Multimedia Systems (1994) 2:103–117

Multimedia Systems

A performance analysis of personal computers in a video conferencing environment

Khoa D. Huynh[1]*, **Taghi M. Khoshgoftaar**[2]**

[1] IBM Corporation, Internal Zip 1430, P.O. Box 1328, Boca Raton, FL 33429-1328, USA
[2] Department of Computer Science & Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA

Abstract. New intelligent adapters, advanced bus architectures, and powerful microprocessors have resulted in a new generation of personal computers with true multimedia capabilities. Collaborative applications are the most demanding applications of multimedia technologies today. We present a performance analysis of how effective video conferencing applications can be supported with personal computers connected through a local area network (LAN). We also evaluate the performance impact of an advanced, peer-to-peer I/O protocol. The key factor in the performance of a video conferencing system over a LAN is the video compression and decompression algorithms. At high video frame rates, the peer-to-peer I/O protocol performs better than the traditional, bus-master, interrupt-driven I/O protocol.

Key words: Video conferencing – Subsystem control block architecture – Peer-to-peer I/O – Interrupt-driven I/O – Move mode – Locate mode

1 Introduction

A government study reports that interactive video instruction improves achievement by an average of 38% over conventional instruction methods while reducing the time needed to become proficient by 31% (Fletcher 1990). Another study (Isaacs and Tang 1993) finds that, compared with audio-only, a video channel adds or improves the ability to show understanding, forecast responses, give nonverbal information, enhance verbal descriptions, manage pauses, and express attitudes. These findings suggest that full-motion, interactive video applications may be particularly useful for handling conflict and other interaction-intense activities. Recent advances in communications technology have made large bandwidth available at reasonable cost, and the advances in computer technology have produced powerful personal computers with built-in video and audio capabilities, such as the IBM UltiMedia[1] family of Personal System/2[1] (PS/2[1]) computers. The advent of such multimedia computers has given rise to many computer supported collaborative applications. An important class of collaborative applications is video conferencing between two or more persons (participants) (Vin et al. 1991).

In a typical video conferencing environment, each participant is equipped with a personal computer, a video camera, and a voice-recording device, such as a microphone. The participants' computers are connected via a network. This network can be a local area network (LAN), such as Ethernet or Token Ring, or it can be a wide area network, such as the Integrated Services Digital Network (ISDN). At the heart of each participant's station is the personal computer, which contains a video adapter card. The video adapter can receive the video input from the camera, compress the video frames, and send the compressed frames across the network to other conference participants. At the same time, the personal computer must also be able to receive compressed video frames from other participants. Once the (compressed) video frames are received from the network, the video adapter in the computer must mix the video frames from all participants into composite frames, decompress the composite frames, and display them on the personal computer's monitor. The process of mixing video frames from multiple participants (sources) into composite frames involves some level of image processing (Vin et al. 1991). Several proposals for multisource, multiuser video mixing (compositing) are discussed in Yun and Messerschmitt (1993). In addition to video information, voice (audio) and text information (chalkboards) must be communicated among all conference participants. The disk storage of video, voice, and data, as well as their retrieval and playback, must also be supported in some video conferencing environments. The disk storage and retrieval of conference information is important if the conference participants wish to go back and review what was discussed during an earlier segment of the conference (Gibbs et al. 1987; Little and Ghafoor 1990).

* e-mail: khoa@vnet.ibm.com
** e-mail: taghi@cse.fau.edu
Correspondence to: K.D. Huynh

[1] IBM, Personal System/2, PS/2, Micro Channel, UltiMedia, ActionMediaII, 386SLC, 486SLC2, P2P are trademarks or registered trademarks of International Business Machines Corp.

In parallel with the research and development of multimedia applications, the IBM Subsystem Control Block (SCB) architecture has been defined to fully exploit the functional and performance capabilities of increasingly powerful microprocessors, advanced system bus designs, such as the Micro Channel[1], and a whole new array of intelligent adapters (also called bus masters). The SCB architecture has two operating modes, the Locate Mode and the Move Mode (IBM 1991). The Locate Mode represents the more conventional I/O processing model where the host processor and system memory are involved in facilitating data transfers between two I/O adapters in the system. This mode is being used in the current generation of personal computers. In contrast, the Move Mode specifies a new peer-to-peer I/O protocol between various intelligent adapters in a system – without involving the host processor and system memory. Data transfers are initiated and take place directly between two adapters in the system. The critical issue here is how effective these SCB operating modes function in I/O-intensive environments, such as video conferencing and other multimedia applications. In this study, we evaluated, in a video conferencing environment, (1) how effective the SCB architecture was in using the main components of a personal computer system, such as the processor, the Micro Channel, and intelligent adapters, (2) how it impacted the time delays in sending and receiving video information to and from the network connecting the conference participants, and in particular, (3) how the two SCB operating modes, the Locate Mode and the Move Mode, compared with each other.

The organization of this paper is as follows. We first discuss the type of video conferencing environment considered in our study. We then present an overview of the SCB architecture and its two operating modes. Our simulation models for the two SCB modes operating in a video conferencing environment are described next. The performance analysis is presented after the description of the models. Finally, a summary of these results is provided in the last section.

2 System under consideration

We considered a video conferencing system with two or more conference participants. Each participant can be a single person or a group of people. At the location of each participant, there is at least a personal computer, equipped with a video adapter card, and a camera for video recording. There is no consideration of audio (voice), since audio typically takes only approximately 10% of the video's bandwidth, storage capacity, and processing power. Audio can usually be included with full-motion video with little, if any, additional cost (Harney et al. 1991). In addition, we are only interested in the system architectures required to support a video conferencing system, rather than the software architectural models for providing conference connection services and configuration management.

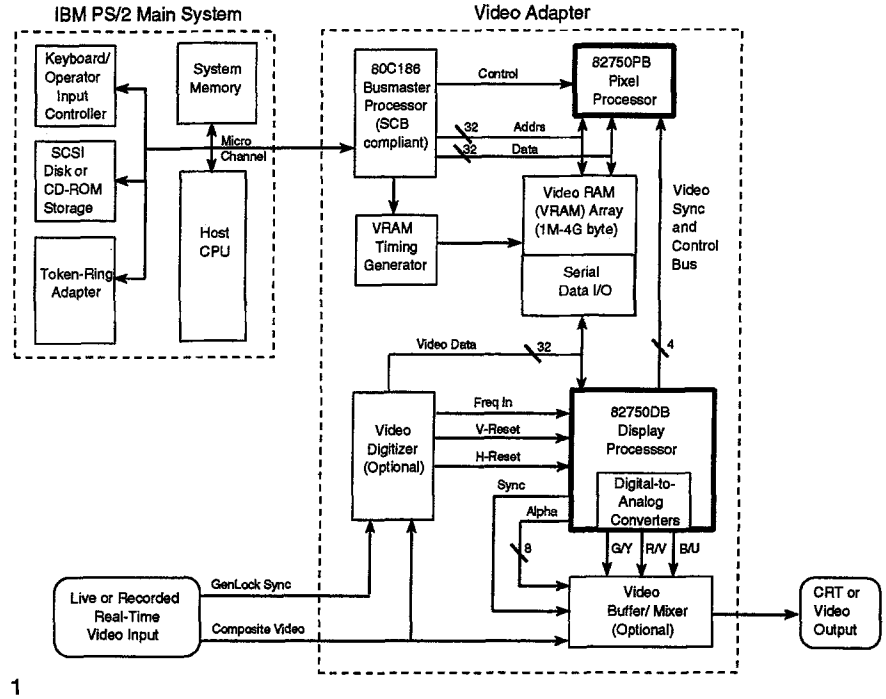
In our study, we considered each conference participant's computer as shown in Fig. 1. In the upper left portion of Fig. 1 are the usual components of the IBM PS/2 system, including the system (host) CPU, the system memory, various I/O devices, such as disks, network controllers, etc., and the Micro Channel. Also connected to the Micro Channel is an advanced video adapter, called the Digital Video Interactive[2] (DVI) subsystem. This subsystem is based on the second-generation Intel i750[2] video processor (Harney et al. 1991). It is similar to the IBM ActionMedia II[1] adapter that was introduced in 1992. However, what differentiates this adapter from the ActionMedia II adapter is the existence of an Intel 80C186[2] processor to support the SCB architecture. In addition, we have also assumed a higher-performance 50-MHz i750 video processor (the ActionMedia II uses the 25-MHz version of the i750, which is currently available). The 80C186 processor primarily acts as the central processor of the video subsystem. It handles the SCB protocol for the subsystem, such as decoding the SCBs (Locate Mode) or control elements (Move Mode) received from the host processor or other adapters. In the Move Mode, it also generates control elements to other adapters in the system, and processes replies from them. The Video RAM (VRAM) plays the role of a general-purpose buffer for the entire video subsystem. It is in this buffer that video images are kept to be compressed, decompressed, mixed, or manipulated. The VRAM is multiported, and has very fast memory cycle time (less than 100 ns).

