.

Most likely, when checking CDS directories, for example the root directory /.:, you will notice that some CDS directories still seem to be replicated to a non-existing replica server.

The first step you should try in such a situation is to clean (erase) the CDS caches on the involved machines. The problem could also be a temporary situation due to invalid cache entries or credentials. If you can, you should wait a certain time, say 12 to 24 hours, and check if the problem still exists.

If the problem still exists, follow these steps. Assuming master_ch is the current, valid master clearinghouse, stale_ch is the non-existing (invalid) clearinghouse; and stale is the hostname of this machine:

1. Log in as root and cell_admin on the Master CDS Server.
2. Check and find all CDS directories that have an invalid replica reference to stale_ch. For example, on AIX run the following shell script:

   ```
   #!/bin/ksh
   for dir in /.: $(cdsli -R); do
     cdscp show directory $dir CDS_Replicas
   done
   ```

3. For each directory found in step 2 having a reference to the invalid clearinghouse, remove (exclude) it from the replica list:

   ```
   # cdscp set directory <directory> to new epoch master /.:/master_ch
   readonly <any valid replica clearinghouse> exclude /.:/stale_ch
   ```

4. Skulk each modified directory:

   ```
   # cdscp set directory <directory> to skulk
   ```

5. Remove the reference to the invalid clearinghouse:

   ```
   # cdscp clear clearinghouse /.:/stale_ch
   ```

   (This command may fail if the reference is already removed.)

6. Delete the object associated with the non-existing clearinghouse:

   ```
   # cdsdel -o /.:/hosts/stale/cds-server
   ```

7. Delete the clearinghouse object:

   ```
   # cdscp delete object /.:/stale_ch
   ```

8. Remove the principal for the invalid clearinghouse:

   ```
   # rgy_edit
   rgy_edit=> do p
   Domain changed to: principal
   rgy_edit=> delete hosts/stale/cds-server
   Please confirm delete of name "hosts/stale/cds-server" [y/n]? (n) y
   ```

9. On the machine where the invalid clearinghouse was located, log in as root and remove the keytab entries for the replica CDS Server:

   ```
   # rgy_edit
   rgy_edit=> ktd -p hosts/stale/cds-server -v 1
   rgy_edit=> ktd -p hosts/stale/cds-server -v 2
   ```

After this procedure, your CDS namespace should be cleaned up. You might, however, be required to wait until some credentials on the Master CDS Server expire before you can reconfigure the just removed clearinghouse with the same name. If you cannot wait, stop the Master CDS Server, remove the credential files and restart it.

11.1.10 AIX Filesystem /var/dce I-Node Limit

DCE credential files are stored in /var/dce/security/creds. Every login creates three files in this directory. These files are usually small (some 100 to 1000 bytes), but should be removed periodically (see 6.2, “AIX DCE Clients” on page 130, for an example). If they are not removed, they will use-up i-nodes from the filesystem, either /var, or /var/dce if you have configured such a separate filesystem.

As an example, if /var/dce is 8 MB in size, its i-nodes will be used up after about 440 logins to DCE although the filesystem appears to still have several megabytes of free space.

In such a condition, login to DCE may fail without any further explanation other than the well-known message: `You entered an invalid principal name or password`.

It is therefore important that old credential files are removed, and, on the other side, any file system monitoring should check the i-node usage in addition to the file space usage.

11.1.11 Language Setting During Installation and Configuration

In the release code for AIX DCE 2.1, there were some problems with scripts assuming the language setting to be `En_US` or `C`. Therefore, you could experience problems if you ran with anything else. An example is that the DFS client part did not show up as an option when you tried to configure a client with SMIT.

The problem has been corrected in PTFs and should be gone now. The scripts now set the `LANG` variable to `C` themselves if the `LANG` is not `En_US` or `C`. Nevertheless, this is a something to look out for. To be on the safe side, you can set `LANG` to either `En_US` or `C` before doing DCE configuration.

Example:

```
# export LANG=C
# mkdce -n itsc.austin.ibm.com -s ev1 all_cl
```

11.1.12 DCE Does Not Start

In most cases when DCE (typically the `dced` daemon) does not start, you either have a full file system (usually /var/dce on AIX), low paging space, or the time skew is too great (see 11.1.1, “Time Skew Too Great” on page 203).

If a DCE daemon core dumps on AIX, the core file is put in the daemon’s directory, which is under the /var/dce structure. This can fill the file system. The obvious solution is to free-up space in the file system or to increase the file system size. Another solution is to create symlinks to another file system for all potential AIX core dumps.

There are many other reasons why DCE could not be started. From observation, there is a chance that a second try sometimes solves the problem when some leftovers from a system or software crash prevented DCE from starting up. In most cases, DCE cannot start because the Security Server(s) is (are) not available due to a network-related problem or if the server(s) is (are) down.

If DCE fails to start even when all other components seem to be okay, you might have an inconsistent configuration, either on the server side or on the client side

because a file was unintentionally deleted. In such cases, unconfiguration and reconfiguration of the failing component solves the problem and does not even need much time.

11.1.13 Cannot Log-In in an Integrated Login Environment on AIX

The bridge between DCE and AIX is a daemon called dceunixd. This must be running in order to use integrated login, to see user names instead of IDs when listing files, and so forth.

One common reason for not being able to log in is that the dceunixd daemon stopped running. It is also peculiar about being started after the other DCE services. Sometimes it can help to kill and restart the daemon.

A good way to start dceunixd is from /etc/inittab. If you use respawn, the daemon will automatically be restarted if it dies.

Example line for dceunixd in /etc/inittab:

```
dceunixd:2:respawn:/usr/bin/dceunixd -l 60 -d 1 >/dev/console 2>&1
```

If you are using the Common Desktop Environment (CDE), the problem can also be in one of the CDE start-up scripts. If the CDE encounters a problem executing anything in these, it will log out the user.

11.1.14 Locating CDS on a WAN

If a DCE client is located on a remote location, the CDS advertisements (broadcasts) do not normally propagate to the remote location. To make the DCE client aware of a CDS Server placed anywhere, there is an option to manually (or by an external procedure) add configuration into the DCE client CDS cache. This is performed by the dcecp cdscache create command.

For example, a DCE client is placed remotely, and the CDS Server placed centrally has the address 9.3.1.120 and host name ev2; the following command would be entered at the DCE client machine

```
dcecp> cdscache create ev2 -binding ncacn_ip_tcp:9.3.1.120
```

This enables the DCE client to find any other CDS Server available because information on available clearinghouses is stored in the root directory of a CDS Server, and this information is always replicated to any CDS replica server.

Note that this step is automatically done during DCE client configuration with the CDS Server that was specified during the installation.

11.1.15 DCE Services Fail During System Backup

Sporadically, DCE server processes may get disturbed during periods when systems are backed up. Some modern, high-speed backup products have the nasty peculiarity of raising the process priority of some of their processes. If such prioritized processes stuck or take too much CPE time for some reason, DCE (and other, normal prioritized applications) may be hindered seriously in their normal execution.

Such backup tools should only be used with care on server systems.

11.1.16 TCP/IP Endpoint Mapper Port Restrictions

By default, the endpoint mapper allocates a port number when requested by an application. Having other applications that require fixed but non-well-known port numbers available (for example, the IBM LAN Network Manager products on OS/2), the endpoint mapper honors an environment variable setting to limit the use of ports to certain ranges of port numbers. This is performed by setting the RPC_RESTRICTED_PORTS environment variable.

The following example restricts the port mapper to use only ports ranging from 3000 to 3100 by UDP and from 6100 to 6200 by TCP:

```
SET RPC_RESTRICTED_PORTS=ncadg_ip_udp[3000-3100]:ncacn_ip_tcp[6100-6200]
```

Using the RPC_RESTRICTED_PORTS environment variable can resolve port assignment problems related to the endpoint mapper. The variable must be set before DCE is started, for example, in /etc/environment on AIX and in config.sys on OS/2.

11.1.17 DCE Protocol Sequences

Sometimes, for certain reasons, the use of a specific protocol (UDP or TCP) causes problems and administrators want to limit the use of the protocols. Using DCE RPC, the binding information contains protocol sequence information. If the protocol sequences that should be used must be limited, the limiting options are:

- The RPC_SUPPORTED_PROTSEQS environment variable. An example setting of the environment variable is as follows:

  ```
  SET RPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp
  ```

  This example would restrict DCE to use only TCP.

- On OS/2 only: The protseqs.rpc file in optdcelocaletc, which could contain the line:

  ```
  ncacn_ip_tcp:ncadg_ip_udp
  ```

- The start of the dced daemon with a protocol option parameter, where the protocol sequences to be used can be specified. For example:

  ```
  dced ncacn_ip_tcp:ncadg_ip_udp
  ```

If binding problems occur, validate the protocol sequences and make changes in the selected protocol sequences if required.

If an OS/2 DCE server is started, and the exported protocol sequences include ncacn_nb_stream, AIX machines will not be able to make RPC calls in some cases. In other cases, the calls will succeed because the binding import will select a random available binding that may include ncacn_ip_tcp and ncadg_ip_upd, which are supported by AIX.

11.2 Serious Problems

Identifying the exact problem can many times be very difficult. It is often by experience and not by some easy checks that you find the source of the problem.

The best way to protect yourself against these problems is to have recent backups of the databases. You should have a cron job saving the databases to

disk no matter what other backup systems you use (see also 9.1.3, “Data Backup” on page 171). This way, they are always available and can be easily restored.

All databases are online databases, which means that you have to make sure that they are not written to at the time of the backup. Also make sure that all data in memory is flushed to disk. The procedures for backing up the databases is explained in chapter 6.3 in the *Administering IBM DCE and DFS Version 2.1 for AIX (and OS/2 Clients)* redbook (SG24-4714) or in the *DCE Administration Guides* or AIX and OS/2 and the *DFS Administration Guide and Reference for AIX* (all shipped as softcopy with the products).

11.2.1 Corrupted Registry Database

The registry database is in the /var/dce/security/rgy_data directory on AIX and in optdcelocalvarsecurityrgy_data on OS/2.

A corruption in the actual database is very unlikely. If you can do a DCE login and a `klist`, everything should be ok.

If you back up the database without making sure all data is flushed to the disk and that there are no updates written to it during the backup, you can get corruptions (as with any other online database). Run the `dcecp registry disable` (or the `sec_admin state -maintenance`) command before doing the backup, and you are safe (see also 9.1.3, “Data Backup” on page 171).

If there are network problems, or a server that has been down for some time, you can get security replicas out of sync. The way to resynchronize the replicas with the master is to run the `dcecp registry synchronize` command. You can check the version numbers of the master and the replicas by running the `dcecp registry show` command with each, the `-master` and the `-replica` options.

11.2.2 Corrupted CDS Database

If CDS stops working, anything that uses CDS will stop to function. CDS is very reliable, and normally you will not have any corruptions. If a corruption occurs, it is probably caused by a foreign influence, such as an interrupted CDS configuration, or a system crash.

A way to find out if there is a corruption in CDS can be by running the `cdsli` command. If it hangs or shows the same entry over and over again, you probably have a corruption. Note, if the command hangs, this is more likely to be due to a temporary network problem or to a replica server that is unavailable.

Once you are sure that the CDS database is corrupted and you do not have a recent backup to restore, you can create another clearinghouse, replicate all entries, and make them read/write, thus making the new clearinghouse the new master CDS Server in the cell. Finally, you remove the old, corrupted clearinghouse. Now you have a healthy CDS in your cell again.

