Design and Analysis of a Multimedia Network Architecture

Robert W. Christie
University of Virginia

Computer Science Report No. TR-95-27
May 12, 1995

Design and Analysis of a Multimedia Network Architecture

Robert W. Christie

Department of Computer Science
University of Virginia
Charlottesville, VA
rwc9u@Virginia.edu

Abstract

A paradigm shift is underway in how computer networks are used. The new breed of applications that incorporate multimedia add a new dimension of complexity to network management because their service requirements are quite different from those of older applications. The problem lies in the service discipline of the network, where routers generally use a single queue which operates in a first-come-first-served manner. As traffic through the router increases, packet delays become longer to the point where the router's internal queue overflows and packets are dropped. This project addresses the design of a packet-switched network that can support soft real-time applications such as multimedia. We focus on the communications protocol, the router design, and the resource reservation and admission control policies that will allow the overall system to operate. We provide a survey of related work on design issues concerning network service paradigms, traffic policing mechanisms, resource administration mechanisms, and resource reservation protocols. We then discuss the design and implementation of our router, end-systems, and admission control policies. We show that with the proper choice of protocols, traffic policing, and resource reservation within the router, a packet-switched network can support guaranteed quality-of-service for multimedia communications.

Acknowledgments

There are a number of people deserving appreciation for their help in bringing this work to fruition. First, I would like to thank our sponsor, NRaD, for funding this research. Second, I would like to acknowledge my colleagues in the Networks Lab. Specifically, I want to thank Dallas Wrege and Bert Dempsey for many fruitful discussions; Fraser Street and Matt Lucas for the use of their video distribution system; and James McNabb for his invaluable assistance on this project. In particular, I would like to thank James for the use of his implementation of XTP 4.0, his work on the initial packet-switch, and countless discussions. Thanks are especially due to my advisor, Alf Weaver, whose guidance and straight talk helped to shape this project. I specifically appreciate the chance he took when hiring me as an undergraduate. Above all I would like to thank my parents for their support; and my wife, Jenny, whose support, love, and patience helped me more than she knows.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Historically, computer networks such as the Internet have been used primarily for file transfer and electronic mail. These applications are characterized by a need for accuracy (that is, the bits that are transmitted must arrive correctly), but they do not have any stringent latency requirements (that is, the correctness of the data does not depend upon the time required for its delivery). This is not true, however, of the new breed of applications that utilize multimedia.

Multimedia streams add a new dimension of complexity because their delivery requirements are quite different from those of file transfer. For multimedia, the emphasis is more on timely delivery than on absolute accuracy. A single bit error in the data that defines a video frame buffer is unlikely to be noticed at all, but the late delivery of video frames causes the video image to alternately freeze and then jump ahead, a process which rapidly degrades picture quality.

In a local area network environment, digitized and compressed multimedia streams can be made to perform well by careful system design which ensures that:

(a) the transmitting CPU is not overloaded, so that the transmitted data stream is emitted with low jitter,

(b) the receiving CPU is not overloaded, so that incoming data can be processed and displayed as soon as it is received, and

(c) the aggregate requirements of all the network traffic are well below the capacity of the network, again to minimize jitter in the transit time of the data.

However, when we switch from local area networks to wide area networks by introducing network routers, the situation changes dramatically and the system designer loses control. Most commercial Internet routers simply run the Internet Protocol (IP) in a first-come-first-served fashion; that is, packets are routed in the order they are received, without

regard to their intrinsic importance to the overall system. A congested router drops data when its internal queues are full, thereby causing gaps in the delivered data stream. This lost data translates into incomplete frame buffers which in turn means that, depending on hardware design decisions in the receiver, video frames are either displayed incorrectly or are simply skipped.

The solution to transmitting high-quality multimedia streams in packet-switched networks is four-fold:

(1) As stated above, the overall capacities of the transmitting CPU, receiving CPU, and network must be properly sized by the system designer.

(2) The communications protocol must have some mechanism whereby it can mark the relative importance of the network traffic it handles; for example, it may wish to mark time-sensitive multimedia traffic as being inherently more important than non-time-sensitive file transfer operations.

(3) The network routers must operate in a priority-sensitive manner, rather than just FCFS, so that higher priority traffic is given preferential treatment.

(4) The end-systems and routers must operate as a unified system in which the end-systems can reserve the resources of the router and the router can apply admission control to the end-systems that wish to use its services.

Our goal, then, is to design a packet-switched network that can support soft real-time applications such as multimedia. We assume that system capacities are intelligently allocated by the system designer (point number one above), and thus we focus on the requirements of the communications protocol, the design of a router that recognizes multiple priority traffic streams and responds to them in accordance to their defined importance, and the resource reservation mechanisms and admission control policies that make the overall system work.

1.2 Background Information

Before continuing with a discussion of the overall system design considerations, we want to present a general communication model, and discuss relevant issues concerning computer networking and communication protocols.

A *communication network* is a system that is used to transfer information between two or more devices. A communication network usually consists of end nodes known as *hosts*, and

internal nodes, also known as *network nodes*. Communication occurs between two hosts over a physical medium such as coaxial cable or fiber optical lines. In large computer networks hosts may not be directly connected, so data is routed between two hosts via the internal network nodes.

Communication networks are usually categorized by the way the nodes in the network exchange information. Two categories of networks exist, switched networks and broadcast networks. *Switched communication networks* transfer data from host to host via intermediate nodes. The purpose of these intermediate nodes is to facilitate the moving of the data from the source host to the destination. *Broadcast communication networks* do not use intermediate nodes in the transfer of data. Each host contains a transmitter and receiver, and communicates over a shared medium.

Switched communication networks are further subdivided into circuit-switched networks and packet-switched networks. *Circuit-switched networks* are characterized by the establishment of a dedicated link between two hosts where each link receives certain resources. The typical example of a circuit-switched network is the telephone network. *Packet-switched networks* send data in chunks, or packets, with each packet being routed from node to node until the packet reaches its destination. In contrast to circuit-switching, each packet is received, stored, and then routed to its next destination, a *store-and-forward* approach. Computer networks primarily consist of packet-switched networks.

Packet-switched networks may be classified as either virtual circuit networks or datagram networks depending on how the path between two hosts is selected. *Virtual circuit networks* determine a route between two hosts during the setup of a communication. All packets are then routed along that path resulting in-sequence delivery of all packets. *Datagram networks* make routing decisions on a packet-by-packet basis, which could lead to out of sequence packet delivery. The re-sequencing of the data must then be handled by the destination.

Because the communication process is complex, it is useful to subdivide the process into smaller tasks which act in conjunction to allow computers to communicate. These tasks are known as protocols. A *protocol* is a set of rules and conventions that all systems use in order to communicate. A structured set of protocols implements a *communications architecture* which allows data to be passed between multiple computers. The

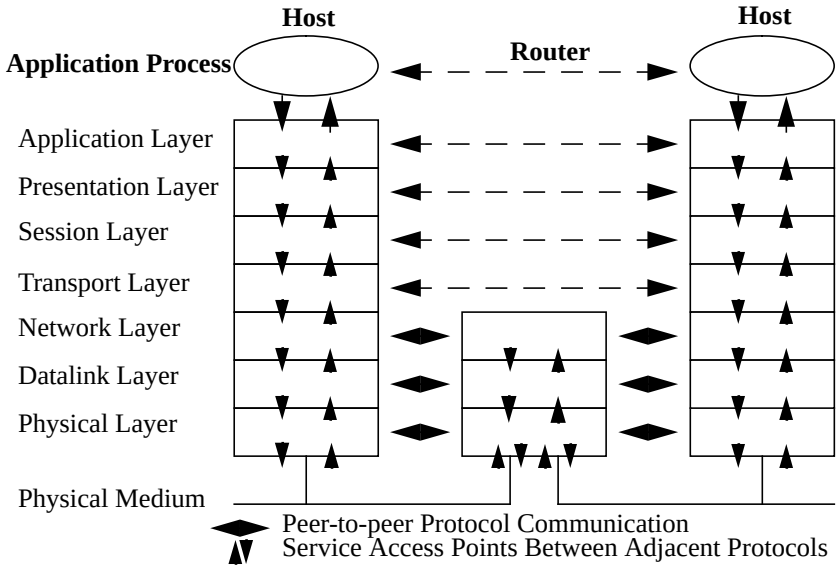

Figure 1. OSI Reference Model

Figure 1 shows the interactions of the OSI Reference Model protocol stack. Two host nodes are shown which are connected via a network node. Adjacent protocols in the stack communicate through specified service access points. The logical communication between peer protocols is represented by dotted arrow lines.

communications architecture is generally viewed as a layered hierarchy of services where each layer, N, only uses the services of the lower layer, N-1, and only provides services to the next layer above, N+1. This architecture is also referred to as the protocol stack, where peer protocols on different hosts logically communicate, but the services the protocol provides are only accessed and used by the adjacent protocols within the stack. A widely used model for describing network communication architectures is the Open Systems Interconnect (OSI) Reference Model [14]. Figure 1 shows an example data path in terms of the OSI model.

The OSI Reference Model defines a seven layer architecture for computer communication. A brief summary of the seven layers follow:

- The *application layer* provides access to the protocol stack for application processes, while also structuring the lower level services to suit the needs of the application process.
- The *presentation layer* ensures that information delivered to the application layer is useful by resolving differences in syntax between heterogeneous host systems. Encryption may also be performed in this layer.
- The *session layer* provides for the organization and synchronization of data exchanges between applications. Establishment, management, and termination of connections is done here.
- The *transport layer* provides a reliable transfer of data between two hosts. Seg-

mentation and reassembly of higher layer messages is done at this level, and mechanisms for end-to-end flow and error control are also provided.

• The *network layer* provides the switching and routing functionality used to connect systems, and also shields the upper layers from differences in the underlying data transmission facilities. The layer is also responsible for establishing, maintaining, and terminating connections between communicating end-systems.

• The *datalink layer* provides for the reliable transfer of data over a physical link. This layer provides services such as framing of data, error detection and correction, flow control, and error control.

• The *physical layer* provides access to the physical medium of transmission. This layer encodes and decodes the bit stream to/from the signalling mechanism used over the physical media.

The result of using the protocol stack is the availability of a group of services that applications on heterogeneous systems may use in order to communicate. Data may be passed from one application down through the protocol stack and then across the physical media, through (possibly multiple) internal network nodes, and then on to the destination host where it will be passed back through the destination host's protocol stack before reaching the application. If the *data path* or flow of the information goes through any network nodes, then the data will be passed up to the network layer of each node's protocol stack so that a decision may be made on where next to route the data. As seen in Figure 1, the outcome of following this data path is the logical interconnection and communication of two applications.

While the OSI Reference Model gives a good overall view of computer communications, actual applications tend to mesh multiple layers together. The lower four layers of the OSI reference model represent the modern protocol stack found in most computers. Typically, applications access the transport layer through a well defined interface such as Berkeley Sockets, Streams, or some other interface. The application then supplies what upper layer functionality it needs. Similarly, lower layer protocols do not always have well-defined subdivisions. Often protocol optimizations for specific systems blur the barriers between the protocols. For instance, it makes little sense to copy a data message each time one passes the message to the next protocol in the stack, because the overhead incurred by these multiple copies would slow the communications process.

The rest of chapter one will address certain considerations that affect the design of a communications architecture that is suitable for multimedia traffic.

1.3 Design Considerations

During the design process of this multimedia system, certain considerations had to be kept in mind because of their effect on overall system performance. Such issues as latency, network access, quality of service, and end system design interact with each other and thus affect overall system performance.

1.3.1 Latency Considerations

Multimedia applications differ from many other applications because of their *soft real-time* requirements. *Soft real-time* refers to the system performance being time sensitive, whereas a *hard real-time* system has performance that is time critical. Both digital video and voice communication must have periodic updates of information, otherwise the picture or sound quality will degrade. This behavior is markedly different from an application such as a file transfer which has no latency restrictions, and only needs to be delivered without error. In order to better understand the process involved in delivering a high quality video image, it is instructive to step through the data path of a typical video transfer application.

A video stream will typically begin as an NTSC (National Television Standards Committee) signal from some video source such as a video-camera, VCR, television, or laser disk player. An NTSC video produces 30 frames per second, with each frame consisting of 525 lines. The video signal is then changed from an analog to a digital representation via a hardware analog-to-digital (A/D) converter. Each pixel in the image is represented by a number of bits; today that number is typically 8, 16, or 24. Assuming a video display of 800x600 pixels with 8 bits/pixel, the resulting frame takes 480 kilobytes (kB) of memory. If frames of this size were transferred between computers, then the bandwidth needed for transmitting thirty of these frames per second is over 100 Mbits/sec. Therefore, most multimedia systems use a compression algorithm to reduce the size of the frame buffer.

Various compression algorithms exist, with JPEG and MPEG being two of the more widely used algorithms. The basis of all compression strategies is a fast algorithm that allows for most of the information contained in an image to be encoded in a smaller format. For example, JPEG (Joint Photographic Experts Group) uses a Discrete Cosine Transform to change an 8x8 pixel tile into a frequency representation. This frequency representation of

the image is then passed through a filter which removes frequencies that will not be missed. The frequencies are also quantized in order to save more space. The remaining frequency values are then processed via a Huffman encoding algorithm which reduces redundancy in the representation of the frequencies. Depending on the amount of quantization done, the resulting compression generally decreases the size of the frame buffer by an order of magnitude or more.

Once a frame has been digitized and compressed, the network protocol stack receives that buffer so that it may be transmitted to the destination host. Once the buffer is at the transport layer it is segmented into well-formed packets, then passed down through the lower protocol layers, and finally transmitted on the physical medium. Depending on the network setup, the packets might have to pass through several intermediate nodes before reaching the final destination. Once at the destination host, the process just discussed is followed in the reverse order, resulting in a frame being displayed on a video device.

One factor of great importance that was not specifically mentioned in the discussion of the multimedia datapath was the latency constraints that bound the entire process. In order for video to appear normal each frame must be digitized, compressed, transmitted, received, uncompressed, and then converted to the proper signal for the video display device within 1/30 of a second. If network load is heavy and frames are lost, then they either have to be retransmitted (and the destination buffers possibly reordered) to meet the deadline for presenting the frame, or the process must proceed even with some data missing. Depending on the equipment and compression algorithms being used, lost data may only result in slight differences in color from the original image or a slight fuzziness in previously sharp lines. However, the loss of data might have more severe consequences such as the loss of a full frame. Thus, the delay of a frame buffer in its arrival at the receiver can only have negative consequences.

1.3.2 Network Access Considerations

Another consideration that will affect the performance of the multimedia system is the type of network that will be used. From an engineering viewpoint there are four main possibilities from which to choose[1]:

- Ethernet
- Token Ring
- FDDI
- ATM

Ethernet is probably the most widely known type of local area network. The network medium access control (MAC) uses a CSMA/CD (Carrier Sense Multiple Access with Collision Detection) protocol for access on a 10 Mbit/sec coaxial cable. This type of access control for the network means that each host may attempt to send a packet at anytime. Also, the host will always be listening to the medium in order to detect any other transmissions. If the host detects that another host is currently transmitting then it will wait until the physical medium is no longer busy before transmitting. If two or more hosts transmit and there is a collision of packets, then each host will detect the collision and back-off for a certain amount of time, and then attempt to retransmit the packet. Due to the nature of the network, the upper bound on the time it takes to detect a collision is twice the round-trip delay (the time it takes the signal to travel from one end of the medium to the other)[31].

Ethernet is widely used because the hardware is inexpensive, but reliable, and most people's current networking bandwidth requirements do not exceed Ethernet's 10 Mbits/s capacity. However, a multimedia application using JPEG compression and a Q factor (a number representing the quantization level) of 60 needs approximately 5 Mbits/s. Therefore, even a two video streams can saturate Ethernet's bandwidth. Another shortfall of Ethernet is that there is no support for guaranteed service inherent in the media access. For instance, multiple packet collisions could cause long delays between the sending of two packets from the same host.

There are currently two versions of token ring available, the first version

1. These choices would be equivalent to choosing a certain a datalink layer, physical layer, and physical medium in terms of the OSI Reference Model. There are other possibilities, but these are the most widely used, and are readily available.

IEEE802.4, runs at 4 Mbits/s, and the second version, IEEE802.5, runs at 16 Mbits/s. The token ring standards run over a number of mediums both copper and fiber based. The token ring protocol uses a “token” packet to decide who has access to the ring. If no station desires to transmit then the token is passed around the ring in a “free” state. If a station then desires to transmit, it must wait until it sees the free token. Upon seeing the free token the station changes it from “free” to “busy”. The station then releases the now busy token followed immediately by its packet. No one else may transmit data at this point, because the token is busy. The data packet and busy token will circle the ring, and will then be removed by the sending station, which will then insert a free token onto the ring so long as the following two criteria have been met:

- transmission of the packet has been completed by the station, and
- the transmitting station has received the busy token.

The previously transmitting station may now release a free token which will let another station transmit. The Token Ring standard also supports a capacity allocation scheme using different priority levels. Token Ring has eight priority levels which may be assigned to outgoing packets. A reservation scheme allows a station with the highest priority packet to reserve the next free token [32]. One problem with the reservation policy is that it can result in starvation of ring access for stations trying to send lower priority packets. This situation could occur if there is always another station that has a higher level packet to send. However, if there are no requests to send higher priority traffic then the last station to send packets at that priority level will lower the priority of the token to its previous level so that lower priority packets may be sent.

The Fiber Distributed Data Interface (FDDI) uses an optical medium, and employs a token ring protocol. Operating at 100 Mbits/s, an FDDI ring is faster than Ethernet (10 Mbit/s) by an order of magnitude. Originally, FDDI was developed for use as a high speed link between main frames and large storage devices (a back-end interface), as a LAN backbone, and as a front-end for LANs which needed more performance than was available via Ethernet [28].

FDDI uses a token ring algorithm based on the IEEE 802.5 standard. However, the FDDI algorithm differs in some ways so that it may maximize its efficiency. FDDI stations contend for ring access via the use of a small “token” packet that circulate the ring. When

there is no contention for the ring, then the token packet is labeled as free. If a host wishes to transmit then it waits until it sees the token packet. Once the host receives the token packet it may begin transmission of its frame. The host is limited in how long it may transmit by timers within the FDDI media access control layer. When the host finishes its transmission it posts a new free token immediately after its last frame. Once this occurs the next host downstream from the sender will receive a free token giving it the opportunity to transmit data. By using this notion of a free/busy token only one host will ever have control of the ring. FDDI also supports multiple traffic priorities through the use of restricted and unrestricted control tokens, and a capacity allocation mechanism that supports both synchronous and asynchronous traffic [32].

Synchronous traffic uses certain mechanisms in FDDI to guarantee a bounded latency between two receptions of a free token. *Asynchronous traffic* offers no such guarantees. The guarantees of the synchronous mode result from the use of the Timed Token Rotation protocol, and a set of timers and counters located in each host. During an intitialization phase of the ring a bidding process occurs to decide the value of the target token rotation time (TTRT), which is the lowest value bid from all stations. The TTRT is the desired latency for the free token to rotate around the ring once. Each host must request a synchronous allocation block via the station management protocol (SMT). If a host does not support synchronous allocation (it is optional), then the host may only transmit in the asynchronous mode. In synchronous mode, if a host has synchronous frames that are ready to be transmitted, then it may capture the next available token, and transmit for a certain allocated amount of time. In asynchronous mode the host may only capture a token if the time that has passed since the last time it saw a token is less than the TTRT [29].