The i750 processor includes two chips, the 82750PB pixel processor and the 82750DB display processor. The 82750PB pixel processor was designed on reduced instruction set computing (RISC) principles (Harney et al. 1991). It can support single-cycle instruction execution, real-time compression and decompression of digital motion video at up to 30 frames/s. At 50-MHz clock speed, the pixel processor can compress digital motion video at 30 frames/s using only 50% of its clock cycles; motion video decompression takes only 25% of the total clock cycles. In general, the pixel processor compresses and decompresses video images from VRAM, generates graphics and special effects. The 82750DB display processor handles real-time display functions and drives the monitor.

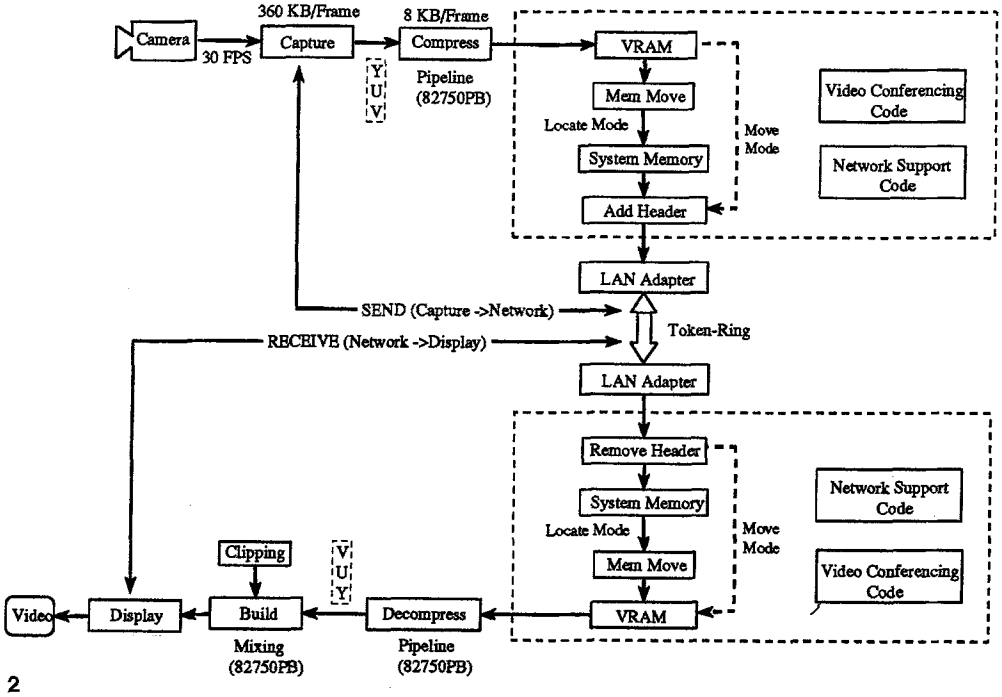
The data flows within a video conferencing system with only two participants are illustrated in Fig. 2. In the upper half of Fig. 2, we see how the video frames are captured, compressed, and sent to the other (remote) participant across the token-ring network. In the lower half of Fig. 2, the video frames are received from the token-ring network, transferred to the video adapter, decompressed, and then displayed on the monitor. It is interesting to note that the SCB Locate Mode uses system memory as an intermediate destination for transferring compressed video frames between the VRAM on the video adapter and the token-ring adapter, while the SCB Move Mode bypasses system memory. We will revisit this later as we discuss the SCB architecture in more detail. It should be emphasized that Fig. 2 only shows video data moving from one participant to another in one direction; in reality, the data flows occur in both directions between two participants.

---

[2] Intel, i386, i386DX, i486, i486DX, i750, 80C186, Digital Video Interactive, DVI, Real Time Video, RTV are trademarks or registered trademarks of Intel Corp.



1



2

Fig. 1. The system under consideration
Fig. 2. The data flow within a video conferencing system with two participants

Both midrange and high-performance PS/2 systems were studied. Below are some of the component characteristics that were implemented in the models used in our study:

The host processor. For the midrange PS/2 Model 70 system, the host processor is the Intel i386DX[2] processor running at 25 MHz with 64 kbytes of external cache. For the high-end PS/2 50 MHz Server, the host processor is a 50-MHz i486DX[2] processor with 256 kbytes of second-level cache. From now on, we refer to the midrange PS/2 Model 70 as simply the 386 system, and the high-end PS/2 50 MHz Server as the 486 system.

The system memory. In our work, we assumed that the amount of physical memory was quite large so that we would never reach the memory overcommit state; there was no memory swapping as a result of memory shortage. However, we did take into account the memory access contention in our models. The memory access speed is dependent on the host processor's cycle time. We assumed that, on both the midrange and high-

end PS/2 systems, the memory controller used the efficient dualpath design. This design implements separate data paths and first-in-first-out (FIFO) queues for the host processor and the Micro Channel. This enables both the host processor and the Micro Channel to buffer memory requests simultaneously and perform other tasks while the memory controller processes those memory requests in the FIFO queues.

The Micro Channel. The midrange PS/2 Model 70 has a 32-bit Micro Channel with a 300-ns basic cycle time, giving an effective data transfer rate of 13.33 MB/s. The high-end PS/2 50 MHz Server supports the 32-bit data streaming mode with an effective data transfer rate of 40 MB/s.

One (hypothetical) DVI video subsystem. As discussed previously, this video subsystem implements the Intel 80C186 processor to handle the SCB protocols. It also has the 82750PB pixel processor to handle the compression and decompression of digital motion video, among other tasks. The actual IBM ActionMedia II adapter, on which this DVI subsystem is based, runs the Real-Time Video^2 (RTV^2) microcode to perform compression and decompression on the i750 pixel processor. The RTV microcode has a five-stage pipeline for video compression and a three-stage pipeline for video decompression. The RTV compression pipeline is run at the frame rate while the decompression pipeline is run at twice the frame rate. The pipelines are designed to keep the video frames moving smoothly from one conference participant to the others across the token ring. This is necessary because there are many variable factors that can adversely affect the delays of motion video frames, such as host processor utilization, network traffic and bandwidth, etc. The pipelines act as a series of buffers to ensure that video frames are not lost if the network is busy or the host processor cannot schedule the software at the required time.

The negative impact of these pipelines is that they dramatically increase the capture-to-playback time delays, which are very critical in video conferencing systems. Since the i750 pixel processor can compress and decompress motion video frames at 30 frames/s, we assumed that both the compression and decompression pipelines were single-stage in our study. There were several reasons for us to assume single-stage pipelines (which would be the case for more advanced video conferencing hardware):

1. The ActionMedia II adapter uses the 25-MHz version of the i750 video processor. Our hypothetical DVI video subsystem uses the higher-performance, 50-MHz version of the i750 processor. As a result, the video compression and decompression can be completed in a single frame time (at 30 frames/s) with comfortable "dead" time for other activities.

2. The pipelines would not be needed for buffering purposes if we could implement normal memory buffers instead. We assumed that the microcode would be responsible for handling these buffers in our study.

3. The single-stage pipelines would allow us to have minimal video compression and decompression time delays. These delays are independent of the system architectures, which are the real focus of our study. Thus by minimizing these delays, we can see the impact of the SCB architecture in the capture-to-playback delays more easily.

In addition to compression and decompression of motion video, we also use the DVI video subsystem for video mixing if there is more than one incoming video stream from the network (in the case of fully distributed video mixing scheme, to be discussed later). Derived from empirical measurements obtained for the ActionMedia II adapter, our DVI subsystem based on the 50-MHz i750 video processor should be able to compress a full-motion video frame in 16.66 ms (half the frame time at 30 frames/s) and decompress a video frame in 8.33 ms (a quarter of the frame time at 30 frames/s). We also assumed that the overhead of mixing multiple video frames into a composite frame would be 8.33 ms. These assumptions, in addition to the fact that our DVI subsystem has the 80C186 processor and uses single-stage pipelines for compression and decompression, lead to the term "hypothetical" used in this paper, since there is no such video adapter currently available.

One SCSI-2 I/O subsystem. This includes the IBM PS/2 SCSI-2 adapter, which can support both the Locate and Move Modes of the SCB architecture, a small computer system interface (SCSI) bus, and one SCSI disk. The adapter has an Intel 80C186 processor to support the SCB protocols, just as in our DVI video subsystem discussed previously.

One (hypothetical) 32-bit token-ring adapter. We assumed that the conference participants were connected via a LAN using the 16-Mbits/s token-ring network technology. There have been several token-ring adapters introduced in the past several years for the PS/2 systems. However, there is no token-ring adapter currently available that can support the SCB Move Mode. As a result, we had to make some assumptions about the architecture of this adapter. We assumed that the token-ring adapter had the same general architecture as our DVI video subsystem discussed previously in this paper. Like the DVI video subsystem, it would have an Intel 80C186 processor to handle the SCB Locate and Move Modes, as well as a general-purpose memory buffer to manipulate (assemble/disassemble) data for transmitting across the token ring. As such, it would have the same processing overhead as the DVI video subsystem in supporting the SCB protocol and handling the memory buffer. We also assumed in our study that the token ring was not saturated, so the token waiting time was negligible.