If you have a client which, for any reason, got its time set back, it can—in rare situations—send requests to CDS, which causes the transaction log to become corrupted. CDS can get confused when it sees the time stamps and cannot process the log. To solve this you have to correct the time on the client first, and then remove the log file. This is the file ending with `.tlog` in /var/dce/directory/cds on AIX. On OS/2 the filename of the log file in the /opt/dcelocal/var/dir/cds directory can be found by typing the `cdsfiles.map` file.

On the CDS Server, stop cdsd, and then restart it. You have lost all changes that were in the log, but CDS is reestablished.

Since entries in the CDS clerk's cache will never be removed, if you want to be sure that the cache does not contain any invalid entries, you have to stop CDS client, remove the actual CDS cache files, and restart CDS client. See 11.1.8, "CDS Cache Contains Invalid Data" on page 208, for information how to do this.

11.2.3 Corrupted DFS Database

In every fileset in a LFS-aggregate there is a fileset header. If an `fts` command doing a change is interrupted before it is finished, information in the FLDB can differ from the information in the fileset header on the File Server machine.

This is usually noticed as a fileset that suddenly is not available anymore. You can also compare the output from `fts lsfldb` with the output from `fts lsheader` if you suspect a problem.

To resynchronize the FLDB and the fileset headers, first run the `fts synchfldb` command and then the `fts synchserv` command.

11.2.4 DFS Databases Out of Sync

If you have more than one FLDB Server, which you should have, they will keep the information consistent among themselves by having the master FLDB Server send updates to the others.

In some rare cases, for example if you have started to configure an FLDB Server and interrupted the process before it was finished, or if the clock skew between FLDB Servers is larger than 10 seconds, the FLDBs will not be able to synchronize with each other.

To be able to add or change data in the FLDB, the FLDB Servers need a quorum. That means that more than 51 percent of the existing FLDB Servers must be available. If they are not, you will probably notice this as a problem when creating new filesets.

If you encounter this problem, the first thing you should check is the clocks on all FLDB machines. Synchronize them if necessary.

You can use the `udebug` command to analyse the FLDB and check if the different machines have the same version of the database. For example:

```
# udebug -rpcgroup /.:/fs -long
```

11.2.5 Corrupted DFS Data

DFS is very good at taking care of the files and filesets. If a user has problems with DFS, it is more likely that it is some other problem, like the user has lost the authentication, than actual problems in the DFS file system.

If your server, for any reason, stopped abruptly, or you had to turn if off without doing a proper shutdown, DFS usually comes up with no problems. If it does not, a controlled reboot can help DFS to clean and check all the databases and log files. DFS will normally come up after this, but maybe it says that it is salvaging some part of the file system during startup. Due to the internal logging, DFS File Servers will be able to recover from most situations.

If you really are experiencing problems with DFS data, you should go through the same troubleshooting procedure as you would do with any other file system. Check the hardware; look in the AIX error log, and look in all DCE/DFS logs, and so forth.

The DFS salvager (see the *DFS Administration Guide and Reference* for more information about the program) can check the LFS log, analyse an aggregate, and try to salvage data on an aggregate. When DFS starts up and exports the aggregates, the salvager runs in recovery mode. If it finds an error in the LFS log, it will try to unroll the actions that lead up to the problem.

Before you run the salvager manually, you must make sure the aggregate you are checking is not exported. If it is, run the `dfsexport` command with the `-detach` option to make it not exported.

If you suspect a problem in an aggregate, you can run the DFS salvager in verify mode. This corresponds more or less to the AIX `fsck` command. For example:

```
# dfsexport -aggregate lfsdata1 -detach
# salvage -verify -aggregate lfsdata1
```

If it encounters errors, you can try to correct them by either using the `-salvage` option, or no option at all, to the `salvage` command. If you use no options, the salvager will first try to unroll the LFS log before it tries to salvage any data. This is usually the best way to go.

As always with utilities like this, use the DFS salvager with caution; make sure you know what the expected outcome is going to be, and be sure you have backups.

Note, the DFS salvager can only be used for LFS aggregates. Use the normal `fsck` command for problems with non-LFS aggregates.

11.3 OS/2 DCE-Specific Problem Resolutions

There are some DCE-related problems on the OS/2 platform that can be solved by either minor parameter modifications or by the means of advanced tools. The following sections describe some common problems related to OS/2 DCE.

11.3.1 OS/2 Base Problem Resolutions

If an OS/2 system for some reason has a performance degradation or a number of `SYS0008` messages are received, check the available memory. This can be done by checking the disk space available for the OS/2 paging data file (`swapper.dat`) and the size of the paging data file itself in regards to the amount of physical memory. The available disk space could be decreased by printer jobs (spool directory), by other application programs or by data. Removing files or moving the `swapper.dat` file to another drive location (the `swappath` parameter in `config.sys`) often resolves this type of problem.

If the system still has performance problems, the use of the IBM System Performance Monitor/2 (SPM/2) program product can be used to pinpoint the problem (see also 4.3.2, “System Performance Monitor/2” on page 70). A typical reason could be the activation of additional concurrent programs that cause paging activity, thus slowing down the system.

Another use of the SPM/2 is the memory leak detection function. Suspect applications can be analyzed for memory leaks which, over time, can cause performance degradation within a system.

If, a system for any reason will not activate programs, the number of OS/2 threads may be to small. To resolve this situation, the OS/2 pstat can be used. Using pstat with the /c parameter will list the active threads. Counting the number of threads and comparing the value of THREADS= in config.sys will normally explain the problem. By increasing the value of the THREADS parameter, this type of problems normally disappear.

The normal installation of an OS/2 Warp system will set a FAT file system cache size on HPFS-formatted systems only. The default parameter is in some cases set to:

```
DISKCACHE=D,LW
```

The D parameter indicates that 10 percent of memory is allocated for FAT disk cache on systems having more than 8 MB of memory. The upper limit of the automatic cache size allocation is 4 MB. On systems using no FAT-formatted drives (for example HPFS only), the FAT cache (diskcache parameter) can be removed, thereby releasing unused memory to other applications.

11.3.2 Resolving NetBIOS Resource Problems

If communication cannot be established using the NetBIOS interface, the LAPSDUMP can be used. The LAPSDUMP utility is part of the applets available in MPTS provided with the OS/2 DSS File and Print Server CD-ROM.

Using LAPSDUMP, detailed information about the LAN Adapter and Protocol Support (LAPS) is available. The key information is the allocation of the NetBIOS resources divided into processes. This information can be used to see who is using what resources. Any application program using the NetBIOS interface does a virtual NetBIOS reset. This operation determines the resources allocated to the application. If the requested resources are not available, the application will either try with a reduced request values or fail. To resolve NetBIOS resource-allocation problems, a calculation of the needed resources must be performed.

If the resources available do not fit into the requirements, another calculation must be performed.

For DCE using NetBIOS, the parameters available are specified in the NetBIOS Sockets section of the MPTS configuration program. The NetBIOS sockets driver does not perform a NetBIOS reset before DCE is started. If resources are not available, DCE will fail if configured to use NetBIOS sockets (ncacn_nb_stream). To resolve this problem, the NetBIOS resources used by all applications must be calculated. The remaining resources should then be available to NetBIOS sockets.

11.3.3 The DCEOS2PQ Utility

The DCEOS2PQ command can be used to check the state of a DCE daemon. Combined with pstat, the status of the OS/2 DCE daemons can be determined. For example, to validate the state of the DCE daemon dced, the following command

```
[C:] dceos2pq dced
Checking State of dced
DCEOS2_PROCESS_INITED
```

displays that the dced process is initiated and running.

11.4 OS/2 DSS File and Printer Server Problem Resolution

The DSS for OS/2 product provides the DCE core functions as available also on AIX. A number of unique features, such as the integration into the OS/2 LAN Server, do not exist on AIX. The following sections describe some common problems related to DCE and their resolutions.

11.4.1 Using DSS Tuning Assistant

Using the apply function of the DSS Tuning Assistant can cause unintended problems. To resolve problems, a backup copy of the configuration files can be used to restore the state before the DSS Tuning Assistant was started.

The DSS Tuning Assistant makes use of a number of assumptions that are used to calculate configuration parameters. The assumptions can be modified to represent the actual environment.

The updated configuration files should be evaluated manually, in respect to the requirements of the DCE Security Server and CDS Server functions. The memory to reserve for the DCE server functions can be extensive depending on the actual number of registry entries and the size of the CDS namespace.

11.4.2 DSS Password Management Servers

The OS/2 DSS File and Print Server can make use of three different password-management servers. They are:

- PWSYNC — which is the DCE password synchronization server. It notifies the password strength servers and the foreign registries that a password is to be validated and synchronized with the DCE Security Server registry.
- LSPWSD — which is the password strength server. It ensures that a password complies with the password-management policies as implemented in the LSPWSD.
- LSPWSYN — which is the password synchronization server. It performs operation on the foreign registry (NET.ACC), which maintains the LAN Server-encrypted format of passwords.

If any of the password-management servers have a problem, the problem can be further analyzed by validating each of the password-management servers. The procedure is to ensure:

1. That DCE account definitions comply to the requirements of the password-management servers. The password-management servers make use of a number of extended registry attributes (ERA).
2. That the DCE account definitions for the password management servers are as required.
3. That the password-management servers are started.

The ERA used for an account controls the way the account is treated by the password-management servers. For example

```
dcecp> principal show USRA86 -xattr
{ls_home_dir {}}
{ls_comment {User in location ITSO DSS}}
{ls_flags 1}
{ls_script_path {}}
{ls_usr_comment {}}
{ls_parms {}}
{ls_workstations {}}
{ls_max_storage -1}
{ls_logon_hours {ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff \
  ff}}
{ls_logon_server {}}
{ls_country_code 1}
{ls_code_page 0}
{default_realm //ITSO10@/.../itso10.austin.ibm.com}
{mcl_policyregions DSS_policy_region}
{ls_encr_passwd {4f e2 64 11 53 9a 4c 5a 6d 14 00 85 ac 39 7e b6}}
{ls_passwd_hist {79 c4 38 00 3c c9 fa 41 75 af ed 44 8f 8d d4 1d 7d f9 10 56 \
  3e 2f 82 58 ca a8 32 53 3a db cc da 60 15 e2 56 1c 62 7d 50 53 \
  cc 09 d8 a2 60 5e 93 53 cc 09 d8 a2 60 5e 93 53 cc 09 d8 a2 60 \
  5e 93 53 cc 09 d8 a2 60 5e 93 53 cc 09 d8 a2 60 5e 93 e6 9f e2 \
  2e 82 fd 9b ed 7a 8e 38 56 b7 0e d8 a9 af 2c 01 47 e9 da 6a 34 \
  0b 12 b5 09 4e 53 4b b0 53 cc 09 d8 a2 60 5e 93 53 cc 09 d8 a2 \
  60 5e 93 43 51 9a b9 13 de 4e 26 53 cc 09 d8 a2 60 5e 93}}
{password_strength lspwsd}
{pwd_mgmt_binding {{dce pwsync cdmf secret name}
  /.../itso10.austin.ibm.com/subsys/dce/pwd_mgmt/pwsync}}
{pwd_val_type 1}
{plaintext_password present}
{foreign_registry lspwsync}
{logon_asn {6b 00 00 00 01 00 01 00 00 00 67 00 00 00 00 00 00 00 66 00 \
  00 00 62 00 00 00 57 00 00 00 00 50 00 00 00 27 00 00 00 41 4c \
  49 41 31 31 40 49 54 53 4f 31 30 40 2f 2e 2e 2e 2f 69 74 73 6f \
  31 30 2e 61 75 73 74 69 6e 2e 69 62 6d 2e 63 6f 6d 00 41 4c 49 \
  41 31 31 00 4c 41 4e 20 53 65 72 76 65 72 00 49 42 4d 00 00 57 \
  00 ba 14}}
```

Without the extended registry attributes, the principal cannot perform an OS/2 LAN Server logon and cannot use DCE services.

