♦ An Overview of the MultiMedia Communications eXchange (MMCX) and Its Performance Characterization

Anwar I. Elwalid, Glen G. Freundlich, Paul M. Gerhardt, Hassan Hagirahim, K. G. Ramakrishnan, and David Tse

MultiMedia Communications eXchange (MMCX), a new offering from Lucent Technologies, supports mixed media calling between end points on a local area network (LAN), such as Windows-based personal computers and any H.323 standards-compliant end point (for example, an Internet phone). The MMCX client/server architecture also provides access to the wide area network (WAN), allowing LAN end points to call users on other networks. The MMCX enables users to communicate and collaborate in a manner that best suits their needs by supporting multiparty conferencing with any type and combination of mixed media. MMCX calls can consist of any mix of voice, data, image, and video media types, with at least one of the calling/called parties directly connected to a packet transmission and/or switched network. A party to the call may be using one or more communications media, but it is not necessary for all parties on a multiparty conference call to be using the same media mix. For example, three parties may be communicating visually using LAN-connected desktop computers equipped with a videoconferencing option (audio and video), and a fourth party may be conferenced via a telephone, but he or she may only be able to speak and listen to the other parties on the call. To provide high-quality real-time mixed media collaboration in a LAN and WAN environment, important performance requirements for packet delays (such as call post-dialing delays) and media blocking must be met. This paper presents an overview of the MMCX and its performance analysis. A design tool based on our analysis has been developed to run in a PC/Windows-based environment that can be used interactively to design and reconfigure MMCX-based networks.*

Introduction

Intranet/Extranet/Internet networks are emerging throughout the enterprise. Users expect these networks to deliver real-time multipoint collaboration and conferencing that support mixed media applications. Successful implementation of this new environment depends on the following criteria:

- *Universal connectivity*. Customers of mixed media applications, including users of a telephone or a legacy integrated services digital network (ISDN) mixed media end point, need the ability to call anyone and bring them into a collaborative conferencing session in real time without worrying about matching protocols or media encodings. The MultiMedia Communications eXchange (MMCX) allows users to build collaborative sessions in real time by conferencing in work group members on MMCX end points, H.323[1] standards-

compliant end points, H.320[2] ISDN end points, and wired or wireless telephones.

- *Quality of experience.* One focus of MMCX is to make mixed media calling easy and intuitive for users. The MMCX platform brings mixed media to the desktop, while making it as easy to use as making a phone call. MMCX also provides Lucent Technologies' high-quality connections.
- *Managed environment.* The MMCX platform ensures that system or network managers can control and manage their resources. For example, wide area network (WAN) bandwidth management is inherent, as are techniques such as media prioritization and multicasting. MMCX offers a remote maintenance board, which automatically relays any system alarms and permits Lucent service personnel to diagnose and resolve troubles from any Lucent location.
- *Outstanding service.* Lucent offers a full range of services that deliver end-to-end support and a single point of contact for the design, integration, maintenance, and management of MMCX. Lucent's NetCare® professional and consulting services also provide specialized support tailored to customers' needs.

The MMCX has a robust set of traditional business telephony features and real-time management of mixed media calling to the data and videoconferencing environment, enhancing the usefulness of these networks for users and network/system managers. It also supports a client/server architecture that provides voice, data, and video collaboration between end points (for example, personal computers and UNIX* workstations) on a local area network (LAN). The MMCX client/server architecture also provides access to the wide area network (WAN), allowing LAN end points to call users on other networks, such as telephones on the public switched telephone network (PSTN) and mixed media end points connected to another LAN or the Internet. Any LAN end points supporting the H.323 standard[1] will also be able to place and receive calls to and from desktop and group video systems based on the H.320 standard.[2]

The MMCX product has calling features for mixed media end points similar to those available for telephones in a private branch exchange (PBX). The MMCX server also acts as a platform for emerging mixed media applications such as Internet call centers and telephony gateways. An Internet call center allows customers to simultaneously browse the Internet, place an order, or ask questions of an agent, all using a single phone line.

Panel 1. Abbreviations, Acronyms, and Terms

ATM—asynchronous transfer mode
CDF—cumulative distribution function
CPU—central processing unit
CT—cut through
CTD—cut-through delay
DLL—dynamic link library
ECS—Enterprise Communications Server
FCFS—first-come first-served
GUI—graphical user interface
HDLC—high-level data link control
IP—Internet protocol
ISA—Industry Standard Architecture
ISDN—integrated services digital network
ITU—International Telecommunication Union
LAN—local area network
LANE—local area network emulation
LST—Laplace-Stieltjes transform
MASI—multimedia application/services interface
MMCX—MultiMedia Communications eXchange
MP—multilink protocol
OA&M—operations, administration, and maintenance
OC-3—optical carrier digital signal rate (in this case, 155 Mb/s) on a synchronous optical network facility
PBX—private branch exchange
PCI—peripheral component interconnect
PD—post dialing
PDD—post-dialing delay
PPP—point-to-point protocol
PRI—primary rate interface
PSTN—public switched telephone network
SNMP—simple network management protocol
TCP/IP—transmission control protocol/Internet protocol
TD—tear down
UDP—user datagram protocol
WAN—wide area network
WILDcard—WAN-into-LAN device

An Internet telephony gateway provides LAN-based telephony applications with access to the PSTN, allowing LAN-based users to call traditional voice telephones, and vice versa. Ahuja et al.[3] have detailed the benefits of mixed media collaboration and the client/server architecture on which the MMCX product is based.

Call control and media streams are subject to important performance requirements. Call control has stringent requirements on *post-dialing delay*, the elapsed time between when the calling party completes the call request and when he or she receives an acknowledgment, and also on *cut-through delay*, the elapsed time between when the called party answers the call request and when the speech path is established. The performance requirements at the media-stream level are based on the packet delay and media blocking requirements for each of the media. To analyze the performance of MMCX at the call level, we abstract the complex message flows into a multiclass queuing system and develop efficient computational algorithms for its analysis. We consider the operating system message scheduling policies and derive an optimal scheduling policy that satisfies the performance criteria and maximizes the MMCX call processing capacity. At the media-stream level, we determine how many primary rate interface (PRI) links are required to satisfy the call blocking requirements for each of the media. We then evaluate the total packet arrival rate at the server and the total load that MMCX traffic will place on the customer's LAN and WAN.

An interactive tool that incorporates the call control and media-stream analyses is implemented with Visual Basic* to run on a Windows*-based PC. This tool is used by technical marketing and sales personnel to build and reconfigure customers' networks.

The remainder of this paper is organized as follows. In the "Overview," we describe the hardware and software of the MMCX client/server architecture, MMCX interfaces to the LANs and WANs, the server hardware organization, the software organization, a call model, and the media streams. In "Performance Requirements," we present the overall performance requirements, including those for call control and media streams. "Call Mix Model" details an MMCX call mix model that has been formulated to analyze performance. We analyze and evaluate the performance of call control and media streams in "Performance Analysis," then give a brief description of the "Design Tool," and summarize our findings in "Conclusions."

Overview

This section describes the MMCX client/server architecture, network connectivity, server hardware and software organizations, a call model, and the media streams.

Client/Server Architecture

The MMCX client/server architecture provides distributed operation, yet supports services that should be centralized because of cost or control considerations. As **Figure 1** shows, the MMCX server is the central point of control for calling features. The server also enables services such as management, security (including admission control), bandwidth management (that is, controlling the amount of LAN traffic an end point may create), and billing to aggregate in a single location. The server is the point of access to hardware or facilities that would be too expensive to provide at each end point. For example, Figure 1 shows an ISDN facility connecting an MMCX server and a private branch exchange (PBX).

The MMCX server also supports the multimedia application/services interface (MASI), a protocol that tightly couples the MMCX product and the DEFINITY® Enterprise Communications Server (ECS). With MASI, the DEFINITY ECS acts as a feature server for the MMCX, providing many of the sophisticated features of DEFINITY ECS on MMCX-enabled end points. In the future, MMCX will also work with other servers located on the LAN to implement additional services. One example shown in Figure 1 is a messaging server that may be used to store and retrieve mixed media messages in the manner currently familiar to voice mail users.