According to our analysis and others (Little and Ghafoor 1991), the main problem with the host processor in a video conferencing system is that the software code providing the conference services may not be scheduled within a strict timing constraint when required because the processor is busy performing other tasks. This is a typical problem in other real-time systems as well. The problem has more to do with processor scheduling than processor utilization. As a result, we did not consider background activities that could require substantial CPU processing and had nothing to do with video conferencing. An example of the type of background activities that we would like to avoid is the nonvideo-conferencing graphics operations that rely heavily on the host processor to perform most

of their work. However, we considered the background file transfers of video information. Conference participants might want to review an earlier segment of their conference or some video images stored on disk. In this case, the video information recorded earlier on disk must be retrieved and transferred across the network linking all participants. We assumed that the disk from which video information was retrieved resided locally within the system under consideration.

Our study focused on two types of systems: a video conferencing system with centralized video mixing, and a system with fully distributed video mixing. There are many techniques for mixing multiple video streams into a single composite stream: centralized mixing, fully distributed mixing, and hierarchical mixing (Vin et al. 1991; Yun and Messerschmitt 1993). In centralized mixing, each participant's computer sends a video stream to a central video mixer. The central mixer forms a composite video stream from all video streams coming from all participants. This mixer then sends this composite video stream to each participant for playback. Thus each computer must be able to support two separate video streams: one outgoing to the central mixer, and one composite stream coming from the mixer. However, fully distributed mixing requires that each participant uses the network's multicast capability to send its video stream to all other participants. It also requires that video streams from all other participants can be received, mixed together to form a composite stream, and played back on the monitor. This approach is the simplest and easiest to implement, but it does incur the duplication of processing power and network bandwidth to perform video mixing at each participant's station.

Video mixing requires some amount of image processing (Ramanathan et al. 1992; Vin et al. 1991). We assumed here that the mixing could be done by microcode running on the i750 pixel processor, and that the mixing overhead was the same as that of video decompression. Regardless of the number of video streams, there is always one image to be decompressed and displayed on the monitor of each conference participant.

In this study, each participant in a centralized mixing system should be able to handle three different data streams:

- An incoming (perhaps composite) video stream from the central mixer across the network,
- An outgoing stream to be sent across the network to the central mixer, and
- A continuous video file transfer stream at 30 frames/s.

However, if the video mixing is done at each participant's station as in a fully distributed mixing system, each participant must be able to support at least four data streams concurrently in a conference involving more than two participants. In the case of the *three-participant conference*, each computer must be able to support the following four data streams:

- Two incoming video streams from the other two participants,
- One outgoing video stream to be sent across the network to the other participants (perhaps using the network's multicast capability), and
- A continuous video file transfer stream at 30 frames/s.

All video streams are multiplexed while moving across the network. Within a participant's computer, the video streams also share the Micro Channel. In a fully distributed mixing system, multiple incoming streams from the network are parsed and then mixed together to form a composite stream by the microcode running on the Intel i750 pixel processor.

On some systems, the average size of the (compressed) video frames and the packet size used to transfer data across the LAN are not the same. Many frames can be combined into a larger data packet to transfer across the network. However, this tends to introduce additional time delays between video capture and playback on a remote system. These delays must be minimized. With a 45:1 video compression ratio and the RTV microcode, the average frame size is about 7–8 kB, which is also found to be the case in Crimmins (1993). The maximum packet size on the standard 16-Mbits/s token-ring network is 8 kb. As a result, each video frame will be sent on the network as a separate packet, so the time delays between video capture and playback should be minimal.

The Motion Picture Experts Group (MPEG) standard specifies that a video signal (and its associated audio) can be compressed to a bit rate of about 1.5 Mbits/s with acceptable quality (Le Gall 1991). For video services using the ISDN, the Consultative Committee on International Telegraphy and Telephony (CCITT) Recommendation H.261 states that a single standard, $p \times 64$ kbits/sec ($p = 1, 2, 3, \ldots, 30$), can cover the entire ISDN channel capacity (Liou 1991). With $p = 30$, the maximum bit rate as specified by the H.261 standard can be up to 2 Mbits/s. In this study, we assumed a maximum bit rate of 2 Mbits/s per video stream.

3 The Subsystem Control Block architecture in video conferencing

The Subsystem Control Block (SCB) architecture enhances the potential of intelligent I/O adapters (bus masters) by defining the logical protocols and control structures that are used to transfer command/control information, data, and status information between the host processor and an I/O adapter, or directly between I/O adapters themselves (peer-to-peer) (IBM 1991). The architecture provides command chaining, data chaining, signaling, and synchronization of commands and status information. It separates the delivery of control information and data to increase the system performance, raise the level of functional capability, and provide more design flexibility. This key feature was considered carefully in our study. The SCB architecture has two operating modes: the Locate Mode and the Move Mode.

3.1 The SCB Locate Mode – traditional approach

Most current I/O adapters used in personal systems today support the Locate Mode (or some variants) of the SCB architecture. It is a bus-master, interrupt-driven I/O protocol. In the Locate Mode, the control structure is a relatively fixed format structure, called the command control block. This structure allows the command, control, and status information, as well as

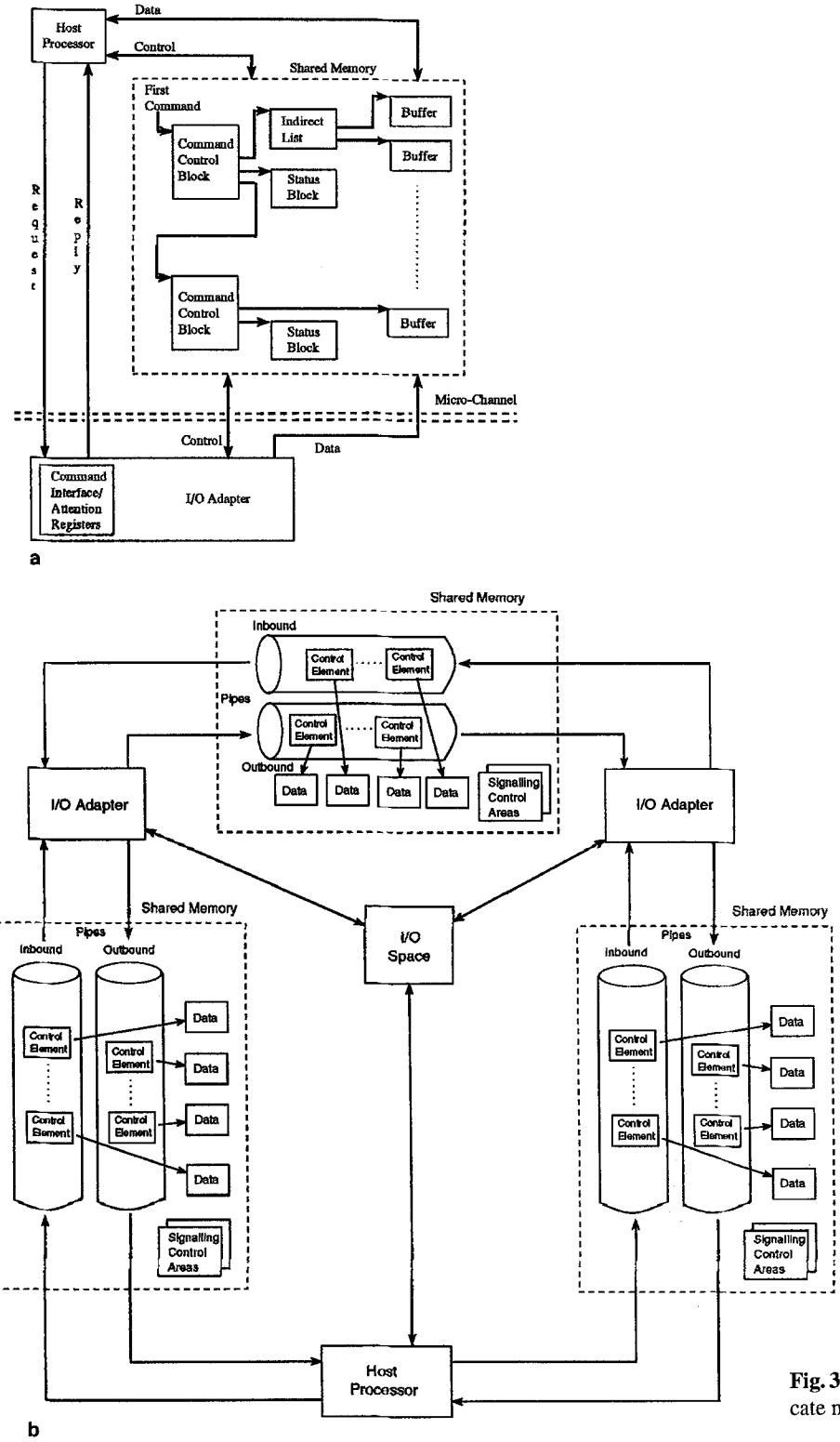


Fig. 3. Subsystem control block architecture: **a** Locate mode; **b** Move mode

pointers to data buffers in system memory, to be passed from the host processor to an I/O adapter (Fig. 3a).