The DSS password-management servers have entries within the CDS namespace. If these entries are not present, unconfiguration and reconfiguration of the password-management servers can resolve this problem. Similarly, if the keytable entries are out of synchronization with the registry entries, the reconfiguration of the password management-servers will solve the problem.

The DSS password-management servers can be restarted by the following commands:

```
DCESTART PW_STRENGTH_SVR
DCESTART PW_SYNC_SVR
NET START LSPWSYNC
```

11.4.3 The PWUCFG Utility

If the PWSYNC (DCE password synchronization server), LSPWSYNC (foreign registry) and LSPWSD (password strength server) need to be unconfigured because the OS/2 machine on which these DSS password-management servers run is not recoverable for some reason, the PWUCFG utility can be used.

The command does have an location specifier that specifies the DCE host name of the machine on which PWSYNC, LSPWSYNC and LSPWSD are to be configured. For example:

```
PWUCFG /SERVER:warp4Osrv
```

11.4.4 The DSSFIXUP Utility

The OS/2 DSS File and Print Server makes use of a number of data structures and attributes in the DCE Security Server and CDS Server components. This data can by corrupted, deleted or become out of synchronization with other related attributes.

The DSSFIXUP utility can correct inconsistencies in the data structures and attributes that are used by the OS/2 DSS File and Print Servers. This data structure and attributes can be corrected by the normal DCE administration commands, but it is a very complex task. To ease the DSS consistency check, the DSSFIXUP is used.

11.4.5 The DIRSYNC Utility

The DIRSYNC server function of the OS/2 DSS File and Print Server synchronizes alias definitions stored in the DCE CDS namespace into the DCDB database (Domain Controller Database). The DIRSYNC utility is used as long as there are LAN Requesters (LAN clients) making use of the alias information stored in the DCDB database.

If, for some reason, alias additions or modifications are not available for LAN Requesters relying on the information in the DCDB database, the following may solve the problem:

```
NET STOP DIRSYNC
NET START DIRSYNC /FULLSYNC
```

The failing alias can then be retried on the LAN Requester machine.

11.4.6 The SERVERPW Utility

The OS/2 DSS File and Print Server makes use of a keytable entry that stores the password (key) the server uses. If for some reason the password is out of synchronization with the server password stored within the DCE registry, the server will not be able to start. To resolve this problem, the SERVERPW utility can be used.

To make use of the SERVERPW utility, an account that is member of the acct-admin group is required. The password the server makes use of is not displayed in this process, but the SERVERPW utility will add a server password entry into the keytable file and modify the server account's key in the DCE Security Server registry.

11.4.7 DSS Uppercase User ID

Because the OS/2 DSS File and Print Server makes use of accounts and passwords in all uppercase letters, problems related to DCE login or LAN Requester (LAN Client) can be resolved by carefully specifying the account and password requirements.

A LAN Server-defined user ID always requires a user ID and password in uppercase. For any other DCE account, the case can be mixed. By using the foreign registry capabilities of the OS/2 DSS File and Print Server, accounts that require access to both DCE applications and DSS functions must be defined in uppercase.

Any password change must match the requirements of the account; otherwise the DCE login or the LAN Requester (LAN Client) logon can fail.

Password changes performed by the cell administrator normally results in lowercase account and password definitions. Therefore, the account cannot make use of services provided by the OS/2 DSS File and Print Server.

Part 3. Appendices

Appendix A. Using the DCE Debug and Messaging Facilities

The DCE serviceability interface (SVC) provides a standard method of displaying messages and performing logging. The DCE serviceability interface works on message catalogs that either are the standard DCE message catalogs or application-specific message catalogs. The SVC message catalogs are created by the DCE `sams` utility.

Using the serviceability interface, the following can be achieved for application messages:

- SVC messages need not to be hard-coded into DCE applications.
- Internationalization issues can be handled by generic interfaces.
- SVC message routing can be handled.

The DCE serviceability interface services are also discussed in 5.7, “Planning for SVC, EMS, and Auditing” on page 118.

The serviceability messages can be routed to various output destinations, such as the console, a printer or a file. The routing destination for the serviceability messages can be specified by:

- The contents of the DCE routing file
- The contents of a routing environment variable
- The DCE application use of `dce_svc_routing()` routine
- The administrators use of `dcecp log modify` commands

Using the DCE serviceability routing to a file, the `outform` specifier can be used to control the number of entries and files for a routing entry.

This is performed by setting two integer numbers. The parameters are:

- `gens`, which is the number of generations of a file (number of files) that should be kept
- `count`, which specifies the count of entries to be written into each file

When the number of message entries in a file reaches the maximum number specified, the message file is closed. The file generation number is incremented, and the next file is opened. When the maximum number of file generations is reached, the generation number is reset to 1. A new file is created, and if a file with this name already exists, it is overwritten.

For example, an OS/2 DCE routing file in X:optdcelocalvarsvcrouting can contain the following entry:

```
ERROR:TEXTFILE.10.200:/opt/dcelocal/var/svc/error.log
```

The error.log file can be in 10 generations and have 200 DCE serviceability messages in each.

The disk space consumed using DCE serviceability to file can be controlled by using the number of generations and message entries. If this is not used, the message files can grow uncontrolled.

The serviceability interface provides, beside messaging, a debug message interface. This SVC debug message interface acts very similarly to the SVC message interface with respect to the message routing.

To activate the SVC debug and trace message routing in a DCE application program, a number of places can be used to set the debug message routing control parameters. These controlling debug message routing parameters can be set in either of the following locations:

- By calling the dce_svc_debug_routine() in a DCE application program. This routine is used internally by the DCE application, or can be set by the remote management interface of a server.
- By the contents of environment variables that are built by the following structure, SVC_*COMPONENT*_DBG. The *COMPONENT* part of the environment variable is the three-character code of the component, converted to uppercase (see the following text for a list of these codes). For example, SVC_RPC_DBG will set the debugging messages for the RPC component. The value of the environment variable is called the debug routing-level specifier.
- By the contents of the routing file. The routing file is normally located in /opt/dcelocal/var/svc/routing.

The debug environment variables and the normal SVC routing file make use of a debug routing-level specifier. The debug routing level specifier is used to specify the component that is supposed to generate debug messages. To pinpoint more details and avoid many debug messages, a subcomponent specifier and a debug level can be added. By using this, the amount of debug messages generated can be reduced significantly.

The general format for the debug message routing-level specifier is as follows:

```
component:sub_comp.level,sub_comp.level,...
```

where

- component is a three-character specification for the component code of the program. The following subcomponent code are available for the standard DCE services:
  - dhd (DCED)
  - sec (Security)
  - gss (GSSAPI)
  - dcp (DCECP)
  - dts (DTS)
  - rpc (RPC)
  - xds (XDS)
  - cds (CDS)
  - gds (GDS)
- sub_comp is a subcomponent name. If the subcomponent name is unknown or any subcomponent should be included, the * can be used.
- level is a debug level from 1 to 9. Using a higher-level value, the amount of debug information is normally increased.

By adding multiple subcomponents and debug levels into the same component debug specifier, different levels of debug messaging can be obtained.

The routing information is added to the debug message routing-level specifier by separating the routing destination specifiers by semicolons (;). For example, adding the following statement into the default SVC routing file

```
rpc:*.9:STDERR:-;FILE:/opt/dcelocal/var/svc/rpc.log
```

will activate the debug messages for the rpc component. The messages will be routed to the standard error device (console) and to a text file called rpc.log.

The similarly setting for the SVC environment variable for AIX is:

```
export SVC_RPC_DBG=rpc:*.9:STDERR:-;FILE:/opt/dcelocal/var/svc/rpc.log
```

For OS/2 Warp, the same setting would be:

```
SET SVC_RPC_DBG=rpc:*.9:STDERR:-;FILE:/opt/dcelocal/var/svc/rpc.log
```

The normal AIX directory separator is also used by OS/2. The default OS/2 directory separator () can be used on OS/2 systems.

A.1 Debug Messaging Option Example

To obtain the DCE SVC message log option settings remotely, the following dcecp command can be used:

```
dcecp> log show /.:/hosts/warp40srv
{FATAL {{STDERR -} {FILE /opt/dcelocal/var/svc/fatal.log}}}
{ERROR {{STDERR -} {FILE /opt/dcelocal/var/svc/error.log}}}
{WARNING {{STDERR -} {FILE /opt/dcelocal/var/svc/warning.log}}}
{NOTICE DISCARD}
{NOTICE_VERBOSE DISCARD}
```

To obtain the debug message log options remotely, the -debug argument of the dcecp log show command can be used. In the following example, the SVC routing file on warp40srv (OS/2) specifies:

```
rpc:*.9:FILE:/opt/dcelocal/var/svc/rpcdebug.log
```

The dcecp log show command does not retrieve the SVC log file information, only the debug level and the debug component.

```
dcecp> log show /.:/hosts/warp40srv -debug
{rpc {* 9}}
```

Depending on the routing specifiers, the debug information is sent to the console or a file. Every debug message is appended to the output device as specified in the output specifier.

The contents of one debug message entry is divided into a number of sections. For example, the following debug message entry (on an OS/2 machine), which was logged into a file, is divided into a number of information sections.

```
1996-12-09-11:49:21:000-06:00I----- PID#116 DEBUG9 rpc mem ? 0 0x152b2acc
msgID=0x00000000 "[time: 000026] [thread: 152b2acc.00000001] (rpc__mem_free)
type 34 @ 1f6050"
```

The sections each provide some information, which is as follows for OS/2 DCE:

- The sections are for a TEXTFILE type log file.
- The time stamp (1996-12-09-11:49:21:000-06:00I-----).

- The Process Id (PID) of the process owning the thread (PID#116).
- The debug level (DEBUG9). This is the name registered by the application using the dce_svc_set_progname() function.
- The component and subcomponent (rpc mem).
- The source code file name (?).
- The line number within the source code (0).
- The address of the thread (0x152b2acc).
- The associated message Identifier (msgId=0x00000000).
- The debug message string related to the debug entry. The debug message string is enclosed within the double quotes.

The same kind of information is available on AIX machines, but the message contents may differ between the two implementations.

Using the debug messages, it is possible to suppress the prolog (non-message text) part of all serviceability text messages. This is performed by setting (export in AIX) the SVC_BRIEF environment variable. For example, on OS/2 do the following:

```
SET SVC_BRIEF=1
```

which will disable the SVC prolog messages.

The following is an example of the brief form of a SVC debug message string:

```
"[time: 000026] [thread: 152b2acc.00000001] (rpc__mem_free) type 34 @ 1f6050"
```

The debug message string contains information provided by the DCE application programmer.

A.2 DCE Daemon Command Line Debug Parameters

Some of the DCE core services have debug options available as a command line parameter. The format and information availbale for the DCE daemons having debug command line options available differs. The following is a list of the DCE core service daemons:

- auditd -d — which specifies debugging. The -d parameter value is used to specify which component to perform debugging on.
- cdsadv — does not have a debug argument.
- cdsclerk -D — which specifies debugging. This parameter has no additional parameter value to be specified.
- cdsd — does not have a debug argument.
- dced — does not have a debug argument.
- emsd — does not have a debug argument.
- dtsd -d — which will have the dtsd daemon to run in foreground. No additional parameter values can be specified.
- gdad -D — (AIX only). Not additional parameters are available for the debug argument.
- secd — does not have a debug argument.

The use of the command line debug parameters is limited compared to the DCE SVC debug message facilities available.