Client applications on the end point provide access to the server applications. Client-side functions can be loosely grouped into the categories of call control and applications. Call control software includes the end point's graphical user interface (GUI) and communica-

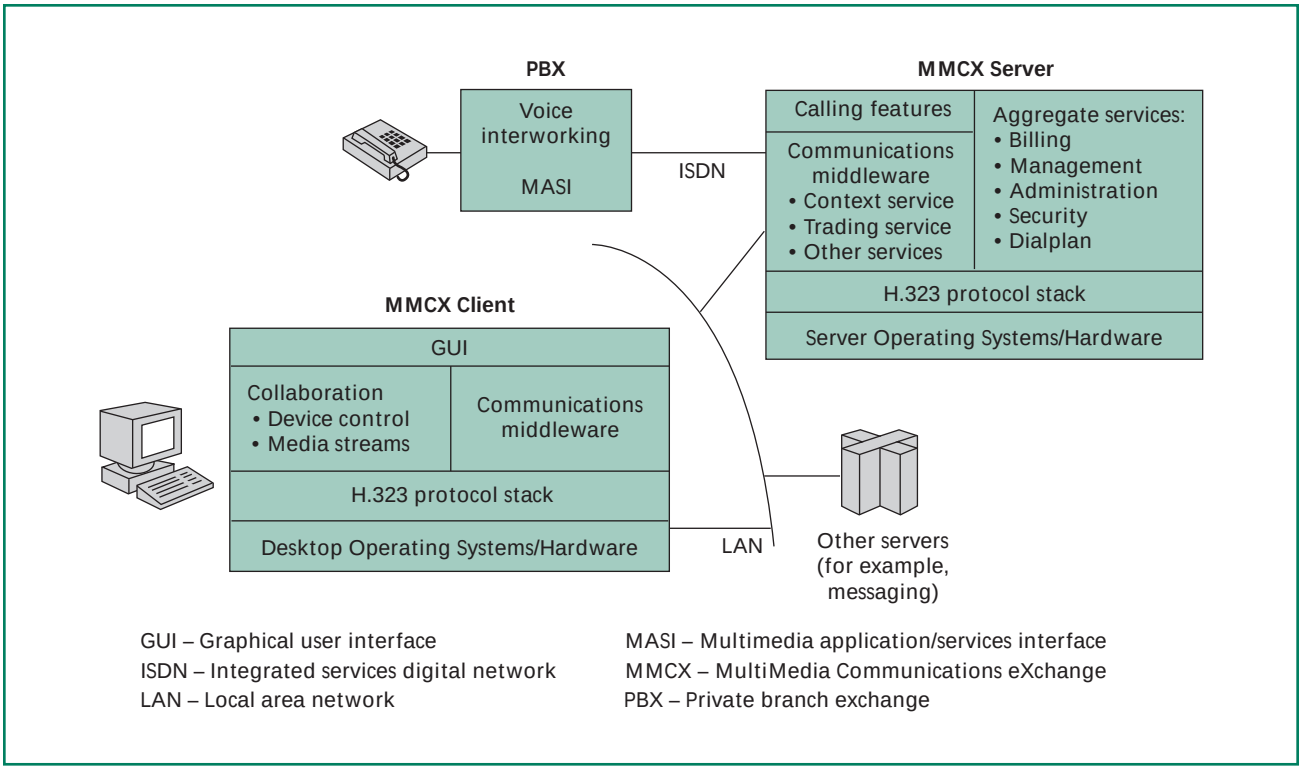


Figure 1.
MMCX client/server architecture.

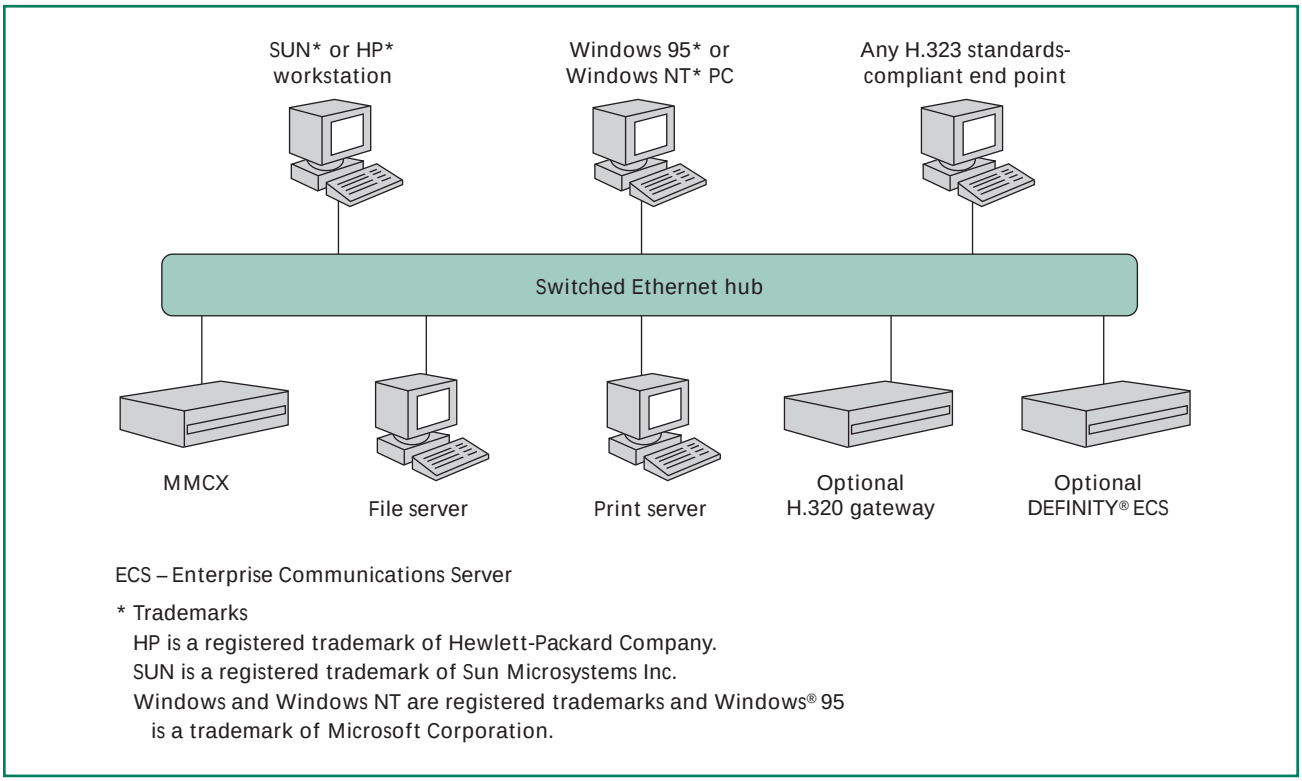
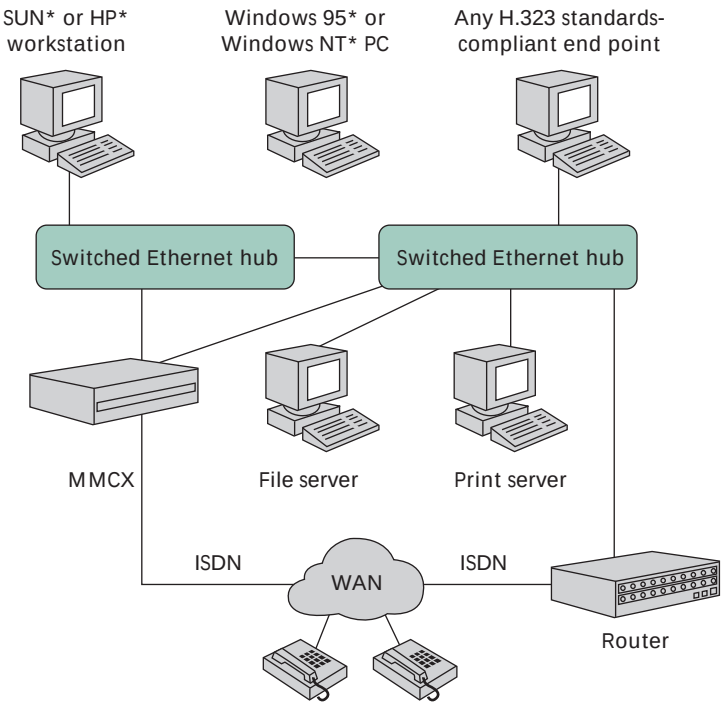


Figure 2.
LAN network connectivity for an MMCX server.



ISDN – Integrated services digital network
MMCX – MultiMedia Communications eXchange
WAN – Wide area network

* Trademarks
HP is a registered trademark of Hewlett-Packard Company.
SUN is a registered trademark of Sun Microsystems Inc.
Windows and Windows NT are registered trademarks and Windows® 95 is a trademark of Microsoft Corporation.

Figure 3.
WAN network connectivity for an MMCX server.

tions middleware. Applications on the end point produce the media streams associated with the audio, video, and data applications. Over time, rich programming interfaces will exist to open up the end point.

Network Connectivity

The MMCX server is added to the network in much the same manner as a file server. **Figure 2** shows the simplest case. In this example, all MMCX end points reside in the same hub as the MMCX server, and no connectivity exists outside this network. The switched hub could also be a network of switched hubs or an asynchronous transfer mode (ATM) hub running ATM LAN emulation (LANE).

Each user has an administered home MMCX server. If the number of users exceeds the maximum supported by one MMCX server, another MMCX server can be added to the network. The entire system scales as MMCX servers are added.

ISDN interfaces in the MMCX server connect with the WAN, as shown in **Figure 3**. A voice call between an MMCX end point and a standard telephone (basically, any telephone in the world, including H.323 standards-compliant Internet phones) can be established through ISDN interfaces. This figure also shows multiple network interfaces for the MMCX server. In fact, the MMCX server can have as many as four interfaces to switches or subnetworks.