In the Locate Mode, only the host processor can send requests to an I/O adapter. To send a request to an I/O adapter, the host processor first builds the control block in system memory to describe the I/O operation that needs to be performed (an I/O request). It then writes the physical memory address of the control block to the adapter's command interface registers, and a device identifier to the adapter's attention register, which interrupts the adapter's on-board processor. After being interrupted, the adapter's processor uses the memory address in its command interface registers to locate the control block in system memory and fetches it into its own storage area for execution. After the I/O request is completed, the adapter

interrupts the host processor, and supplies it with status information. The adapter's reply to the host processor must be synchronized with the request.

As an example, let us consider the case where a video frame must be sent to the token-ring network. First, the video frame is captured from the recording camera into the VRAM of the video adapter. The frame is then compressed by the i750 pixel processor on the adapter. After the frame is compressed, the video adapter sends an I/O interrupt to the host processor. The host processor formulates a control block in system memory and interrupts the video adapter, instructing it to transfer the compressed video frame into system memory. The 80C186 processor on the video adapter then fetches the control block from system memory, decodes it, and initiates a data transfer from the VRAM to some data buffer in system memory. The address of this buffer is specified in the control block. Once the compressed frame is in system memory, the video adapter sends the I/O-complete status back to the host processor. Upon receiving the status, the host processor formulates another control block in system memory and interrupts the token-ring adapter, asking it to get the compressed video frame from system memory. The token-ring adapter then fetches the control block from system memory, decodes it, and transfers the frame from system memory to the token-ring network. After the data transfer is completed, it sends the I/O-complete status back to the host processor, indicating that the request has been completed. As a result, both the host processor and system memory are involved in moving the compressed video frame between the VRAM on the video adapter and the token-ring network.

The operation of the Locate Mode is serial in that the host processor cannot send another request to the same I/O device (disk, LAN connection, etc.) until the current request has been completed. There can only be one request active per device at any given time, because requests to the same device cannot be "tagged". The host processor, however, can send requests to other devices through the same or another I/O adapter.

3.2 The SCB Move Mode – new approach

The Move Mode supports I/O data transfers by using shared memory interfaces to deliver requests and control-related information between two I/O adapters, or between the host processor and an I/O adapter, in the system (Fig. 3b). This provides true peer-to-peer relationship between all system components. The key feature in the Move Mode is that an I/O adapter can send requests to, and accept requests from, another I/O adapter, not just the host processor. In the Locate Mode, an I/O adapter can only receive requests from the host processor.

The Move Mode uses control elements instead of control blocks. The control elements are variable in length and can contain I/O requests, status, or error notifications. They can be used by the host processor or an intelligent I/O adapter to deliver requests or replies to another adapter. The control elements are moved between I/O adapter pairs, or between an adapter and a host processor, through a pair of delivery pipes. Each pipe behaves as a FIFO queue, and allows for the delivery of control elements in only one direction. Full duplex operation thus requires a pair of pipes. There is one pair of delivery pipes for each pair of I/O adapters (or for each adapter and the host processor) that want to communicate with each other. Fig. 3b shows three pairs of delivery pipes used by two I/O adapters and the host processor to communicate with one another. A delivery pipe must be in memory shareable by the sending and receiving adapters. This does not mean that the pipes must reside in system memory; they can be in an I/O adapter's memory buffers, provided that any other adapter or host processor that wants to communicate with it can access those memory buffers. In this study, we assumed that the delivery pipes between the I/O adapters were not in system memory. The pipes allow multiple control elements to be queued and processed asynchronously to each adapter or host processor. Unlike the Locate Mode, the Move Mode is not serial in nature: it can support the queuing of many requests to the same I/O device in the delivery pipes.

Let us consider the case of moving a video frame to the token-ring network. First, the video frame is captured and compressed in the VRAM of the video adapter. After the video frame is compressed by the i750 pixel processor, the 80C186 processor on the video adapter performs some processing on the compressed frame, such as attaching a header to the frame, and so on. It then builds a control element in VRAM, and sends it to the token-ring adapter through a delivery pipe. Once the token-ring adapter processes the control element extracted from the delivery pipe, it sets up a direct data transfer of the compressed video frame between the video adapter (VRAM) and itself. The video frame is eventually fed onto the token-ring network. The host processor and system memory are not involved at all in this data transfer. However, the host processor does have to send a control element to the video adapter at the start of each video stream (not frame) to initialize the video adapter and supply it with necessary information about the video stream. After that, the transfer of frames in the video stream is handled completely by the video and token-ring adapters.

4 Modeling methodology

In our work, we used the IBM Research Queuing Package (RESQ) (MacNair and Sauer 1985; Sauer and MacNair 1979, 1984). We have decided to use the simulation approach mainly because the SCB architecture and the system hardware modeled here are highly sophisticated, and thus, analytical solutions could not be easily obtained. In addition, the models need some capabilities that are only provided with simulation (not analytical modeling) in RESQ, such as the capability to make routing decisions based on the status of simulation conditions as well as probabilities, etc.

4.1 Description of models

The focus of our work was the construction of simulation models for the Locate Mode and Move Mode of the SCB architecture. The models were written in the RESQ language. Each

model is a set of open chain submodels, simulating multiple concurrent video data streams and file transfers. Depending on the particular video conferencing system that we want to simulate, we can enable or disable each of the submodels individually. Deterministic interarrival times are used for frame rates, and exponential distribution with a mean value of 8 kB is used for frame sizes. For a more detailed description of the models, refer to (Huynh and Khoshgoftaar 1993).

The access contention for the system shared memory and the Micro Channel is modeled by representing these components as "active" service centers, rather than "passive" resources, in RESQ. The models have separate memory queues for the memory traffic initiated by the host processor and the adapters on the Micro Channel, reflecting the dualpath memory controller design. Arbitration on the Micro Channel is modeled with the processor-sharing discipline in RESQ.

In the Move Mode model, the delivery pipe from a source adapter to a destination adapter is modeled as a queue in front of the destination adapter. Each queue is dedicated to the control elements sent from each particular source adapter; each source adapter has its own queues to the same destination adapter, as specified by the SCB architecture. The pipe is in the destination adapter's memory. The SCB Move Mode also recommends that all data transfers between two adapters are made between the memories on those adapters. In the case of the video adapter and the token-ring adapter, for instance, the data transfers are made between the VRAM and the token-ring adapter's memory buffers. The system memory is not used in these data transfers.

In the Locate Mode model, the queues in front of service centers representing adapters are not used (that is, they all have the queue length of 1), since requests (control blocks) are queued at the device-driver (operating system) level running on the host processor, not at individual adapters. This is to reflect the fact that there is no delivery pipe in the Locate Mode. Tokens are used in the Locate Mode model to ensure that the host processor cannot send another request to a device (and its adapter) before the current request to the same device is completed. There is no such token used in the Move Mode since control elements are queued in the delivery pipes. However, some tokens are used in both models to calculate the time delays in RESQ.

4.2 Confidence interval methods

For confidence interval estimation, we used the independent replication method. Following are the specific parameters that we specified for the simulation runs of the models:

- Number of replications: 5.
- Confidence interval: 90%.
- Simulation length: 2000 frames (over 1 s of full motion video).

We used the same simulation parameters for simulation runs of both models, so the results can be directly compared to one another.

4.3 Model validation

We have validated our Locate Mode models by empirical performance measurements collected on real hardware. The file transfer submodel's results are comparable to those obtained on the real PS/2 systems under similar workload conditions. For the video conferencing workload, we have also obtained empirical performance data for an existing person-to-person (P2P1) software package running on the IBM ActionMedia II card. The P2P software is the basic video conferencing application between two persons. The ActionMedia II card uses the RTV 2.0 microcode to compress and decompress full-motion video at 15 frames/s. To compare our Locate Mode model's results against the real P2P measurements on ActionMedia II, we modified our Locate Mode model to use the five-stage compression pipeline and the three-stage decompression pipeline. The compression pipeline runs at the frame rate (15 frames/s), but the decompression pipeline is run at twice the frame rate (30 frames/s). In this case, the model's results are very similar to those obtained on the real P2P software.

There is no existing hardware that can support the SCB Move Mode, so it is much harder to validate our Move Mode model directly. However, the same modeling techniques and many modeling assumptions used in the Locate Mode model are applied to the Move Mode model, so we are very confident about the Move Mode model.