A.3 Performance Impact of the Serviceability Debugging

Using serviceability debugging routines within an application, the following will happen to each debugging routine when called in the execution flow of the application program

- The routine is called and output is generated. This happens because the debug level associated with the debug message was enabled.
- The routine is called, but no output is generated. This happens because the debug level associated with the debug message was disabled.
- The routine is not present in the application code. This happens because the DCE_DEBUG switch was disabled during compilation.

In the first two cases, the debug message routine is called in any case, thus causing some performance degradation in the application.

The performance cost of a serviceability logging operation normally amounts to one mutex lock operation and one file operation that requires lock access for a output file.

To avoid this situation, the application programmer has an C-language-level macro facility available. The purpose of this macro (DCE_SVC_DEBUG_ATLEAST) is to check the current debug level and make a call to the debug message routine whenever the debug routine message is required. This causes less impact of the overall performance of the application than calling debug messaging routines without generating output.

Appendix B. Code Listings of the fst Test Program

For some of the figures provided in Chapter 8, “Application Development With Performance in Mind” on page 155, a DCE RPC test program was created. The fst program makes use of an array because the size of date should be varied. The DCE RPC server procedure returns the data immediately.

B.1 Environment Preparation

To make the test program work, the following steps are required related to CDS.

Create a CDS directory for the fst test programs:

```
cdscp> create directory /.:/sample
```

Create an RPC entry:

```
rpccp> add entry /.:/sample/fst_srv
```

The fst_srv must be defined as an account into the cell. This is done with the following steps:

```
dcecp> principal create fst_srv
dcecp> group add none -member fst_srv
dcecp> organization add none -member fst_srv
dcecp> account create fst_srv -group none -org none -mypwd -dce- -password \
fst_pwd
```

Create a default keytab file in the directory of the fst_srv program. This is performed by:

```
# rgy_edit
Current site is ...
rgy_edit==> ktadd -p fst_srv
Enter Password:*
Reenter password to verify:*
rgy_edit==> quit
```

The test was performed by using the cell administrator context. To start the test program, just start the fst_srv without any parameters:

```
# fst_srv
```

The fst_clt needs a few arguments. The syntax of the fst_clt test program is:

```
fst_clt iterations buffersize protection_level protocol
```

where the parameters have the following usage:

- iterations — the number of times the RPC procedure is called.
- buffersize — the size of the actual data transferred. The size can be from 1 byte to 100000 bytes.
- protection_level — the packet protection level. The following numerical values are assigned:
  - 0 = rpc_c_protect_level_default
  - 1 = rpc_c_protect_level_none
  - 2 = rpc_c_protect_level_connect

  - 3 = rpc_c_protect_level_call
  - 4 = rpc_c_protect_level_pkt
  - 5 = rpc_c_protect_level_pkt_integ
  - 6 = rpc_c_protect_level_pkt_privacy
  - 7 = rpc_c_protect_level_cdmf_priv
- protocol — can be the protocols available in the protocol sections of the bindings. Examples of protocols are:
  - ncacn_ip_tcp
  - ncadg_ip_udp

The following is an example of a start:

```
# fst_clt 100 1000 7 ncadg_ip_udp
```

B.2 fst.idl

The IDL source code used for the test programs. The remote procedure call makes use of an array.

```
/* IDL for fst interface */
[
uuid(8d33b761-3f98-11d0-917e-10005a250d1b),
pointer_default(ref),
version(1.0)
]
interface fst
{
/* A context handle used to access remote storage */
 typedef [context_handle] void* store_handle_t;

 /* The xfer buffer for do_fst */
 typedef byte fst_data_t[*];

 void do_fst(
                [in] handle_t binding,
                [in, max_is(*themax)] fst_data_t fstdata,
                [in]   unsigned long       *themax

        );
}
```

B.3 fst.acf

The fst.acf file is used for the interface definition.

```
[explicit_handle] interface fst
{
        do_fst([comm_status, fault_status] status);
}
```

B.4 fst_clt.c

This is the fst_clt source code. To make use of the packet-level protection, the rpc_binding_set_auth_info() function must be used.

```
/*****************************************************/
/* FST client code for test of data transfer         */
/*****************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <time.h>
#include <string.h>
#include <pthread.h>
```

```
#include <dce/dce_error.h>
#include <dce/rpc.h>
#include <dce/sec_login.h>
#include <dce/keymgmt.h>
#include <dce/uuid.h>
#include "fst.h"

#define EXPORT_ENTRY "/.:/sample/fst_srv"
#define MAX_FST_BUF 100000
void print_error(char *, error_status_t);

int main(int argc, char *argv[])
{
 handle_t binding;                /* The binding handle */
 unsigned32 count;
 error_status_t status;
 unsigned32 themax=10;
 unsigned_char_t *string_binding; /* String binding conversions.*/
 rpc_ns_handle_t import_context;  /* Context for importing bindings*/
 unsigned_char_t * charBuffer;
 unsigned_char_t  * charSrvName;
 unsigned32 givemeprotocol=1;
 unsigned32 iterations=1;
 unsigned32 buffersize=512;
 unsigned32 protection_level=rpc_c_protect_level_default;
 time_t ltime;
 char line[256];
 char *result;
 sec_login_handle_t login_context;

 /* Check the command line */
 if (argc < 3) {
  /* fst_clt 1000 1024 0 ncacn_ip_tcp */
  fprintf(stderr, "Use: fst_clt iterations buffersize<protection_level><protocol>\n");
  return 1;
 }
 iterations = atoi(argv[1]);
 buffersize = atoi(argv[2]);
 if (argc > 3) {
  protection_level = atoi(argv[3]);
  if (protection_level > rpc_c_protect_level_cdmf_priv)
   protection_level = rpc_c_protect_level_cdmf_priv;
 }

 if(buffersize > MAX_FST_BUF + 1) {
  buffersize=MAX_FST_BUF;
 }

 if(buffersize == 0) {
  buffersize=1; /* Not that clean */
 }

 themax = buffersize;
 charBuffer = malloc(buffersize);
 if (charBuffer == 0) {
  fprintf(stderr, "No memory available\n");
  return 99;
 }
 memset(charBuffer,'F',buffersize);
 /* Start importing servers. */
 fprintf(stdout, "fst_clt: Calling rpc_ns_binding_import_begin()...\n");
 rpc_ns_binding_import_begin(
                rpc_c_ns_syntax_default,
                EXPORT_ENTRY,
                fst_v1_0_c_ifspec,
                NULL,
                &import_context,
                &status);
 if (status != rpc_s_ok) {
  print_error("rpc_ns_binding_import_begin()", status);
  return 1;
 }

 if (argc > 4) {
  while( givemeprotocol==1)  {
```

```
 /* Import a server binding */
 fprintf(stdout, "fst_clt: Calling rpc_ns_binding_import_next()...\n");
 rpc_ns_binding_import_next(import_context, &binding, &status);
 if (status != rpc_s_ok) {
  print_error("rpc_ns_binding_import_next()", status);
  return 1;
 }

 /* Convert the binding to a readable string */
 fprintf(stdout, "fst_clt: Calling rpc_binding_to_string_binding()...\n");
 rpc_binding_to_string_binding(binding, &string_binding, &status);
 if (status != rpc_s_ok) {
  fprintf(stdout, "fst_clt: Requested protocol not found\n");
  print_error("rpc_binding_to_string_binding()", status);
  return 1;
 }

 /* Print it for validation of the requested test scenario */
 fprintf(stdout, "fst_clt: Imported resolved binding == %s\n", string_binding);
 if (strstr(string_binding, argv[4]) != NULL)
  givemeprotocol = 0;

 /* Free the string binding space */
 fprintf(stdout, "fst_clt: Calling rpc_string_free()...\n");
 rpc_string_free(&string_binding, &status);
 if (status != rpc_s_ok) {
  print_error("rpc_string_free()", status);
  return 1;
 }
} /* end while */

/* Free the import context */
fprintf(stdout, "fst_clt: Calling rpc_ns_binding_import_done()...\n");
rpc_ns_binding_import_done(&import_context, &status);
if (status != rpc_s_ok) {
 print_error("rpc_ns_binding_import_done()", status);
 return 1;
}
} else {
/* Import the first server binding */
fprintf(stdout, "fst_clt: Calling rpc_ns_binding_import_next()...\n");
rpc_ns_binding_import_next(import_context, &binding, &status);
if (status != rpc_s_ok) {
 print_error("rpc_ns_binding_import_next()", status);
 return 1;
}

/* Convert the binding to a readable string */
fprintf(stdout, "fst_clt: Calling rpc_binding_to_string_binding()...\n");
rpc_binding_to_string_binding(binding, &string_binding, &status);
if (status != rpc_s_ok)  {
 print_error("rpc_binding_to_string_binding()", status);
 return 1;
}

/* Print it for validation of the requested test scenario */
fprintf(stdout, "fst_clt: Imported resolved binding == %s\n", string_binding);
/* Free the string binding space */
fprintf(stdout, "fst_clt: Calling rpc_string_free()...\n");
rpc_string_free(&string_binding, &status);
if (status != rpc_s_ok) {
 print_error("rpc_string_free()", status);
 return 1;
}

/* Free the import context */
fprintf(stdout, "fst_clt: Calling rpc_ns_binding_import_done()...\n");
rpc_ns_binding_import_done(&import_context, &status);
if (status != rpc_s_ok) {
 print_error("rpc_ns_binding_import_done()", status);
 return 1;
}
} /* end if (argc> 3) *?

/* Print the protection level value in use */
```

```
fprintf(stdout, "fst_clt: Protection level in use == %x\n", protection_level);
/* Set the protection level */
rpc_binding_set_auth_info(binding,
                          "fst_srv",
                          protection_level,
                          /* rpc_c_authn_none, */
                          rpc_c_authn_default,
                          NULL,
                          /* rpc_c_authz_dce, */
                          rpc_c_authz_name,
                          &status);
if (status != rpc_s_ok) {
 print_error("rpc_binding_set_auth_info()", status);
 return 1;
}

/* Variants, for diverse timing are commented out or is as required */
/* fprintf(stdout, "fst_clt: outside the loop\n");
   result = gets(line); */
/* time(&ltime);
   printf("The starting time is %s\n", ctime(&ltime));
*/
/* test loop */
for (count=1; count <= iterations; count++) {
 /* fprintf(stdout, "fst_clt: before do_fst count=%d\n", count);
    result = gets(line);
 */
 do_fst(binding,
        charBuffer,
        &themax,
        &status );
 if (status != rpc_s_ok) {
  print_error("do_fst()", status);
  return 1;
 }
 /* fprintf(stdout, "fst_clt: after do_fst count=%d\n", count);
    result = gets(line);
 */
}
/* time(&ltime);
   printf("The ending time is %s\n", ctime(&ltime));
*/
/* fprintf(stdout, "fst_clt: after loop\n");
   result = gets(line); */
return(0);
}

/* print_error-- Client version.
 * Prints text associated with bad status code.
*/
void print_error(char *caller, /* Who got the error   */
                 error_status_t status) /* The status */
{
 dce_error_string_t error_string;
 int print_status;
 dce_error_inq_text(status, error_string, &print_status);
 fprintf(stderr," Client: %s: %s\n", caller, error_string);
}
```