The benefit of using the timed token rotation (TTR) protocol is that a host can negotiate bandwidth and latency guarantees for its synchronous traffic. The initialization process of the ring accepts the lowest bid as the TTRT, thus ensuring that there is a low guaranteed response time for synchronous traffic. The worst case will have the token arriving no more than two times the TTRT after the last token arrived [29]. Typically, the TTRT is set to 8 ms, because this value was found to provide good service and performance for varying loads and system configurations [20].

A newer technology that has generated much interest in the last few years is

Asynchronous Transfer Mode (ATM). ATM offers a connection-oriented datalink service that provides guarantees for bandwidth and delay. Unlike both Ethernet and FDDI which use a datagram networking facility for routing packets through multiple internal nodes, ATM uses virtual circuits. Thus, during a connection setup ATM negotiates a path between the two end hosts, and then all packets for that connection are routed over that path, unless there is a failure along that path which means the route must change. Because of the flexible nature of the ATM specification, there are a variety of maximum bandwidths available depending on the type of the physical medium, and the signalling architecture used. For instance, using optical fiber and the OC-3 signalling architecture yields a maximum bandwidth of 155 Mbits/sec with ATM.

ATM was designed with the objective in mind of supporting a variety of different traffic types. Its small, fixed packet size (53 bytes) minimizes the delay experienced by each packet in transit from end host to end host, and the virtual circuit connection setup can set certain service parameters so the connection has certain guarantees. A key feature of ATM is the notion of the virtual circuit. In ATM, bandwidth reservation is flexible and dynamic. It does not reserve specific positions for data in its stream, the way that a time division multiplexed (TDM) system would. Instead, the virtual circuit reserves bandwidth over a period of time without specifying any position in the stream. With these types of guarantees, ATM seems to be the ideal mode of communication for traffic such as voice, video, and distributed real-time control systems [1].

Each of the four networking access methods mentioned above have their positive and negative sides. However, Ethernet offers no mechanisms to support either latency or bandwidth guarantees, which are needed in a soft-real time communication system. This leaves either ATM, Token Ring, or FDDI as the possible choices for network access. A shortfall of FDDI and Token Ring is that they both use packet-switching for routing over multiple subnets. Most current routers use the Internet Protocol (IP) for the routing of packets. IP is a connectionless, best-effort protocol which allows for little (or no) QoS support within the routers. So even though the FDDI standard supports deterministic latency guarantees on a single subnet, this functionality is lost when one sends information across multiple subnets. ATM, on the other hand, features a connection-oriented datalink that uses virtual circuit switching and has mechanisms for guaranteeing bandwidth and prioritizing traffic between two end hosts.

However, no matter how good ATM might be, there is still a vast amount of communication equipment and software that is currently in use. These legacy LANs will not be replaced overnight. Thus, a challenge for the ATM vendors and network planners is how to integrate a connection-oriented service like ATM into the connectionless LANs that are in use today [1].

In terms of trying to implement a multimedia communication system, ATM seems to be better suited for the task then does a packet-switched datagram network. However, ATM is still in its infancy. The standards are not completely set, so many of the products use proprietary systems, and as a result are not fully interoperable. While the specification supports rate control and prioritization, some implementations are only now supporting this functionality. It appears that acceptance of ATM is growing, but it will be years before it becomes ubiquitous. For instance, the U.S. Navy has spent the last ten years refining the SAFENET architecture (MIL-STD-2204) which uses FDDI. Instead of changing directly to ATM, they have taken a wait-and-see attitude. Token Ring is a viable access method for a multimedia communication architecture, however; we did not have immediate access to equipment, and FDDI offers almost the exact service with certain deterministic guarantees for its synchronous traffic class. With regard to these considerations, our choice for network access was FDDI, resulting in a system design focusing on delivering guarantees for services in a packet-switched datagram network.

1.3.3 Quality of Service Considerations

Quality of Service is the guarantee of a certain level of performance in the network system. In general, it is difficult to make guarantees in a network due to the problem of just trying to answer the question of whether the network can provide the services requested by a new connection without violating the current QoS guarantees made to existing connections within the network. In order to answer the question the network must know if it can handle the new request, and also what effect this new request will have on services provided to other connections in the network [22]. This type of problem is known as one of *performance oriented admission control* [40]. Once the network has decided to accept a connection there must be certain underlying mechanisms that ensure that the service guarantees are met. These mechanisms are usually implemented in the lower four layers of the OSI Reference

Model. In general, you must have a service guarantee at each of these layers in order for any guarantee to be made at the next higher level.

Various mechanisms are used to ensure service guarantees such as prioritizing certain connections, reserving resources along the data path between two end hosts, and utilizing traffic control mechanisms like modifying the rate of flow through the network, and between end nodes. Prioritizing certain connections in the communication architecture will decrease the latency incurred in transfer for those connections when compared with non-priority data as long as there are mechanisms throughout the communication architecture that make prioritization possible. Resource reservation in a communication architecture usually refers to the need to have buffer space and other protocol dependent structures available for that connection at all times. Traffic control is the ability to modify the shape and flow of traffic in the network system. Without this ability a host or node could overrun a receiver with too much information.

Today's packet-switched networks such as the Internet use the TCP/IP protocol stack for most of their networking needs. IP has little in the way of QoS, because it is based on a connectionless, best-effort type of service, and was developed years before QoS was even an issue. TCP is a connection-oriented protocol, but also has very little QoS support. However, there are many proposals for QoS protocols and architectures. Three such efforts are noteworthy because they have progressed past paper designs.

ST-II is a connection-oriented internetwork protocol with several running implementations, although certain parts of the protocol such as the *FlowSpec (*which indicates resource requirements) and real-time admission control algorithms are still undefined [36].

Another newer protocol, RSVP, which is to be used as a companion protocol for IP, allows for the exchanging of resource reservation messages. This protocol is currently in the implementation stage [42].

The Tenet Real-Time Protocol Suite is currently running and being ported to a number of new testbeds. The Tenet protocols use admission control, connection-oriented communication, and channel rate control to support *real-time channels.* However, one simplification that was made for this suite of protocols was to provide only unicast real-time channels. An implementation with support for multicast channels is currently underway [2].

The general theme found throughout the related work is that if packet-switching is used for communication between two end hosts, then it is necessary to reserve resources for that communication connection if QoS is desired. In order to reserve the resources, admission control to the network is needed.

1.3.4 End System Considerations

For simplicity in design, end systems were assumed to behave in a normal way, i.e. they are not overloaded. However, a protocol stack with mechanisms to support QoS must have an operating system with services that facilitate the use of the protocol's mechanisms. For instance, a protocol might have mechanisms for prioritization of data, or guarantees of delay bounds, but if the operating system has no sense of *real-time*, then an application might not realize any of these guarantees.

The original UNIX operating system has different processes that share the CPU in a *time-shared* manner, where each process runs for a certain period of time and then the kernel of the OS suspends that process, and schedules a new process to run for a certain time period [3]. Many of today's versions of UNIX still use some sort of time-sharing scheduling. However, for an OS to support real time applications it must have a bounded worst case delay for the servicing of the real-time processes. One way to get real-time support is to design the operating system so that all processes are preemptible and have scheduling done on the priority of each process. In this way the highest priority process will always gain control of the system, and therefore bounds on latency can be determined for this process. Without this support for a multimedia application, there is no guarantee that a frame buffer that is received before its deadline will update a video output device by its deadline. Thus, in a multimedia communication architecture there is a need for a real-time operating system.

Current work at the University of Virginia involves the porting of an entire communication protocol stack to a real-time OS. After an extensive study [30] of the available real-time operating systems, LynxOS was chosen to be the operating system for the port. Lynx already supports a modified version of the BSD Tahoe TCP/IP protocol suite. The work at UVA involves porting the Xpress Transport Protocol (XTP) version 4.0 to the LynxOS environment. XTP is a next generation transport protocol with functionality

that is needed by real-time applications. XTP supports multiple priority traffic, multicast connections, rate control, and various other policies that give the user rich functionality from which to choose.

1.4 Organization of Thesis

In this chapter we reviewed background information on network communication systems and multimedia system design considerations. Chapter 2 discusses related work in the areas of multimedia system design and packet-switched network architectures with QoS. In chapter 3 we describe the router design, and analyze its bottlenecks. We discuss the admission control strategy used in the router in Chapter 4. Finally, we present our conclusions and future work in Chapter 5.

Chapter 2

Related Work

2.1 Overview

Many efforts have been documented in the literature on multimedia systems, including the incorporation of QoS into a communication architecture. The two topics tend to be intertwined, because many multimedia applications are sensitive to the quality of service that their packets receive from the underlying communication architecture. This chapter reviews related work in multimedia system design, emphasizing the architects' choices which relate to the design considerations outlined in the first chapter of this thesis.

In the first section of the chapter we review the mechanisms which allow a network to offer deterministic service guarantees. This section focuses on the admission control tests and policing mechanisms which ensure the deterministic guarantees. In the second section of the chapter we discuss the Internet Protocol (IP) and its shortcomings in offering support for QoS. Then we discuss two internetworking reservation protocols that do offer

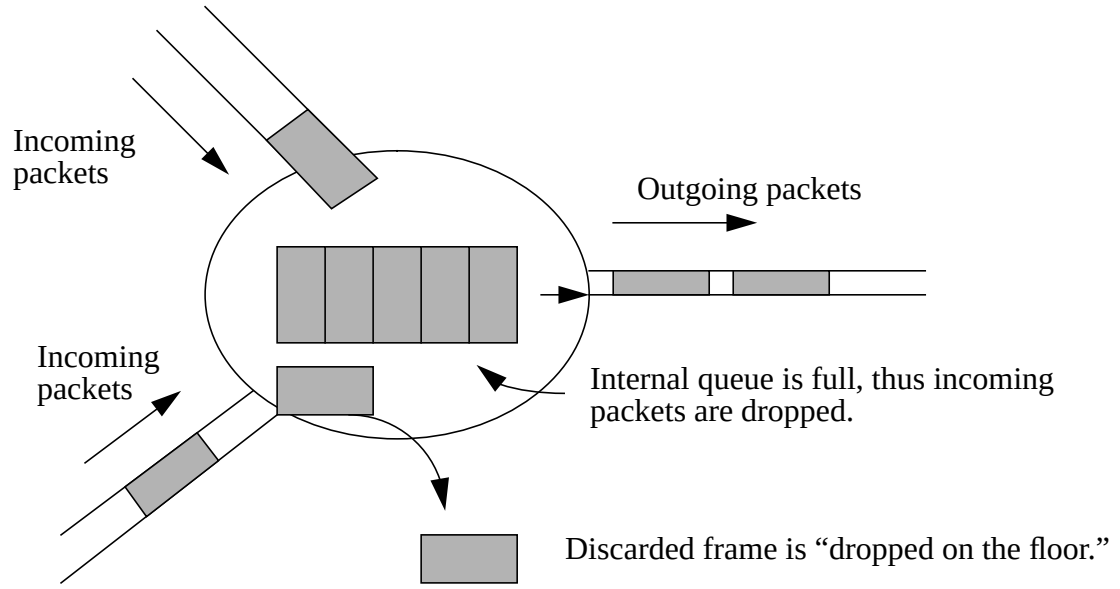

Figure 2. Visual Depiction of Congestion in a Router

Figure 2 depicts congestion in a router caused by two higher bandwidth links attempting to send traffic over a link with less capacity.

support for QoS, the Internet Streams Protocol Version II (ST-II) and the Resource Reservation Protocol (RSVP). The third section of this chapter describes previous designs for multimedia communication architectures. The amount of literature on this topic is quite large, so in order to facilitate a manageable discussion of the topics, we have tried to limit the review to areas that have resulted in a working implementation of a system or protocol. We describe two communication architectures which offer various levels of service for multimedia data streams. These two architectures are the Tenet Real-Time Protocol Suite, and the Capacity-Based Session Reservation Protocol (CBSRP) in conjunction with the Advanced Real Time System (ARTS).

2.2 Admission Control and Traffic Policing Mechanisms

Once information leaves the confines of a packet-switching Local Area Network (LAN), then there is a need for a special purpose piece of hardware or software called a router for the transmission of information from one subnet to another. With the introduction of this new hardware into the communication architecture one also introduces the potential of having new bottlenecks in the system. One potential bottleneck is that the router can become congested to the point where delays are extremely long, or packets are getting dropped by the router due to lack of buffer space within the router. Figure 2 shows a visual depiction of this congested state.

In [19] Jacobsen describes the “congestion collapses” that occurred on the Internet in 1986 which caused throughput to drop from 32 Kb/s to 40 b/s. These collapses were caused by misbehavior in the TCP implementations and by the inability of end system protocols to detect a congested situation at a router. This resulted in end systems continuing to transmit the same amount of information even though the router was congested and dropping packets. Jacobsen proposed seven changes to TCP in order to allow it to deal with the dynamic nature of the networks, and avoid congestion. The result of these changes was that TCP could dynamically detect congestion and then adjust the window sizes and retransmission times of its sender in order to throttle the amount of information being sent over a link in order to avoid congestion. These reactive techniques were implemented through the use of a number of new algorithms including slow-start, round-trip-time variance estimation, dynamic window sizing as a result of congestion, and exponential

retransmission backoff.

While these techniques helped to avoid congestion in networks, they are not completely efficient because they do not communicate directly with the point of the bottleneck, but instead infer from round-trip estimations and time-outs of timers that the network route is suffering from congestion. Other work such as Chiu and Jain's research on increase/decrease algorithms for congestion avoidance [9] and Floyd and Jacobsen's work on RED (Random Early Detection) gateways [13] suggest the use of direct feedback from the routers to dynamically detect and avoid congestion. However, all of these methods are still reactive, adjusting to the increase in network use and the onset of congestion by throttling the sources in one manner or another. This dynamic change of flow and rate might be fine if the application is retrieving a file from a remote host, because the result would be a delay in retrieval time. However, if the application is a video teleconferencing tool, then the delay will cause frames to miss their deadlines, resulting in a lower frame rate. Thus, the application needs service guarantees from the network if lower quality or large changes in quality are not acceptable. In the past, research on guaranteeing service in a network has followed two paths, *deterministic* guarantees and *stochastic* guarantees. A stochastic guarantee defines services that have a certain probability of being violated, whereas deterministic guarantees are never violated. Typically, the reason for choosing a stochastic guarantee is that higher network utilization can be achieved; however Knightly et al. [21] demonstrate that a "considerably high network utilization is achievable by a deterministic service." For example, a guarantee on a delay bound for a specified connection is a deterministic service. We have focused on deterministic guarantees because of the soft-real time requirements of our system design.

In order to offer a deterministic guarantee, network resources need to be reserved for use. Thus, there must be an entity which controls access to the network accepting requests if resources are available, and either rejecting requests or negotiating with the user if resources are not available. Admission control tests are the mechanism used to determine if the network can allocate the needed resources for a request without violating the service guarantees of any current connections. Once a connection is admitted, traffic policing guarantees that a connection does not violate the parameters on which admission was granted. The following two sections discuss past admission control tests and traffic policing implementations.

2.2.1 Admission Control Tests

Admission to a network can be based on a number of factors such as the required throughput, the needed buffer space, or the maximum delay that packets for a connection need. The complexity of the admission control test will depend on the desired service, the accuracy of the test, and the type of service discipline or packet scheduling used at the internal network nodes. For instance, conventional networks use a First-Come-First-Served (FCFS) discipline for servicing packets at a network router. The mechanism for implementing this discipline is not complex, since it can be done with a FIFO queue. However, the trade-off for using this discipline is that only one delay bound could be guaranteed for the entire network. Also, in most conventional networks admission control tests are not performed, so as mentioned previously congestion may develop in the network. These deficiencies in the FCFS discipline have resulted in the design of a number of new service disciplines such as Delay Earliest-Due-Date [12], Stop-and-Go [18], Rotating Priority Queue [24], and Rate Controlled Static Priority Queue [44], each with admission control tests.

2.2.2 Traffic Policing Mechanisms

A traffic policing mechanism ensures that connections admitted to the network do not violate the parameters of service under which they are admitted. For instance, if a network admits a multimedia stream then it must ensure that it can handle the *bursty* nature of the traffic. A multimedia stream such as MPEG uses interframe encoding to compress the video stream. Interframe encoding takes advantage of the similarities between successive frames in a typical frame sequence. However, every few frames (the number is dependent on parameter settings) a frame is sent using only intraframe encoding which typically causes it to be much larger then the surrounding frames. If this frame is sent through a network with little or no interpacket gap, then this burst of traffic could overflow buffers within the network. Thus, the network must have prepared for this situation by allocating for the worst case, and the network policing mechanisms must ensure that the traffic is shaped to conform with the parameters of service that the user requested. Another effect of the policing mechanism is that the closer it maps to the real traffic, then the higher the network utilization may be using that mechanism. A number of traffic policing

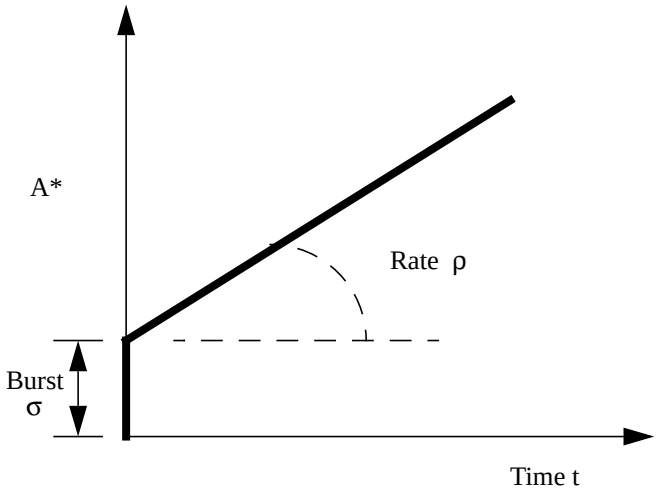

Figure 3. Upper Bound on Traffic for Leaky Bucket

Figure 3 shows the worst case (upper bound) burst and rate possible for a leaky bucket policing mechanism.

mechanisms exist to enforce different admission control requirements. In this section we show three: the leaky bucket, the jumping window, and the moving window.

The Leaky Bucket (LB) mechanism [37] uses a counter and a timer to control the outgoing flow of traffic from a node. In the basic case the counter is incremented by one each time a packet or cell is transmitted, and then the counter is decremented periodically so long as the counter remains above or equal to 0. If the counter reaches a predefined threshold, then it is not allowed to send anymore (the threshold being set via service parameters or admission control requirements). The effect of the leaky bucket mechanism can be described by a traffic constraint function [21]:

$$A^*(t) = \sigma + \rho t$$

A*(t) is an upper bound on the amount of traffic seen during any time interval, t. σ represents the maximum burst that the bucket will allow, and ρ is the rate at which the bucket is updated. Figure 3 shows the worst case scenario (i.e., the upper bound) for a leaky bucket policing mechanism.