5 Results

The models require many input parameters relating to the functional and performance characteristics of the host processor, system memory, Micro Channel, adapters, disks, video processing hardware, and the token-ring network. These parameters were obtained empirically on actual hardware available commercially or under development. The software overhead, such as the path lengths for the file system, network support code, and video conferencing software, were measured using the DEKKO software analysis tool running on the IBM Operating System/2. For more details on the input parameters, please refer to Table 1.

Table 1. General assumptions (input parameters for models)

Compression ratio = 45:1 = 360 kB:8 kB
Bit rate = 2.0 Mbits/s.
Token-ring packet size = 8 kB
File transfer (background) size = 4 kB
No audio (10% of processing bandwidth) considered
– Compression = 50% cycle, one-stage pipeline
– Decompression = 25% cycle, one-stage pipeline
– Video mixing = 25% cycle
Software overhead:
– File system = 5 KLOCS/request.
– Network support = 5 KLOCS/request.
– Initial code (Move Mode) = 5 KLOCS/stream.
– Video conferencing = 15 KLOCS/frame (Locate)
1 KLOC/frame (Move)

5.1 Micro Channel utilization

It is more advantageous to reduce the Micro Channel utilization, since the channel is the main data path connecting the host processor, system memory, adapters, and other I/O devices. Fig. 4a shows that the Move Mode uses the Micro Channel much less than the Locate Mode in both 386 and 486 systems. The data indicate that the Move Mode demands 50% less bandwidth from the Micro Channel for the same video conferencing workload running at the same frame rate on the same hardware, thus giving the channel higher data transfer capacity. The data also show that the Micro Channel is not the system bottleneck: its utilization never exceeds 20% in our study. The Micro Channel's 40-MB/s data streaming mode supported in the 486 system exhibits a threefold improvement in the channel utilization and hence, its data transfer capacity, over the normal data transfer mode used in the 386 system.

5.2 Host processor utilization

One of the key system considerations in video conferencing applications is the use of the host processor. In a three-way conference with fully distributed video mixing scheme, Fig. 5b shows that the i386 microprocessor in the midrange PS/2 Model 70 would be nearly 100% utilized if the video conferencing software (to be executed every frame) exceeded 60 kilolines of assembly code (60 KLOCS) in the Locate Mode, with the video file transfers running in the background. On the 486 system (PS/2 50-MHz Server), the host processor utilization approaches 100% if the video conferencing software is approximately 240 KLOCS in the Locate Mode. Based on empirical measurements obtained for the existing P2P software, the path length for video conferencing software is only 15 KLOCS/frame in the Locate Mode, so the host processor is not likely to be a system bottleneck.

In the Move Mode, the CPU utilization is less than 0.01% in all cases. This shows that, at least in terms of CPU utilization, the Move Mode is superior to the Locate Mode for the i386-based systems, since the host processor does not get involved at all in the Move Mode, except for some initialization at the beginning. Much of the work that the host processor must do in the Locate Mode is done by the adapters in the Move Mode. As the data in Fig. 5 shows, the receive/decompress time delay increases dramatically if the video conferencing software running per frame on the Intel 80C186 processor on the video adapter exceeds 16 KLOCS. However, this can be improved by having a faster processor on the video adapter. Furthermore, the software path length for video conferencing that the processor on the video adapter must execute per frame is estimated to be only 1 KLOC in the Move Mode.

Across various frame rates, but in the same video conferencing environment, the host processor utilization does not change in a linear fashion in the Locate Mode, according to Fig. 6a and 6b. This is due to the impact of the video file transfers running at 30 frames/s in the background. The impact of these background video file transfers on the host processor utilization is shown in Fig. 7a. Given the fact that the video file transfers are running at 30 frames/s, their impact on the host processor utilization is much larger at lower frame rates. At 8 frames/s, background video file transfers account for almost 50% of the total host processor utilization, while at 30 frames/s, they only account for roughly 20%.

Comparing the data of Fig. 6a and 6b, it can be observed that the centralized video mixing scheme uses much fewer system resources than the fully distributed mixing scheme. This is quite expected.

5.3 Utilization of other system components

Since memory access speeds are dependent on the host processor's cycle time, the data in system memory in the 486 system can be accessed much faster than in the 386 system. This is reflected in the data shown in Fig. 7b. At 30 frames/s, the effective memory utilization approaches 35% in the 386 system, as compared to only 10% in the 486 system (with its burst mode). As expected, the higher the frame rate, the higher the effective memory utilization is. This is true for the Locate Mode only. The Move Mode does not involve system memory in data transfers, so the system memory use is near zero in the Move Mode. The data in Fig. 7b also indicate that the disk utilization by the background video file transfers is 24%.

Fig. 7b also shows that a three-way video conferencing environment, with concurrent video file transfers and fully distributed mixing scheme, can use up to 53% of the token ring's bandwidth at 30 frames/s and 20% at 8 frames/s. This assumes the compressed frame size of 8 kB (about 45:1 compression ratio), and a maximum bit rate of 2 Mbits/s. The 16-Mbits/s token ring can indeed handle multiple, concurrent compressed video data streams.

5.4 Video compress/send time delay

The video compress/send time delay refers to the time interval from the moment a video frame is captured in VRAM until it is sent on the token-ring network to other conference participants. In a centralized video mixing system, there is no significant difference between the Locate Mode and the Move Mode in this delay (Fig. 8a). This is because most of the time is spent in compressing the video frames, even though we assumed single-stage compression pipeline. Since the pipeline runs at the frame rate, the lower the frame rate, the longer it would take to compress a video frame, and thus, the longer the compress/send delay would be. As a result, the slopes of the curves are negative as the frame rate increases. The performance of the host processor, system memory, and the Micro Channel does not seem to have much effect. Consequently, video compression algorithms are the key factor in the overall time delay in a video conferencing system.

In a fully distributed video mixing system, the system resource requirements, and thus resource contention, are higher than in a centralized video mixing system. Consequently, the Move Mode, with its lower resource utilization, can deliver better performance than the Locate Mode in a fully distributed

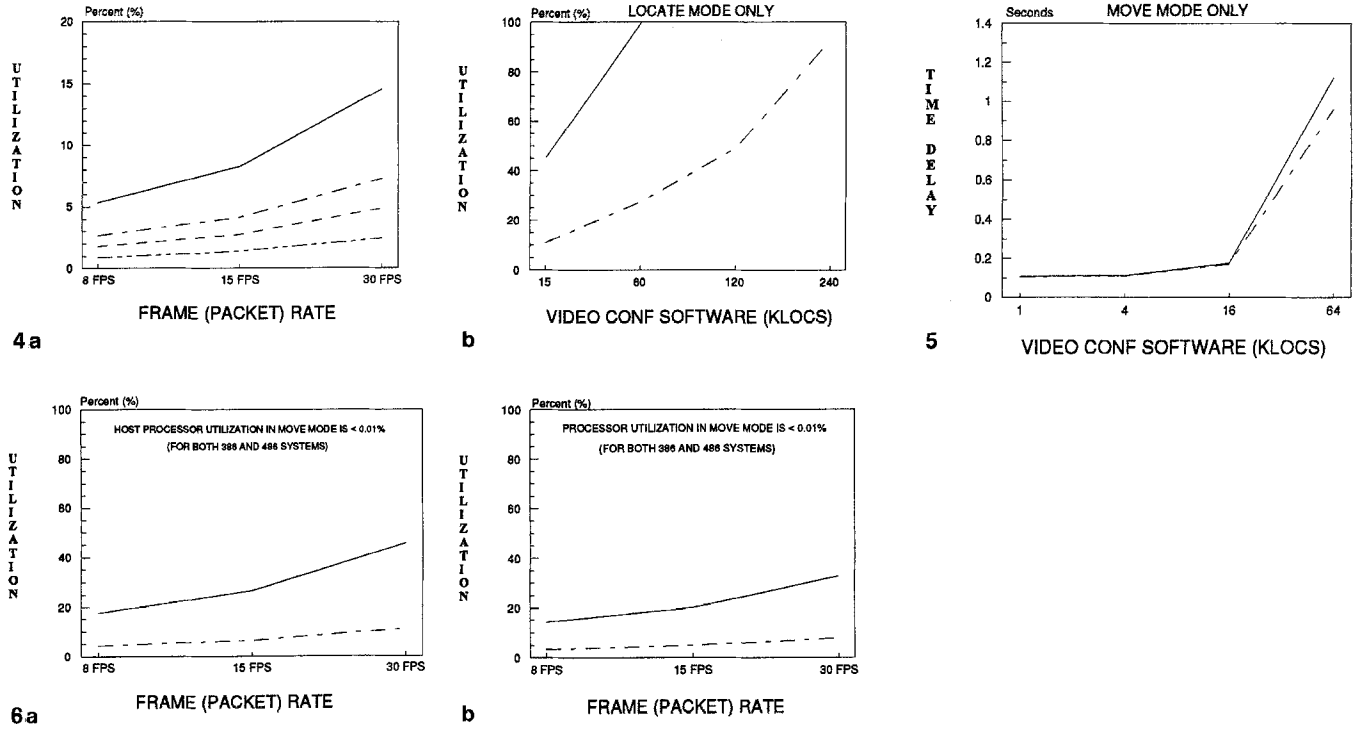


Fig. 4. Fully distributed video mixing. **a** Micro Channel utilization. **b** host processor utilization. Frame rate = 30 frames/s. —, Locate Mode (386 system); – -, Locate Mode (486 system); - - -, Move Mode (386 system); – - -, Move Mode (486 system)