B.5 fst_srv.c

This is the fst_srv source code. The DCE remote procedure returns the data immediately as the only action.

```
/*****************************/
/* FST server type           */
/*****************************/
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <time.h>
#include <pthread.h>
#include <dce/dce_error.h>
```

```
#include <dce/rpc.h>
#include <dce/sec_login.h>
#include <dce/keymgmt.h>
#include <dce/uuid.h>
#include "fst.h"

#define EXPORT_ENTRY "/.:/sample/fst_srv"
#define NCACN_IP_TCP "ncacn_ip_tcp"
#define NCADG_IP_UDP "ncadg_ip_udp"
#define NCACN_NB_STREAM "ncanc_nb_stream"
void print_server_error(char *, error_status_t);

int main(int argc, char *argv[])
{
 unsigned32 status;
 unsigned32 nca_select=0; /* Protocol to use */
 char line[256];
 char *result;
 rpc_binding_vector_t *binding_vector;

 fprintf(stdout,"fst server: Starting up . . .\n");
 /* Set Srv principal name and authentication services */
 fprintf(stdout, "fst_srv: Calling rpc_server_register_auth_info()...\n");
 rpc_server_register_auth_info( /*"/.:/sample/fst_srv", */
                                "fst_srv",
                                /* rpc_c_authn_none, */
                                rpc_c_authn_default,
                                NULL,
                                NULL,
                                &status);
 if (status != error_status_ok) {
  print_server_error("rpc_server_register_auth_info()", status);
  return(1);
 }

 /* Register interface with RPC runtime */
 fprintf(stdout, "fst_srv: Calling rpc_server_register_if()...\n");
 rpc_server_register_if(
                fst_v1_0_s_ifspec,
                NULL,
                NULL,
                &status);
 if (status != error_status_ok) {
  print_server_error("rpc_server_register_if()", status);
  return(1);
 }

 /* Use all protocol sequences that are available */
 fprintf(stdout, "fst_srv: Calling rpc_server_use_all_protseqs()...\n");
 rpc_server_use_all_protseqs(
               rpc_c_protseq_max_reqs_default,
               &status);
 if (status != error_status_ok) {
  print_server_error("rpc_server_use_all_protseqs()", status);
  return(1);
 }

 /* Get the binding handles generated by the runtime */
 fprintf(stdout, "fst_srv: Calling rpc_server_inq_bindings()...\n");
 rpc_server_inq_bindings(&binding_vector, &status);
 if (status != error_status_ok) {
  print_server_error("rpc_server_inq_bindings()", status);
  return(1);
 }

 /* Register assigned endpoints with endpoint mapper */
 fprintf(stdout, "fst_srv: Calling rpc_ep_register()...\n");
 rpc_ep_register(
            fst_v1_0_s_ifspec,
            binding_vector,
            NULL,
            (unsigned_char_p_t) "fst server version 1.0",
            &status);
 if (status != error_status_ok) {
  print_server_error("rpc_ep_register()", status);
```

```
 return(1);
 }

 /* Export the bindings */
 fprintf(stdout, "fst_srv: Calling rpc_ns_binding_export()...\n");
 rpc_ns_binding_export(rpc_c_ns_syntax_default,
               (unsigned_char_t *)EXPORT_ENTRY,
               fst_v1_0_s_ifspec,
               binding_vector,
               NULL,
               &status);
 if (status != error_status_ok) {
  print_server_error("rpc_ns_binding_export()", status);
  return(1);
 }

 /* Start listening for calls */
 fprintf(stdout, "fst_srv: Listening for calls. . .\n");
 rpc_server_listen(rpc_c_listen_max_calls_default, &status);
 if (status != error_status_ok) {
  print_server_error("rpc_server_listen()", status);
  return(1);
 }
 return(0);
}

/* print_server_error-- Prints text for none good status code */
void print_server_error(char *caller, /* Who got the error*/
                        error_status_t status)  /* Status */
{
 dce_error_string_t error_string;
 int print_status;
 dce_error_inq_text(status, error_string, &print_status);
 fprintf(stderr," Server: %s: %s\n", caller, error_string);
}

/* do_fst RPC, which does nothing */
void DCEAPI do_fst(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t binding,
    /* [in] */ fst_data_t fstdata,
    /* [in] */ idl_ulong_int *themax,
    /* [out] */ error_status_t *status
#endif
)
{
 return;
}
```

Appendix C. Future Performance Improvements

This book describes some DCE and DFS conceptual performance issues in a general way, but also contains detailed performance-related data specific to current products. The exact level of software used for any tests described in this book can be found in appendix E.1, “Test Environment” on page 245.

This Appendix briefly describes how further performance improvements may be incorporated into products.

C.1 Products Updates

IBM constantly improves products and makes these improvements available to customers in various ways. Although any new release or version of a product has some minor or major performance improvements compared to previous releases and versions, such improvements can also be incorporated without special notice in product updates called Program Temporary Fixes (PTFs) on AIX, or Service Packages on OS/2 Warp.

Product updates are released several times a year. Their primary target, as the name implies, is to fix problems that were encountered either by customers or by IBM itself. Product updates may also contain functional improvements. For example, the audit function was introduced in DCE for AIX with such a product update.

Product updates may also contain performance improvements in certain areas. In performance-critical environments, it might be worth it to regularly check new updates for such improvements, especially when exact measurements have been done for planning, as the figures shown in Appendix E, “Lab Testing Results” on page 245 provide.

C.2 Product Releases and Versions

New releases or versions of products incorporate new functions and/or major reworks on current functions. Most often, performance is affected somehow by such additions or modifications.

To improve the performance of a product is an ongoing aim in software development. If, however, new functionality requires the base services to incorporate additional functions as well, overall performance can be affected as a tradeoff to the new function. If, for example, a future release of DFS incorporates full encryption for data transfer over the network, server performance will be somewhat different from current releases without such encryption.

As a general rule, however, new releases and versions can be expected to have performance improvements incorporated. As with product updates, if performance is a critical issue in a specific environment, new releases and versions of the products should be tested and performance measurements may be required with the new product version or release.

C.3 New DCE Releases from The Open Group

The Open Group, formerly the Open Software Foundation (OSF), is responsible for the primary development of DCE and DFS releases. Vendors such as IBM build up their products on the code releases available from the The Open Group.

The current DCE products for AIX and OS/2 are based on OSF DCE 1.1. The most recent releases of DCE and DFS from The Open Group are 1.2.1 and 1.2.2, both introduced in 1996. They both incorporate performance improvements in certain areas:

- Security Server

  Improvements to the Security Server allow customizable checkpoint intervals and partitioning the database such that checkpointing—a time when the Security Server's response time is degraded due to internal work—is more controllable and does not take much time even with large databases.

- Distributed File System

  There are several improvements to DFS that increases its performance:

  – An optimized token manager decreases memory requirements and improves performance drastically.

  – Improved Vnode/VM management scales better in heavily loaded DFS servers and operates more efficiently in multiprocessor machines.

  – Enhanced replication services scale better with a growing number of files.

  – Bulk transfers for directory browsing enhances directory operation performance significantly.

  – DFS backup performance is improved through several changes.

There are other improvements that may lead to performance improvement in certain areas due to better adaptability to current processes. Other improvements introduced in DCE 1.2 have already been implemented by IBM in the current versions of the DCE products and made available to the customers.

Appendix D. Setting Preferred Security Server

Setting a preferred Security Server for DCE clients not only helps to coordinate and separate network traffic in distant networks but also results in a remarkable performance improvement. This is achieved by controlling access to the Security Server by using the pe_site file. For further details on the pe_site file, see also 7.2.1, “Security Service, the pe_site File” on page 142.

D.1 OS/2 Full Client

When OS/2 DCE clients and servers are started, by default the pe_site file is updated with information of the binding information of Security Servers within the cell. The pe_site file can be updated with information after DCESTART has been performed by having a procedure or application performing operations on the pe_site file. The pe_site file for AIX is updated by the chpesite command.

The pe_site file will control which Security Server to contact. The search order and priority of which Security Server to contact can be established in the CDS using a RPC profile entry.

This is performed by the DCE client loockup into the /.:/hosts/<hostname>/profile, where the profile contains an entry describing which LAN profile to use. For example, for the *ev2* machine:

```
dcecp> rpcprofile show /.:/hosts/ev2/profile
{{00000000-0000-0000-0000-000000000000 0.0}
/.../itsc.austin.ibm.com/cell-profile 0 }
{{6f264242-b9f8-11c9-ad31-08002b0dc035 1.0}
/.../itsc.austin.ibm.com/lan-profile 0 LAN}
```

The default group of Security Servers to select from are listed in the /.:/sec entry of the CDS namespace.

```
dcecp> rpcgroup list /.:/sec
/.../itsc.austin.ibm.com/subsys/dec/sec/master
/.../itsc.austin.ibm.com/subsys/dec/sec/ev3
/.../itsc.austin.ibm.com/subsys/dec/sec/warp40srv
```

To change the search list and to prioritize, the profile of a defined host must be modified to include a LAN profile, that has defined the search order and the priorities. To do this, the LAN profile must be created.

Assuming the lan-profile_center is a profile intended for locally attached machines on a campus LAN and that remote destinations must be excluded, the following must be performed:

Create the RPC profile entry into the CDS namespace

```
dcecp> rpcprofile create /.:/lan-profile_center
```

Add selected Security Server entries into the created profile entry:

```
dcecp> rpcprofile add -a rs_bind
                      -i d46113d0-a848-11cb-b863-08001e046aa5,2.0 \
                      -m /.:/subsys/dce/sec/ev3 \
                      -p 0 \
                      /.:/lan-profile_center

dcecp> rpcprofile add -a rs_bind \
                      -i d46113d0-a848-11cb-b863-08001e046aa5,2.0 \
                      -m /.:/subsys/dce/sec/master \
                      -p 1 \
                      /.:/lan-profile_center
```

To make use of the newly created lan-profile_center, the host profile must be modified for those hosts that are going to make use of the newly created LAN profile lan-profile_center. This is performed by the following command:

```
dcecp> rpcprofile add -a rs_bind
                      -i d46113d0-a848-11cb-b863-08001e046aa5,2.0 \
                      -m /.:/lan-profile_center \
                      -p 0 \
                      /.:/hosts/<hostname>/profile
```

Having multiple Security Server replicas, the method of preferred Security Server allows the cell administrator to apply different LAN profiles for different use.

Assuming the warp40srv Security Server replica is located at a remote location, with a number of DCE clients placed in the same location, a LAN profile prioritizing the local Security Server replica would be preferred. To do this, the cell administrator could create a LAN profile specific for the DCE machines placed near the warp40srv Security Server machine. For example:

Create the RPC profile entry into the CDS namespace:

```
dcecp> rpcprofile create /.:/lan-profile_remote
```

Add selected Security Server entries into the created profile entry:

```
dcecp> rpcprofile add -a rs_bind
                      -i d46113d0-a848-11cb-b863-08001e046aa5,2.0 \
                      -m /.:/subsys/dce/sec/warp40srv \
                      -p 0 \
                      /.:/lan-profile_remote

dcecp> rpcprofile add -a rs_bind \
                      -i d46113d0-a848-11cb-b863-08001e046aa5,2.0 \
                      -m /.:/subsys/dce/sec/master \
                      -p 1 \
                      /.:/lan-profile_remote
```

The DCE clients on the remote location must have modified the host profile entry to contain the lan-profile_remote entry.

```
dcecp> rpcprofile add -a rs_bind
                      -i d46113d0-a848-11cb-b863-08001e046aa5,2.0 \
                      -m /.:/lan-profile_remote \
                      -p 0 \
                      /.:/hosts/<hostname>/profile
```

With this concept of preferred Security Server replica, the cell administrator assigns a LAN profile for the hosts. If changes into the number of Security Server replicas are necessary, the only place the cell administrator has to apply changes is in the LAN profiles defined.

D.2 OS/2 DCE Slim Client

The OS/2 DCE slim client does not have host entries in the CDS namespace. To allow the OS/2 slim client to make use of the preferred Security Server design, entries must be added in the CDS namespace.