The Jumping Window mechanism [18] controls the maximum number of packets or cells that may be sent within a given time interval. During a fixed interval (the window) the transmitter may send up to N packets or bytes. A counter keeps track of the number of packets sent, and stops transmission if the counter reaches 0. The counter is then reset at the end of the interval (size of window). The trade-off between the jumping window and

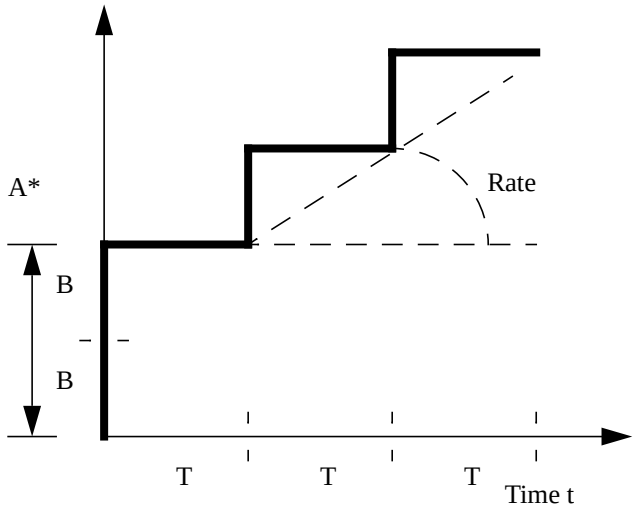

Figure 4. Upper Bound on Traffic For Jumping Window

Figure 4 shows the upper bound on traffic for a jumping window policing mechanism.

leaky bucket is that the leaky bucket requires smaller timer intervals for updates of the counter whereas the jumping window can have a worst case burst of two times the maximum burst per window interval. Figure 4 graphically shows the upper bound (worst case) scenario for a jumping window policing mechanisms.

The equation below shows A* [41], an upper bound on the amount of traffic that may be admitted into the network during a given time, t, when using a jumping window policing mechanism. B is the maximum burst of packets that can be sent in one interval, and T is the size of the window being used.

$$A^*(t) = 2B + \left\lfloor \frac{t}{T} \right\rfloor B$$

Another method of traffic policing is called the Moving Window mechanism. One variant of the moving window mechanism is the X_{min},X_{avg},I,S_{max} [12] policy. This policy enforces an average rate (X_{avg}, the average packet interarrival time) over a fixed interval, I for a maximum packet size, S. During any interval the maximum rate that may be sent is bounded by X_{min} (the minimum packet interarrival time). As with the other mechanism an implementation typically uses a counter to keep track of the number of packets that have been sent within the last interval. The policy also uses a circular queue to keep track of the number of packets that have been sent in sub-intervals of I in order to ensure that X_{min} is not violated. The benefit of using X_{min},X_{avg},I,S_{max} is that it is more flexible than the

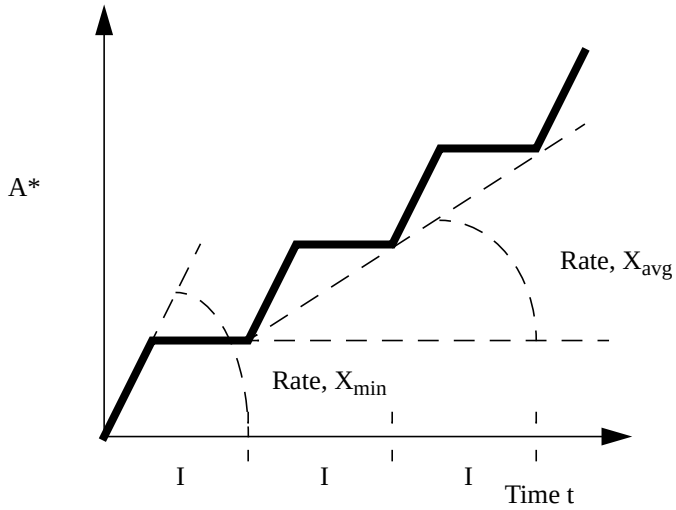

Figure 5. Upper Bound on Traffic For Moving Window

Figure 5 shows the upper bound (worst case) traffic model for a moving window policing mechanism.

jumping window mechanism, because the maximum burst is more tightly bounded. The disadvantage of using the moving window mechanism is that the timers may be of very small granularity if the rate specified by X_{min} is high. The other disadvantage of the moving window mechanism is that it requires more overhead then the two previously mentioned traffic policing mechanisms because of the need to update the circular queue values.

The equation below shows A* [24], an upper bound on the amount of traffic that may be admitted into the network during a given time, t, when using a moving window policing mechanism. X_{avg} is the average rate per interval I that is acceptable, and X_{min} is the maximum rate that may occur in a given interval.

$$A^*(t) = \left\lfloor \frac{t}{I} \right\rfloor (I \cdot X_{avg}) + min\left(\left(t - \left\lfloor \frac{t}{I} \right\rfloor I \right) X_{min}, X_{avg}, I\right)$$

There are a number of tradeoffs that affect the choice of the policing mechanism used in an admission control protocol. For instance, the closer the traffic constraint function is to the generated traffic, the higher the efficiency will be for the system. However, the more accurate traffic constraint functions also require more complex mechanisms for their implementation, which could affect overall system performance. If some of the bottlenecks of the communication architecture are within the router (as is usually the case), then the result of using a less efficient policing mechanism is that router

resources are tied up in each connection, but are not on average being used.

2.3 Networking and Reservation Protocols

The Internet Protocol [26] was developed over twenty years ago by the Defense Advanced Research Projects Agency (DARPA). The overriding goal of ARPA was to develop an effective way to interconnect multiple networks, originally the ARPANET and ARPA packet radio. Below this main goal were seven second-level goals with the three most important being:

- Survivability— communication should be able to continue even with the loss of a network or gateway.
- Types of Service—the architecture should be able to support a variety of services at the transport level.
- Variety of Networks— the architecture should be able to support multiple types of networks.

Of lesser importance were support for distributed management of resources, cost effectiveness, minimizing the cost of attaching to the network, and accountability of the resources [7]. The order of importance of the above goals had a profound effect on the resulting TCP/IP protocol stack. IP provides a connectionless service which needs very little state information to route from one host to another. The result is that the criteria of survivability is met, because packets may be re-routed around a non-operative gateway.

However, during the past twenty years the maximum bandwidth available to the desktop has increased by approximately two orders of magnitude, along with similar increases in processor speed. These increases in computational power and network bandwidth have facilitated the creation of new applications that use audio and video, and which ideally require certain service guarantees when transported between two hosts on a network. Although distributed management and accountability of resources were goals of the original Internet architecture, they were low on the list of goals, and as a result, did not receive that much attention. IP does have a type of service field which is supposed to offer hints to the network nodes as to the precedence and delivery requirements of the packet, however this field is usually ignored [33]. Thus, IP has no mechanisms for resource reservation or its negotiation, guaranteeing delay bounds, or, until just recently, no mechanism for multicast (RFC 1112).

These inadequacies have lead to the development of a number of new protocols called reservation protocols. These protocols provide a means for negotiating the quality of service parameters associated with network resources and are a means to establish and maintain state within network nodes along the path between two hosts, as opposed to the administration of the resources [10].

2.2.1 Resource Reservation Protocol (RSVP)

RSVP is a new protocol design that emerged in 1993 from researchers at Xerox PARC and USC. Implementations of RSVP are currently in progress. Functionally, RSVP acts as a companion protocol to IP or another network layer protocol, with RSVP controlling the way in which IP sends packets. RSVP is known as a simplex protocol, meaning that it reserves resources in only one direction. It is also considered to be receiver-oriented, i.e., the receiver is responsible for initiating the resource reservation for the data stream or flow. Another feature of RSVP is its scalable support for *multicast*, the distribution of information from a single transmitter to multiple receivers. RSVP allows a multicast group to be formed from heterogeneous receivers, each of which can utilize a different amount of network resources; in addition, each receiver can participate in one or more data streams sent to the same multicast address. Another result of these design goals is that a multicast sender need not have knowledge of all receivers, because an internal node in the network may have all the information about a stream to which a receiver wants to connect. RSVP also allows the receiver to dynamically switch between different streams. The routers within the network also maintain “soft state” which supports dynamic membership changes and automatic adaptation to changes in routes [45].

RSVP interfaces to three other modules: the next higher layer in the protocol stack, typically the application program or session layer; the network routing protocol; and the network admission control, which decides whether a new flow may be accepted. The requirements of a data stream or flow are represented by the *flowspec*, which is passed from the application to RSVP, and is then used for determining admission control. RSVP leaves the *flowspec* undefined [45].

As described in chapter one, latency, network access, quality of service, and end-system design are four topics that must be considered in the design of a multimedia

communication architecture. RSVP fulfills some of the needs discussed under QoS. In order to provide resource reservation in a network architecture, Delgrossi et al. [10] state that two components are needed:

- an end-to-end resource reservation protocol, and
- a resource administration mechanism.

RSVP fulfills the first component mentioned above, and can support a best-effort QoS, meaning that all resources that might ever be needed are reserved before the sending of data. RSVP cannot give a more strict QoS guarantee because there is no direct relation between it and the routing and data transmission protocol. Thus, it is possible for a route to change during transmission of a stream, creating a new route which does not have all of the needed resources available [10].

2.2.2 Experimental Internet Stream Protocol Version 2 (ST-II)

ST-II is the successor to the ST protocol [15] which was designed in the late 70's. ST-II was developed for the most part at BBN, and was released as RFC 1190 in 1990. There are a number of implementations of ST-II currently working at various locations. ST-II is a connection-oriented network layer protocol that supplies mechanisms for data transfer as well as end-to-end system resource negotiation along simplex routes. ST-II uses an integrated approach to resource reservation and data transfer as opposed to the more modular approach used by RSVP. ST-II also uses a sender-oriented approach to resource reservation, where the sender initiates a request for a certain stream specification, which then passes through any intermediate nodes before reaching its destination. Both intermediate nodes and the destination have the opportunity to modify the stream specification before it is sent back to the initiating transmitter [36]. Multicast streams with reserved resources are also permitted, although the sender must be aware of each receiver which could create bottlenecks at the source for large receiver groups [10].

ST-II interfaces to both the transport layer and a higher layer interface which specifies the flow or stream specification. The current stream specification (*flowspec)* has three purposes. During the setup phase of a stream it specifies the minimum packet size and rate required by the transmitter. This information is used by the protocol in order to reserve resources in the network nodes. When the *flowspec* reaches the destination it also contains the

packet size and rate actually obtained from the network, and the average delay and delay variance expected for a packet transmitted along that path. The receiver may then reduce its resource demand if it still has that option, and if it accepts the connection, then it returns the updated *flowspec* to the transmitter which then must decide if it still wants the connection considering the updated flow specification. The RFC states that the *flowspec* is still under development, and will no doubt change as a result of more experience with QoS networks and multimedia applications [36].

Like RSVP, ST-II also fulfills some of the needs discussed in chapter one concerning QoS. ST-II provides an end-to-end resource reservation protocol through which a guaranteed level of QoS can be provided so long as an internal node along a stream route does not fail [10]. Also with the current *flowspec* definition, the end nodes are given an idea of the network delay they should expect to see.

2.3 QoS Communication Architectures

In this section we discuss two communication architectures which offer varying levels of performance or qualities of service. Unlike the protocols in the previous section, these communication architectures offer mechanisms for resource administration as well as for resource negotiation. The discussion of each architecture will focus on how its services apply to the transmission of multimedia traffic.

2.3.1 Tenet Real-Time Protocol Suite

The Tenet Real-Time Protocol Suite is an on-going research project of the Tenet Group at the University of California at Berkeley. The protocol suite consists of a set of communication protocols that can be used to transmit real-time data streams with a guaranteed quality over packet-switching networks. The suite includes a network layer protocol, the Real-Time Internetwork Protocol (RTIP), two transport layer protocols, the Real-Time Message Transport Protocol (RMTP) and the Continuous Media Transport Protocol (CMTP), and an administration protocol, the Real-Time Channel Administration Protocol (RCAP) as discussed in [2].

The Tenet suite is based on earlier work (mathematical algorithms) by Ferrari, the founder of the group. This work contains a set of mathematically provable algorithms for

offering network hosts a number of service guarantees based on *idealized* models of the network components. The implementation approach taken by the group was to allow the suite to coexist with the Internet protocols so that non real-time traffic may be sent via TCP or UDP. The protocol suite is currently running on a number of platforms under different operating systems, such as the HP9000/700 under HP/UX, SPARCstations under SunOS, SGIs under IRIX, and 80x86's under BSDI BSD/386 with the networking software derived from BSD UNIX [2]. The Tenet report [2] mentions that timing behavior in the workstations is complicated by factors such as process scheduling, interrupt handling, and CPU load. Thus, testing was extremely important in order to accurately calculate admission control parameters. In order to gain a better understanding of the suite it might be beneficial to walk through the Tenet protocol stack.

The Tenet protocol suite is designed to run over any datalink layer that can provide guaranteed performance services to the network layer. Currently it runs over FDDI, but will soon be ported to an ATM network.

The Real-Time Channel Administration Protocol (RCAP) interfaces with the datalink, network, and transport layers. It has functionality similar to the resource administration protocols mentioned in section one, such as the establishment and termination of the real-time channels, plus a facility for making status inquiries about the established connections. RCAP is similar to ST-II in that the sender initiates a request for channel establishment which then passes through internal networking nodes, which perform admission control tests. If these tests succeed then the request will reach the destination which also performs an admission test. If any test fails then that node sends a message to the initiator notifying it of the failure. Otherwise, the destination would send a message back through the same route on which the request was made so that any node may relax the amount of reserved resources it had set aside for the connection in the event that it had overallocated in the first place. Unlike ST-II and RSVP, RCAP has a defined set of QoS and traffic parameters giving the service requirements for the connection. The parameters are:

- upper bound on end-to-end message delay,
- lower bound on probability of timely delivery,
- an optional upper bound on delay jitter,

- lower bound on probability of no loss due to buffer overflow,
- minimum inter-message time,
- minimum average inter-message time,
- averaging interval, and
- maximum message size.

The use of these parameters provides an abstract interface to allow different admission control tests to be performed on heterogeneous networks and end-systems.

RTIP is a connection-oriented unreliable network protocol. The RTIP protocol performs rate control, jitter control, and scheduling based on the quality of service associated with the connection. The RTIP protocol ensures that packets arriving at a node do not violate any delay constraints on the amount of time the packet may spend in the node, or any bounds on rate that may be set for the connection. RTIP also uses information received from RCAP in order to allocate the needed buffer space for the connection, and to police the inter-packet intervals.

RMTP is a light-weight transport protocol that gives applications a message-based abstraction. Unlike TCP which provides a reliable service with mechanisms for acknowledgments, retransmissions, or flow control, RTMP assumes that if these mechanisms are needed then a higher layer protocol can provide these services. Thus, RTMP relies on the services provided by RTIP such as rate-based flow control, and in-order delivery due to the connection-oriented nature of the underlying protocol [2].

CMTP is another light-weight transport protocol that gives applications a different paradigm for the sending and receiving of data. CMTP offers a periodic, time-driven service for the sending and receiving of information. The designers envisioned this protocol being used for multimedia traffic.

The Tenet protocol suite is a full protocol suite designed and implemented to support real-time traffic. The Tenet suite meets all of the design considerations discussed in chapter one, at least in theory. However, as mentioned in [2] the worst case analysis used to make guarantees is based on idealized models of the network, whereas the actual workstations and operating systems have timing behavior that is affected by the operating system, interrupt handling, and CPU load. So, the area of real-time communications has been explored by this group as well as others, however, there still remains a number of questions concerning the interaction of “real-time communication architectures” in real

systems, as well as the feasibility of certain real-time admission control and policing algorithms in real systems.

2.3.2 Capacity Based Session Reservation Protocol (CBSRP) Using ARTS and FDDI

The Capacity Based Session Reservation Protocol was developed at Carnegie Mellon University in order to minimize the variance of delay for continuous media such as video in a local area network. CBSRP provides resource negotiation, administration, and admission control for hosts on a LAN. By reserving buffers, CPU cycles, and network bandwidth, the protocol is able to offer bounded delays for end-to-end host communications [34].

In [34] the authors relate their experiences with using CBSRP in conjunction with ARTS [35], a distributed real-time system, and FDDI to deliver continuous media with bounded delay guarantees over a LAN. By using FDDI, they took advantage of the *synchronous* class of traffic which bounds the time between network accesses for a host to 2*TTRT (Timed Target Rotation Time), where the TTRT is set by the user. Under the synchronous class of traffic each user must reserve synchronous allocation space in order to transmit. The synchronous class delivery latency is then bounded as mentioned above, resulting in deterministic delay bounds at the datalink layer. In order to meet delay bounds at higher layers the operating system must also support deterministic guarantees. For this project the ARTS system used a deadline monotonic policy for scheduling, which meant that if it was possible to schedule the task then its deadline was guaranteed to be met. Admission control tests to determine if a desired connection could be guaranteed service were performed through the use of CBSRP.

CBSRP allows sessions to be requested by either the sender or the receiver. In order to create a session, the application must make a request with a set of parameters specified by the protocol that map to the needed resources for the connection. The parameters which are listed below are based on the idea that audio and video have temporal and spatial qualities which characterize the continuous media stream. The parameters are:

- minimum and maximum temporal resolution, which would correlate to frames/s in video,
- minimum and maximum spatial resolution, which would correlate to the number

of bits/pixel that a video application would receive,

• an array of possible discrete temporal values the class of traffic might take on,

• an array of possible discrete spatial values the class of traffic might take on,

• importance,

• maximum end-to-end delay, and

• maximum packet loss rate.

With these parameters the session manager could possibly have a number of equally admissable points in a two dimensional plane of temporal and spatial values for a given connection. This flexibility allows the manager to dynamically reallocate resources during the course of a connection so long as the reallocation does not cause the admission test of any stream to fail. If the session manager is unable to meet a connection request's criteria, then the request is rejected. Criteria for admission to the network are shown below,

Given the following definitions:

• $C_{i,j}$: execution time needed to transmit all data of node i, session j

• $D_{i,j}$: network deadline of node i, session j

• $L_{i,j}$: worst case gap between the first and last packet issued from node i, session j to the network

• SA_i: synchronous allocation allotted for node i

• σ : transmission overhead (note that this definition of σ differs from section 2.2.2)

First, the sum of all synchronous allocations in the network must be less than TTRT minus any transmission overhead. Expressed mathematically,

$$\sum_{i=1}^{n} SA_i \leq TTRT - \sigma$$

In addition, the minimum network deadline must be at least twice the value of TTRT. If the deadline was less than twice the TTRT there is no way to guarantee that the data can be sent because each network node is only guaranteed to see the token once in every 2*TTRT period. So a second condition is

$$D_{i,j} \geq 2 \cdot TTRT$$

If the message must be fragmented, then the deadline must be greater than twice the TTRT plus the time it takes to send the whole message. Otherwise, if the last packet has not been sent by this time, then there is no way to guarantee that the full message will make it to the network before its deadline, because the transmitter might not get control of the ring for

another 2*TTRT time units. This would mean that the last packet would arrive late.

So

$$D_{i,j} \geq L_{i,j} + 2 \cdot TTRT$$

The final requirement is that the sending node must be able to transmit all packets from all sessions originating at that node at least 2*TTRT before the earliest deadline of a session. Again, if all packets are not transmitted before 2*TTRT there is the possibility that some packets could arrive late to the network because the node might not receive access to the ring for another 2*TTRT time units. This means that synchronous allocations must be allocated based on $D_{i,min}$, the earliest occurring deadline. Synchronous allocation is based on $D_{i,min}$ because the host system uses a FIFO queue, which can only make a single delay guarantee. This results in the following equation which assumes a worst case possibility that all sessions send packets within TTRT.

$$SA_i \geq \frac{\sum_{j=1}^{n} C_{i,j}}{\lceil (D_{i,min} - TTRT) / (TTRT) \rceil}$$

If the previous criteria can be met then the session is granted. If the unused network resources are not sufficient to grant the session, then the session manager will relax the requirements of any sessions that are using more than their minimum requirements, and have an importance level less than the requesting session. If this relaxation fails then the session manager will reject the session. Before final acceptance of the session, the request must also pass the schedulability requirements of the operating system which are also explained in [2].