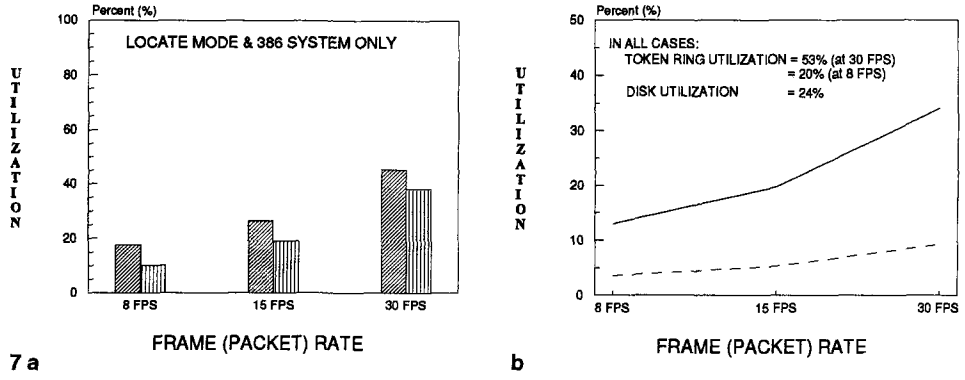
Fig. 5. Video receive/decompress delay (Fully distributed video mixing). —, 386 system; – -, 486 system

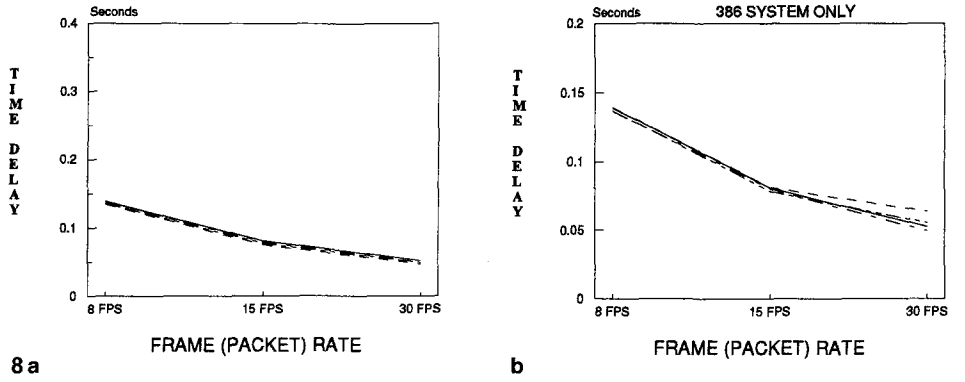
Fig. 6. Host processor utilization. (Video conferencing software is 15 KLOCS in Locate Mode and 1 KLOC in Move Mode). —, Locate Mode (386 system); – -, Locate Mode (486 system); **a** fully distributed video mixing; **b** centralized video mixing

video mixing system. As can be observed in Fig. 8b, the compress/send delay is essentially the same for both the Locate Mode and the Move Mode at 8 frames/s and 15 frames/s. However, as we approach 30 frames/s, the resource contention becomes much higher, so the Move Mode with its direct peer-to-peer data transfer protocol results in lower time delays. This is because the utilization of system resources, such as the host processor, memory, Micro Channel, etc., is lower in the Move Mode than in the Locate Mode. It can also be observed that, with one additional video stream and the video mixing overhead required in a fully distributed mixing system, the time delays are higher than those of the centralized mixing system at 30 frames/s. At lower frame rates (8 frames/s and 15 frames/s), there is really no difference in the video compress/send time delays, regardless of the SCB modes and video mixing schemes. It is reasonable to expect that, as more participants are added to a video conferencing system, the time delay differences between the Locate Mode and Move Mode, and between centralized and fully distributed video mixing schemes, will be more apparent.

5.5 Video receive/decompress time delay

The video receive/decompress time delay refers to the time interval from the moment a video frame is read from the token-ring network until the moment it is ready to be displayed on the computer's monitor. Although the video decompression usually requires only 50% of the processing required for video compression, its impact on the receive/decompress delay is still great. This is because both compression and decompression use single-stage pipelines, which all run at the frame rate. As a result, in a video conferencing system using the centralized video mixing scheme, there is no significant difference in this type of delay between the Locate Mode and the Move Mode, or between the 386 and 486 systems (Fig. 9a). However, it can be observed that the Locate Mode on the 386 system, with its slower host processor and Micro Channel, is slightly worse than the Move Mode. Fig. 9b examines the receive/decompress delay more carefully on the 386 system. The data shows that the Move Mode performs better than the Locate Mode at all frame rates. This is true for both centralized and fully distributed mixing schemes. So on a slower system, the direct peer-to-peer transfer protocol offered by the Move Mode delivers better performance than the more traditional I/O approach used in the Locate Mode. Figure 9b also indicates that the delays in a centralized video mixing system are slightly lower than those in a fully distributed video mixing system. This is quite expected because a system with fully distributed video mixing must handle more concurrent video streams than a system with centralized video mixing. In addition, in the fully distributed mixing system, the video mixing must be performed at each participant's location while this task is performed by a central video mixer in the centralized video mixing system. At 30 frames/s, the time delay difference between the centralized and fully distributed video mixing systems is larger than at lower frame rates. This is be-





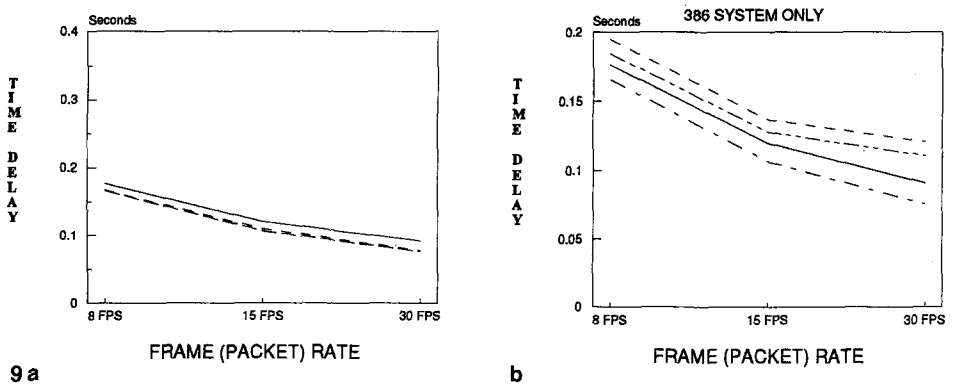


Fig. 7. Fully distributed video mixing. **a** Host processor utilization. ▨, with file transfers; ▥, with no files transfers; **b** host memory utilization. —, Locate Mode (386 system); - -, Locate Mode (486 system)

Fig. 8 Video compress/send delay. Video conferencing software is 15 KLOCS in Locate Mode and 1 KLOC in Move Mode. **a** Centralized video mixing. —, Locate (386 system); - -, Locate (486 system); – -, Move (386 system); – - -, Move (486 system); **b** —, Locate (centralized); - -, Locate (distributed); – -, Move (centralized); – - -, Move (distributed)

Fig. 9 Video receive/decompress delay. Video conferencing software is 15 KLOCS in Locate Mode and 1 KLOC in Move Mode. **a** Centralized video mixing. —, Locate (386 system); - -, Locate Mode (486 system); – -, Move Mode (386 system); – - -, Move Mode (486 system); **b** —, Locate (centralized); - -, Locate (distributed); – -, Move (centralized); – - -, Move (distributed)

cause the system must handle more video frames per second at 30 frames/s, and the need to handle more concurrent video streams in the fully distributed video mixing system increases the resource contention, and thus, the time delays.

As in the case of the video compress/send time delays, the lower the frame rate, the longer it would take to decompress a video frame, and thus, the longer the receive/decompress time delay would be. This is because the decompression pipeline runs at the frame rate. For instance, at 8 frames/s, the time delay at the single-stage decompression pipeline is 125 ms (without any queuing at the pipeline), while the same type of delay is only 33.33 ms at 30 frames/s.

Figure 10 shows the time delays for a single video stream, either outgoing or incoming. The data shows that the compress/send overhead is actually smaller than the receive/decompress overhead, even though the compression overhead takes more processing power from the pixel processor. There are several reasons:

– The overall time delays for video compression and decompression are essentially the same because both use single-

stage pipelines (even though video compression takes more processing from the pixel processor than video decompression). The 50-MHz i750 pixel processor is able to handle both compression and decompression within a frame time.