The host directory for slim hosts must be created first. This needs to be done only once for all slim clients with the following command:

```
dcecp> directory create /.:/slim.hosts
```

The following steps must be done for each slim client to make use of the preferred Security Server replica:

```
dcecp> directory create /.:/slim.hosts/<hostname>

dcecp> rpcprofile add -a slim_client \
                      -i 6f264242-b9f8-11c9-ad31-08002b0dc035,1.0 \
                      -m /.:/lan-profile_remote \
                      -p 0 \
                      /.:/slim.hosts/<hostname>/profile \
```

Additional Information related to the preferred Security Server design can be found on the following Web pages:

- http://ps.boulder.ibm.com/pbin-usa-ps/getobj.pl?/pdocs-usa/dssspr.html
- http://www.software.ibm.com/is/sw-servers/directory/dsssup.html
- http://www.software.ibm.com/

Appendix E. Lab Testing Results

This Appendix lists some detailed results from the various tests that have been done in order to measure or verify a certain network flow or behavior of specific processes or components. The discussion of these can be found in Chapter 2, “Process Flow and Components Affecting Performance” on page 7.

E.1 Test Environment

The test environment used for most tests is as illustrated below:

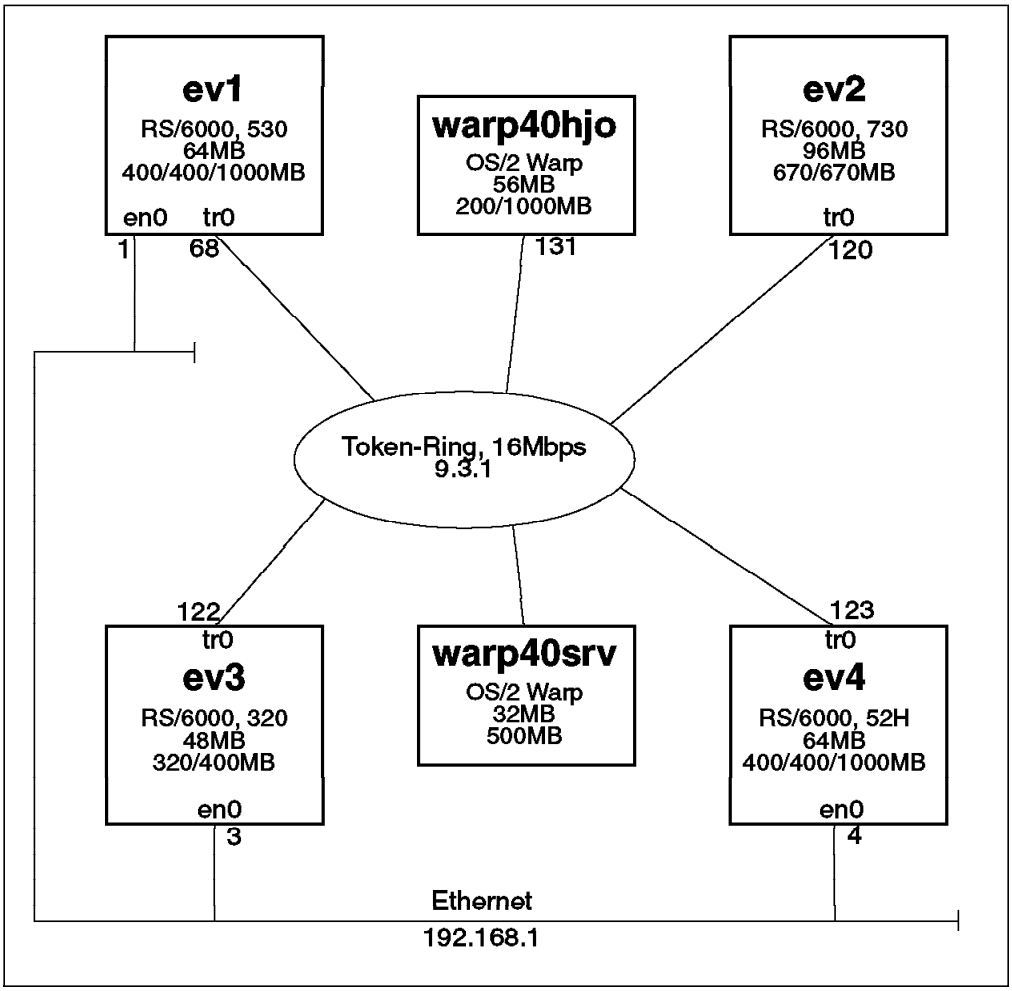


Figure 48. Lab Test Environment

The test environment included a number of shell scripts on the AIX machines. For the OS/2 machines, they represented of different scenarios, where they either were configured especially for the scenario or started with the DCE daemons as needed in a test run.

Some of the resulting figures in the AIX test scenarios where reconfirmed on corresponding OS/2 scenarios.

The hardware and software configuration of the AIX systems were:

Table 24. IBM RS/6000 Hardware and Software Used in the Test Environment

System Name	ev1	ev2	ev3	ev4
Model	Model 530	Model 730	Model 320	Model 52H
Memory	64 MB	96 MB	48 MB	64 MB
Disks [MB]	400/400/1000	670/670	320/400	400/400/1000
IBM AIX	Version 4.1.4 plus latest PTFs (as of October 1996)			
IBM DCE for AIX	Version 2.1 plus PTFset 12			

The two OS/2 systems where configured with a large amount of memory. The warp40srv system was primarily used as a client system, whereas the warp40hjo system functioned as various DCE servers.

Table 25. IBM PC Systems Hardware and Software Used in the Test Environment

System Name	warp40hjo	warp40srv
Model	PS/2 77	PS/Valuepoint
Memory	56 MB	32 MB
Disk	400/1000 MB	400 MB
Warp	Warp Server 4.0	Warp Server 4.0
DCE	OS/2 DSS 4.0	OS/2 DSS 4.0

Unless reconfigured for specific tests, the basic system configuration of the AIX systems was as shown in the following table.

Table 26. Test Environment DCE System Configuration

DCE Service	ev1	ev2	ev3	ev4	OS/2
Master Security Server	✓				
Initial CDS Server		✓			
DTS Local Server	✓	✓	✓		
DFS System Control Machine		✓			
DFS Fileset Database Machine		✓		✓	
DFS File Server Machine		✓		✓	
Security Client	✓	✓	✓	✓	✓
CDS Clerk	✓	✓	✓	✓	✓
DTS Clerk				✓	✓
DFS Client		✓	✓	✓	✓

The test runs were performed as described in the following sections. Unless required for their specific traces, the DTS and DFS services were not running, because they may have caused additional, unrelated traffic which might have interfered with the traces of the DCE core services.

E.2 Network Tracing

Most traces were taken with the AIX built-in iptrace facility. This tool can be started multiple times per machine in order to trace traffic to and from specific destination systems.

The following example command starts tracing on *ev1* for traffic between *ev1* and *ev3*:

```
# iptrace -a -s ev1 -b -d ev3 -b -i tr0 /tmp/ev1_ev3.trc
```

This command stores the traced data in the file /tmp/ev1_ev3.trc. After completion of the tests, iptrace needs to be stopped by means of the kill command (iptrace could also be started and stopped under control of the System Resource Controller using the startsrc and stopsrc commands, respectively).

After the iptrace was stopped, the file containing the raw data of the trace (/tmp/ev1_ev3.trc, in this case) must be formatted with the ipreport command in order to process the information.

The output from the ipreport command was then used by the various scripts to extract and summarize the required information.

Some tests were also traced with a Personal Computer running the IBM DatagLANce LAN and protocol analyzer application. In some DFS test cases, only DatagLANce could be used because iptrace was unable to catch up with the high volume and reported dropped packets (it is believed that the DFS kernel processes run at a higher priority than iptrace and therefore may not allow iptrace to have enough CPU time).

Most tests were performed five or more times in order to have reliable data.

E.3 How to Read the Test Results

The test results of the major parts of the test scenarios are presented in two tables each. The first table is a summary table on a per-system-bases, for instance it tells you how much traffic flew through a network interface of the specific machine in either direction. It gives you the average numbers of packets and the average packet sizes in bytes per test run, as well as the maximum total network traffic in bytes involved in a certain scenario. Average values are calculated as a mean from all single test runs, whereas the maximum traffic is the highest observed figure in any single test run. These figures can be used to describe the processes and the resource usage of them.

Table 27. Traffic Summary for Starting CDS Daemon (Sample)

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	49	90	14805
Security Server	145	91	149396
CDS Server	241	92	134591

The second table shows the observed distribution of the network traffic in the test scenario on a per-connection-basis. In the example case shown in Table 27 and Table 28 on page 248, the DCE client (Cli) does have some network conversations to the Security Server (Sec) and to the CDS Server (CDS), but there is no traffic between the two servers (Sec and CDS). The minimum, average and maximum figures again can be used to estimate the type of traffic generated. For the traffic between the client and the CDS Server, a large range of resulting figures have been observed in this example.

Table 28. Traffic between Machines for Starting XYZ Daemon (Sample)

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	24	49	101	62	90	218	2164	9211	14805
Cli ↔ CDS	6	241	360	62	92	218	1208	85855	134591
Sec ↔ CDS	0	0	0	0	0	0	0	0	0

The figures obtained in the following sections represent base scenarios. Repeating the scenarios may generate other figures for several reasons. Other scenarios can easily be accomplished, traced and analyzed. Please note that although the tests have been run several times, the values always only represent the absolute (min, max) or average (avg) traffic for one single run of the test; they never show total traffic for more than one test run. For example, the first row from Table 28 reads like: There was at least one test run with only 24 packets, at least one with 101 packets, and the average number of packets from all test runs was 49. The smallest packet size in any of the test runs was 62 bytes; the largest was 218, and the average packet size from all test runs was 90 bytes. At least one test run created only 2164 bytes of network traffic; at least one used 14805 bytes, and the overall average is 9211 bytes.

E.4 Test Scenarios for DCE Core Services

In the following sections, every test is briefly described, followed by the summarized and detailed test results in form of tables. Where necessary, additional comments about special observations or about special provisions have been added. The general discussion on the test results can be found in Chapter 2, “Process Flow and Components Affecting Performance” on page 7.

E.4.1 Start of the DCE Client Daemons

The following tests give you an overview on how much network traffic is generated by starting DCE clients. This can be important if a large number of client systems are expected to be started in a short period of time.

Start of the DCE Daemon (dced): Whenever the DCE client is started, there will be some traffic involved. The local machine principal is the first to obtain credentials for the machine’s principal:

```
/.:/hosts/<machine name>/self
```

The test was performed by stopping DCE on *ev3* and *ev4*, and then removing all DCE cache files to make sure there was not any related information stored in cache.

The command rc.dce dced was run on an AIX client machine five times while the network traffic was captured. The following table lists the results of the test on a per-system basis for the start of dced:

Table 29. Traffic Summary for Starting dced Daemon

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	81	413	33999
Security Server	81	413	33999
CDS Server	0	0	0

Note that dced was stopped five minutes after being started, but this traffic is not included in the tables.

The following table summarizes the results on a per-connection basis.

Table 30. Detailed Traffic Between Machines for Starting dced Daemon

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	80	81	81	130	413	1378	32775	33366	33999
Cli ↔ CDS	0	0	0	0	0	0	0	0	0
Sec ↔ CDS	0	0	0	0	0	0	0	0	0

For a discussion on these results, see 2.3.1, “Client Machine Startup” on page 15.

Start of the CDS Advertiser (cdsadv): The cdsadv command starts the advertisements and solicitation daemon on the local client system. The test scenario traced the network traffic when the CDS advertiser was started.