For transmission of multimedia streams, the system uses the UDP protocol. UDP was chosen because it was believed that a lightweight protocol with high throughput was needed, rather than a reliable protocol that provided retransmission.

The results of the CMU work were that their dynamic QoS control scheme enabled the sending of continuous media over an FDDI LAN in a flexible and predictable manner. However, the feasibility of scaling their approach to multiple FDDI networks across routers is not clear, and would seem to be costly due to the dynamic reconfiguration of synchronous allocation.

2.4 Summary

With the advent of network multimedia communication, the need for different types of service in an internetworking environment has become more apparent, and as a result has been the focus of many researchers.

The general consensus is that in order to avoid bottlenecks in a communication system it is necessary to reserve resources for each connection, and if deterministic guarantees are desired then some sort of admission control to the system is necessary. In the first section of this chapter we discussed admission control tests and the traffic policing measures that enforce adherence to the parameters that were agreed on for entrance to the network. The general trend for these algorithms is that the more efficient they are in network utilization, then the more complex or costly the algorithm is to implement.

In the second section we discussed the shortcomings of a currently used internetworking protocol (IP), and then discussed two new network reservation protocols that have been proposed.

The two systems discussed in the third section show two differing approaches towards researching the field of real-time communications. The Tenet group based their protocol suite on mathematically derived algorithms which guarantee deterministic bounds. The suite is designed to offer real-time service over heterogeneous internetworks; however, the efficacy of these systems with differing support for the real-time communication architecture is still being researched. On the other hand, CMU's research focused on giving service guarantees for multimedia traffic on a single FDDI subnet. Their architecture uses a real-time operating system which supports the underlying guarantees attainable in the communications stack; however, the solution is limited in scope because of its single subnet nature. Our research focuses on the middle road between these two systems. We sought to design and implement an admission control protocol and router that had the capability to give deterministic guarantees to connections on different subnets. However, our design also focuses on the need to manage certain limitations and bottlenecks of our system.

Chapter 3

Router Design

In this section we describe the design of the router used in the multimedia communication architecture. Typically, commercial routers such as the Cisco AGS use hardware and firmware to increase the throughput and decrease the latency through the router. Many of these routers have a "fast path" for regular IP packets which decreases the latency for these packets, while IP packets with options set are passed to a slower path unless the router is specifically configured otherwise [16]. The result is that most commercial routers when configured properly, can offer the maximum throughput allowed by the medium when sending the minimum packet size from certain hosts (who are configured to use the "fastest path"). However, these routers still suffer from congestion when under heavy load, and because of the limitations of FIFO queuing, they can only offer a single delay guarantee assuming no packets are lost. In the ideal situation, we would have chosen to alter the source code of a commercial router to support multimedia traffic. However, no commercial vendor of routers was willing to allow us access to source code, so we have based our design on a personal computer architecture. We do not expect this design to meet the same level of overall performance as a firmware router in such aspects as throughput and latency, but with such added features as static priority queues, admission control, and traffic policing, the router and the multimedia architecture offer service guarantees which are not available in typical networks.

The following section first describes the router architecture and follows the datapath of a packet through the router. The second section discusses the Xpress Transport Protocol which is a next-generation transport protocol that offers new functionality to its users. The main feature of XTP that the router uses is a priority field in the transport layer's header which allows the router to demultiplex the traffic into different priority queues. The third part of the chapter presents performance measurements of the router, and then analyses the bottlenecks of the current design. The admission control protocol and mechanisms will be introduced in this chapter, but the main discussion will be postponed until the following chapter.

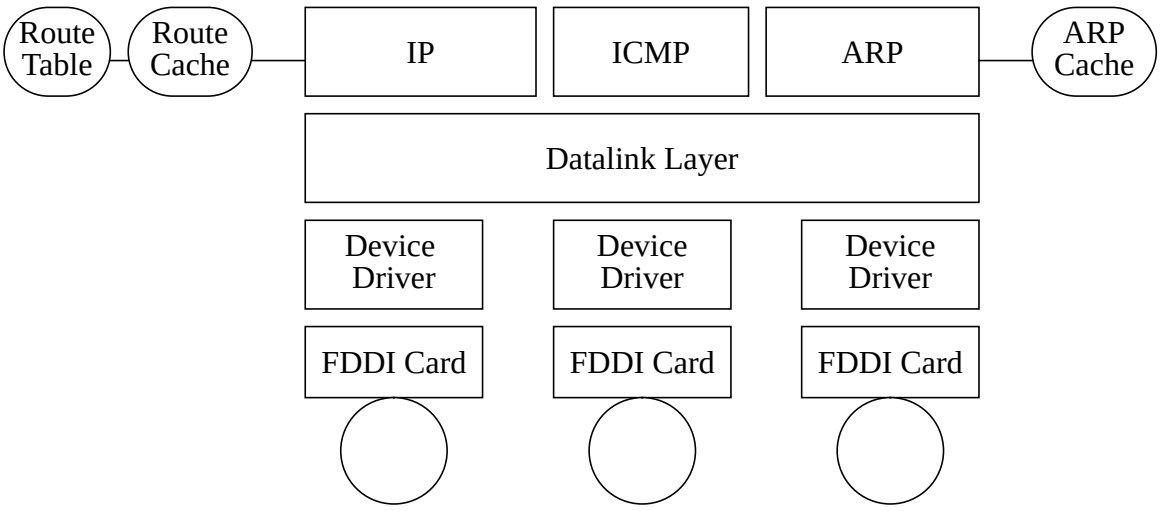

Figure 6. Router Architecture

Figure 6 shows the general router architecture. Route tables are statically set, and then IP destination address routes are cached in the route cache for quick look-ups.

3.1 Router Architecture

The overall design considerations mentioned in chapter one were applied in making decisions on the structure of the router. The result has a firm grip "when the rubber meets the road." When designing the router such issues as the number of data copies, the critical code length, the servicing of interrupts, and load on the CPU were considered.

The router uses a 33 MHz Intel 486-based PC, with the Extended Industry Standard Architecture (EISA) bus. The choice of a relatively slow PC allowed for the overall design to be stressed more easily by end systems on multiple subnets. For instance, it takes less traffic to congest the router when it is a 33 MHz PC as opposed to a 66 MHz PC. We use single attached Network Peripheral (NP) EISA bus FDDI cards. The choice of the NP card was based on its availability, and the availability of device driver source code for the card.

The choice of the EISA bus ensures that bus speed will not be the bottleneck on performance for a router supporting a small number of Network Interface Units (NIUs). The EISA bus provides a maximum transfer rate of 33 MBytes/s (or 264 Mbits/s) which is over four times the transfer rate of an ISA bus (8 MBytes/s). The EISA bus is also wider at 32 bits, and supports up to fifteen slots; however, the PC we use only has eight slots. The software router design will support any number of cards, and is currently limited by the physical hardware (the number of available slots in the machine and the available

interrupts that may be used by the network cards). There are a number of other backplanes available such as VME and PCI. However, we had no access to any VME hardware, and the PCI bus architecture was new, with no FDDI network adapters available for use at the start of this project. Thus, the EISA bus architecture was used.

Figure 6 shows the router architecture. IP [26], ARP [25], and ICMP [27] each run as separate prioritized threads within the router. All three protocols meet the specifications found in their respective RFCs. Extensions to the IP protocol were made to allow for multiple queues to handle prioritized traffic. In this way preferential treatment may be given to certain streams of traffic. At this time the router demultiplexes traffic via the priority field found in the header of an Xpress Transport Protocol (XTP) packet.

The software within the datalink layer is used to abstract the underlying FDDI hardware and device drivers from the protocols used within the router. Thus, various network adapter cards could be used in the router so long as device drivers were written that conformed with the calls presented at the datalink layer.

In designing the router such issues as the number of data copies, the critical code length, the servicing of interrupts, and load on the CPU had to be considered. In order to explain these issues in more depth it is instructive to follow the data flow through the router.

When a frame arrives at one of the network interface cards (NICs), the card interrupts the CPU and switches to the FDDI device driver interrupt subroutine. Because of the hardware architecture of the FDDI card's DMA chip, it is necessary to copy the frame from the FDDI board to the router's buffer space. Thus, because of the FDDI card architecture the router must do two DMA transfers when storing and forwarding packets. Switching to a new card at a later date might enhance performance if the card would support a *zero-copy* architecture, i.e., frames are DMAed directly from one FDDI card to another. However, the drawback of a *zero-copy* architecture is that the hardware queues limit the number of packets that can be stored on a board, and the hardware queues must somehow facilitate the preemption of service on lower priority packets until all higher priority packets have been served.

Once the packet has been copied into router buffer space it is enqueued in the proper queue, and the device driver then returns control to the highest priority active thread. The highest priority task is user-defined, but for testing purposes it was the IP routing engine. The IP task processes all packets of the highest priority first, and then moves to progressively

lower priority packets. However, once a packet is dequeued it will be processed, so the time between the arrival of a packet of the highest priority into an empty queue and its servicing is bounded by the time it takes to service a currently dequeued packet. Assuming that the router never reaches congestion in the highest priority queue, then the highest priority packets have a bounded worst case delay through the router. The derivation of the worst case delay will be discussed in the chapter on the admission control protocol.

In order to further reduce time in the critical path, both an ARP cache and a route cache are used. Route tables are statically set, and are used to make routing decisions when an IP frame with a previously unseen destination field arrives. Once the routing decision has been made, the destination is placed in a route cache for quick lookup. The route cache also holds the new destination's MAC address which was either found in the ARP cache or resolved via an ARP request. Thus, the next packet traveling between the same two end nodes will bypass the routing tables and ARP cache, because all the needed information will be found in the route cache.

The servicing of interrupts was also a design consideration for the router. A renegade user can seriously degrade the performance of the router. For testing purposes it was assumed that all users behaved in a bounded fashion. In practice, all users must adhere to a common admission control policy discussed in the next chapter.

In order to simplify the design issues the router is currently the only task running on the PC; however, the router could be ported into an operating system that supports such functionality as prioritized threads and fine-grained timers.

3.2 Xpress Transport Protocol 4.0

The Xpress Transport Protocol 4.0 [39] offers a variety of service paradigms to the user because of its design emphasis on separating policy or paradigm issues from mechanism or implementation issues. Thus, a user may choose to use a service similar to that of TCP (reliable and connection-oriented) or a service similar to UDP (unreliable and best-effort). The choice depends on the setting of bits within the options field of the XTP header.[1] XTP also offers the ability to create other service paradigms through the use of

1. The actual bit settings were done through the application programming interface of the XTP version used in our lab. We use a commercial version of XTP 4.0 from Network Xpress, Inc.

mechanisms such as:

- Rate and Burst Control,
- Multicast,
- Multicast Group Management,
- Priority,
- Selectable Error Control, and
- Selectable Flow Control.

The use of XTP at the end systems in our testbed allows the router to demultiplex traffic via the XTP SORT (priority) field which may be set by the user. This results in an end-to-end path that receives preferential treatment based on the value of the SORT field. Other protocols could have been used; however this would have increased the complexity at the end-systems because protocols such as TCP, UDP, and IP do not support multiple priority levels in their defined form. It would have been necessary to alter the source code of these protocols in order to take advantage of the availability of prioritized traffic. With our design the router supports regular IP traffic by sending it at the lowest priority level. Thus, TCP, UDP, and IP traffic may be sent in the same manner as before. XTP traffic may also be sent at this base level, or it may be prioritized.

Our admission control protocol and traffic policing mechanisms also take advantage of certain XTP features. We use XTP's rate and burst mechanisms to help police traffic entering the network. This policing mechanism will be described in the next chapter.

3.3 Performance Measurements

In this section we discuss the baseline performance measurements of the router. In order to make service guarantees one must ensure that the guarantee can be met. Therefore, the experimental testing of the basic router is extremely important because it must be known what resources are bottlenecks in our design so that they may be managed via the admission control protocol. The first part of this section discusses the latencies through the router. The second section presents data on the throughput that the router may sustain with varying levels of background traffic. In the third section we summarize the results of the performance measurements. With the resulting information we then designed an admission control protocol and traffic policing mechanism. These mechanisms allow end-systems sending traffic through this router to have certain service guarantees on worst case delay and throughput for their connections at the highest priority level.

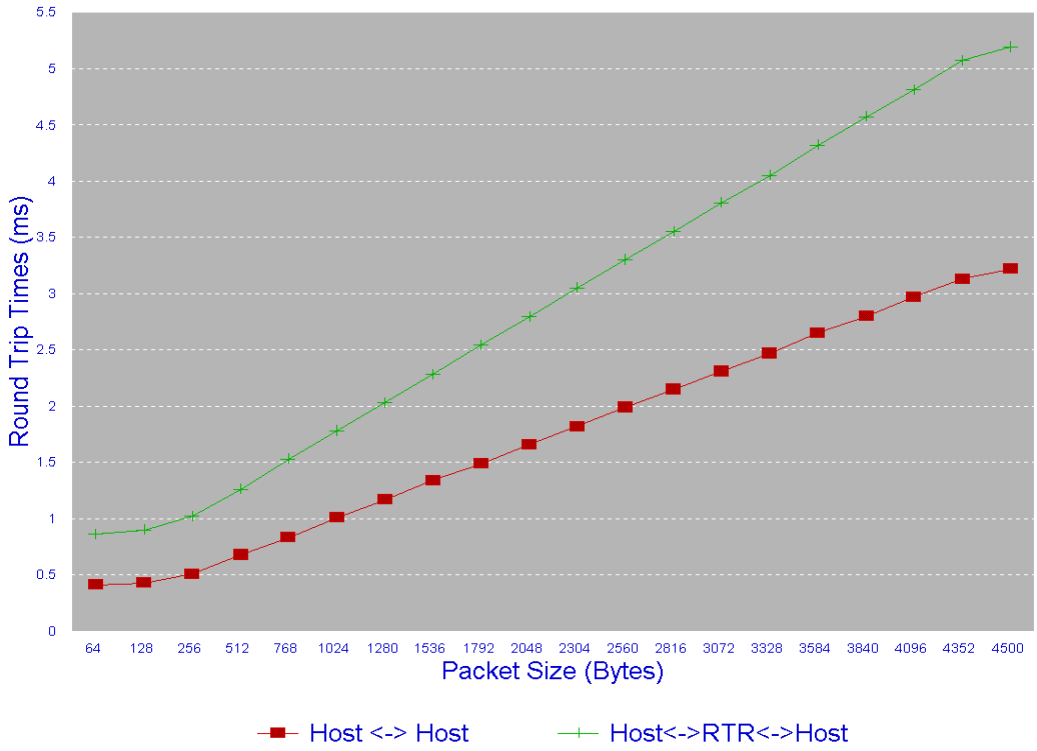

Figure 7. Packet Size Vs. Round Trip Times

Figure 7 shows the effect of packets size on round trip times between two hosts on a single subnet, and then between two hosts connected via the router. 10,000 packets were sent for each test with the result in the graph representing the time average.

3.3.1 Latency Measurements

The delay a packet incurs at the router as it passes from an end-system on one subnet to an end-system on another is an important quantity because it is part of the overall latency of a packet between end systems. Ideally, the best way to measure the one-way latency through the router would be to record the time it enters the router, and then record the exit time. However, due to the architecture of the PC, creating a timer function with the granularity needed to measure the one way-latency of a packet is too costly in terms of overhead for the router.

The 80x86 architecture uses a number of 8254 programmable interval timers (3 timers per chip) which may be set by the user to interrupt after a certain delay. With the use of one of these timers, a timing system was developed for the router, which allows multiple timer requests to be serviced from a single interval timer. Although there are a number of interval timers in the PC, only one may be set by the user. Thus, the timer must be able to

interrupt at the minimum granularity of the desired clock. Too small a granularity can affect system performance because the timer interrupt calls code which checks to see if any user-set timers have expired every time an interrupt occurs. Thus, the router latency was not determined by timers within the router.

Some systems have equipment that supports a synchronized clock which allows events on separate subnets to be timed. With this type of equipment the one-way latency may be measured by finding the difference between the time the packet was seen on the source subnet, and then on the destination subnet. Unfortunately, this equipment was not available because of its prohibitive cost.

Because of the problems associated with the two previous methods of measuring latency, we used a solution which would give us the latency indirectly. We measured the time it took to send a packet from an end-system on one subnet through the router to an end-system on another subnet, and then back to the original sender. This value is known as the round trip time (RTT). Assuming the sender and receiver are matched, then the following equation would yield an approximation of the one way latency:

$$\text{Router Latency}_{\text{one-way}} = \frac{RTT_{\text{through router}} - RTT_{\text{between hosts on a single subnet}}}{2}$$

Figure 7 shows the effects of varying packet size on the RTT. As packet size increases the round trip times increase for both the single subnet test and the test through the router. This increase is due mostly to the memory copies or Direct Memory Accesses (DMA) that occur when a packet is received or when a packet is being sent. A DMA transfers a large block of data from an external device (in this case an FDDI card) to the computer's main memory. This type of transfer frees the CPU to do other processes. However, there is usually an initial setup time needed before using a DMA, and then a variable cost is incurred depending on the amount of data being transferred. So in a round trip between two nodes there are currently four DMA's; each node does a DMA on the send, and each node does a DMA on the receive. In a round trip between two nodes through the router there are a total of eight DMA's. This difference in the number of DMA's is the main reason for the differing slopes of the two lines.

The result of the latency test shows that an increase in packet size increases the time of processing for a single packet within the router due to increased latency of each DMA transfer.

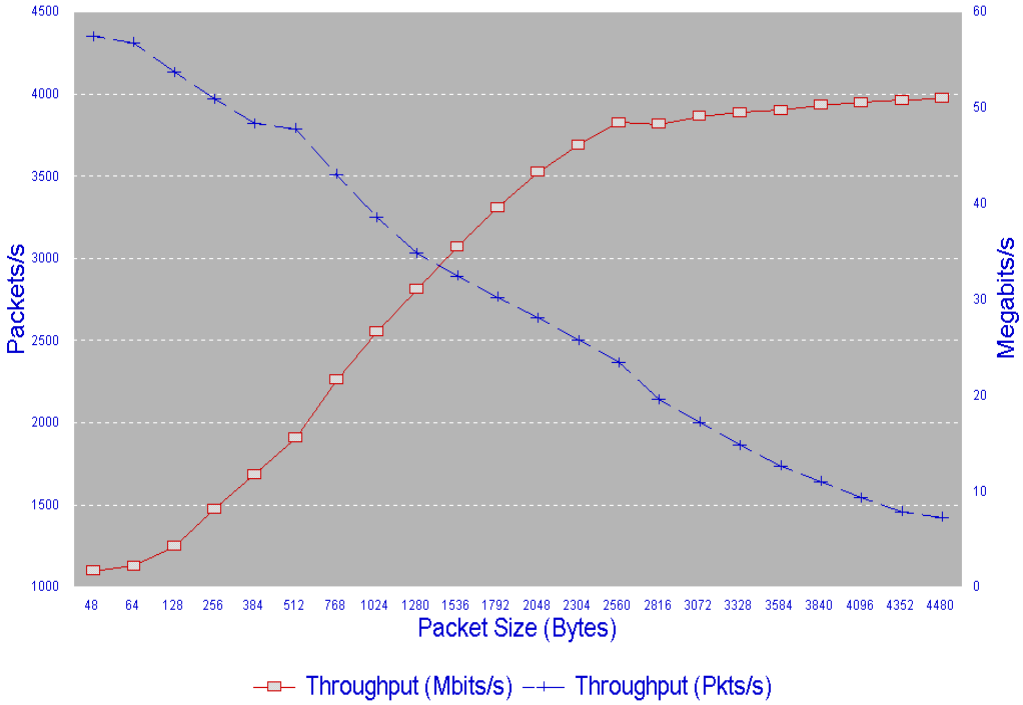

Figure 8. Throughput Vs. Packet Size For The IP Router

Figure 8 shows the effect of packet size on the throughput the router can support without dropping packets. The graph shows throughput in both Megabits/s and Packets/s.