An MMCX network can be created by connecting MMCX servers in separate locations, as shown in **Figure 4**. This example shows servers connected over ISDN facilities. The MMCX servers use point-to-point protocol (PPP) and multilink protocol (MP) to create a logical pipe that carries mixed media traffic between servers. The MMCX servers dynamically expand and

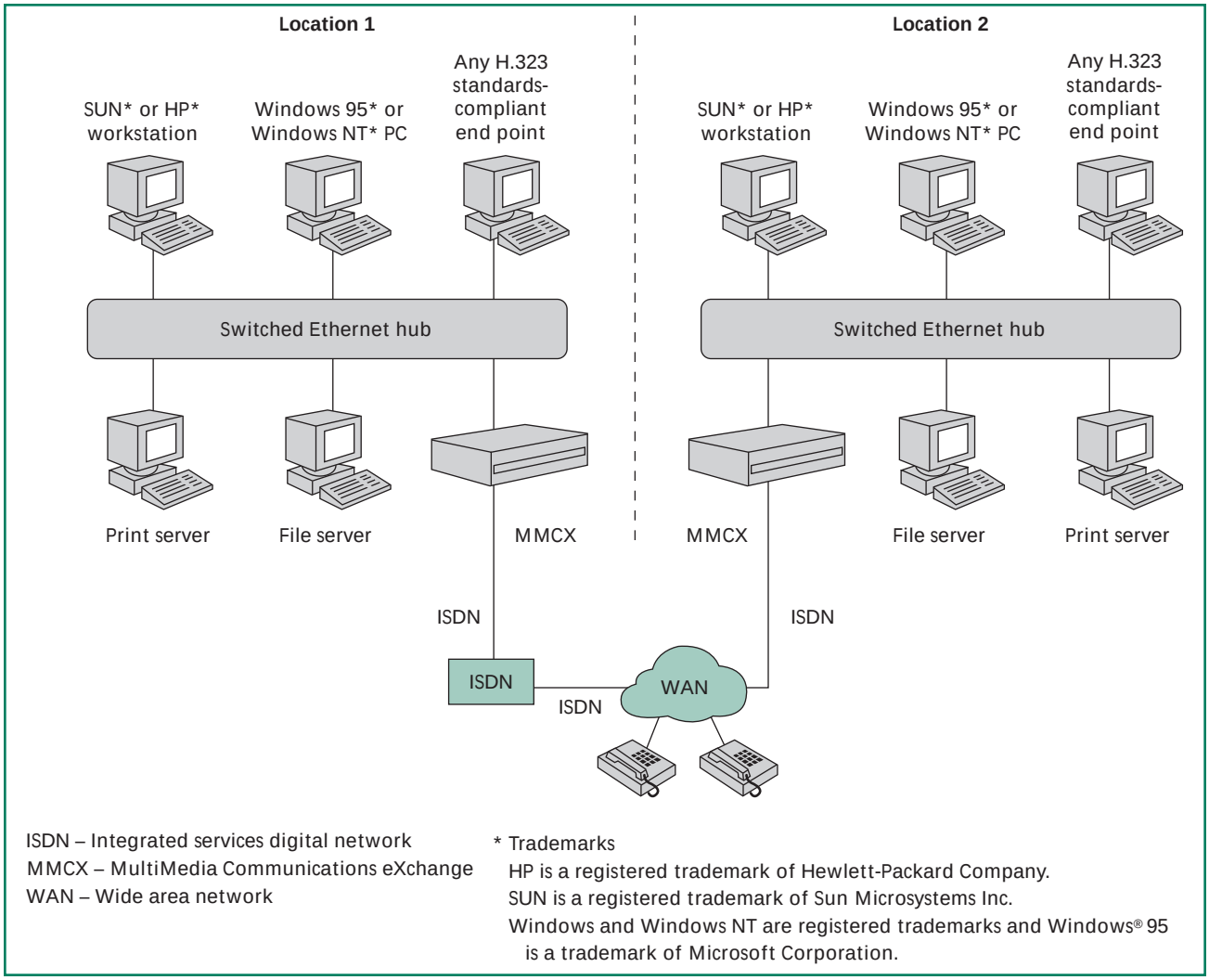


Figure 4.
LAN/WAN network connectivity for an MMCX multiserver environment.

contract the bandwidth of the pipe as users make and drop interserver calls. We describe interserver connectivity in more detail in "Media Streams."

Server Hardware Organization

The MMCX server is built on a PC platform. The central processing unit (CPU) is a plug-in Pentium processor board (not a motherboard) with memory and logic that accommodates connection to Industry Standard Architecture (ISA) and peripheral component interconnect (PCI) buses. The server platform also includes serial ports, floppy controllers, and hard disk controllers. The optional remote maintenance board provides remote diagnostic and restart capabilities. As many as four ISDN primary rate interface (PRI) boards, each of which supports two interfaces, may reside in the server. A maximum of four 10-Mb/s Ethernet network interface circuit boards may reside on the PCI bus, which can accommodate a single 155-Mb/s OC-3 ATM network interface circuit. The WAN-into-LAN device (WILDcard), a custom protocol processing board, has devices that encode high-level data link control (HDLC), play out and conference synchronous audio streams, and provide echo cancellation. Firmware controls the hardware, terminates the audio packetizing protocol, and terminates the PPP and MP used for interserver communication. A maximum of two WILDcards may reside in the server.

Software Organization

The MMCX product is based on an object-oriented model in which the objects can be freely distributed between the server and end points using an object request broker that manages remote objects as if they were local. This design allows MMCX to follow current open standards and recommendations and to easily migrate towards new standards as industry accepts them.

The basis for the call processing model is a concept known as the "communications context." The context, which consists of people and services that support communications, can be compared to a meeting: it involves people, services (for instance, a whiteboard), and a location. It can also persist over time. People may leave or reenter as desired. In fact, all participants may leave together, then return to find the meeting in the same state as when they left. In MMCX, a service that is brought into a context may reside or be located in some other network accessible from the MMCX.

The intrinsic value of this communications model is its generality. The model supports familiar communications system features with parties and transport. It enables more elaborate communications in which multiple parties and a rich array of user services, such as types of mixed media, can be added and removed dynamically. User services in a context are likely to be independent of one another; each can be added or removed at any time without affecting the others.

The software for the model is built on the following major elements.

- *Communications middleware*. Defined as a layer between applications (that is, elements that implement calling features or broker the connection of media streams) and transport-specific services, such as ISDN PRI or transmission control protocol/Internet protocol (TCP/IP). The communications middleware in MMCX consists of a number of services, including:
  - *Context service*—Described at the beginning of this section.
  - *Virtual transport service*—Provides an abstraction of transport that presents a common model for a variety of underlying communications networks, hiding the transport from the applications.
  - *Name service*—Supports virtual addressing for virtual transport.
  - *Trading service*—Enables dynamic registration of services and dynamic location of registered services based on a request model.
  - *Event collection service*—Supports maintenance, engineering, and field support activities, and generates traffic and accounting information.
- *End point*—Provides an interface between the user and the system, implemented through the end point's GUI. End-point software also connects hardware (such as audio or video devices) with the system.
- *Registration service*—Allows a user in MMCX to address another user, not a physical location, as is the case for a telephone. A login sequence associates a user with a physical location. The registration service also controls access to the system.
- *Call control service*—Provides for call origination, address translation, media brokering, and feature sequencing.
- *Audio mixer*—Conferences audio when two or more end points are involved in a call.
- *Interworking service*—Translates between the LAN presentation and some other form (for example, ISDN voice). The interworking service also provides proxy representation (a virtual end point) for end points not in the LAN.
- *Interserver routing service*—Supplies the facilities for setting up a call between servers.
- *Maintenance services*—Provide for system initialization and recovery and monitor system health. They also include facilities for upgrades and remote diagnostics.
- *System administration*—Provides a simple network management protocol (SNMP) agent for operations, administration, and maintenance (OA&M) functions.

These major elements are implemented as a set of distributed objects accessible through an object request broker. Although these objects can be highly distributed, the audio mixer, interworking service, interserver

Table I. Creating a point-to-point call.

Originating client	Server	Destination client
1. The originating user selects a party to call.		
2. Client software requests that a context be created.		
	3. Server software creates the context.	
4. Client software requests that the originator be added to the context.		
	5. Server software processes the request.	
6. Client software requests that the media managers (for example, voice, video, data) be added to the context.		
	7. Server software processes the request.	
8. Client software requests that the destination be added to the context.		
	9. Server software processes the request, possibly applying a termination feature (for example, call coverage).	
10. Client software observes context state changes and applies some form of feedback (for example, audio ringback tone) to the originating user.		10. Client software observes context state changes and applies some form of notification (for example, audio ringing tone) to the destination user.
11. Client software sends media capabilities (for example, voice, video, data) to the media managers.		11. Client software sends media capabilities (for example, voice, video, data) to the media managers.
	12. Media managers identify the "best match" of media capabilities for the two end points and direct the endpoints to begin transmission of media streams.	

routing service, and most of the call control service currently reside in the server.

Call Model

As we mentioned above, the "communications context" is the basis for the MMCX call model. For example, **Table I** shows the steps used to create a simple point-to-point call, with both parties on the same LAN.

At this point, the end points are exchanging media streams, meaning that a user hears the other user's voice, sees the other user's video, and collaborates with the other user on a data application. The communications context consists of each user and the managers of the media involved in the call. A user may add or drop a medium at any time without affecting other media in the call and may also indicate beforehand the media to be added during origination.

Media Streams

On the LAN, the Internet protocol (IP) carries the mixed media traffic generated by the mixed media end points. The user datagram protocol (UDP) rides on top of IP and carries the voice and video packets. The TCP also rides on top of the IP and carries call control signaling packets and packets associated with data applications.

In Figure 4, suppose an end point in location 1 places a mixed media call to an end point in location 2. In the call control signaling phase, the end point and

server in location 1 exchange TCP/IP packets that inform the server whom the end point wishes to call. The server then exchanges TCP/IP packets with the server in location 2, which likewise exchanges TCP/IP packets with the called end point. The calling user will then hear a ringback tone, indicating that the request is being processed, and the called user will hear a ringing tone. When the called user answers, more TCP/IP packets are exchanged between the end points and servers, the tones stop, and the servers prepare to process the audio packets associated with the call.