With only the dced daemon running on *ev3* and *ev4* (clients), the CDS advertiser was started with the command rc.dce cdsadv and stopped five minutes later. The stop of cdsadv also caused some network traffic. However, the resulting numbers in the following two tables contain only traffic generated by the start of cdsadv:

Table 31. Traffic Summary for Starting the CDS Advertiser

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	132	270	36060
Security Server	35	136	5105
CDS Server	97	318	31017

The test was repeated five times. Table 32 on page 250 shows the details.

Table 32. Detailed Traffic Between Machines for Starting the CDS Advertiser

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	32	35	39	62	136	1326	4435	4747	5105
Cli ↔ CDS	95	97	99	62	318	1298	30801	30932	31017
Sec ↔ CDS	0	0	0	0	0	0	0	0	0

The above tables exclude the very first start of cdsadv on every machine. This first start of cdsadv after a removal of the cache files caused remarkably more traffic than the subsequent tests. It was found that the traffic to/from the Security Server was up to two times higher and to/from the CDS server even up to four times higher during such an initial start.

Start of the DTS Clerk Daemon (dtsd -c): Starting the DTS client causes the local time to be automatically synchronized with the DCE cell time represented by DTS Servers.

The following test was performed on *ev4* (DCE client, DTS clerk). The DTS client was started by the command `rc.dce dtsd`.

After the test scenario was executed, the analysis of the traces provided the following figures:

Table 33. Traffic Summary for DTS Clerk Daemon Start

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	294	260	102112
Security Server	65	159	21869
CDS Server	196	312	76267

And the detailed data are:

Table 34. Detailed Traffic Between Machines for DTS Clerk Daemon Start

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	25	65	117	62	159	1440	2210	10324	21869
Cli ↔ CDS	168	196	244	62	312	1514	53158	61308	76267
Cli ↔ DTS-Srv	33	33	33	62	140	1386	4577	4604	4710
Sec ↔ CDS	0	0	0	0	0	0	0	0	0

The second-to-last row shows the network traffic between *ev4* (DTS clerk) and *ev3* (DTS Server). It must be noticed that the figures in the tables are valid only after the DTS clerk had already been started and stopped on *ev4* (Cli) at least one time after all DCE caches were erased. The first start of `dtsd` after a cache

clean-out imposed about 50percent more network traffic as compared to the figures in the tables above.

Start of the DFS Client: For analyzing the traffic involved when starting a DFS client, we use a different system setup by adding a separate machine that houses all the DFS server processes (DFS System Control Machine, DFS Fileset Database Machine, and DFS Fileset Server), in the following tables simply referred to as the *DFS Server*. Since we are mostly interested in the network traffic between the client machine and the servers, we did not trace the traffic between the servers. The tests have been run five times in order to have comparable values and to calculate meaningful average values.

Table 35. Traffic Summary for DFS Client Start Up

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	215	256	58414
Security Server	70	182	13916
CDS Server	97	277	28991
DFS Server	48	324	17119

Table 36 shows the details:

Table 36. Traffic between Machines for DFS Client Start Up

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	59	70	77	62	182	1346	11093	12764	13916
Cli ↔ CDS	56	97	106	60	277	1310	14755	26740	28991
Cli ↔ DFS	48	48	49	130	324	1514	14133	15637	17119

Starting a DFS client actually starts the `dfsbind` and the `dfsd` processes. The former, `dfsbind`, does not create any network traffic when being started.

E.4.2 User Authentication (dce_login)

The DCE login command validates the principals identity and obtains network credentials for that principal.

The test scenario performs a series of DCE logins from an AIX DCE client machine (*ev3*) with a DCE user created for this purpose. This user belongs to only one group and, as such, represents a minimum.

The command that was used to log in the user on *ev3* (DCE client) was:

```
# dce_login <dce user> <password>
```

The following tables summarizes the results per login of a rerun of the DCE login scenario 20 times.

Table 37. Traffic Summary for dce_login

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	65	278	19968
Security Server	19	340	8214
CDS Server	46	252	11862

The following second table lists the traffic details per login between the involved server and client machines.

Table 38. Traffic Between Machines for dce_login

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	16	19	43	62	340	886	6148	6395	8214
Cli ↔ CDS	44	46	49	62	252	786	11552	11698	11862
Sec ↔ CDS	0	0	0	0	0	0	0	0	0

Note: In some cases, the CDS Server did not get involved at all (for example, there was no traffic between the client and the CDS Server). Table 38 lists only those cases where the CDS Server was involved; otherwise the second row from the bottom would have all zeros in the 'min' fields. However, most DCE logins involve the CDS Server, unless special configuration has been done.

E.4.3 User Reauthentication (kinit)

The `kinit` command is used to refresh the TGT (Ticket Granting Ticket) of a principal. The tickets in the local cache file are replaced by the refreshed TGT.

The test scenario was performed on *ev3* (DCE client). The command `kinit <dce user>` was used to do the tests. The resulting numbers of the test scenario are in the following two tables:

Table 39. Traffic Summary for kinit

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	73	227	17739
Security Server	30	270	8917
CDS Server	43	197	8822

And the details are:

Table 40. Detailed Traffic Between Machines for kinit

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	17	30	37	62	270	1360	7762	7970	8917
Cli ↔ CDS	34	43	45	62	197	786	5432	8405	8822
Sec ↔ CDS	0	0	0	0	0	0	0	0	0

E.4.4 User Deauthentication (kdestroy)

Whenever a DCE login is performed using a valid principal, there will be a login context created. This login context is used by any spawned processes from this DCE login process.

The kdestroy command destroys the principal DCE login context and the principal's credentials, in other words, it deletes the credential files.

Doing this using the kdestroy command does—not surprisingly—not create any network traffic, since kdestroy only removes the local credential files from the credential cache directory.

E.4.5 Stopping DCE Client Daemons

After the previous tests, we want to know how much packets of information are involved in the disincorporation of the DCE services on a machine. To achieve that we stopped all the DCE services from *ev3*. The command used, after all client services had been started, was `dce.clean all`.

Table 41. Traffic Summary for dce.clean all

Machine	Avg Nbr of Packets	Avg Size of Packets	Max Traffic (Bytes)
Client	114	424	29385
Security Server	32	95	4626
CDS Server	82	299	24821

Detailed traffic:

Table 42. Detailed Traffic between Machines for dce.clean all

Connection	Number of Packets			Size of Packets			Total Traffic (Bytes)		
	min	avg	max	min	avg	max	min	avg	max
Cli ↔ Sec	26	32	51	62	95	218	2560	3035	4626
Cli ↔ CDS	79	82	85	62	299	1300	24449	24659	24821
Sec ↔ CDS	0	0	0	0	0	0	0	0	0

There is no network traffic between the Security Server and the CDS Server. Stopping the DFS client processes does not cause any traffic.

E.4.6 Security Server Under Heavy Load

The purpose of this test was to characterize the type of application a Security Server represents and to gain some estimate on how many user logins a certain configuration is able to process.

The test environment consisted of a Security Server under test and up to 11 RS/6000 systems (including an SP2) as clients, each running six instances of a test script in parallel, simulating user logins. The test script simulated users logging in to DCE using the `dce_login` command. The security registry database contained 3500 user accounts and 3500 groups. Each user account had its own and unique primary group; thus, there was no group to which all the users belonged to.

The test script used was:

```
#!/bin/ksh
i=0
cnt_file=counter.$$
export TRY_PE_SITE=1
typeset -Z5 rnd
typeset -L4 index
while true; do
  rnd=$RANDOM; index=$rnd                # generate 4-digit random number
  /usr/lpp/dce/bin/dce_login user$index user <<EOF
  /usr/lpp/dce/bin/kdestroy
EOF
  ((i=i+1))
  echo $i >$cnt_file
done
```

Figure 49. Script Listing for Testing a Heavily Loaded Security Server

The test script picks a random principal name in the range from *user0000* through *user3276* and performs a `dce_login` with that particular user id. The user accounts and passwords were set up accordingly such that the `dce_login` succeeded. A counter value was written to a file in order to count the login attempts.

This setup simulates 66 users performing logins to DCE as fast as the Security Server can process them, which means that the CPU on the Security Server was 100 percent busy during the tests. The tests were run for 10 minutes and the `netpmon` tool was used on the AIX Security Servers to report the usages of the system's resources.

The Security Servers tested were run on different RISC System/6000 and PS/2 hardware in order to compare their performance:

- An RS/6000 Model 320 (Power, 20 MHz)
- An RS/6000 Model 550 (Power, 41.6 MHz)
- An RS/6000 Model 43P (604 PowerPC, 133 MHz)
- An SP2 Wide Node, equivalent to a RS/6000 Model 590 (Power2, 66 MHz)
- A PS/2 Model 77i (Intel 486 DX4, 33/100 MHz)

Sufficient memory was installed in these server machines such that there was no paging (swapping) activity during the tests. No tuning, other than the increased number of listener-threads on AIX, was done on the server machines.

The following table shows the number of logins a server was able to process within 10 minutes (10 m), and the same number expressed in logins per second (lps):

Table 43. DCE Login Performance Comparison

Server Model, Protocol Used	Model 320		Model 550		Model 43P		SP2 WN		PS/2 77i	
	10 m	lps	10 m	lps	10 m	lps	10 m	lps	10 m	lps
UDP (ncadg_ip_udp), 5 Listener-Threads	1236	2.1	2971	5.0	6742	11.2	6668	11.1	UDP:	
TCP (ncacn_ip_tcp), 5 Listener-Threads	1358	2.3	2848	4.7	6659	11.1	6793	11.3	3124	5.2
UDP (ncadg_ip_udp), 15 Listener-Threads	1406	2.3	2592	4.3	6846	11.4	7091	11.8	TCP:	
TCP (ncacn_ip_tcp), 15 Listener-Threads	1372	2.3	3339	5.6	6956	11.6	7041	11.7	2868	4.8

The tests have been done with both UDP and TCP as the transport protocol since we have seen in earlier tests that the protocol sequence does have an impact on the performance. This was done by moving the corresponding line to the top in the pe_site file. Also, the only tuning parameter for secd, which is the number of listener-threads, has been varied between the lowest value of 5 (equals to the default) and the maximum number of 15.

Note: Since secd on OS/2 Warp does not support a customizable number of listener-threads, only two results are shown in Table 43, for UDP and for TCP protocols, respectively.

The following Figure 50 on page 256 shows the CPU utilization, along with rough figures for disk and network I/O as observed during the tests.

The graph shows clearly that secd is a CPU-bound process. Only if there is not enough physical memory available to keep the registry, paging activity will slow down the performance. The percentages of CPU utilizations shown in Figure 50 on page 256 are medium values from all tests; they did not change remarkably among the different tests and hardware systems. The values shown for disk and network I/O are rough figures from sample measurements; they have not been verified exactly in every test run because they are not likely to induce a considerable load requiring a closer focus. At some times, the network I/O was higher, which is believed to be due to base load on the network, DNS traffic, and due to retries, caused by UDP buffer overflows.