– The video compression process performed on the video frames shortly after their capture in VRAM has minimum queuing. This is due to the fact that video frames are captured at precisely the frame rate, and so as long as the compression process does not take longer than the frame time, there is no queuing at the compression stage. However, the video decompression process is not started until after the frames have been received from the network, manipulated, and mixed. Many variable factors associated with the delays at the host processor, memory, Micro Channel, and other system components, cause the video frames to arrive at the decompression pipeline at irregular intervals, not precisely at the frame rate. This leads to a higher queuing level at the decompression pipeline, and consequently, higher time delays.

So far we have seen no significant difference between the Move Mode and the Locate Mode in the time delays. However, the Move Mode is much better in handling video file transfer operations (Fig. 10). This is because the video file transfer operations involve two slow subsystems, the disk subsystem and the token-ring network. In the Locate Mode, each data transfer must travel from the source subsystem to system memory, and then, from system memory to the destination subsystem; thus, from the standpoint of a single data transfer, the two subsystems operate serially, not in parallel. On the other hand, the Move Mode, with its direct peer-to-peer transfer protocol, allows these two slow subsystems to operate in parallel (one sending and the other one receiving data), and thus, significantly cuts down the overall time delays.

However, a video conferencing data stream involves only one slow subsystem, and that is the token-ring network. The video adapter's VRAM is very fast, and data transfers between the VRAM and host memory can be performed relatively quickly. Therefore, not having the data transfers to and from host memory in the Move Mode does not result in a significant performance improvement over the Locate Mode. In general, the SCB Move Mode works best when the data transfers involve two slow adapters.

5.6 Breakdown of the time delays

In this section, we will look at the video compress/send and video receive/decompress time delays in more detail. Figure 11a–d shows the time delays for a system using fully distributed video mixing. Figure 12a–d illustrate the same time delays for a system using centralized video mixing.

In Fig. 11a, only the delays in the Locate Mode are shown. The two vertical bars on the left represent the video compress/send delay and video receive/decompress delay on the 386 system. The two vertical bars on the right show the same delays obtained on a 486 system. It can be observed that the time delays, especially the video receive/decompress delay, are lower on the 486 system. Since there is little or no queuing at the compression pipeline (as discussed in the preceding section), and the compression is performed by microcode running on the i750 pixel processor on the video adapter, the compression overhead is the same for both the 386 and 486 systems. The video conferencing software overhead, which runs on the host processor in the Locate Mode, is smaller on the 486 system, as expected. Because of the i486's higher processing power, there is less queuing at the decompression pipeline, so the decompression overhead is much smaller on the 486 system. The queuing time in the operating system's network device driver in the compress/send delay is larger than the queuing time in the operating system's SCSI disk device driver in the receive/decompress delay. This is because the throughput of the 16-Mbits/s token-ring network is smaller than that of the SCSI disk I/O subsystem. Overall, the receive/decompress time delays are higher than the compress/send delays because (1) there is more queuing at the decompression pipeline than at the compression pipeline (as discussed in the preceding section), and (2) there is the video mixing overhead in the receive/decompress delays. There is no need for video mixing in the compress/send delays because each participant needs only to send his or her own video out to the network.

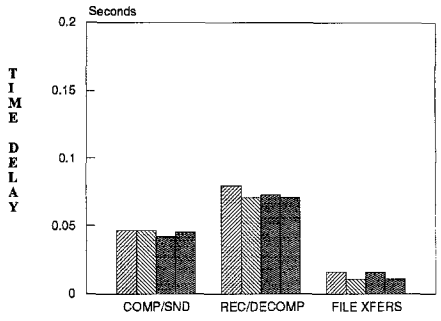


Fig. 10. General time delays, single data stream. Video conferencing software is 15 KLOCS in Locate Mode and 1 KLOC in Move Mode. Maximum bit rate = 2 Mbits/s, frame size (compressed) = 8 kB (mean), frame rate = 30 frames/s. ▨, Locate Mode – 386 system; ▧, Move Mode – 386 system; ▩, Locate Mode – 486 system; ■, Move Mode – 486 system

Figure 11b shows the time delays in the Move Mode only. One interesting observation is that the time delays on the 386 and 486 systems are approximately the same. This is because the host processor does not have to do much work in the Move Mode. There is no queuing in the operating system's device drivers because the I/O commands are queued in the delivery pipes between the adapters in the Move Mode.

Figure 11c and 11d compare the time delays between the Locate Mode and the Move Mode on the 386 and 486 systems, respectively. The time delays in the Move Mode are smaller than those in the Locate Mode, due in part to the smaller video conferencing and network support overhead in the Move Mode. The time delays for video compression and decompression are about the same for the two SCB modes, which is quite expected because these operations are performed by microcode running on the video adapter, not by the host processor. The video conferencing and network software overhead is smaller in the Move Mode because the 80C186 pro-

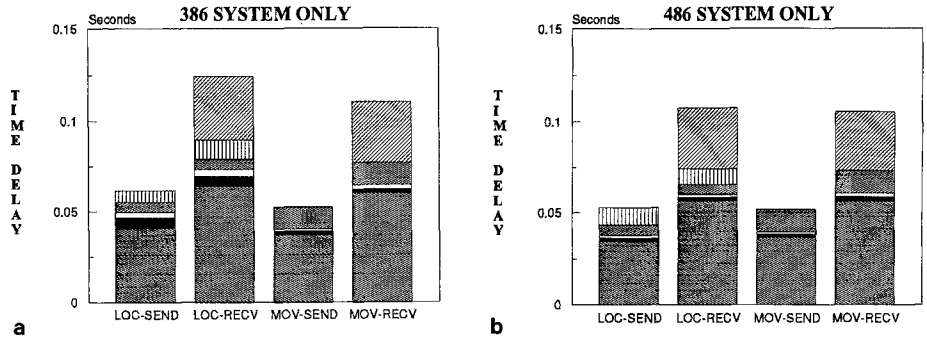


Fig. 11a–d. Time delay components (fully distributed video mixing). Video conferencing software is 15 KLOCS in Locate Mode and 1 KLOC in Move Mode. Frame rate = 30 frames/s, frame size (compressed) = 8 kB (mean), stream length = 2000 frames (packets). ▨, video mixing; ▥, operating system queuing; ▩, data transfers; □, network software; ■, video conferencing software; ▩, compress/decompress: **a** 386 system only; **b** 486 system only; **c** Locate Mode only; **d** Move Mode only

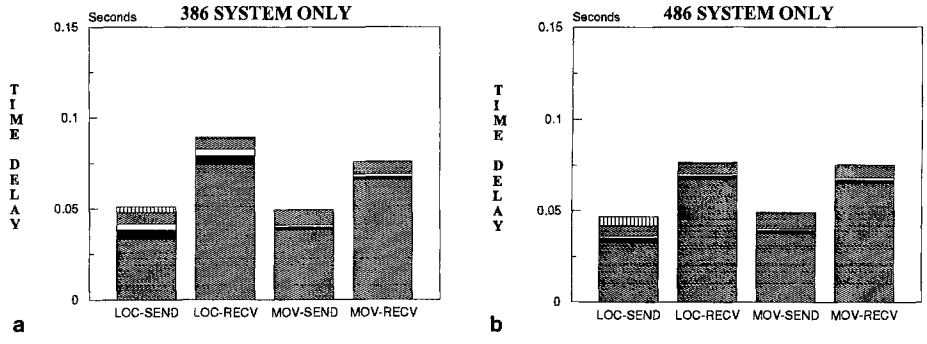


Fig. 12a–d. Time delay components (centralized video mixing). Video conferencing software is 15 KLOCS in Locate Mode and 1 KLOC in Move Mode. Frame rate = 30 frames/s, frame size (compessed) = 8 kB (mean), stream length = 2000 frames (packets). ▨, video mixing; ▥, operating system queuing; ▩, data transfers; □, network software; ■, video conferencing software; ▩, compress/decompress: **a** 386 system only; **b** 486 system only; **c** Locate Mode only; **d** Move Mode only

cessor on the video adapter executes less code for each frame in the Move Mode than the host processor executes in the Locate Mode. This is true even though the 80C186 processor is slower than the i386 or i486 host processor. The overhead of data transfers in the Move Mode is greater than that in the Locate Mode because it includes the queuing time in the delivery pipes between the video adapter and the token-ring adapter in the Move Mode. There is no delivery pipe in the Locate Mode, but the I/O requests are queued inside the operating system's device drivers; the queuing overhead in the device drivers are shown separately from the data transfer overhead in the Locate Mode. However, it is clear from the figures that the sum of the data transfer overhead and the queuing time in the device drivers in the Locate Mode is greater than the sum of the data transfer overhead and the queuing time in the delivery pipes in the Move Mode. Note again that the video mixing overhead only appears in the video receive/decompress time delays.