As each user speaks into his or her audio headset, the end point converts the speech into UDP/IP packets and sends those packets over the LAN to its server. Each server forwards the speech across the WAN to the other server, which generates UDP/IP packets containing the speech and sends those packets over the LAN to the destination end point. The end point then plays the speech into its user's headset.

Suppose the users add video and a data application (for example, a shared whiteboard) to the call. Each end point generates UDP/IP and TCP/IP packets containing its encoded video and data traffic, respectively, and sends those packets to its server. Each server sends those packets across the WAN to the other server via the PPP/MP, and the far-end server sends those packets across the LAN to the far-end end point.

Performance Requirements

From the perspective of performance characterization, the examples described above show that the MMCX server provides two types of services for its mixed media end points. First, it handles call control requests, including requests to place and tear down calls and to add and drop users and media on existing calls. Second, it handles the actual mixed media streams and protocols for calls that interconnect with the WAN.

The MMCX server has important performance requirements for both call control and media stream services. For example, a user who places a new call must receive feedback within the brief period of time that the system is processing the request. The telephony analogy is called *post-dialing delay*, that is, the delay that occurs between the time the user dials the last digit of a telephone number and hears ringing in the telephone handset.

The performance requirements associated with handling media streams include the maximum end-to-end delay that the system can impose on each of the media. Years of human factors experience[4,5] show that audio information can be delayed at most 400 ms end to end, that is, between the talker's microphone and the listener's speaker, without interfering with the quality of service that customers expect. We take this for granted with the traditional telephone, but meeting this requirement in the LAN/WAN environment requires careful engineering of all the system components, including the end points, the MMCX server, and the LAN and WAN networks.

Call Control Requirements

The call control performance has been characterized by three parameters—post-dialing delay, cut-through delay, and tear-down delay.

Post-dialing delay. For intraserver calls involving only one MMCX server, at least 99% of the calls should experience a *post-dialing delay* (the time between when a user dials the last digit of the called party and when he or she receives an indication from the network that the called party was just alerted) of less than 1.5 seconds. For interserver local and long distance calls, at least 99% of the calls should experience a post-dialing delay of less than 4 seconds, given that a call was established between two MMCX servers. If this is the first interserver call on the link between the two MMCX servers, at least 99% of the calls should experience a post-dialing delay of less than 6.5 seconds. (The additional 2.5 seconds are needed to establish a server-to-server MMCX link between the two MMCX servers. A second call origination is then needed to establish the bearer channels for the initial call. Therefore, two sequential calls are needed to set up the initial server-to-server link.)

The post-dialing delay requirements apply to:

- Establishing a point-to-point audio call;
- Adding a party to a call;
- Adding media to a call, such as video or data; and
- Transferring a call.

Cut-through delay. At least, 99% of the calls should experience a *cut-through delay* (the time between when the called party answers the audio call and when the speech path is established) of less than 750 ms. This requirement is also applied to adding media cut-through (for example, video or data) and party cut-through.

Tear-down delay. The 50% and 99% *tear-down delay* (the time between when a party requests that a call be disconnected and when the party receives an indication that the call was torn down) for any type of media should be less than 300 ms and 500 ms, respectively.

Media Stream Performance Requirements

This section describes media stream performance requirements for audio end-to-end delays and end-to-end video or data delays.

Audio end-to-end delay. The studies conducted so far[4,5] have indicated that the degree of difficulty caused in an audio conversation with adequate echo control will not be objectionable if the one-way end-to-end audio delay does not exceed 400 ms.

End-to-end video or data delay. We have provided no specific requirement for end-to-end video delay because the delay required at the end points to compress the video signal is significantly longer than the audio encoding delay. However, we have included a requirement for end-to-end network delay of any media that excludes the delays incurred at the end points. The end-to-end network delay is set to be less than 100 ms at the worst case, where two MMCX end points are connected across the WAN.

Call Mix Model

An important aspect of performance characterization is the *call mix model*, that is, how the customer plans to use the system. The call mix model takes into account such factors as the total number of users connected to the system, the average number of calls per user per hour, and the distribution of number of parties per call. These factors influence the offered load on the MMCX server and how its components are engineered.

A call mix model has been defined for MMCX to help provide a framework that characterizes the performance of a network incorporating MMCX technology. The model contains three submodel components, namely, the call classification model, call generation model, and call arrival rate model.

Call Classification Model

The MMCX traffic is partitioned into four classes of traffic:

- *Class I*—Point-to-point audio calls, in which the number of parties for each call is always equal to two.
- *Class II*—Multipoint audio calls, in which the number of parties is always greater than two.
- *Class III*—Short-collaboration mixed media calls, in which the maximum call duration does not exceed 30 minutes. Calls in this class are further subdivided into point-to-point and multipoint calls having a mix of audio, video, and data media.
- *Class IV*—Long-collaboration mixed media calls, in which the minimum call duration is greater than 30 minutes. Similarly, the calls in this class are further subdivided into point-to-point and multipoint calls having a mix of audio, video, and data media.

Each class has several types of attributes, such as the call holding time, number of media, and number of parties. These attributes are application dependent and may differ from customer to customer. **Table II** shows an example of call classification parameters being defined for the concurrent engineering application, which typically involves having design engineers in different geographical locations collaborate on designing a component using multimedia software packages.

Call Generation Model

The call generation model is defined by three parameters, namely, the percentage of incoming, outgoing, and intraserver calls.

Incoming calls are those arriving from other sites through the WAN(s). These calls can be initiated from any remote end points (for example, MMCX or non-MMCX end points). *Outgoing calls* are initiated from any MMCX end points at the given site to any other remote end points via the WAN. *Intraserver calls* are local calls that are made within

Table II. An example of call classification parameters in concurrent engineering.

	Call density	Average call duration (minutes)	Distribution of number of media	Distribution of number of parties
Class I	25%	5	Audio	2
Class II	5%	30	Audio	3
Class III	60%	15	50% audio and data 50% audio and video and data	50% of 2 parties 30% of 3 parties 10% of 4 parties 5% of 5 parties 5% of 6 parties
Class IV	10%	120	50% audio and data 50% audio and video and data	50% of 2 parties 30% of 3 parties 10% of 4 parties 5% of 5 parties 5% of 6 parties

the same site. They also include calls made or received to and from their local PBX.

Call Arrival Rate Model

The *call arrival rate* is the total number of calls that will be made and received by a given MMCX end point during the busy-hour period. The busy-hour period encompasses the intervals from 9:00 a.m. to 11:30 a.m., and 1:00 p.m. to 3:30 p.m., a five-hour period. In the call mix model, the default value of the call arrival rate is set to five calls per day. During the busy-hour period, each MMCX end point will be involved in five different conversations (both making and receiving calls).

Performance Analysis

We have conducted a detailed analysis of the MMCX server to ensure that it meets the call control and media stream performance requirements. Two distinct performance analyses corresponding to two distinct performance criteria have been used to analyze the MMCX—an analysis of MMCX call control and an analysis of media streams.

Call Control Modeling and Analysis

In this section, we present a queuing model and analysis of call control used to:

- Derive a scheduling policy that
  - Meets the performance requirements for processing messages involved in the call flow,
  - Is simple to implement, and
  - Achieves an acceptable call throughput for the MMCX server;
- Determine the call processing capacity of MMCX;
- Study the relationship between the call processing capacity and CPU requirements and quantify the gain in call processing capacity as a function of increase in CPU power; and
- Find the admissible set of different types of call arrival rates that can be supported.

The basic call control model, which is abstracted from complex message flows, is a single-server multiclass queue. Each MMCX call can be thought of as comprising three elements: post-dialing (PD), cut-through (CT), and tear-down (TD) components. Because the CPU time requirements for the TD component are negligible compared to what is required for the PD and CT components, we can ignore this in our analysis. Therefore, we only consider PD and CT components of the call in the remainder of this paper. Each component in turn contains a sequence of messages with random service times. Calls of different types are composed of different message sequences and therefore have different overall service times.

Description of the model. First, we list the assumptions behind the call control queuing model:

1. The time taken for end point interactions is negligible. Because the call flow consists of messages processed either by the end points or the call server, the service times of messages processed by the end points are small compared to those for

messages processed by the server. In our model, as soon as a message is processed by the call server, the next message for the call will instantaneously arrive in the queue.

2. The context switching time of an operating system is negligible compared to the processing time of the messages. The total service time of a call is therefore just the sum of the service times of the individual messages that make up the call flow.
3. Calls of type s arrive at a Poisson rate λ_s, and the arrival processes of calls of different types are independent of each other. For a given scheduling policy, the admissible set of call arrival rates alluded to earlier is then the set of all 4-tuples $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$, such that the delay requirements are satisfied.
4. Within each call type, s, the arrival streams of the two components (PD, CT) are modeled as independent Poisson processes, each with rate λ_s. In practice, some correlation exists, especially between the PD and the CT streams, because the PD component ends when ringing begins and the CT component is initiated when the recipient accepts the call. However, since the time lag between the two components is likely to be much larger than their service times, this is a reasonable assumption.
5. Service times of individual messages are assumed to be exponentially distributed with possibly different means. Thus, the total service time of a component is distributed as a sum of exponentials.
6. The bearer traffic uses a constant fraction of the call server's CPU processing power. This essential assumption is conservative, assuming that priority is always given to the bearer traffic and the bearer traffic load is always at the busy-hour peak rate.