3.3.2 Throughput Measurements

In this section throughput measurements for the router are discussed, with emphasis being placed on identifying some of the bottlenecks in the router. The first set of throughput tests used non-prioritized IP traffic through the router. In this case, the router was allowed to use all available internal buffers for this stream. The tests used a minimum interpacket gap that did not cause the router's packet queue to overflow. The second set of tests sent two streams of traffic through the router, one at high priority and one at low priority. The low priority stream consisted of constant length packets being sent at a rate that would congest the router. The packet size of the high priority traffic was then varied, with the maximum throughput being found for a given packet size that did not cause any of the high priority packets to be dropped within the router. Discussion of each set of tests follows.

3.3.2.1 Single Traffic Type

Figure 8 shows the results of the tests using a single stream of non-prioritized IP traffic through the router. Peak throughput, when incurring no packet loss due to queue overflow in the router, is approximately 50 Mbits/s. Testing showed that the first bottleneck the router would encounter was due to the servicing of interrupts. If the interpacket gap was too small, then the router would only have time to enqueue the packet before being interrupted with notification that another packet had arrived. This bottleneck limited the number of packets per second that the router could serve at the minimum packet size. As packet size increased, the latency incurred due to the two memory copies (DMA transfers) needed to store and forward the packets from the source subnet to the destination subnet became more of a factor. The minimum interpacket gap had to be increased between packets of larger size in order for the router to have time to service each packet. From the results in Figure 8 it can be derived that the router needs approximately 700 μs to receive a 4,500 byte packet, find the destination subnet, and then send the packet out on the destination subnet. This measurement also assumes that the Route and ARP caches hold the information needed to process the packet.

3.3.2.2 Two Traffic Types, High and Low Priority

Figure 9 shows the results of the throughput tests that sent two streams of traffic through the router. One stream was sent as a low priority stream. The low priority stream was sent at a rate such that it loaded the router to the point where some packets of the low priority stream were being dropped even without the second high priority stream being sent through the router. The packet length of the low priority traffic was held constant as successive iterations of the throughput testing varied the packet size of the high priority traffic. Then the low priority packet sizes were changed to a new constant value and the testing was repeated. Figure 9 shows results from tests where the low priority traffic was held at constant packet sizes of 128, 1,024, 2,048, and 4,500 bytes while the size of the high priority traffic was varied. The legend associates a high priority stream with the constant packet size low priority stream that was contending for router resources.

The results of Figure 9 show that as background traffic (low priority) packet size increases it has less of an effect on the performance of the high priority traffic, i.e. the

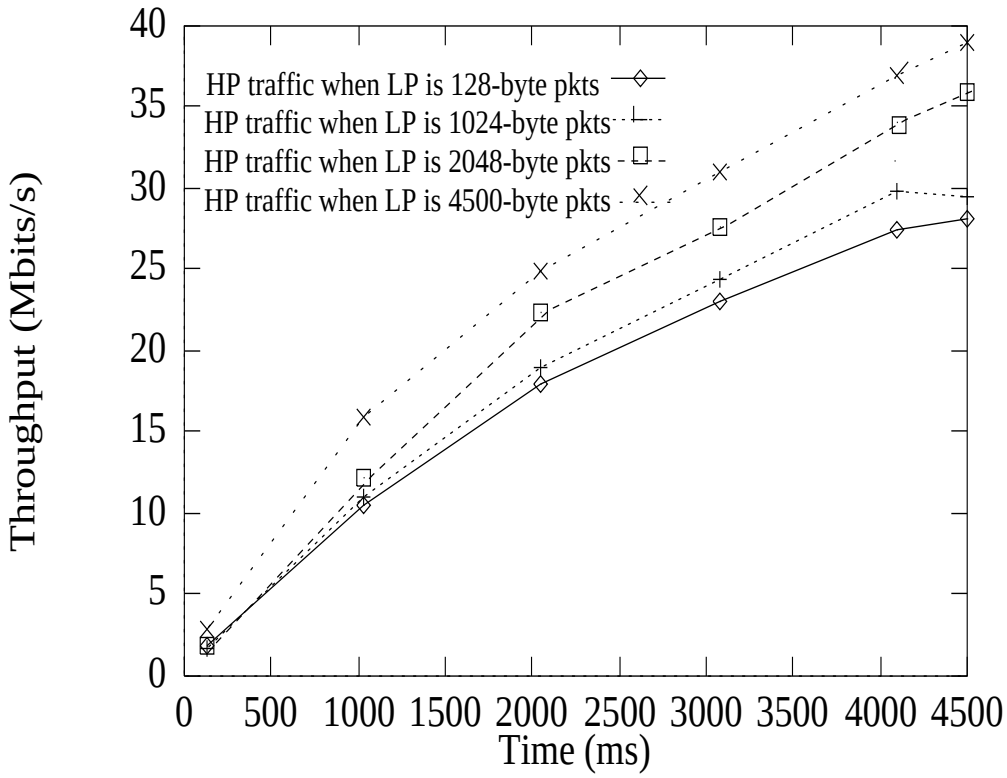

Figure 9. High Priority Traffic Throughput When the Router is Congested with Low Priority Traffic

Figure 9 shows plots of sustained throughput (no losses) through the router for high priority traffic when competing with an offered load of low priority traffic (not shown). The high priority traffic varies its packet size, while the low priority traffic is kept at a fixed packet size for each round of testing. This results in a constant level of background (low priority) traffic being presented to the router, along with the maximum amount of high priority traffic that the router can sustain.

router can sustain a higher throughput for the high priority traffic stream without dropping any high priority packets, because it is receiving less interrupts during a given time interval from the low priority packets. For instance, the router can only sustain 28 Mbits/s of high priority traffic when sending 4,500 byte packets, and competing for CPU time with receptions of 128 byte low priority packets (offering 5800 pkts/s or 5.5 Mbits/s). However, the router can sustain approximately 40 Mbits/sec of high priority traffic sending 4,500 byte packets when competing with low priority packets of equal size (offering 1411 pkts/s or 50.8 Mbits/s).

The difference in sustainable throughput occurs because the router must service more interrupts for the smaller packets because more can arrive within a given time period. Even though the router has separate buffers reserved for use with the two classes of traffic,

the CPU is taken away from processing high priority packets in order to process the interrupt of an incoming packet which could be a low priority packet.

The results of this set of tests show that the performance of traffic designated as high priority may be adversely affected by the amount and size of packets sent at a lower priority level. (For simplicity, we only used two priority levels for our testing).

3.4 Summary

In this chapter we discussed the design of the router, and the Xpress Transport Protocol (the protocol used in conjunction with the router), and then presented performance measurements of throughput and latency tests. With the given router design, the performance measurements indicate that there is an upper bound on performance that the router can support for high priority traffic that is less than the physical media speed of FDDI (100 Mbit/s). The measurements also indicate that high priority traffic may be adversely affected by lower priority traffic due to the increased number of interrupts that must be served in order to receive and process the packet to the point where it is placed in its appropriate priority queue. In general, the effect of increased background traffic is an increased latency in the time it takes to service high priority traffic, which then decreases the maximum throughput that may be sustained by the router for the highest priority traffic. This means that in order to guarantee service for the highest priority level, one needs to limit the amount of traffic that is admitted to the network. This topic will be discussed in the next chapter.

Chapter 4

Admission Control

In this chapter we discuss the design of the admission control protocol and the associated resource reservation used in our multimedia communication architecture. From the results of the previous chapter it is apparent that our router, like many commercial routers, can become a bottleneck depending on the traffic load. We can generalize the results to the following:

- An increase in packet size increases processing time associated with that packet within the router due to the increase in latency for each DMA.
- As the number of interrupts due to packet arrivals at the router increases, the performance of the router decreases because it spends more processing time within its interrupt subroutine.
- As router performance decreases the latency associated with the processing of a packet will increase. This action can result in packets being dropped due to consumption of the finite amount of buffer space within the router.

Thus, in order to ensure that the router does not become congested we must limit the amount of traffic that can pass through the router. We discuss our network admission control design in the first section of this chapter. The second section discusses how we propose to ensure that once a host has been admitted to the network, it will be able to gain access to the medium of the network. Then, in the final section we present results from the testing of our communication architecture by the simultaneous transmission of video and with background traffic.

4.1 Admission Control Design

In this section we discuss the mechanisms for resource administration used in our communication architecture, followed by a discussion of the end-to-end resource reservation protocol.

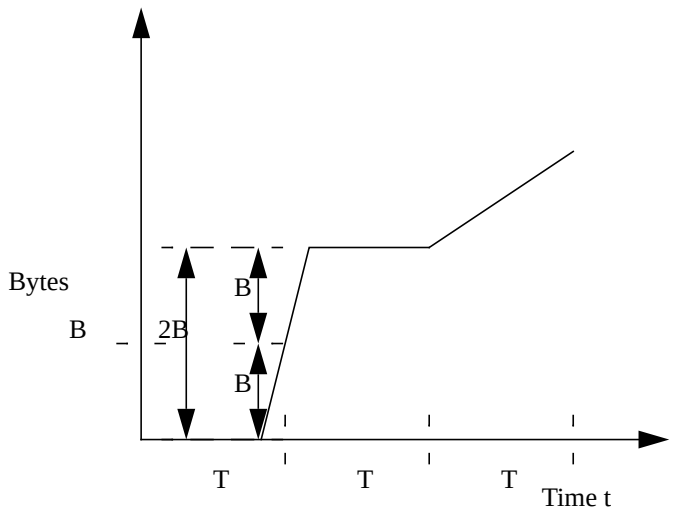

Figure 10. Worst Case Burst for the Jumping Window Policing Mechanism

Figure 10 shows the worst case burst that can occur when using the Jumping Window policing mechanism. The worst case can occur when a burst of size B occurs at the end of one period T, and is immediately followed by another burst, of size B, in the next period.

4.1.1 Resource Administration Mechanisms

As mentioned in the introduction of this chapter, the only way to ensure a certain level of performance in the router is to bound the number of interrupts that the router must process within a given interval. In order to bound the interrupts we decided to police the traffic at the end-systems, thus limiting the amount of traffic that the router would handle. Once the number of interrupts has been limited, the next factor affecting router performance is the size of the packet which it must process.

In the previous chapter we described the data path through the router, with the current implementation needing two DMA operations per packet, one to move the packet from the FDDI card to router-controlled memory and one to move the packet from the router memory to the FDDI card on which it is retransmitted. We showed via measurements of round trip times through the router that as packet size increases the processing time within the router also increases. Thus, we must either bound the packet size used by a connection, or assume the worst case in which all packets sent are the maximum size (4,500 bytes for FDDI). We currently assume the worst case of all packets being 4,500 bytes for all calculations.

4.1.1.1 End-System Resource Administration

Our system uses the Xpress Transport Protocol version 4.0 in the end-systems. As part of XTP's functionality, it offers *rate and burst* control for its data stream. However, the XTP mechanism is not "strict enough" in its enforcement of traffic policing for our needs. XTP uses the Jumping Window mechanism for its rate control. One of the limitations of this mechanism is that the worst case burst one might see is twice the burst allocated for a single interval as shown in Figure 10. Another limitation of the XTP rate control mechanism is that it only polices *data* traffic, assuming that all control traffic is negligible.

Problems can also arise due to certain limitations of the router. The router's performance is affected first by the number of packet arrivals per interval, and then by the size of the packets that have arrived. The XTP rate and burst mechanism is not specified in units of packets, rather in bytes/s and bytes/burst. A burst of 8,000 data bytes on an FDDI network can consist of two large 4,000-byte XTP packets, or it can be eight 1,000-byte XTP packets, or in the worst case, it may be 8,000 one-byte packets. The performance of the router would vary dramatically depending on how the burst was sent.

Thus, if we only use the XTP rate and burst mechanism for policing traffic entering our network, then we must allocate buffer space for packets within the router based on a burst of twice the amount specified by the XTP connection, plus some added constant (derived from observations) to take into account any control packets that might be emitted during that interval, plus some more to handle the case when packets are less than the maximum size (thus requiring more of them to carry a fixed amount of data).

For example, the router has buffer space to hold 32 high priority packets at any time, and from the results in chapter three, we know that the router can support a rate of approximately 50 Mbits/s when the packets are 4,500 bytes in length. If we have four connections through the router, each with an XTP-defined burst of six packets (27,000 bytes) and a maximum rate of 10 Mbits/s, then at any time the router should ideally have no more than 24 packets in its queue and should be able to handle the requested rates. However, if any of the connections send control packets or have two maximum bursts of packets back-to-back, then the router's queue can overflow. Thus, a more strict traffic policing mechanism is needed in the end-systems.

Based on the availability of rather fine-grained timers offered by the Intel 8254

chips that are on each PC, we have decided to use a Leaky Bucket [37] mechanism for policing connections that go through the router. The policing mechanism limits all traffic entering the network to a specified maximum burst, and then periodically updates a credit counter based on the specified rate. The rate and burst requests are specified in Mbits/s and bytes respectively. The implementation assumes that all packets entering the network are 4,500 bytes, and makes its calculation for the credit counter update rate based on this value, and the knowledge that the current version of the protocol stack has access to timers of 1 ms granularity, plus the value of the requested rate from the admission control protocol.

$$\text{Update Period (s)} = \left\lfloor \frac{\text{4,500 (bytes)}}{\text{Requested Rate (Mbits/s)} / 8 \text{ (bits/byte)}} \right\rfloor$$

The result of this calculation is that the update period of the credit counter may allow a larger throughput than the requested rate because the smallest timer granularity is 1 ms. The resource reservation component of the admission control protocol then uses this re-factored rate (gained by using the update period) in its request to the router for admission to the network. If the rate request results in an update period that is smaller than 1ms, then the request is currently denied by the end station at which the request originated. Burst requests are also assumed to be in 4,500 byte packets, which results in a request to the router being in units of packets/burst (this is a resource the router can manage). The router manages its buffer space on a packet by packet basis, and has enough room to hold up to 32 packets. We assume the worst case packet size of 4500 bytes when calculating our burst parameter because the router performance is sensitive to packet size. Thus, by assuming worst case size we know the router can handle a burst of smaller packets.

$$\text{Resultant Burst} = \left\lceil \frac{\text{Requested Burst (bytes)}}{\text{4,500 (bytes)}} \right\rceil$$

Assuming that the router accepts the request, the leaky bucket mechanism disables the sending of packets for a connection whenever its credit count drops to zero. Figures 11-14 show the result of traffic policing an XTP connection using the leaky bucket mechanism with a requested rate of 10 Mbits/s (which translates into an allocated rate of 12 Mbits/s)

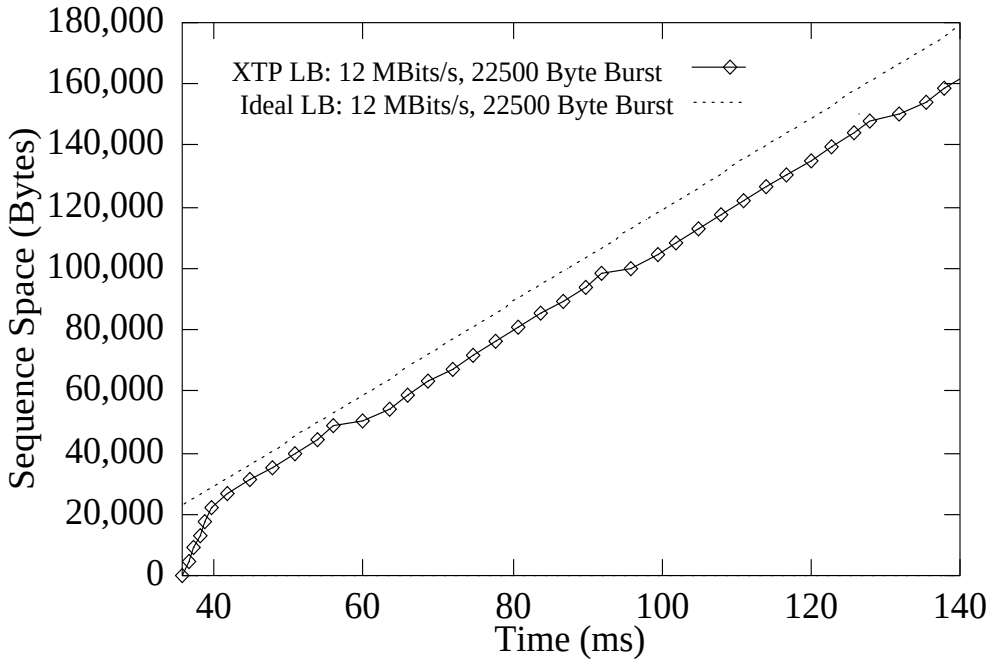

Figure 11. Leaky Bucket Traffic Policing in XTP

Figure 11 shows the performance of leaky bucket traffic policing in XTP. The counter is being updated once every 3 ms; and maximum burst is 5 packets. Due to timer granularity, a request for a rate of 10 Mbits/s had to be increased to 12 Mbits/s.

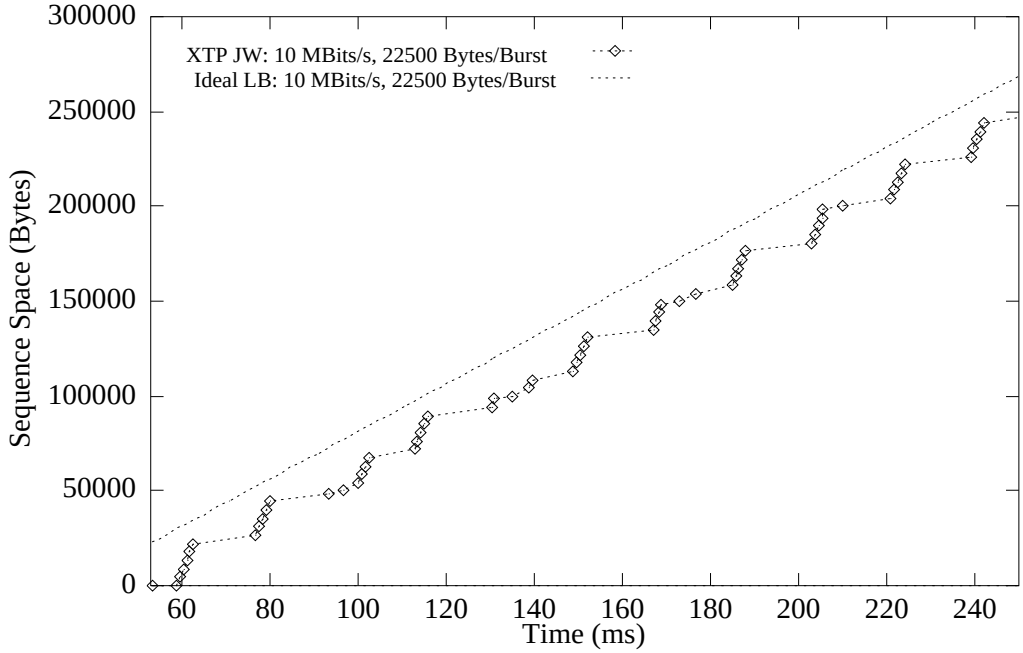

Figure 12. Jumping Window Traffic Policing in XTP

Figure 12 shows the performance of XTP's original rate and burst control, a jumping window mechanism. The jumping window mechanism uses a larger granularity timer, and as a result the protocol may set the maximum rate to be 10 Mbits/s with a burst of 22,500 bytes.