Figure 12a–d shows the time delays for a video conferencing system using centralized video mixing. There is no video mixing overhead in the time delays because the video mix-

ing is performed by the central mixer located somewhere on the network, so the video receive/decompress time delays are lower than those in the fully distributed video mixing system. The video compress/send time delays are approximately the same as in the fully distributed video mixing system. Except for the video mixing overhead, we can make the same general observations from this figure as from Fig. 11a–d.

6 Conclusion

In this paper, we provided an overview of the IBM PS/2 and SCB architectures. The SCB architecture is designed to efficiently manage the communication and data transfers between the processor, intelligent adapters, and the Micro Channel. This architecture has two operating modes, the Locate Mode and the Move Mode. The Locate Mode embodies the more conventional I/O processing model, while the Move Mode offers a true peer-to-peer I/O protocol between intelligent adapters on the Micro Channel. Simulation models of the PS/2 systems were constructed to determine how effective midrange and high-end PS/2 systems can support one of the most I/O-intensive multimedia applications today, the video conferencing environment. From the modeling data, the peer-to-peer I/O protocol in the Move Mode uses much fewer system resources (host processor, memory, Micro Channel) than the Locate Mode. This increases the overall data processing capacity of the system. This suggests that, on a slow system, the direct peer-to-peer transfer protocol offered by the Move Mode delivers better performance than the more traditional I/O approach used in the Locate Mode. In the Locate Mode, the 25-MHz i386 processor in a midrange PS/2 system would be 100% used if the video conferencing software running on the host processor exceeded 60000 assembly instructions (60 KLOCS)/frame. Even with the peer-to-peer I/O protocol in the Move Mode, the end-to-end time delay between the capture and playback systems would increase dramatically if the video conferencing microcode running on the Intel 80C186 processor on the video adapter exceeded 16 KLOCS/frame. This can be improved by having a faster processor on the video adapter card. However, based on empirical measurements obtained for the existing P2P software (basic video conferencing between two participants), the path length for video conferencing software is only about 15 KLOCS/frame in the Locate Mode. In the Move Mode, this path length is estimated to be approximately 1 KLOC.

The Micro Channel utilization never exceeds 20% in our study, so the Micro Channel is not really the system bottleneck. With a video compression ratio of about 45:1, and a maximum bit rate of 2 Mbits/s per video stream, the token ring is 53% utilized in a three-way video conference with fully distributed video mixing and background file transfers.

Since, in our study, the video compression and decompression pipelines run at the frame rate, the higher the frame rate, the lower the time delays would be. We also found that the key to minimize the video compress/send and receive/decompress time delays is to optimize the video compression and decompression (video codec) algorithms. Nothing else matters much. We feel that significant research should be dedicated to the development of fast and efficient algorithms for full-motion video codec. Unlike other multimedia applications, video conferencing environments require real-time video compression. There are currently three standard video codec techniques: JPEG for full-color, still-frame applications, MPEG for motion-intensive applications, and px64 standard (CCITT Recommendation H.261) for video-based, real-time telecommunications. These standards combine several approaches for compressing and decompressing video, such as the discrete cosine transform, vector quantization, and differential pulse code modulation. In order to maximize the performance of these algorithms and minimize costs, we need to determine which aspects of these algorithims should be implemented in hardware and which aspects should be done in software. The implementation of video codec algorithms can generally be classified into three different categories:

– The hardware approach: the video codec algorithms are implemented in specilized hardware to maximize performance (an example is C cube).

– The software approach: the video codec algorithms are completely implemented in software to be run on the system (host) processor. An example of this approach is the MPEG encoder for X Window systems (Patel et al. 1993). The encoder algorithms are implemented in C and are executed on the system processor.

– The hybrid approach: the video codec algorithms are implemented on microcoded, programmable processors. An example of this approach is the Intel i750 with the RTV microcode discussed previously in this paper.

The hardware approach tends to provide better video codec performance at the expense of flexibility in implementing different algorithms. On the other hand, the software approach emphasizes flexibility over performance. However, with the tremendous progress made in the performance of general-purpose processors, the software approach now delivers respectable performance. Although the hybrid approach achieves some commercial success with the Intel i750 and other programmable processors, pure software implementations of video codec will be significantly improved with the introduction of RISC and multiprocessor technologies to personal computers.

At the high rate of 30 frames/s, with fully distributed video mixing scheme, the Move Mode does perform better than the Locate Mode. The Move Mode is more effective if the data transfers involve two slow subsystems, such as the disk subsystem and the token-ring network. In this case, the direct peer-to-peer data transfer capability in the Move Mode is much faster than the two-part transfer (from the source adapter to host memory *and* from host memory to the destination adapter) required by the conventional I/O protocol in the Locate Mode. Consequently, the Move Mode seems to handle video file transfer operations much better than in the case where video data streams are moved between fast VRAM on the video adapter and a slow token ring in a video conferencing environment.

As expected, the time delays on the 486 system are lower than those on the 386 system in the Locate Mode. However, in the Move Mode, the time delays on the 386 and 486 systems are approximately the same. This is because the host processor does not have to do much work in the Move Mode. On the same hardware, the receive/decompress time delays are higher than the compress/send delays because (1) there is more queuing at the decompression pipeline than at the compression pipeline, and (2) there is the video mixing overhead in the receive/decompress delays. There is no need for video mixing in the compress/send delays because each participant only needs to send his or her own video out to the network.

The time delays in the Move Mode are smaller than those in the Locate Mode, due in part to the smaller video conferencing and network support overhead in the Move Mode. In addition, the sum of the data transfer overhead and the queuing time in the device drivers in the Locate Mode is more than the sum of the data transfer overhead and the queuing time in the delivery pipes in the Move Mode.

There is no video mixing overhead in a centralized video mixing system because the video mixing is performed by the central mixer located somewhere on the network, so the video receive/decompress time delays are lower than those in the fully distributed video mixing system. In addition, a participant in a fully distributed video mixing scheme must handle more concurrent video streams than a participant in a system with centralized video mixing. The video compress/send time delays in this type of systems are slightly less than those in the fully distributed video mixing system.

Acknowledgements. The authors thank Richard Mendelson, Ralph Pipitone, and John Sierra of IBM Corporation, and Dr. Neal Coulter of Florida Atlantic University for their wonderful support. Their input and constant encouragement have made a significant difference in the progress of this work. Special thanks are also due to Ranga Anumulapally of Florida Atlantic University for his active participation in this project.

References

Crimmins S (1993) Analysis of video conferencing on a token ring local area network. Proceedings of the First ACM International Conference on Multimedia, Anaheim, Calif., pp 301–310

Fletcher JD (1990) Effectiveness and cost of interactive videodisc instruction in defense training and education. Institute for defense analysis paper P-2372, Washington

Gibbs S, Tsichritzis D, Fitas A, Konstantas D, and Yeorgaroudakis Y (1987) Muse: a multi-media filing system. IEEE Software 4: 4–12

Harney K, Keith M, Lavelle G, Ryan L, Stark D (1991) The i750 video processor: a total multimedia solution. Commun ACM 34: 65–78

Huynh K, Khoshgoftaar T (1993) A Performance Analysis of Video Conferencing On Personal Computer Systems. Technical Report TR-CSE-93-26, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Fla

IBM (1991) Subsystem Control Block I/O Architecture Technical Reference, 1st edn. IBM, IEEE Computer Society, Los Alamitos, CA

Isaacs E, Tang J (1993) What video can and can't do for collaboration: a case study. Proceedings of the First ACM International Conference on Multimedia, Anaheim, Calif., pp 199–206

Le Gall D (1991) MPEG: a video compression standard for multimedia applications. Commun ACM 34: 47–58

Liou M (1991) Overview of the px64 kbit/sec Video Coding Standard. Commun ACM 34: 59–63

Little T, Ghafoor A (1990) Network considerations for distributed multimedia object composition and communication. IEEE Network 4: 32–49

Little T, Ghafoor A (1991) Multimedia synchronization protocols for broadband integrated services. IEEE J Selected Areas Commun 9: 1368–1382

MacNair E, Sauer C (1985) Elements of practical performance modeling. Prentice-Hall, N.J., pp 46–80

Patel K, Smith BC, Rowe LA (1993) Performance of a software MPEG video decoder. Proceedings of the First ACM International Conference on Multimedia, Anaheim, Calif., pp 75–82

Ramanathan S, Rangan P, Vin H, Kaeppner T (1992) Optimal communication architectures for multimedia conferencing in distributed systems. Proceedings of the 12th International Conference on Distributed Computing Systems, pp 46–53

Sauer C, MacNair E (1979) Queueing network software for systems modeling. Software – Practice and Experience 9: 369–380

Sauer C, MacNair E (1984) The evolution of the research queueing package. Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis, pp 5–24

Vin H, Rangan P, Ramanathan S (1991) Hierarchical conferencing architectures for inter-group multimedia collaboration, p 17

Yun L, Messerschmitt D (1993) Architectures for multi-source multi-user video compositing. Technical report number CS91–210, Department of Computer Science and Engineering, University of California at San Diego, September 1991

For photographs and biographies of the authors see Volume 2, Number 1, 1994, p 50.