Scheduling policies. In this section we examine more carefully the class of scheduling policies of interest. Suppose a message has just been processed and the policy has to decide which among all the active messages to process next. (*Active* messages are those ready to be processed.) Each active message is associated with an originating call. Let us first consider the case where all messages correspond to one class of call. We can then show that given a chosen class, the optimal scheduling policy is to process those active messages of the call that arrived the earliest. In other words, we will always finish processing all the messages of a call before we start to work on a call that arrives later.

Formally, *Stim-to-completion* is a schedule in which all messages corresponding to one call are processed before any message of another call that arrives later. In addition, messages within a call are processed according to a first-come first-served (FCFS) discipline.

Theorem: *Consider the cost function, which is 0 if the call delay requirements are satisfied, and a convex increasing function of the delay otherwise. Stim-to completion minimizes the expected value of this cost function.*

The above result greatly reduces the state space of the queuing model. First we assume that each call arrives as two independent jobs (see assumption 4 above), corresponding to PD and CT, respectively. Thus, a job class is defined by its call type (such as point-to-point audio or multipoint audio) and its component type (PD or CT). Consequently, we have an eight-class single-server queuing system corresponding to four call types and two components. Within each class, jobs have identical service time distributions and delay requirements. **Figure 5** shows the eight-class single-server queuing system, in which each ellipse represents a job and the vertical lines inside the ellipse represent the sequence of messages. (The bearer traffic is not shown in the figure.)

The scheduling policy in this abstract queuing model has to make decisions at two levels: at the first level, which non-empty queue to serve; and at the second level, which job to serve within the selected queue. The result we obtained above proves that once the queue is selected, the order of jobs to be served within the queue is FCFS. Let us now consider the scheduling policy of which non-empty queue we should serve. We examine two candidates: strict priority rules and earliest due date.

The *strict priority* policy assigns a static priority ordering on the queues. It always picks from the non-empty queue with the highest priority. Of all possible scheduling policies, strict priority is the extreme, because the jobs in the queue that are given the highest priority will have the smallest possible delays, and

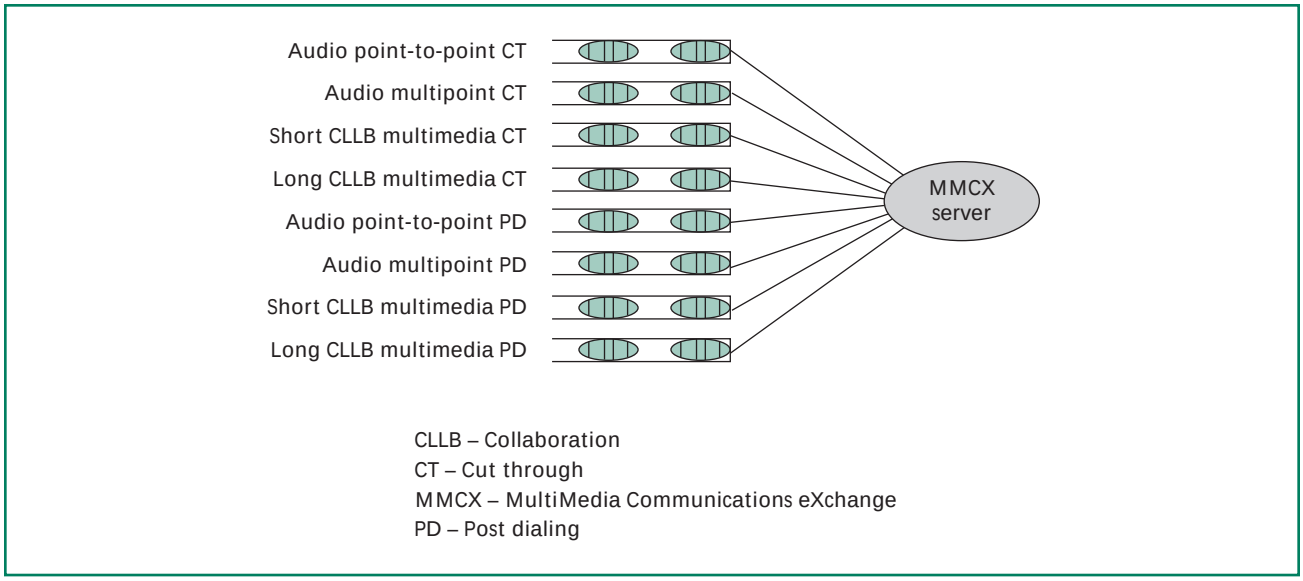


Figure 5.
Queuing model of call control.

those that are given the lowest priority will have the largest possible delays.

The *earliest due date* policy is a dynamic priority rule that serves the job with the earliest due date, that is, the time at which the job must be finished to meet the delay requirement.

Call control performance analysis. In this section, we summarize the performance analysis of the queuing system shown in Figure 5 and derive efficient computational procedures. Let us consider the strict priority preemptive/resume scheduling policy. First, we assign numbers from 1 to 8 to the queues, with the implicit understanding that k has higher priority than k+i. The main computational effort of the analysis is the inversion of the Laplace-Stieltjes transform (LST) of the waiting time distribution for a given queue. We give below the LST of the waiting time distribution for queue k ($1 \le k \le 8$) job in the preemptive resume priority system.

Let $B_k^*(s)$ be the LST of the service time of a job of queue k in the preemptive resume priority system, and let $1/\mu_k$ be the corresponding mean. The service time distribution is the sum of the service times of the constituent messages. There can be as many as 25 messages, depending on the message flow. The LST of the waiting time distribution of the job in queue k is[6]

$$\mathbf{W}_k^*(s) = \left(1-\rho_k^+\right)\sigma_{k-1} / \left(s - \lambda_k + \lambda_k \mathbf{B}_k^*(\sigma_{k-1})\right) \quad , \quad (1)$$

$$\sigma_{k-1} = s + \lambda_{k-1}^+ - \lambda_{k-1}^+ \theta_{k-1}^+(s) \; , \quad (2)$$

and $\theta_{k-1}^+(s)$ satisfies

$$\theta_{k-1}^+(s) = \mathbf{B}_{k-1}^*\left(s + \lambda_{k-1}^+ - \lambda_{k-1}^+ \theta_{k-1}^+(s)\right) \quad . \quad (3)$$

Here

$$\lambda_k^+ = \sum_{i=1}^k \lambda_i ; \rho_k^+ = \sum_{i=1}^k \rho_i ,$$

where $\rho_i = \lambda_i / \mu_i$; and λ_i is the mean arrival rate of jobs to queue i. The numerical procedures for computing the waiting time distribution involves solving the fixed-point (3) and numerically inverting (1).

We will first consider the system performance in the presence of a single call type, namely, audio point-to-point type, and evaluate the call processing capacity of the different scheduling policies. Subsequently, we will consider the performance with multiple call types. **Figure 6** is a plot of the log of the probability that PD and CT exceed their limits as a function of the call arrival rate for the case where PD jobs are given high priority over CT jobs, and for the case where CT jobs are given priority over PD jobs. Thus, there are four curves: two for the PD (one with high priority, and one with low priority) and, similarly, two curves for the CT. The policy that meets the requirement (proba-