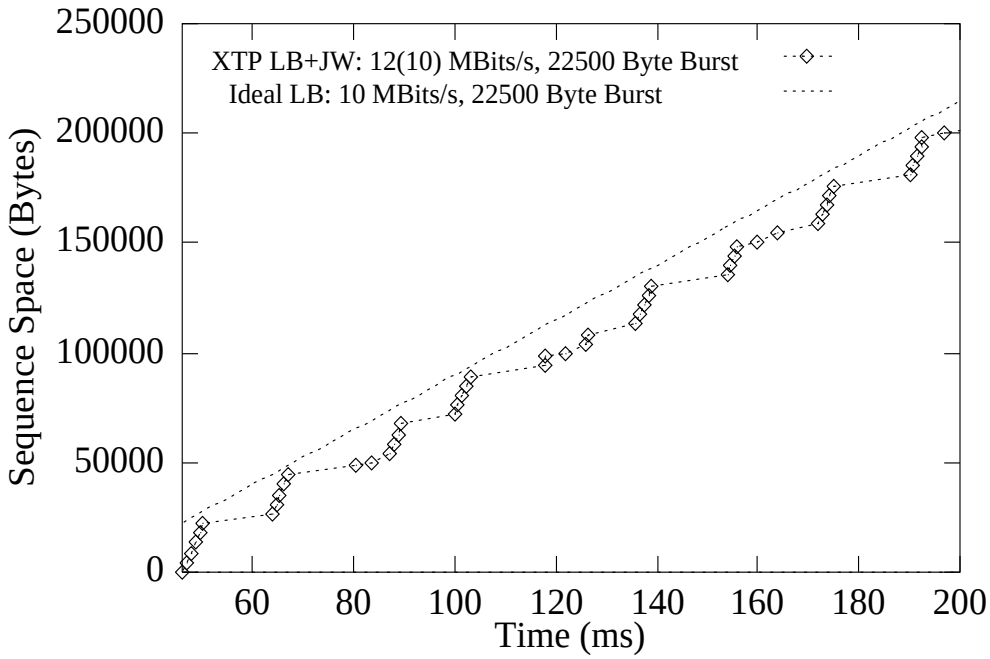

Figure 13. Leaky Bucket and Jumping Window Traffic Policing in XTP

Figure 13 shows the use of both the leaky bucket and the jumping window traffic policing mechanisms in XTP. The leaky bucket has a maximum rate of 12 Mbits/s, and the jumping window has a maximum rate of 10 Mbits/s. Both have a maximum burst of 22,500. However, the leaky bucket polices traffic in terms of outgoing packets, whereas the jumping window is looking strictly at data bytes.

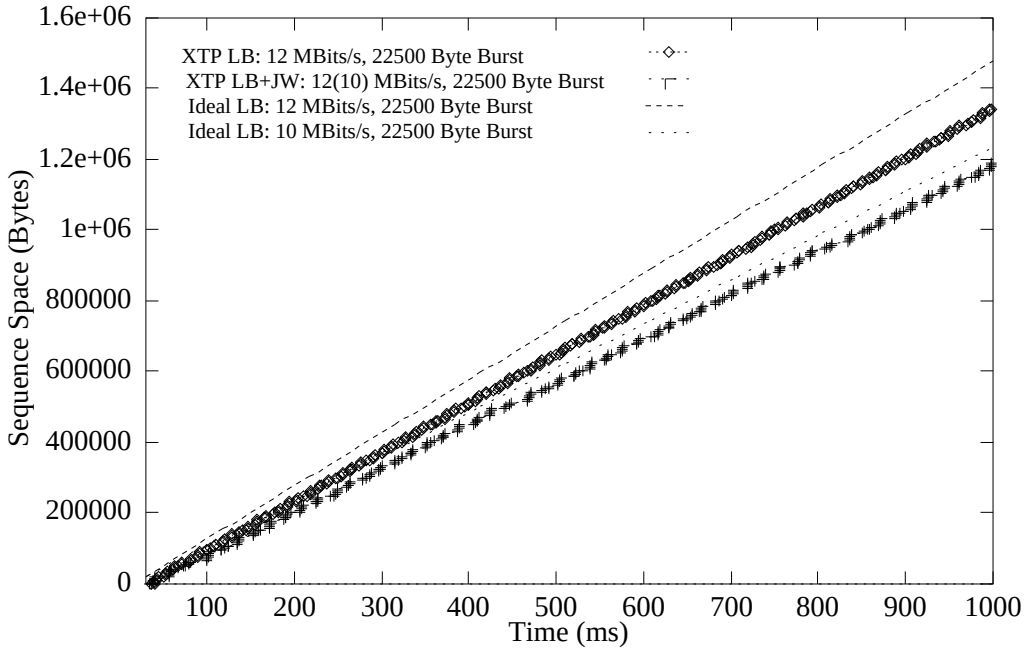

Figure 14. Various Traffic Policing Methods used by XTP

Figure 14 shows the use of a leaky bucket mechanism with and without a jumping window mechanism for the policing of traffic in XTP. In this case using both mechanisms kept the traffic constrained to less than 10 Mbits/s.

and a burst of 22,500 bytes (5 packets/burst), compared with using XTP's rate and burst functionality for policing. The overall result is that both the leaky bucket mechanism and jumping window mechanism do their job, and restrict the flow of packets entering the network such that it never exceeds the requested rate and burst.

One design decision that had to be made was when to notify the protocol that credit was available for sending more packets. The current implementation does not busy-wait until credit becomes available; instead, the XTP implementation looks for other work to be done. Therefore, the XTP connection may send a burst which exhausts its credit, but may not have anything left to send, such that reactivating the XTP transmit engine when the counter is updated only causes the system to waste time finding out that there is really no work to be done. The other possibility is that there are packets to be transmitted each time a credit is regained. If the XTP transmit engine is restarted each time a credit arrives, then this causes *thrashing* of the XTP protocol and any other tasks that are currently active, with the XTP transmit engine waking up and then sleeping each time a timer expires that updates the connection's credit counter. The other extreme is to explicitly restart the XTP transmit engine when the credit counter is equal to its maximum amount. Packets can still be sent in the interim if credit is available and the transmit engine is active for another connection or for some other reason. However, this heuristic can lead to maximum credit bursts followed by a silent period with a duration equal to the maximum burst times the credit update rate. Instead, we have chosen to explicitly restart the XTP transmit engine whenever the credit counter is updated. Our choice was based on initial tests with multimedia streams where large bursts and low rates caused excessive delays between packet bursts if the XTP transmit engine was not restarted each time its credit counter was updated.

Figure 11 shows the results of using our leaky bucket implementation with the XTP transmit engine restart policy mentioned above. In this test case the admission control request was for a rate of 10 Mbits/s and a burst of 22,500 bytes; however, due to the granularity of the timer the protocol had to offer 12 Mbit/s.

Figure 12 shows the results of using XTP's rate and burst options (a jumping window mechanism for traffic control). In this test the upper bound on the rate was set at 10 Mbits/s with a burst of 22,500 bytes. The rate of 10 Mbits/s can be enforced in this case because of the differing implementation between leaky bucket and jumping window. The jumping window

mechanisms updates its byte counter to the requested amount every time interval of Burst/ Requested Rate. For a 22,500 byte burst and a rate of 10 Mbits/s the update interval for the jumping window credit counter is 18 ms. The leaky bucket mechanism must have an update interval of 3 ms for a rate of 12 Mbit/s.

Figure 13 shows the results of sending data with XTP when policing the traffic with both the leaky bucket and jumping window mechanism. In this case the leaky bucket policed at a rate of 12 Mbits/s and the jumping window at a rate of 10 Mbits/s with both having a burst of 22,500 bytes. (Note: The jumping window mechanism polices traffic by counting the bytes of data whereas the leaky bucket polices traffic by assuming a packet size of 4,500 bytes.) The traffic is constrained in this case to be less than 10 Mbits/s; however, in the worst case the traffic can have an average rate of 12 Mbits/s over two update intervals of the jumping window mechanism. This worst case can occur if two back-to-back bursts of packets are sent (see Figure 10). The jumping window mechanism would allow this to occur, and then the actual number of packets sent would be constrained by the leaky bucket mechanism.

Figure 14 shows the results of an XTP transmission over a period of a second. The results show that the leaky bucket mechanism does throttle the traffic to its requested rate and burst (the policing of the burst is not visible from the graph, but is apparent in the raw data output which is not shown).

4.1.1.2 Router Resource Administration

With traffic flow entering the network being limited at the end systems, the router must ensure that, if it accepts a request for entrance, it has the resources available for the request. The current version of the router supports a number of priority levels depending on how many static priority queues are available within the router. However, for the resource administration we have limited the number of priority levels to two. As the data in chapter three indicates, the router performance for processing high priority traffic degrades as the number of packets/s or interrupts/s increases. We assume that the number of packets/s generated at the lower priority is bounded. With this knowledge, we know (from the test results in chapter three) the upper bound on throughput that the router can support without dropping any packets. Therefore, each time an end-system makes a request for a given rate and burst we determine whether adding the new rate and burst would violate our upper

performance bound for the high priority traffic. For instance, if we assume that we will have no more than 30 Mbits/s of traffic at the lower priority with a minimum packet size of 1,024 bytes, then we know from empirical results (see chapter three) that the router can handle 30 Mbit/s of traffic at high priority when packets are 4,500 bytes in length. Under these conditions we can sum the currently allocated bursts and rates through the router and add the new request. If the total sum for the burst is less than 32 (the number of packet buffers within the router that are allocated for high priority traffic), and the rate is less than 30 Mbits/s then the new high priority connection can be accepted. The admission test is:

$$\sigma_{request} + \sum_i \sigma_i \quad < \quad 32 \text{ packets} \cdot 4,500 \text{bytes/packet}$$

$$\rho_{request} t + \sum_i \rho_i t \quad < \quad (3,750,000 \text{bytes/s}) \cdot t$$

The result of the request (admission or denial) is then passed to the resource reservation protocol so that the end-systems may be notified.

4.1.2 Resource Reservation Protocol

Two resource reservation protocols were mentioned in chapter two, RSVP and ST-II. Unfortunately, the RSVP specification was in its draft stages at the start of this project, and ST-II takes the place of IP as the network protocol. While implementing one of these two protocols would have been instructive, that was not the focus of this project. Instead, we developed our own protocol to be used for admission control which satisfies the needs of our system and allows for testing of the resource administration being done within our system.

Our reservation protocol is an extension to the extant Internet Control Message Protocol (ICMP) [27]. ICMP is the basis for the ping program (PING_REQUEST and PING_REPLY), and also allows hosts and routers to communicate messages such as a destination being unreachable (DESTINATION_UNREACHABLE), or a request for a source to slow its traffic output (SOURCE_QUENCH), plus others. We have added the following messages:

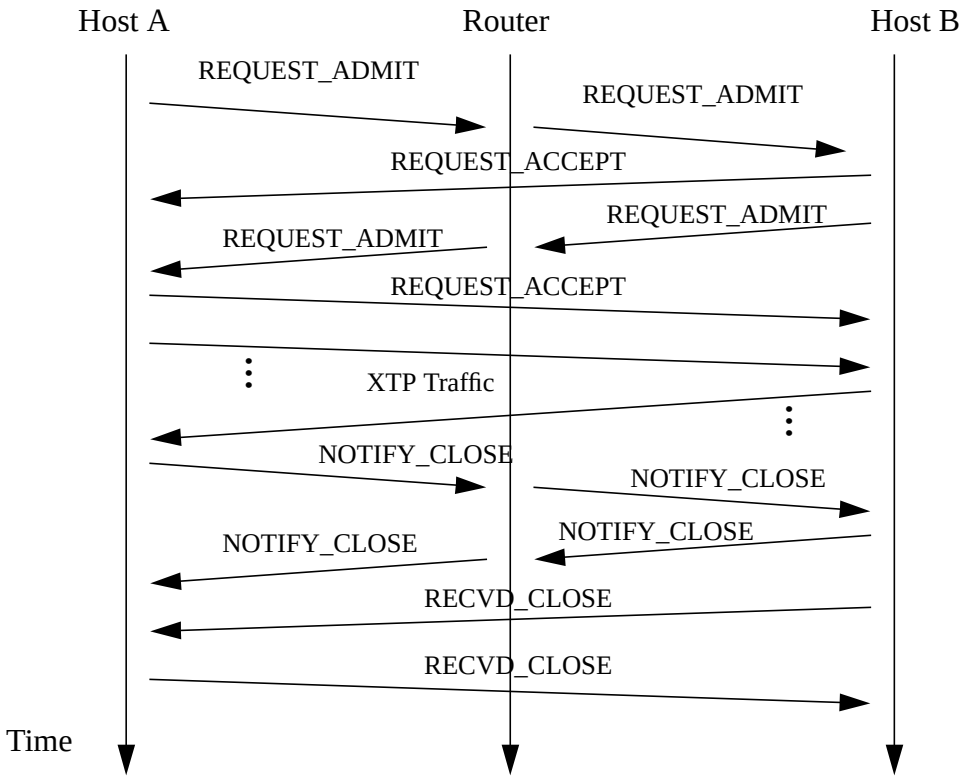

Figure 15. Typical Packet Exchange Using the Resource Reservation Protocol

Figure 15 shows the typical packet exchange that occurs using the defined resource reservation protocol when both the router and hosts accept the two requests. Lines that are broken at the router denote that the router receives these packets, does any required resource administration, and then passes the information along to the required host. Lines that do not break at the router indicate that the router processes these packets as regular IP packets being sent to the required destination.

• REQUEST_ADMIT — The request admit message is passed to the router along with the rate and burst requests. If the router accepts the connection, then the request is forwarded to the destination, otherwise it notifies the initiating host that the request has been denied. If the destination accepts the request, it sends a reply to the original host. If it denies the request then a denial is returned to the router, allowing any tentative resource allocations to be released, and then the packet is forwarded to the initiating host.

• REQUEST_STATUS — The request status message allows an end-system to query the router about its state.

• NOTIFY_CLOSE — The notify close message is issued when an end-station or router wishes to tear down a connection.

• REQUEST_ACCEPT — The request accept message is sent by an end-system if it accepts the associated request.

• REQUEST_DENIED — The request denied message is sent by an end-system if

it does not accept the associated request.

• REPORT_STATUS — The report status message is generated by the router in response to a request for status. It contains state information about the router and current connections.

• RECVD_CLOSE — The received close message is sent by an end-station to the initiating host upon reception of a “notify close” message.

The request for admittance along with its requested rate, burst, and host information will allocate resources for a unidirectional channel between two hosts connected via the router. If bidirectional communication is desired, then the destination host must also make an admission request before communication can begin. Figure 15 shows the logical flow of communication using the resource reservation protocol.

The resource reservation protocol was designed as a companion to the XTP protocol stack used at the University of Virginia Computer Networks Laboratory. The procedure prototypes used in the API for the reservation protocol are the following:

```
int32_t QOS_request_admit(uint32_t context, XTPadr_t *local,
                    XTPadr_t *remote,XTPadr_t *router, uint32_t *priority,
                     uint32_t *rate, uint32_t *burst);
int32_t QOS_close(uint32_t context);
int32_t QOS_request_status(uint32_t context, QOS_t *qos_mib);

typedef struct {
        uint32_t priority;          // priority of the connection
        uint32_t rate;              // rate of the connection
        uint32_t burst;             // burst of the connection
        uint32_t avg_idle_time;     // avg idle time in the router
} QOS_t;
```

The context number that is used in all three calls is a number that the XTP protocol at the local host associates with the given connection. All connections at a given host are guaranteed to have different context numbers. This number is passed to the router and the destination end-station in a “request for admittance” message. By using this number and the initiating host’s IP address the router and end-stations can distinguish among different requests or connections from the same initiating host. All API calls are blocking with return codes defined for success and for a number of error conditions.

4.2 Media Access

In chapter one we presented four alternatives for media access: Ethernet, Token Ring, FDDI, and ATM. We decided to use FDDI based on its availability, the amount of bandwidth it offered, and because its synchronous allocation mechanism offered certain deterministic guarantees.

In order to make any sort of deterministic delay guarantees it is necessary to have service guarantees at all levels of the protocol stack. When a host uses FDDI's synchronous allocation mechanism it is guaranteed to have access to the ring at least once every 2*TTRT. The value of the Target Token Rotation Timer (TTRT) is negotiated by all active hosts when the ring becomes active. The minimum requested value for the TTRT becomes the new value used by the ring. The length of time the host has access to the ring is determined by the size of its allotted synchronous allocation block.

In order to maintain the timing properties of the FDDI ring the amount of synchronous allocation is bounded by the TTRT minus the transmission overhead.

$$\sum_{i=1}^{n} SA_i \leq TTRT - \sigma$$

The process by which one makes a request for synchronous allocation is defined in the FDDI SMT. However, the basic idea is that there exists one host who regulates the allocation, ensuring that the timing properties of the ring are not violated.

In the initial admission protocol design the intent was to have the router act as the manager of the synchronous allocation for all rings to which it was attached. Thus, when a request for the admission to the network was made, the admission control resource manager would also have to consider whether there were enough synchronous allocation units available to give the requesting host on its ring, and enough units to allocate to the router on the destination host's ring.

Unfortunately, the Network Peripheral cards that are available for use do not support the synchronous bandwidth class (it is optional in the FDDI specification). Therefore, our current implementation cannot guarantee access to the ring. In our current testbed the effect of not having guaranteed access is minimal due to the small number of hosts on the rings and the small size of our testbed.

4.3 Performance Testing

In this section we describe the tests used to gauge our multimedia communication architecture's performance. The first part of this section describes the video distribution system used in the testing. The second part discusses the testing and the results of sending high priority video, high priority file transfers, and varying levels of low priority background traffic through our network when using our admission control protocol and modified version of XTP.

4.3.1 Video Distribution System

In this section we discuss the video distribution system used for testing the effectiveness of the admission control protocol and the router. A description of the hardware and software follows, then we discuss the video application end-system protocol. Finally, we describe option settings used in XTP at the end-stations.

4.3.1.1 Video Distribution Testbed

The video distribution testbed consists of six Intel 486-based EISA bus PCs. The machines vary in speed, with two running at 50 MHz, two at 33 MHz, and two at 66 MHz. *Terrapin*, the router, currently maintains two rings. One subnet contains three computers, while the other subnet contains two. Figure 16 shows the testbed setup.

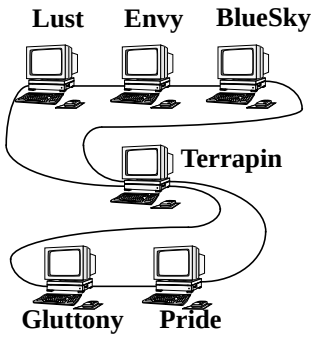

Figure 16. Video Distribution Testbed

Figure 16 shows the video distribution testbed that is used for testing the router and admission control performance. *Terrapin*, the router, connects two FDDI subnets.