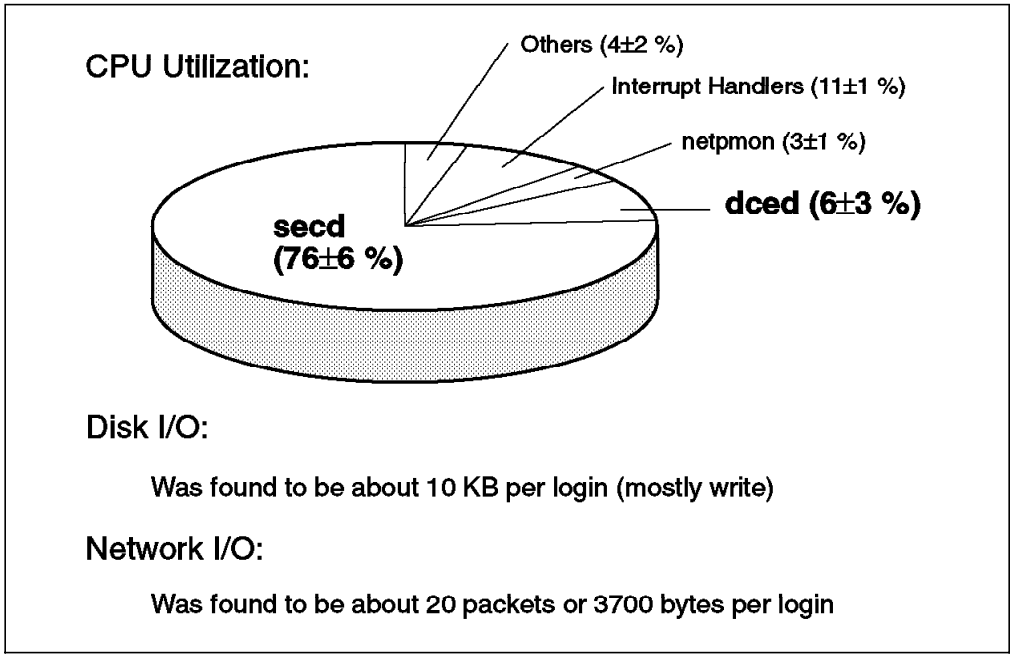


Figure 50. Resource Utilization on a Heavily Loaded Security Server (AIX)

Due to increased disk I/O on heavy login load during the tests, the performance of the 43P and the SP2 node were slightly degraded because there was *I/O wait* reported at an average of about 5 percent of CPU time.

On the clients, the `TRY_PE_SITE` environment variable was set; thus the CDS Server was not involved in the tests. Some additional tests have been done without this environment variable being set. The result revealed that there is no remarkable difference in the Security Server performance. On the other side, the CDS Server became involved as well, which does not happen through the use of this variable.

Similar tests with only a few registry objects defined did not improve server performance notably. In other words, the number of objects stored in the registry database has no significant influence on performance.

The tests may not be representing a real environment for the following reasons:

- Any software releases and fix levels other than the ones used in the tests can lead to other results.
- Besides the monitoring tool, `netpmon`, there was nothing running on the servers. In real environments, most often other tools may be running at the same time.
- The simulated users produced their requests in a loop only as fast as the server could handle them. Only a (theoretic) maximum of 66 requests could have been pending at any given time. In real environments, there is no such dependency, resulting in the unlikely chance of having hundreds of users logging in at exactly the same time, leading to time-out situations that can make login attempts to fail.
- Different device handlers with different characteristics are being used for Token-Ring (as used in the tests) and Ethernet or for other network interfaces.

- Changes might also occur to the registry database at the same time as other users log on, for example due to password changes, which might result in a reduced performance.

E.4.7 CDS Server under Heavy Load

A CDS Server has a huge variety of functions that are used in daily business, such as:

- Directory lookup
- Profile lookup
- Object search and retrieval
- Object write (registration)
- Object updates

The way these functions are accessed may vary among different application implementations. Any figures that are measured are valid only for this specific kind of test and cannot be used in general to estimate the performance of a CDS Server. For this reason, we did not measure performance figures, but rather wanted to know how machine resources are being used by cdsd, the CDS Server process. The results look—not surprisingly—very similar to the results from the Security Server tests:

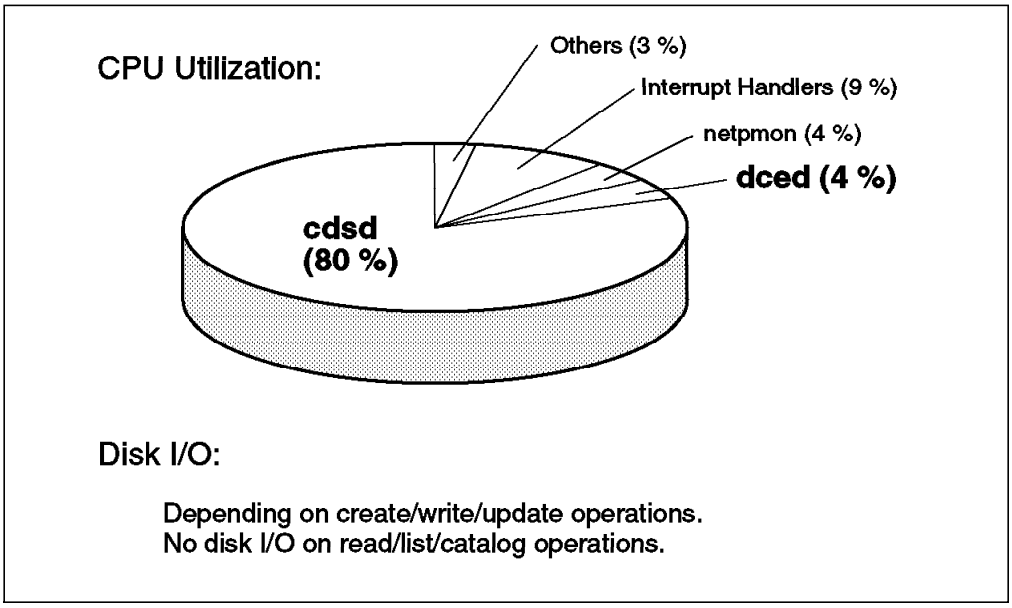


Figure 51. Resource Utilization on a Heavily Loaded CDS Server (AIX)

The tests have been done using a total of eight RISC System/6000 as clients, all running shell scripts which did a mixture of directory and object listings, as well as creation and deletion. Network I/O was found to be low (beneath 100 packets per second), but this is due to the limited test environment and is not related to CDS Server performance.

Appendix F. Special Notices

This publication is intended to help DCE system administrators and planners to optimize their environment for performance. The information in this publication is not intended as the specification of any programming interfaces that are provided by any DCE or DSS product for AIX and OS/2 Warp. See the PUBLICATIONS section of the IBM Programming Announcement for DCE and DSS products for AIX and OS/2 Warp for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ADSTAR®
AIX®
AIX/6000®
AnyNet®
AS/400®
AT®
BookManager®
Current®
DatagLANce
DB2®
DFS
FFST/2
First Failure Support Technology
IBM®
InfoExplorer
LANStreamer®
NetFinity®
NetView®
OpenEdition®
OS/2®
POWERparallel®
PowerPC®
PROFS®
PS/2®
RISC System/6000®
RS/6000
SP2®
System/390®
SystemView®
VM/ESA®

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix G. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

G.1 International Technical Support Organization Publications

For information on ordering these ITSO publications, see “How To Get ITSO Redbooks” on page 265.

- *IBM DSS and DCE Cross-Platform Guide*, SG24-2543
- *Administering IBM DCE and DFS Version 2.1 for AIX (and OS/2 Clients)*, SG24-4714
- *The OS/2 Debugging Handbook* Series, SG24-4640 through SG24-4643
- *Inside the Directory and Security Server for OS/2 Warp*, SG24-4785 (in press)
- *The Distributed File System (DFS) for AIX/6000*, GG24-4255
- *Understanding OSF DCE 1.1 for AIX and OS/2*, SG24-4616
- *DCE Cell Design Considerations*, SG24-4746
- *RS/6000 Performance Tools in Focus*, SG24-4989 (in press)

G.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RISC System/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RISC System/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

G.3 Other Publications

These publications are also relevant as further information sources:

Softcopy documentation shipped with the DCE and DSS products on the AIX platform, including:

- *Introduction to DCE*
- *DSS Up and Running!*
- *DCE for AIX Administration Guide*
- *DCE for AIX Administration Command Reference*
- *DCE for AIX Application Development Guide - Introduction*

- *DCE for AIX Application Development Guide - Core Services*
- *DCE for AIX Application Development Guide - Directory Services*
- *DCE for AIX Application Development Reference*
- *DCE for AIX DFS Administration Guide and Reference*
- *DCE for AIX NFS/DFS Authenticating Gateway Guide and Reference*
- *DCE Error Message Reference*
- *IBM AIX High Availability Cluster Multi-Processing Guide for DCE and DFS*
- *DCE Manager User's Guide*

Softcopy documentation shipped with the DCE and DSS products on the OS2/Warp platform, including:

- *Up and Running!*
- *DCE for OS/2 Warp: Getting Started*
- *Client User's Guide*
- *Commands and Utilities*
- *DCE for OS/2 Warp: Administration Guide*
- *Administration Command Reference*
- *DSS Administrator's Reference*
- *DFS Client Guide and Reference*
- *MPTS Configuration Guide*
- *DSS Programming Guide and Reference*
- *DCE for OS/2 Warp: Error Messages Reference*
- *Error Messages References*
- *DSS Problem Determination*

General publications:

- *AIX Performance Tuning Guide*, SC23-2365
- *Distributed Computing Environment: Understanding DCE Concepts*, GC09-1478

The following books are published by O'Reilly & Associates:

- *Understanding DCE*
- *Guide to Writing DCE Applications*
- *DCE Security Programming*
- *Pthreads Programming*

Information on the Internet:

- The IBM DCE home page at `http://www.networking.ibm.com/dce/dcehome.html` sporadically carries links to performance related white papers and lab test reports.
- The IBM DCE Support Web page at `http://service.software.ibm.com/dssdce`.
- Transarc Corp.'s Web pages at `http://www.transarc.com/` has lots of valuable technical information about DCE.

- The Open Groups's Web pages at http://www.opengroup.org/.
- The news group about DCE: comp.soft-sys.dce.

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

ACL	Access Control List
ADSM	ADSTAR Data Storage Management
ANSI	American National Standards Institute
APA	All Points Addressable
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BDM	Binary Distribution Machine
CDMF	Commercial Data Masking Facility
CDS	Cell Directory Services
DCDB	Domain Controller Database
DCE	Distributed Computing Environment
DES	Data Encryption Standard
DFS	Distributed File System
DNS	Domain Name Service
DSS	Directory and Security Server
EMS	Event Management Services
EPAC	Extended Privilege Attribute Certificate
ERA	Extended Registry Attributes
FDDI	Fiber Distributed Data Interface
FLDB	Fileset Location Database
HPFS	High Performance File System (OS/2)
IBM	International Business Machines Corporation
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IP	Internet Protocol
ISO	International Standardization Organization
ITSO	International Technical Support Organization
LAN	Local Area Network
LAPS	LAN Adapter and Protocol Support
LFS	Local File System
MIB	Management Information Base
MPTN	Multiple Protocol Transport Networking
MPTS	Multiple Protocol Transport Services
MSS	Maximum Segment Size (TCP/IP)
MTU	Maximum Transfer Unit
NFS	Network File System
NIS	Network Information System (Sun)
NTP	Network Time Protocol
OSF	Open Software Foundation
PAC	Privilege Attribute Certificate
PROFS	Professional Office System
PTGT	Privilege Ticket Granting Ticket
PTF	Program Temporary Fix
SCM	System Control Machine
SMP	Symmetric Multiprocessors
SMIT	Systems Management Interface Tool
SP2	Scalable POWERparallel System 2
SPMI	System Performance Measurement Interface
TCP	Transmission Control Protocol
TGT	Ticket Granting Ticket
UDP	User Datagram Protocol
UTC	Universal Time Coordinated
UUID	Universal Unique Identifier
WAN	Wide Area Network

Index

Special Characters

A

B

C

D

E

F

G

R

S

T

U

V

W

X

ITSO Redbook Evaluation

DCE and DFS Performance Tuning and Problem Determination on AIX and OS/2 SG24-4919-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redeval@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

This soft copy for use by IBM employees only.

Printed in U.S.A.