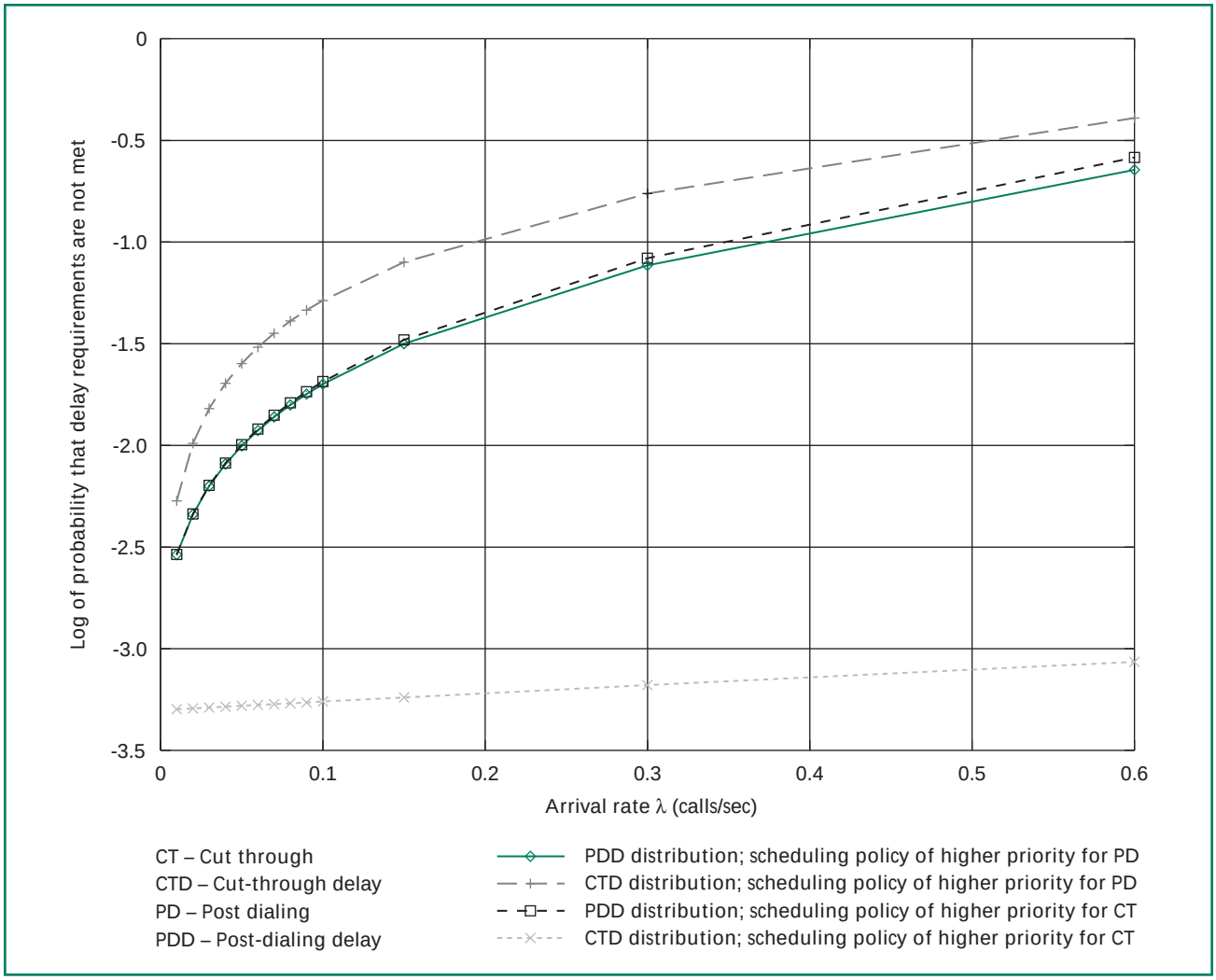


Figure 6.
Audio point to point, with PDD and CTD distribution probability of PDD > 1.5 sec and CTD > 0.4 sec.

bility 0.99) and achieves the highest call arrival rate is the one in which CT is given high priority over PD. The achievable call arrival rate in this case is 0.05 call/sec (that is, 180 calls per hour). More importantly, the two curves corresponding to PD are almost identical, implying that no other scheduling policy, including the earliest due date, can improve the performance of the low-priority PD jobs. We therefore conclude that the policy in which CT jobs are given high priority is optimal in this queuing model, where PD jobs have much larger service time requirements than CT jobs.

In **Figure 7** we consider the four call types and plot the performance as a function of the aggregate call arrival rate, where the individual arrival rate of each call type is determined by the call mix model. The eight curves in the plot correspond to the PD and CT of the four call types, where CT jobs are given high-priority service. The service order among the CT jobs belonging to different call types is arbitrary and is found to be irrelevant from our computational experience; the same is true for the PD jobs. Figure 7 shows that the maximum achievable aggregate call arrival rate is 0.004 call/sec (that is, 14 calls per hour).

Table III demonstrates the insensitivity of the performance to the variability in the service time distribution of the jobs and to the priority ordering among the PD jobs of different call types. The table considers two different distributions—exponential and constant—and examines all possible priority ordering among the four PD queues. The entries in Table III,

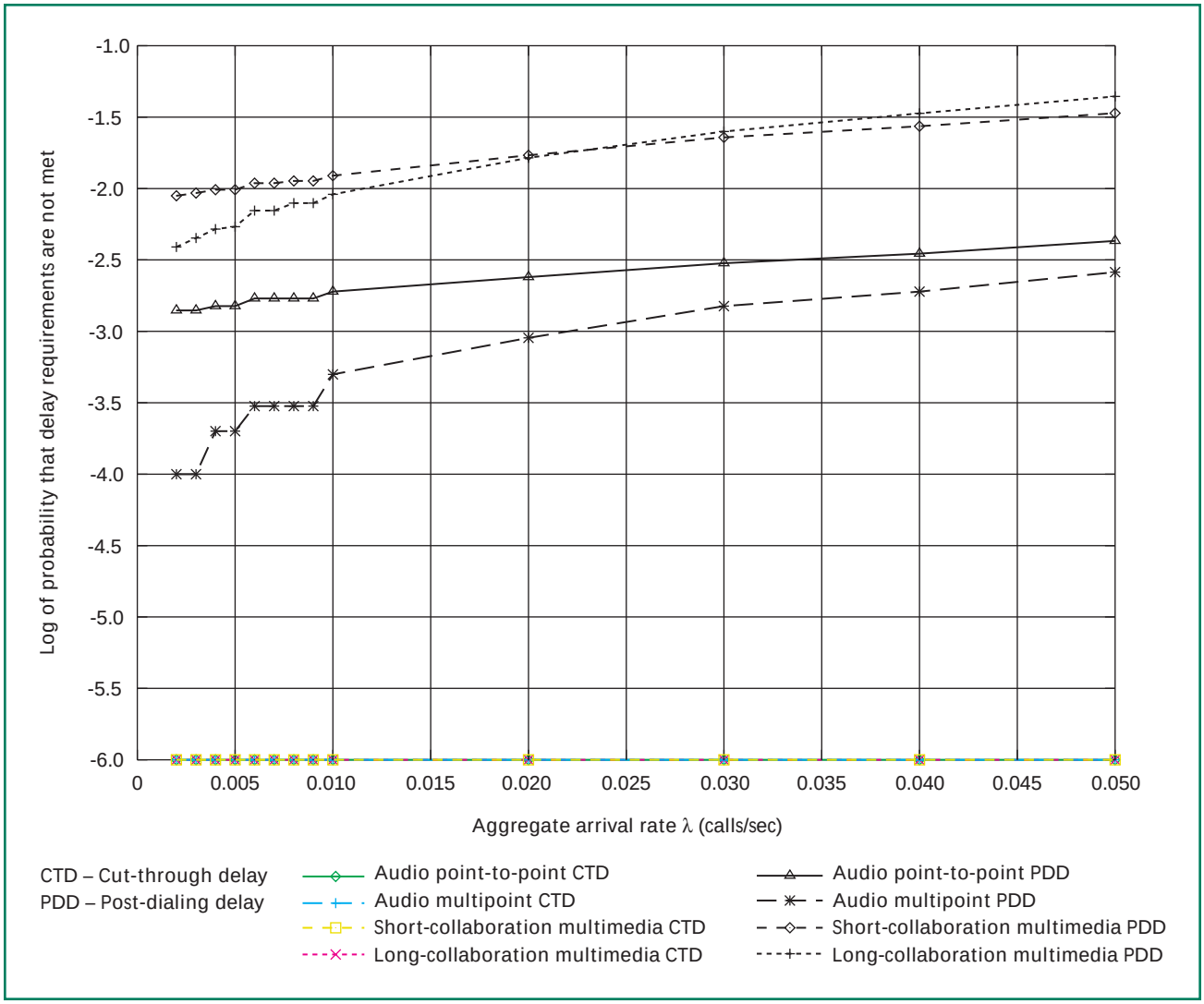


Figure 7.
PDD and CTD distribution probability for PDD > 1.5 sec and CTD > 0.75 sec.

which are the probability that the PD requirements are met, are all higher than the desired 0.99 value.

In **Figure 8**, we examine the effect of scaling the service rate of jobs in the system (or, equivalently, of increasing the CPU processing capacity of the server) on the aggregate call arrival rate. Increasing the CPU capacity by a factor of three increases the call processing capacity by more than a factor of one hundred.

The maximum achievable call arrival rate (that is, the MMCX call processing capacity) is 14 calls per hour, as discussed above. Figure 7 showed that this capacity is achieved where the PDD for short-collaboration mixed media calls achieves the delay bound constraint, which dictates that the delay be less than 1.5 seconds with 99% probability. We can relax this delay requirement either by increasing the delay bound, decreasing the probability of delay satisfying the specified bound, or applying both.

Table IV illustrates the effect of relaxing the delay requirements on the server capacity, where it is evident that the server capacity dramatically increases as we relax the delay requirements. This indicates that the cumulative distribution function (CDF) of PD initially rises very sharply and then flattens considerably.

One conclusion we can draw from this information is that mixed media call processing is much more

Table III. Insensitivity to priorities (within PD and CT) and service time distributions.

	Probability PDD ≤ 1.5 sec for PD queues							
	APP-PD		AMP-PD		SCMM-PD		LCMM-PD	
Priorities	exp	const	exp	const	exp	const	exp	const
5 6 7 8	0.9984	0.9998	0.9997	1.0000	0.9896	0.9982	0.9943	0.9968
5 6 8 7	0.9984	0.9998	0.9997	1.0000	0.9896	0.9982	0.9967	0.9997
5 7 6 8	0.9984	0.9998	0.9982	0.9982	0.9896	0.9982	0.9943	0.9968
5 7 8 6	0.9984	0.9998	0.9997	1.0000	0.9896	0.9982	0.9967	0.9997
5 8 6 7	0.9984	0.9998	0.9982	0.9982	0.9896	0.9982	0.9943	0.9968
5 8 7 6	0.9984	0.9998	0.9982	0.9982	0.9896	0.9982	0.9967	0.9997
6 5 7 8	0.9984	0.9998	1.0000	1.0000	0.9896	0.9982	0.9943	0.9968
6 5 8 7	0.9984	0.9998	1.0000	1.0000	0.9896	0.9982	0.9967	0.9997

APP – Audio point to point
AMP – Audio multipoint
const – Constant distribution
exp – Exponential distribution
LCMM – Long-collaboration multimedia call
PD – Post dialing
SCMM – Short-collaboration multimedia call

complex than audio call processing and thus requires less stringent delay requirements.