Each machine contains either a single-attached or dual-attached Network Peripherals FDDI card. The router contains two NP FDDI cards (version 3), and all other machines use

NP version 2 cards. Each host machine also contains two extra hardware boards used for the display and compression/decompression of the video. The video board is a Truevision Bravado board, which uses eight bits-per-pixel color. The board also has an extra bus connector which allows add-on boards to be connected. Our video system uses a RapidTech JPEG [38] compression/decompression board in conjunction with the Bravado video board.

Video is captured by the Bravado board, with maximum size images being full screen video with 640x480 resolution. The captured frame is passed to the JPEG compression board where encoding of the frame occurs. The RapidTech board uses a Discrete Cosine Transform (DCT) to change the representation of an 8x8 pixel block into a frequency distribution. The board then filters this frequency distribution with regard to a user-defined quantization factor (Q factor). Frequencies that will not be missed are filtered out, and the remaining frequencies are represented in an efficient manner based on the Q factor (a larger Q factor indicates larger quantization intervals, resulting in more compression, and a corresponding degradation in picture quality). These frequency values are then encoded using the Huffman algorithm. The resulting frame is then copied to an application-owned buffer. From this buffer space the frame is passed to XTP for transmission to the destination host. Upon reception by the destination host the frame is passed to the RapidTech board for decompression, and then to the Bravado board for display.

4.3.1.2 Video Application End-System Protocol

The video distribution application was first developed as a multicast video distribution system that used XTP's reliable multicast and group management functionality to deliver a reliable video stream to multiple receivers [11]. Making minor modifications to the application allowed the video distribution to be run as a unicast connection between two end-systems, along a path that included our XTP-aware IP router [6].

Once initialized, the video application waits for interrupts from the RapidTech compression board signalling that a new frame is ready for transmission. The interrupt rate is a function of frame rate. Thus, full frame rate (30 fps) yields interrupts every 33 ms. When an interrupt occurs the application enqueues the frame in a queue of user-defined length. Both the sending and receiving applications maintain frame buffer queues. These queues protect the video stream from jitter due to latency incurred in the application and network. Video

playback begins by accumulating four video frames in the receiver to act as a “cushion” against network jitter. If a frame is delayed or retransmitted enroute, this receiver-based buffer allows approximately 133 ms of recovery time before the playback stream is visibly interrupted. If the receiver’s frame buffer is ever exhausted, then it replays the last good frame until a new one arrives.

The transmitter also maintains a buffer queue. Ideally, the transmitter should always be processing the most recently compressed frame. However, the current version of the application uses a blocking send call to XTP, so it is possible that a new frame may be ready for transmission, but the transport protocol may still be in the process of sending a previous frame buffer. In this case the new frame buffer will be queued for processing at a later time. In the worst case, the transmitter’s queue will fill completely, which currently results in the overwriting of the most recently enqueued frame.

In order to quantify the performance of the video stream we measured the transmission time latencies for frames leaving the sender. We define the transmission time latency to be the time it takes the user level XTP send call to execute and return control to the application program. This time represents how long it took XTP to reliably deliver a single video frame to the receiver. By measuring the performance between two end-systems connected via the router we can characterize the router performance for a soft real-time system.

4.3.1.3 End-to-End Transport via XTP

XTP offers a variety of service paradigms over which a video application may be based [39]. Using XTP gives the application writer such choices as whether to use *go-back-n* or selective retransmission; whether to create a reliable connection between two hosts; and whether to prioritize the data stream. Plus, most of the functionality found in XTP is completely orthogonal.

Many video applications use an unreliable transmission control protocol such as UDP or RTP on top of UDP. If part of a video frame does not arrive, either the frame is dropped or the decompression hardware can handle missing data which will result in degradation of the picture quality. While XTP offers the user a choice of a reliable or an unreliable service, transmission of the video frames uses the reliable delivery service

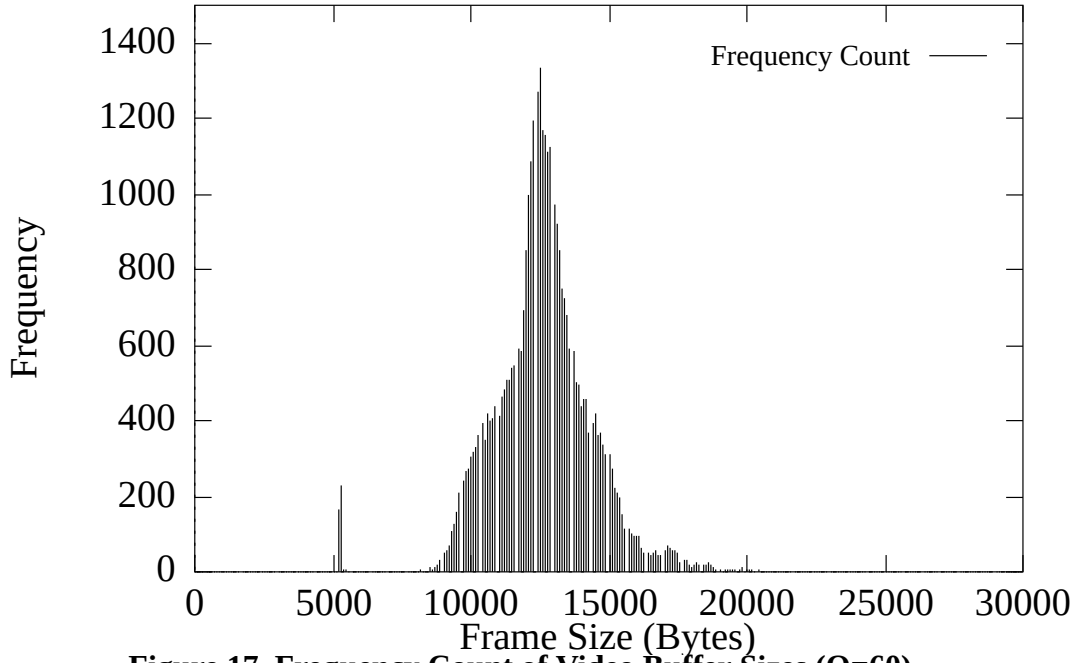

Figure 17. Frequency Count of Video Buffer Sizes (Q=60)

Figure 17 shows the frequency count of all frame buffers from the ten minute video sequence used in testing the performance of the router and admission control protocol. Buffer size is affected by a number of factors including both the quantization(Q) factor and the content of the video frame.

between end-systems, because the decompression hardware is too sensitive to missing data. The current tests also use a *go-back-n* retransmission scheme.

4.3.2 Experiments

In order to quantify the router and admission control performance with the video distribution system, a number of tests were done. For the experiments we established a single unicast video connection between two machines on opposite sides of the router, *Terrapin*. The video sequence was a ten minute clip from the movie, *The Terminator.* The sequence had a number of varied scenes, including both action and “talking head” scenes. *Gluttony* was used as transmitter for the video stream, and opened a connection with the receiver, *Lust*.

The first set of tests varied the allocated rate used to send the video. Using our knowledge of the frequency distribution of frame sizes (Figure 17), we derived the peak rate for the video transmission assuming that a frame is transmitted every 33ms. Under these conditions the peak rate is 21,680 bytes/33 ms * 8 bits/byte = 5.25 Mbits/s. The first test requested and was allocated a rate of six Mbits/s, a burst of 27,000 bytes (six

packets), and high priority status. The burst request was made large enough to handle the sending of the largest frame size plus some extra overhead to cover control packets. The test was then repeated three times with the same burst request and rate requests of 7.2 Mbits/s, 9 Mbits/s, and 12 Mbits/s.

The second set of tests sent varying levels of background traffic while still sending the video stream. First, the connection between *Gluttony* and *Lust* requested and was allocated the following services:

- high priority,
- a rate of 9 Mbits/s, and
- a burst of 27,000 bytes (6 packets).

The allocation choices for the video stream were based on the desire to deliver a high quality video stream to the receiver. For our purposes we defined high quality video as 640x480x8 resolution with a full frame rate of 30 frame/s. The JPEG compression used by the video system hardware allows the user to set the Q factor, so tests were done with the Q factor set at 60. The quality of the video that the destination host receives is a function of a number of factors such as the frame rate, the amount of compression, and the content of the frame. From the results of previous tests we determined that the throughput needed to deliver full frame rate video with a Q factor of 60 had a peak rate of 5.25 Mbits/s. We requested a rate of 9 Mbits/s for reasons that will be discussed in the next section.

During successive tests other traffic was added, which included a high priority file transfer and varying levels of background traffic. The second test involved a video connection, plus a file transfer with the following admission request and subsequent allocation:

- high priority,
- a rate of 18 Mbits/s, and
- a burst of 9,000 bytes (2 packets).

The third test added a low priority stream of background traffic with a throughput of 47 Mbits/s that was generated from sending packets of 4,377 bytes. With this level of background traffic the results of chapter three show that the router should still be able to support 35 Mbits/s of high priority traffic which was greater then the current high priority rate allocations.

The fourth test changed the characteristics of the background traffic such that the

throughput was 7.2 Mbits/s, with a packet size of 128 bytes. With this level of background traffic the results from chapter three indicate that the router is in violation of the amount of background traffic that may be sent with the current allocation of 27 Mbits/s at the high priority level (when sending 27 Mbits/s at the high priority level, the router can only support 5.5 Mbits/s of 128-byte background traffic).

Using the previously mentioned tests our goal was to study the effectiveness of our communication architecture under different traffic loads and look at a number of indicators which could quantify the performance of the system.

First, in order to ensure that the admission control protocol and policing mechanisms were working properly we kept track of the maximum queue depth within the router in the high priority queue. Assuming that background traffic is bounded, the maximum queue depth should never exceed the sum of the allocated bursts in the network system.

Second, we were interested in the effect of the router performance on the delivery of high quality video. In order to quantify the performance of the video we measured the latency of the frame transmission. As mentioned previously, the current version of the video distribution system uses a blocking send call that is supplied by the XTP application interface. Each time a frame is sent this call blocks until the entire frame has been successfully transferred to the receiver. In its current configuration, the XTP transmitter segments the frame into n packets, n = frame size modulo 4,500, and then transmits each packet when the traffic policing mechanisms permit. Then, in the XTP header of the final packet a flag is set requesting that the receiver send its status back to the transmitter, which causes the receiver to respond with a status report that includes where the receiver is in the sequence space of the transmission. If the frame has successfully arrived at the receiver, then the status packet contains this information, and the transmitter will return from its blocking state after receiving the status update. If data is missing then the transmitter will resend the missing information, and request another status update. Once the packet is successfully received by the destination and the transmitter is notified, then the sending call will return. However, if the connection is sending at high priority and is using the resource reservation facilities of the system, then packets should never be lost, because the router should never drop packets. The sender would only resend information if an internal timer

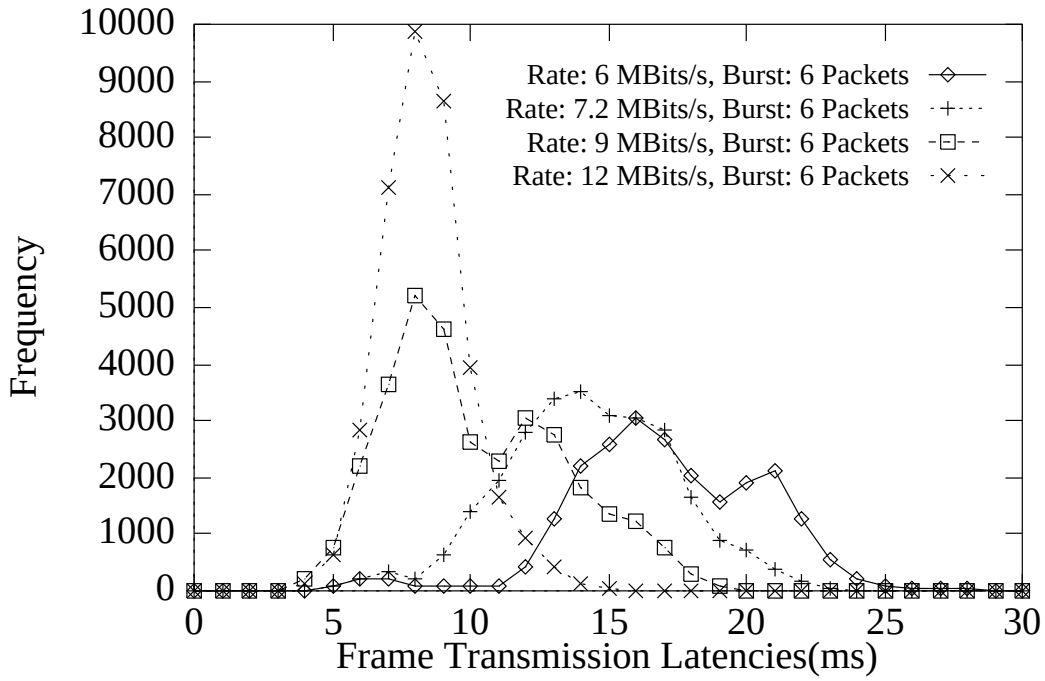

Figure 18. Frame Transmission Latencies Versus the Allocated Rate for the Connection

Figure 18 shows the frame transmission latencies for a ten minute video sequence when sent via a high priority connection through the router to its destination. The figure shows the effect the allocated rate has on frame transmission latencies.

has expired, signifying a large delay since the last response from the receiver. Thus, the latency of the send call represents the sum of the time it takes the transmitter to process a frame, send it, have the router forward all packets to the destination, and receive a response from the receiver.

4.3.2.1 Experimental Results

The transmission time latencies given a requested rate and burst allocation may be seen in Figure 18. The figure shows that as the allocated rate increases the overall latencies of the video transmissions decrease. The percentage of time that a receiver's buffer queue had n packets in it is shown in Table 1. This buffer queue, along with the transmitter's queue help to prevent a loss of synchronization between the source and destination due to jitter. If the queue length is one, then that means the buffer queue has been drained, and the receiver is replaying the most recent frame it has in its queue.

Table 1: Percentage of Time that n Buffers Were Full in the Receive Queue

n	Rate: 6 MBits/s	Rate: 7.2 MBits/s	Rate: 9 MBits/s	Rate: 12 MBits/s
1	37%	25%	9%	1%
2	61%	74%	35%	1%
3	< 0.1%	< 0.1%	10%	75%
4	1%	1%	46%	23%

Even though all four test cases allocated more than the required peak rate for the video stream, the higher rate allocations yielded better performance. The differing performances are a result of the interaction of the buffering policy of the video distribution system, and the update rate of credit for the XTP leaky bucket policing mechanism.

As rate allocation increases the time period between packet updates for the leaky bucket counter decreases. For a rate of 6 Mbits/s the credit counter is updated every 6 ms. So, if the bucket is empty, an outgoing frame of 22,000 bytes (segmented by XTP into packets of 4,500 bytes or less) at the transmitter will take at least 30 ms to transmit completely (and this does not include the protocol processing time at the transmitting workstation). It is easy to envision a case where a number of large frames fall together during video playback. This sequence of frames could cause the receiver to lose its cushion of jitter buffers. The sender at this point has a number of new frames ready to be sent. However, because of the low update rate of its credit counter it is unable to send frames quickly to the receiver even if it is requesting a new frame. Thus, the rate policing keeps the receiver from refilling its jitter buffers. So, each time a frame misses its deadline at the receiver, the receiver must replay the most recent frame it has received. In the test cases where the rate was set higher, then the receiver has a chance to build back its jitter buffers after they have been drained away. Due to this relationship between allocated rate and performance of the video, we decided to overallocate our rate for the video in order to gain the *responsiveness* that the higher rate provides. Thus, in the second set of tests the allocated rate for the video stream is 9 Mbits/s, which is 3 Mbits/s more than the peak rate for this video stream.

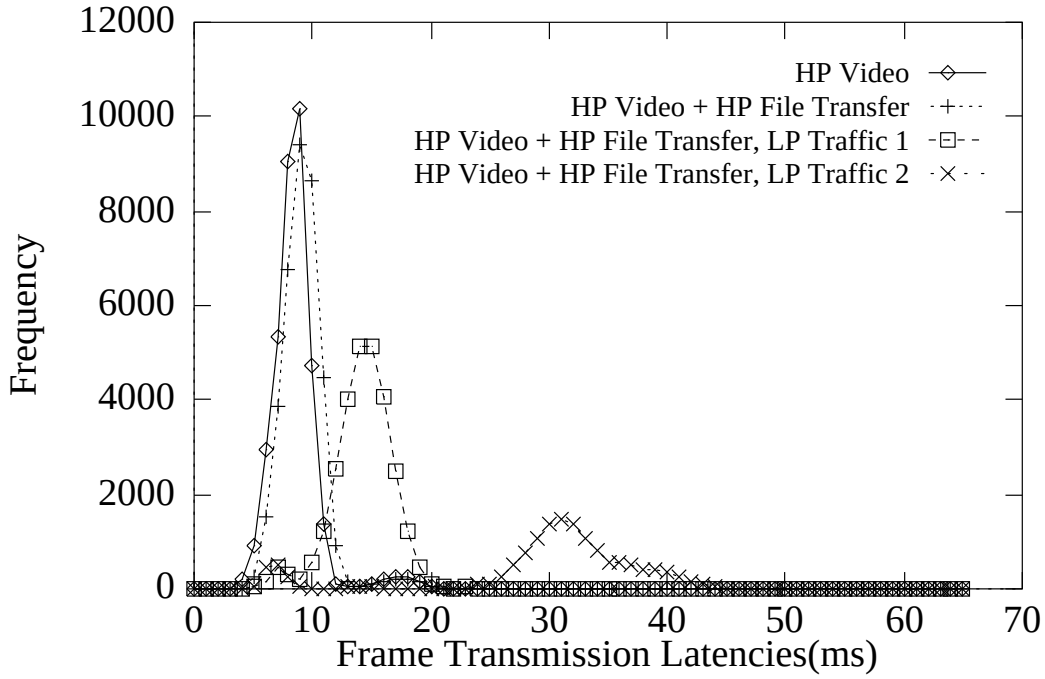

Figure 19. Frame Transmission Latencies for a Ten Minute Video Sequence

Figure 19 shows the frame transmission latencies for a ten minute video sequence when sent via a high priority connection through the router to its destination. The figure shows the effect of varying levels of high priority traffic and background load on the latency of the video transmissions.

The results of the second set of tests showed that the admission control protocol and policing mechanisms worked as expected as long as the background traffic was bounded. The maximum queue depth for high priority traffic was never greater than the sum of the allocated bursts, except in the fourth test case where the background traffic violated its upper bound (given the service allocations made for the higher priority traffic). In this case the maximum queue depth was thirteen for the high priority traffic, but the allocated burst for the two high priority streams was only eight.

Figure 19 shows the results of measuring the video frame transmission latencies. The results show very little difference between the latencies occurring when the video is sent alone versus when the video and file transfer are sent together. The average delay for successful transmission of a video frame was 10.1 ms for the first test and 10.8 ms for the second test, with the worst delay being 22 ms for both the first and second test. The resulting video that was displayed at the receiver appeared smooth with no noticeable signs of any frames missing their deadline for the screen updates (a few, less than 10%, did miss their deadline which results in the replay of one of the frames in the receiver's queue.)