A second approach that can be used to increase the call capacity of MMCX is to increase the CPU processing power of MMCX, as discussed above in conjunction with Figure 8. If CPU power cannot be increased, we can reserve a similar fraction of CPU cycles to process the bearer traffic, thus efficiently increasing the CPU cycle for call processing. **Table V** shows the effect of this on the MMCX capacity, with the delay requirements set at 5 seconds with 0.90 probability. Again, we see a dramatic increase in call capacity when the CPU cycle is increased (such as in instances where we decrease the CPU cycle for the bearer traffic).

Finally, to allow for real-time evaluation of the admissibility of different call mixes, we have obtained the admissible set of call arrival rates of different types. The numerical algorithms for computing the waiting time distributions have been implemented in Visual C++ and encapsulated in a dynamic link library (DLL). The design tool links to this DLL during execution.

Media Stream Analysis

The media stream analysis is similarly based on a detailed description of the MMCX server software components involved in handling media streams. The analysis presented here evaluates the MMCX server performance for a given type of loading condition that is obtained from the call mix model. In particular, the analysis will determine how many calls per second the server will receive, how many bearer packets the server will have to process per second, and how many PRI links it will need to ensure that the probability of call blocking will be met for each of the media.

Call arrival rate at the MMCX server. The average call arrival rate, λ, at the MMCX server is given by

$$\frac{kN}{1800(2a+3b+c+d)},$$

where

$$a=2\alpha_1(\theta_{11}+2\theta_{21}+\theta_{31})$$
$$b=3\alpha_2(\theta_{12}+2\theta_{22}+\theta_{32})$$
$$c=\alpha_3(\theta_{13}+2\theta_{23}+\theta_{33})(2\gamma_2+3\gamma_3+4\gamma_4+5\gamma_5+6\gamma_6)$$
$$d=\alpha_4(\theta_{14}+2\theta_{24}+\theta_{34})(2\eta_2+3\eta_3+4\eta_4+5\eta_5+6\eta_6),$$

and where N is the total number of end points, k is the average number of conversations that each user has per hour during a busy hour, α_i is the fraction of calls in class i, γ_i is the fraction of short-collaboration mixed media calls (Class 3) that have i users, η_i is the fraction of long-collaboration mixed media calls (Class 4) that have i users, and θ_{ij} is the fraction of type i calls in class j given that i takes on values 1, 2, and 3, where 1 represents outgoing calls, 2 is incoming, and 3 is intraserver.

Number of PRI links. The PRI links are partitioned into three separate regions, where each region is dedicated to a given type of medium, such as audio, video,

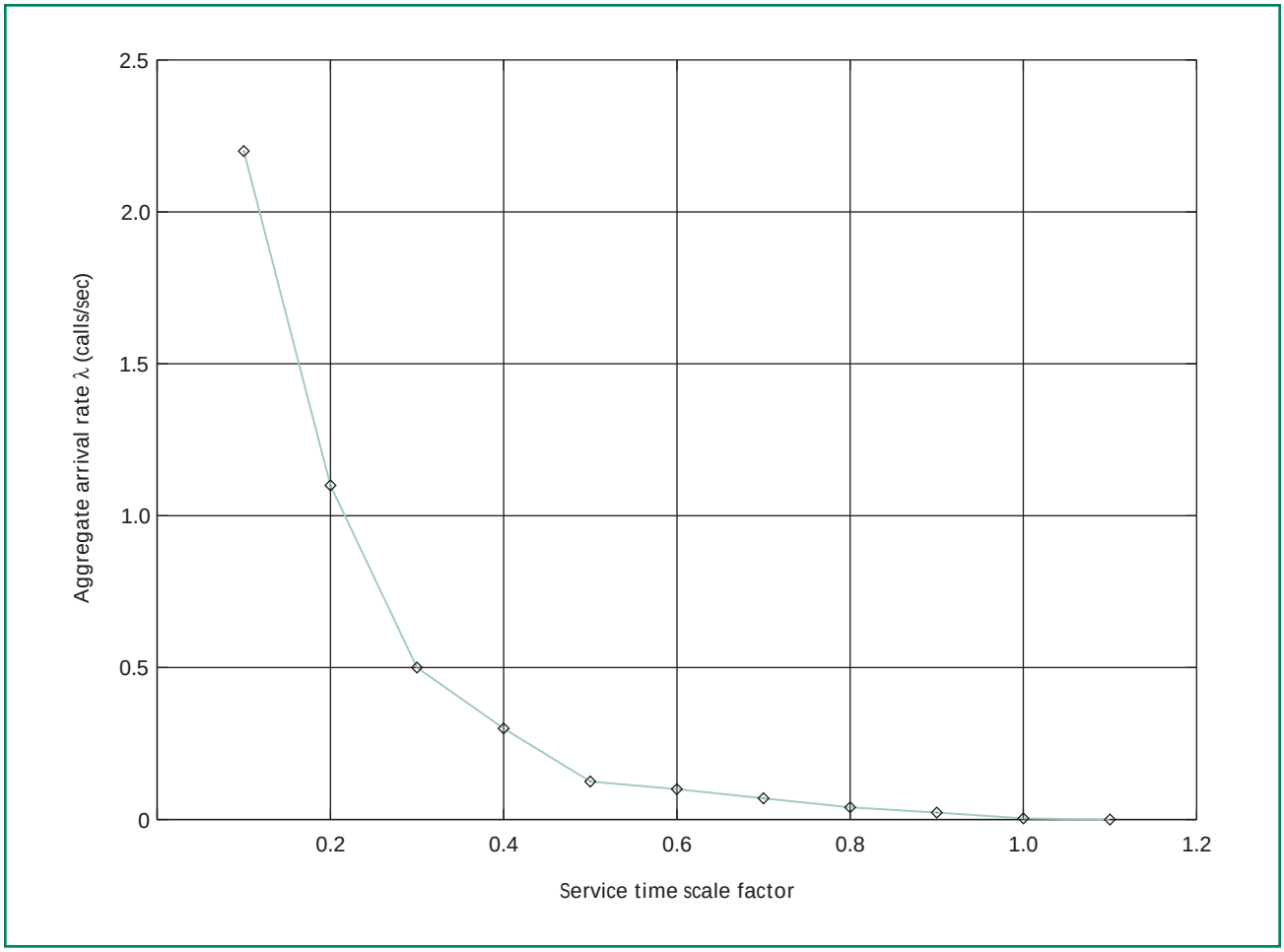


Figure 8.
Service times versus maximum arrival rate of calls.

or data. B-channels are assigned to each region such that the probability of medium blocking will be met for the corresponding medium.

Number of B-channels for audio traffic. For any type of audio call that requires connections on the PRI links (no matter how many parties are involved in the call), we only need to reserve one B-channel on the PRI links. When an MMCX server receives a packet from its PRI interface, it is able to multicast the packet to any number of parties at its local site.

The Erlang Loss-B formula is used to evaluate the probability of blocking for audio traffic. However, two factors have to be considered when evaluating the media blocking for audio traffic. First, the audio medium will always exist in any type of MMCX call. Second, only incoming and outgoing calls will require connections on the PRI links. Thus,

λ (audio) = z*λ,

where z denotes the fraction of calls that are incoming and outgoing.

Number of B-channels for video traffic. Video connections are always point to point. For any type of video call, the number of video channels required on the PRI links is equal to the smaller number of connections being requested at either site. The expression Vchan is derived for the average number of video channels that are required to be assigned for any type of video call on the PRI links.

As we did for audio traffic, we used the Erlang Loss-B formula to derive an expression for the video media blocking probability on the PRI links. The total number of B-channels is obtained from

Table IV: MMCX call capacity with respect to delay bound requirements (default case of reserving 75% of CPU for bearer traffic).

Delay bound requirement (sec)	Probability that PDD ≤ delay bound requirement	MMCX call capacity (calls/hour)
1.5	0.99	14
3.0	0.90	720
5.0	0.90	850

Table V. MMCX call capacity with respect to percent of CPU reserved for bearer traffic (delay bound requirements set at 5 sec with 0.9 probability).

CPU reserved for bearer traffic	MMCX call capacity (calls/hour)
75%	850
80%	540
90%	216

the media blocking expression and is then multiplied by Vchan to evaluate the total number of B-channels for the video traffic.

Number of B-channels for data traffic. For any type of data call, consisting of any number of users, we always reserve p number of B-channels on the PRI links, where p is a parameter that is administered initially for the system.

The analysis of media blocking for the data traffic is similar to that of the audio traffic. Using the Erlang Loss-B formula, we can derive an expression for the data medium blocking probability. From this expression, we will evaluate the number of B-channels for the data traffic.

Packet Arrival Rate at the MMCX Server

After the call is established, the transmission of bearer packets will be followed by the corresponding end points in the call. In general, the MMCX servers act as a gateway. The outgoing bearer packets destined for a remote site will be transmitted to the WAN by way of their local MMCX server. Similarly, the bearer packets coming from the WAN to a given MMCX site will be received first by their corresponding local MMCX server.

Because audio conference bridges are implemented at the MMCX servers, all audio packets that have to be bridged at a given site will have to go through their corresponding local MMCX server. Therefore, the packet arrival rate at the MMCX server is defined as the combination of all the packet arrival rates being received from both the LAN and WAN.