In the third test when 47 Mbits/s of background traffic is introduced, the frame transmission latencies increase. The average latency is now 12.9 ms with the maximum latency being 22 ms. By examining the system one can develop a scenario in which this worst case occurred given the following:

- With a rate of 9 Mbits/s, the transmitter is updating its leaky bucket credit counter every 4 ms.
- With 47 Mbits/s of background traffic, the router can process approximately 39 Mbits/s of high priority traffic with 4,500 byte packet sizes (see chapter three). This translates into 1083 packets/s, or a processing time of about 0.9 ms per packet.

If the transmitter of the video has no credits when it starts to send a 20,000 byte frame, then it will take between 16 and 20 ms until the last packet of the frame enters the network. In the worst case the router could have 8 packets (the sum of the high priority burst allocations) in its high priority queue when this frame arrives. The entering packet may then suffer a delay of up to 7.2 ms before exiting the router. Once the final packet of the video frame is received by the destination host and processed, it must then send a reply back to the transmitter. This control packet could also suffer a 7.2 ms delay on the return path. Thus, without taking into account protocol processing in the end systems the worst case delay due to the network and admission control method is 20 ms + 7.2 ms + 7.2 ms = 34.4 ms.

Qualitatively, the video presented at the receiver in the third test showed little degradation in performance with the frame rate dropping to approximately 25 frames per second (50 fields per second) due to frames missing deadlines.

In the fourth test the background traffic is changed to a rate of 7 Mbits/s, generated by packets of 128 bytes in length. Due to the size and rate of the background traffic, the router is put into a state where it can make no performance guarantees for the high priority traffic. The effect of this change is seen in a large increase in the latencies of transmissions for video frames. The average frame latency for a frame is approximately 26 ms with a maximum latency of 33 ms. The effect of the loss in performance guarantees is also qualitatively seen at the receiver where the frame rate has dropped to approximately 15 frames per second.

Chapter 5

Summary, Conclusions, and Future Work

In this chapter we summarize our work and discuss the conclusions that can be drawn from the design, implementation, and tests of our communication architecture. The section on future work describes possible extensions that can be made to the current architecture.

5.1 Summary and Conclusions

A paradigm shift is underway in how computer networks are used. As computers become faster multimedia applications are becoming more common. The result of this shift is that latency control becomes more of an issue. Different types of networking paradigms offer varying levels of service that can help control latency.While circuit-switched networks can offer dedicated links with a certain level of service, they are both inefficient and costly (in terms of the cost for the bandwidth needed for video distribution). This leaves packet-switched networks with its two paradigms of virtual circuits (associated with ATM) and datagram networks (Ethernet, FDDI). ATM offers a number of service guarantees that are ideal for the transmission of multimedia traffic. For instance, the specification supports rate control and prioritization. However, when this project started the ATM equipment that was available did not offer all of these features. It appears that the acceptance of ATM is growing, but it will be years before its use is widespread. For instance, the U.S. Navy has spent the last ten years refining the SAFENET architecture which uses FDDI, and instead of switching directly to ATM they have decided to wait until ATM becomes a more stable product. For these reasons we chose a datagram networking paradigm over which we would design our networking architecture to support multimedia traffic.

Thus, our goal for this project was to design a packet-switched network that can support soft real-time applications such as multimedia. At the outset we made the assumption that system capacities would be intelligently allocated, allowing our focus to

be on the communications protocol, the router design, and the resource reservation and admission control policies that would allow the overall system to operate.

Our router design was based on a PC architecture. The advantage of using a PC-based architecture was that source code was available for all parts of the protocol stack from the network adapter drivers to the networking protocols. The access to the source code (written by myself and others) allowed for more control over the entire system. The disadvantage of the architecture is that it is vulnerable to excessive interrupts from the network adapters. This vulnerability could be removed by changing the architecture, or by policing all traffic entering the network

In spite of these limitations we developed a router with usable performance. The router is able to support a peak rate of approximately 50 Mbits/s when sending 4,500-byte packets, and a peak packet rate of approximately 1,450 packets/s when sending 48-byte packets. Tests of latency and throughput showed the two main bottlenecks of the router to be:

- a decrease in router performance as the inter-arrival time between packets decreases (corresponding to an increase in the number of interrupts in a given time unit), and
- a decrease in router performance as the packet size increases.

Because of the bottlenecks in the router and the potential for congestion, we developed an admission control policy for our network. The admission control design has a number of components:

- Service Paradigm — the way that routers and end-systems process traffic determines the type of service guarantees that one can make for a connection.
- Traffic Policing Mechanism — the traffic entering the network must conform to some characterization so that the admission control algorithm can make a decision as to whether the network has resources to handle a new request without violating any parameters of current requests.
- Resource Administration Mechanism — there must be a correspondence between the parameters that are used to make requests for service from the network, and the resources that are used by the network to fulfill the requests.
- Resource Reservation Protocol — there must be a method for passing requests for service throughout the network.

Our network uses a multiple static priority queue service paradigm. By using this

service paradigm we could take advantage of the existing support for priority levels in the Xpress Transport Protocol by altering the router to sort incoming packets based on XTP's priority field. Therefore, the router is XTP-aware. A benefit of having multiple queues is that each queue has its own worst case delay bound as opposed to the regular FIFO queue which offers a single delay bound. Our test results indicate that the multiple priority queue approach is an effective mechanism for delivering various levels of service to different traffic types.

We examined several methods of traffic policing before choosing a leaky bucket mechanism. For instance, XTP offers a jumping window mechanism for rate and burst control. However, this method of policing was not suitable for our system because the worst case packet burst was twice the burst of a single interval. This results in over-allocation of buffer space at the router to ensure that packets are not dropped. Instead, by using the leaky bucket mechanism we could allocate for a specific rate and burst within the router. Tests of our implementation of the leaky bucket mechanism within XTP showed that the mechanism did police the traffic correctly and could be used in tandem with XTP's jumping window mechanism if desired.

By studying the performance tests of the router and identifying the bottlenecks we were able to manage the scarce resources. In our system, the scarce resources were processor time in the router followed by buffer space within the router. Processing time was mapped to the rate of traffic allowed to enter the network, and buffer space was mapped to the aggregate burst of packets allocated to all connections through the router.

The resource reservation protocol was developed as an extension to the existing ICMP protocol. The use of this resource reservation protocol facilitated testing of the other aspects of our system, but is by no means a full-fledged reservation protocol.

By testing the performance of the router under various traffic loads of high and low priority we determined the analytic criteria that must be met in order to give guaranteed throughput with a specified burst to a requesting connection.

Assuming that background traffic has a minimum packet size of 1,024 bytes, and that its maximum rate never exceeds 30 Mbits/s, then we know that the router can handle 30 Mbits/s of traffic at high priority when packets are 4,500 bytes in length. Under these conditions we can keep track of the aggregate rate and burst that is currently allocated

through the router, and use these values to determine if a new request may be accepted. The admission test is:

$$\sigma_{request} + \sum_i \sigma_i \; < \; 32 \text{ packets} \cdot 4,500 \text{bytes/packet}$$

$$\rho_{request} + \sum_i \rho_i \; < \; (3,750,000 \text{bytes/s})$$

The final part of the project involved testing our network architecture with a video distribution system that uses XTP. We used this video distribution testbed to illustrate the effectiveness of our network architecture. The results from the tests involving multimedia traffic show that congestion within the router is eliminated so long as all data streams obey their traffic characterizations (via policing or agreement). Under these conditions the worst case delay through the system can be bounded (assuming full access to the network).

The tests also show that there are a number of complex interactions between the video distribution system, the transport protocol, and the traffic policing mechanisms. In most traffic characterizations, control information is assumed to be out-of-band and is ignored. In our system, the traffic policing mechanism polices all packets entering the network. This method has the possibility of introducing delays into the protocol control mechanism which were not there before, and as a result it could change the timing of the protocol. Second, the video distribution system uses synchronous send calls on the transmitting side. These calls simplify the operation of the sending side, but also serialize its operation, thereby slowing down the sender. Changing the design of the video distribution system could improve its performance, which would stop the need for the over-allocation of rate which occurred in the testing for this project. The major benefit of testing our network architecture with the video distribution system is that the tests involving the transmission of a multimedia data stream offer *proof-of-concept* evidence in support of the project.

Thus, after implementing our design and performing experiments testing throughput, latency, and the performance of multimedia traffic under various traffic loads we can draw these of conclusions.

(1) By managing the limiting resources of the system, it is possible to guarantee requests for rate, and implicitly give an upper bound on the worst case delay between two end-systems

due to the service discipline used within the router.

(2) Through the use of a leaky bucket traffic policing mechanism at the end-stations, and a multiple static priority queuing discipline within the router, the admission control mechanism can guarantee that a requested rate and burst of packets at the high priority level will be delivered to the destination host.

(3) By using results from performance tests, it is possible to derive the worst case delay bound for the high priority queue which can then be used to determine an end-to-end bound on the worst-case delay.

5.2 Future Work

Certain design decisions made during the course of this project deserve further attention.

Choice of Service Discipline for the Router — The router currently uses a multiple static priority queue service discipline. Due to the bottlenecks in the router and the nature of the multiple static priority queue, the efficiency of the system is not ideal. If time permitted, implementing other service disciplines such as Rotating Priority Queue (RPQ)[24], Delay Earliest Due Date (Delay-EDD) [12], or Stop and Go [18] along with their respective admission control tests would allow for a quantitative analysis of the tradeoffs between efficiency and the cost of complexity in implementation.

Bounding the Background Traffic — The test results from this project indicate that as long as bottlenecks on system performance exist in the router, then all traffic entering the network must be policed or bounded in some way. Off-the-shelf protocol stacks do not offer the needed policing mechanisms, and even the use of FDDI synchronous class traffic would not bound the number of packets per second that could be sent in a given interval as regular FDDI traffic. This results in the possibility that a rogue or errant user sending low priority traffic could cause violations of service guarantees for high priority traffic in the current system design. A number of options are available in order to strengthen the system’s defense against such users.

The processing of all high priority packets could be done at interrupt time in the interrupt subroutine of the device driver. The benefit of such a move is that incoming low priority packets would not delay the processing of the high priority packet. The trade-off of

such a solution is that the hardware receive queue might overflow because incoming packets would wait there while a high priority packet was being processed. This problem might be alleviated if the FDDI card supported two queues, one for the synchronous class (higher priority) and one for the asynchronous class (lower priority).

Another option for guarding against excessive processing times of low priority packet interrupts is to disable either the board or address which is being overloaded. Interrupts for certain boards could be masked while processing high priority traffic, or one could use multicast MAC addresses for regular traffic. If the second option was used then whenever low priority traffic became a problem, the card could be configured to ignore incoming packets for a period of time. The efficiency and feasibility of each of these options is uncertain and deserves further study.

Single Router System versus a Multiple Router System — By having only one router in our communication architecture we are able to ignore the problem of rate and burst violations that could possibly occur at internal network nodes due to the work-conserving nature of most routers (ours included). Adding multiple routers forces a linear increase in the amount of buffer space that must be allocated along a multi-router path if one is using a non-work conserving service discipline in the routers [44]. Depending on the rate and burst requests for the network, and the number of internal hops, the current service paradigm of multiple static priority queues might suffice. However, there are a number of non-work conserving disciplines which either fully reconstruct traffic patterns between routers, or at least partially reconstruct the traffic in order to police rate and burst violations between routers [18],[44],[24],[43]. There exist many trade-offs between the offered service and the implementation complexity of these service disciplines.

Traffic Policing within the End-Systems — XTP offers a jumping window traffic policing mechanism, and our design incorporated a leaky bucket policing mechanism into XTP as well. Studying the efficiency and performance of the protocol using multiple leaky buckets would be of interest. Also, the current version of the leaky bucket mechanism is interrupt driven. By changing the method of counter update one could take advantage of using the fine grained timers available on a PC without having to interrupt the CPU at a frequency equal to the minimum granularity of the clock. Instead of updating the credit counter via an interrupt, the credit counter value can be calculated by reading the clock

before each attempted send. The difference between the two times plus the previous value of the credit counter will allow determination of the credit counter's new value. Implementation issues such as detecting the timer wrap would have to be resolved. Testing indicates that interrupts indicating a timer wrap are often missed by the PC.

Implementation Considerations for the Multimedia Distribution Application — The current version of the multimedia distribution application uses a reliable service as well as a blocking send call for each frame transmission. By altering the application to use an asynchronous send, the application could work on other tasks instead of remaining idle until a response returns from the receiver. This change would probably decrease the latency of frame transmissions. Another change which would decrease the frame transmission latencies is changing to an unreliable service paradigm. If the compression/decompression hardware could be configured in such a way as to be more fault-tolerant of missing data, then send latencies would be further reduced. The XTP transmitter would not have to wait for an acknowledgment from the receiver before transmitting a new frame. Instead, the transmitter would only be policed by the rate control mechanisms within XTP.

References

[1] A. Alles, “ATM in Private Networking,” Hughes LAN Systems, Tutorial, 1993.

[2] A. Banerjea et al., “The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences,” *Computer Science Technical Report Number TR-94-059*, University of California at Berkeley, November 1994.

[3] M. Bach, “The Design of the Unix Operating System,” Prentice Hall, Englewood Cliffs, NJ, 1993.

[4] A. Cambell, G. Coulson, and D. Hutchinson, “A Quality of Service Architecture,” *Computer Communication Review*, Vol. 24, No. 2, April 1994.

[5] A. Cambell, G. Coulson, F. Garcia, and D. Hutchinson, “A Continuous Media Transport and Orchestration Service,” *Proc. ACM SIGCOMM92*, Baltimore, MD, August 1992, pp. 99-110.

[6] R. Christie, and A. Weaver, “Supporting Multimedia Traffic Via an XTP-Aware IP Router,” Draft, January 1995.

[7] D. Clark, “The Design Philosophy of the DARPA Internet Protocols,” ACM 1988.

[8] D. Clark, S. Shenker, and L. Zhang, “Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism,” *Proc. ACM SIGCOMM92*, Baltimore, MD, August 1992, pp. 14-26.

[9] D. Chiu and R. Jain, “Analysis of Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks,” *Computer Networks and ISDN Systems*, 1989, pp. 1-14.

[10] L. Delgrossi et al., “Reservation Protocols for Internetworks: A Comparison for ST-II and RSVP,” *Fourth International Workshop on Network and Operating System Support for Digital Audio and Video*, Lancaster, UK, November 1993.

[11] B. Dempsey, M. Lucas, and A. Weaver. “Design and Implementation of a High

Quality Video Distribution System Using XTP Reliable Multicast". *Proceedings of the International Workshop on Advanced Communications and Applications for High Speed Networks*, Heidelberg, Germany, September 1994.

[12] D. Ferrari, and D.C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE Journal on Selected Areas in Communications*, April 1990, pp. 368-379.

[13] S. Floyd and V. Jacobsen, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, August 1993, pp. 397-413.

[14] H. Folts, "A Tutorial On The Open Systems Interconnection Reference Model," *Open Systems Data Transfer*, June 1982, pp. 2-21.

[15] J. Forgie, "A Proposed Internet Stream Protocol", *IEN 119*, 1979.

[16] J. Hawkinson, ed. "Cisco Networking Faq Version 1.0," RFC 1153, November 1994.

[17] J. Hyman, A. Lazar, and G. Pacifici, "Joint Scheduling and Admission Control for ATS-based Switching Nodes," *Proc. ACM SIGCOMM92*, Baltimore, MD, August 1992, pp. 223-234.

[18] J. Golestani, "Congestion-Free Communication in High-Speed Packet Networks," *IEEE Transactions on Communications*, December 1991, pp. 1802-1812.

[19] V. Jacobsen, "Congestion Avoidance and Control," *Computer Communication Review*, Vol. 18, No. 14, May 1988, pp. 314-329.

[20] R. Jain, "Performance Analysis of FDDI Token Ring Networks: Effect of Parameters and Guidelines for Setting TTRT," *Proc. ACM SIGCOMM90*, Philadelphia, PA, 1990.

[21] E. W. Knightly, D. E. Wrege, J. Liebeherr, and H. Zhang, "Fundamental Limits and Tradeoffs of Providing Deterministic Guarantees to VBR Video Traffic," To appear in *ACM Sigmetrics*, May 1995.

[22] J. Kurose, "Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks," *3rd International Workshop on Networking and Operating System Support for Digital Audio and Video*, San Diego, CA, November 12-13, 1992.

[23] S. J. Leffler, W. N. Joy, R. S. Fabry, and M. J. Karels, "Networking

Implementation Notes: 4.3 BSD Edition," *Unix System Manager's Manual*, USENIX *Association*, 1986.

[24] J. Liebeherr and D. Wrege, "Design and Analysis of a High Performance Packet Multiplexer for Multiservice Networks with Delay Guarantees," Technical Report CS-94-30, University of Virginia, July 1994.

[25] D. Plummer. "An Ethernet Address Resolution Protocol," *RFC 826*, November 1982.

[26] J. Postel, ed. "Internet Protocol," *RFC 791*, September 1981.

[27] J. Postel, ed. "Internet Control Message Protocol," *RFC 792*, September 1981.

[28] F. Ross, "An Overview of FDDI: The Fiber Distributed Data Interface," *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 7, September 1989.

[29] F. Ross, "FDDI— A Tutorial," *IEEE Communications*, Vol. 24, No. 5, May 1986.

[30] R. Simoncic and A. Weaver, "Choosing The Right Real-Time Operating System For The Navy," NRaD Report, July 1994.

[31] W. Stallings, "Networking Standards: A Guide to OSI, ISDN, LAN, and MAN Standards," Addison-Wesley, Reading, MA, 1993.

[32] W. Stallings, "Data and Computer Communications," Macmillan Publishing Company, New York, NY, 1994.

[33] W. R. Stevens, "TCP/IP Illustrated Volume 1," Addison-Wesley, Reading, MA, 1994.

[34] H. Tokuda, Y. Tobe, S. Chou, and J. Moura, "Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network," *Proc. ACM SIGCOMM92*, Baltimore, MD, August 1992, pp. 88-98.

[35] H. Tokuda, C. Mercer, and Y. Ishikawa, "ARTS: A Distributed Real-Time Kernel," *ACM Operating Systems Review*, July 1989.

[36] C. Topolcic, "Experimental Internet Stream Protocol, Version 2 (ST-II)," *RFC 1190*, October 1990.

[37] J. Turner, "New Directions in Communications (or Which Way to the Information Age?), *IEEE Communications Magazine*, Vol. 24, No. 10, October 1986.

[38] G. Wallace. “The JPEG Compression Standard”. *Communications of the ACM*, 34(4):30-44, April 1991.

[39] A. Weaver, “What is the Xpress Transport Protocol?”, Transfer, XTP Information, December 1994.

[40] G. Woodruff and R. Kositpaiboon, “Multimedia Traffic Management Principles for Guaranteed ATM Network Performance,” *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990.

[41] D. Wrege, Personal communications, 1995.

[42] L. Zhang, D. Estrin, S. Herzog, S. Jamin, and R. Braden,ed. “Resource Reservation Protocol (RSVP) — Version 1 Functional Specification,” Internet Draft, March 1995.

[43] H. Zhang, “Providing End-to-End Performance Guarantees Using Non-Work-Conserving Disciplines,” to appear *Computer Communications Journal*.

[44] H. Zhang and D. Ferrari, “Rate-Controlled Static Priority Queueing,” *Proceedings of IEEE INFOCOM’93*, 1993.

[45] L. Zhang, S. Deering, D Estrin, S. Shenker, and D. Zappola, “RSVP: A New Resource Reservation Protocol”, *IEEE Network*, September 1993, pp. 8-18.