Average audio packet arrival rate at the MMCX server. The overall average audio packet arrival rate at the MMCX server for incoming and outgoing calls is equal to the packet arrival rate at the server per call multiplied by the average number of simultaneous audio calls in progress at any given time, obtained by the following expression:

$$P(\text{audio}) = CIP(\text{audio})*b[2\alpha_1 + 3\alpha_2 + \alpha_3(2\gamma_2 + 3\gamma_3 + 4\gamma_4 + 5\gamma_5 + 6\gamma_6) + \alpha_4(2\eta_2 + 3\eta_3 + 4\eta_4 + 5\eta_5 + 6\eta_6)],$$

where b is the audio packet arrival rate from a given end point and CIP(audio) is defined as the average number of audio calls in progress on the PRI links at any given time. CIP(audio) is obtained by multiplying the audio call arrival density with the probability that the audio calls are not blocked, as shown in the expression:

$$CIP(\text{audio}) = A(\text{audio}) * (1 - \text{BlockingProb(audio)}).$$

Average video packet arrival rate at the server (P[video] packets/sec). The average video packet arrival rate at the server is given by the product of the average video packet arrival rate per call at the server and the average number of video calls in progress, CIP(video). Thus,

$$P(\text{video}) = \frac{CIP(\text{video}) \bullet (h + m)}{\alpha_3 + \alpha_4}$$

where

$$h = \alpha_3 c(2\gamma_2 + 2\gamma_3 + \frac{8}{3}\gamma_4 + 3\gamma_5 + \frac{18}{5}\gamma_6)$$

$$m = \alpha_4 c(2\eta_2 + 2\eta_3 + \frac{8}{3}\eta_4 + 3\eta_5 + \frac{18}{5}\eta_6),$$

and where c is the video packet arrival rate from a given end point and CIP(video) is defined as the average number of video calls in progress on the PRI links at any given time. CIP(video) is obtained by multiplying the video call arrival density with the probability that the video calls are not blocked, using the expression:

$$CIP(\text{video}) = A(\text{video}) * (1 - \text{BlockingProb(video)}).$$

Average data packet arrival rate at the server (P[video] packets/sec). The average data packet arrival rate at the server is given by the product of the average data packet arrival rate per call at the server

and the average number of data calls in progress, CIP(data). Thus,

$$P(data) = CIP(data) * \frac{\omega}{64} * d \text{ packets/sec,}$$

where d is the data packet arrival rate from a given data end point and CIP(data) is defined as the average number of data calls in progress on the PRI links at any given time. CIP(data) is obtained by multiplying the data call arrival density with the probability that the data calls are not blocked, as shown in the expression

$$CIP(data) = A(data) * (1 - BlockingProb(data).$$

Packet arrival rate for intraserver calls. In intraserver calls, only the audio conferencing packets go through the server. The average audio packet arrival rate at the server is given by the expression

$$P(intraserver) = CIP(audio)*b[6\alpha_2 + \alpha_3(6\gamma_3 + 8\gamma_4 + 10\gamma_5 + 12\gamma_6) + \alpha_4(6\eta_3 + 8\eta_4 + 10\eta_5 + 12\eta_6)].$$

Design Tool

As we described above, the standard MMCX server configuration has been engineered to support the offered load associated with a specific call mix model. The default parameters in that model were chosen to be "reasonable" based on the customer input. However, for each specific customer installation, the call mix parameters may differ from the default model in some way, such as variations in the length of mixed media calls, the number of users per call, or the number of calls per user per hour. Thus, we have created an MMCX design tool to configure a server for a specific customer call mix model.

The design tool is an interactive statistical modeling package that incorporates call control and media stream analyses, as we described above. The design tool is implemented with Visual Basic to run on a Windows-based PC.

The design tool takes as input a customer-specified call mix model. Using the analysis described above, the design tool calculates the offered load on the MMCX server for that model and produces a specific server configuration that will support the performance requirements described above. In support of the ordering process, the design tool produces an itemized list of the detailed components for the server, such as plug-in cards for both protocol processing and network interfaces.

Finally, the design tool calculates the average and maximum number of simultaneous calls of each of the media (for example, audio, video, and data) generated by the specified call mix model and the associated LAN and WAN bandwidths. This information enables a customer to plan any necessary network upgrades to handle the mixed media traffic that will be generated when the system is installed.

The MMCX server also supports system management commands that allow customers to control the use of their WAN bandwidth. For example, the design tool calculates the maximum WAN bandwidth generated by video calls for a given type of customer-specified call mix model. If the users of the system were to place more video calls than were assumed in the call mix model, more WAN bandwidth would be required. A customer that is concerned about the WAN bandwidth costs can restrict the MMCX server, via the system management commands, from ever using more WAN bandwidth for video than the design tool predicts is necessary.

Conclusions

In this paper, we presented an overview of MMCX, a premises mixed media communication product being offered by Business Communication Systems of Lucent Technologies. Many important performance design issues arise when MMCX is incorporated into a customer's LAN and WAN networks. For example, strict delay requirements exist on various components of call setup procedure, such as the post-dialing delay, cut-through delay, and tear-down delay. We also identified the requirements on end-to-end delay for transferring the audio, video, and data packets through the local and wide area networks.

We defined a mixed media call mix model to characterize the customer's mixed media traffic. The underlining concept behind developing this model was to partition the traffic into four classes and define some mixed media performance parameters for each class. To evaluate and verify the performance of the customer's system for a given type of network configuration, we developed mathematical models and then extended them to evaluate the impact of MMCX traffic on the customers' local and wide area networks. The

input to these models was based on performance parameters defined in the call mix model.

Using the Microsoft Visual Basic software environment, we developed a performance design tool that incorporated all the analyses. This tool is used to configure an MMCX server when it is incorporated into the customers' networks (for example, to evaluate the number of cards and network interfaces for the MMCX server). The design tool also evaluates the impacts of MMCX traffic on customers' networks by calculating how much load the MMCX will add to the customers' local and wide area networks.

Acknowledgments

We thank Elise Amorosi, Phil Harris, and Kurt Haserodt of Bell Labs for their contributions to the MMCX architecture overview section. Also instrumental in the project were Christine Kleiwerda of Bell Labs and Patrick Opiyo, a student intern from the University of Colorado, who together implemented the MMCX design tool.

*Trademarks

HP is a registered trademark of Hewlett-Packard Company.

Sun is a registered trademark of Sun Microsystems Inc.

UNIX is a registered trademark of Novell.

Visual Basic, Windows, and Windows NT are registered trademarks and Windows® 95 is a trademark of Microsoft Corporation.

References

1. ITU–T Recommendation H.323, "Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service," International Telecommunication Union–Telecommunication Standardization Sector, 1996.
2. ITU–T Recommendation H.320, "Narrow-Band Visual Telephone Systems and Terminal Equipment," International Telecommunication Union–Telecommunication Standardization Sector, 1996.
3. S. R. Ahuja, H. F. Haisch, R. S. Ramamurthy, and L. M. Sanders, "Multimedia Collaboration," *AT&T Technical Journal*, Vol. 74, No. 5, Sept./Oct. 1995, pp. 46–53.
4. "Architecture and Network QOS, Part II: Multimedia Conference Requirements," Section 7.1.1, Multimedia Communications Forum, November 3, 1994.
5. ITU-T Recommendation G.114, "One-Way Transmission Time," International Telecommunication Union-Telecommunication Standardization Sector, 1993.
6. H. Takagi, *Queuing Analysis*, Vol. 1, North-Holland, Amsterdam, 1991.

(Manuscript approved June 1997)

ANWAR I. ELWALID is a member of technical staff in the Mathematics of Networks and Systems Research Department at Bell Labs in Murray Hill, New Jersey. His research interests include networking, multimedia traffic, and queuing and stochastic systems. Mr. Elwalid received a B.S. degree from Polytechnic Institute of New York, Brooklyn, and M.S. and Ph.D. degrees from Columbia University, New York City, all in electrical engineering.

GLEN G. FREUNDLICH is a distinguished member of technical staff in the Multimedia Communications Exchange Department in Lucent's Business Communications Systems in Westminster, Colorado. He is responsible for the architecture, design, and development of multimedia communications systems and is also involved in the definition of standards related to multimedia communications. Mr. Freundlich received a B.S. in computer science from Colorado State University in Fort Collins.

PAUL M. GERHARDT, a technical manager in the Multimedia Communications Exchange Department in Lucent's Business Communications Systems in Westminster, Colorado, received a B.S. in computer science and mathematics from the State University of New York at Binghamton and an M.Eng. in computer science from Cornell University, in Ithaca, New York. His group is responsible for the networking, protocol, and performance aspects of the MMCX platform.

HASSAN HAGIRAHIM, a member of technical staff in the Cross-Platform Customer Solutions Department in Lucent's Business Communications Systems in Middletown, New Jersey, works on multimedia performance analysis, asynchronous transfer mode network performance analysis, and design and implementation of performance tools. He received a B.S. in electrical engineering from Kings' College and a Ph.D. in electrical engineering from Imperial College, both part of the University of London in the United Kingdom.

K. G. RAMAKRISHNAN is a distinguished member of

technical staff in the Mathematics of Networks and Systems Department at Bell Labs in Murray Hill, New Jersey, where he works on mathematical optimization, integer programming, routing in asynchronous transfer mode networks, and wireless applications. He earned a B.S. in electrical engineering from the Indian Institute of Technology, Kanpur, India, and M.S. and Ph.D. degrees in computer science from the State University of Washington, Seattle.

DAVID TSE, a member of technical staff at Bell Labs

when this work was performed, is now an assistant professor in the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. His research interests are in traffic management and resource allocation problems in broadband and wireless networks, focusing on quality of service provisioning. He received a B.A.Sc. degree in systems design engineering from the University of Waterloo, Canada, and M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge. ◆