The XUDP Protocol for Timely Multimedia Transport

Michael J. Andrews and David Cyganski

Convergent Technology Center
ECE Department, WPI
Worcester, MA 01609-2280

Abstract

Current efforts to make networked real-time multimedia data distribution a practical reality have focused primarily on the development of means to exploit the TCP protocol. However, much of the Internet is sufficiently slow or congested as to force most multimedia applications to use large "streaming buffers" to accommodate the highly non-deterministic nature of this communications channel. Some users may prefer to trade full reliability and corresponding fidelity for a variable quality, but low latency and properly paced transmission.

XUDP (eXtended User Datagram Protocol) is a transport layer protocol that provides an alternative, semi-reliable, method for transmitting real-time, multimedia data over an internet. Through a process of "timed obsolescence," an application can deliberately specify a time to discard any given data parcel for which all comprising UDP datagrams have not been received. This allows placement of a value on the preservation of the timing relationships between sequential multimedia data parcels which is higher than the value given to reliable but un-timely delivery. XUDP is currently implemented as a user-space daemon, operating above UDP through Berkeley sockets, taking advantage of UDP's fast payload checksum and multiplexing ability. It offers full flow-control with a number of TCP-like optimizations and a sockets-like API.

Keywords: multimedia data transport, internetworking, loss-tolerant, transport-layer protocol

1 Introduction

Given the large amount of data required to present a realistic motion video stream to the user, it is inevitable that in an internetworked video application, transmitted packets will be lost either through collisions or in overrun buffers when the necessary transmission bandwidth exceeds the available network bandwidth. The TCP/IP protocol suite has provided Internet application developers with two transport layer protocols: a fully-reliable stream protocol (TCP) and a best-effort datagram delivery service (UDP). However, the recent trend toward real-time multimedia data transmission over an internetwork demands a type of service that TCP/IP alone, by design, cannot provide: *semi-reliability.* The notion of semi-reliability stems from the observation that *the most important factor in the user appreciation of real-time, multimedia data is the linear time-wise progression of the time-sensitive data.*

Ideally, multimedia applications should have a certain amount of control over what data needs to get through, and what data can be discarded if network congestion occurs. Recently, real-time multimedia applications have been making use of UDP as an alternative to the more robust TCP. UDP functions primarily as an interface to the best-effort delivery service of IP, and as such offers no guarantees of the reliability of its data transmission service. In fact, UDP offers no sort of flow control whatsoever, frequently resulting in less-than-desirable performance on fast computers where applications are able to overload local networking buffers, losing packets before they even enter the physical network. Using TCP to transmit the data certainly guarantees that all data will be received at the other end of the connection. If no buffering methods were in place in the application, this behavior would result in highly variable delays in the presented video stream. Even utilization of TCP with buffering, or multimedia streaming, can result in a long initial presentation delay at the user's end, and a great deal of abuse of network bandwidth resulting from the multiple segment retransmits that may be needed to enforce TCP's reliability. Protocols such as XTP[8] have focused primarily on multimedia data delivery over high speed networks (ATM primarily) and do little to solve the existing problem of transporting multimedia data over our now heavily congested Internet. RTP (the Real Time Protocol[6]) is designed specifically for internetworking usage, but places the burden of flow control in the application layer if used directly by any application. The recent (1996) ITU-T H.323 standard provides a structured means of regulating video communication, specifically, on internetworks that do not implement Quality of Service (QoS). XUDP provides flexible and configurable services that can accommodate many multimedia transmission requirements and network modalities.

This paper documents the research and development of the eXtended User Datagram Protocol (XUDP), built to fill a "middle layer" between the application and the User Datagram Protocol. It attempts to fit the need for *semi-reliability* by making use of concepts and innovations from standard existing protocols such as the Transmission Control Protocol and recent multimedia-specific protocols such as the eXpress Transport Protocol (XTP). During the design of XUDP, particular attention was paid to the well-documented historical development of TCP. Our proposed solution is built around a concept known as *timed-obsolescence,* which states that multimedia data should have a window of usefulness associated with it, a sort of expiration date for video and audio.

XUDP is currently an in-progress effort, conducted by the Convergent Technology Center at WPI. Early results obtained with a primitive implementation of XUDP are promising.

Section 2 characterizes multimedia data transmission in a context relevant to the concept of timed-obsolescence. Section 3 describes the eXtended User Datagram Protocol and presents certain implementation issues. Section 4 concludes the paper.

2 Characteristics of Multimedia Data

Real-time multimedia data is expensive to transport and must be received in a timely manner, with utmost importance placed on preserving the linear progression of sequential timed data at the user's end.

Two major obstacles stand in the way of high-fidelity, real-time multimedia data transport over an internet: limited available network bandwidth and dynamically varying transmission delays. Understandably, the larger problem of the two is the high bandwidth requirement necessitated by real time audio and video. In the MPEG-2 compression standard, for example, the Main profile at the Low level allows for a 30 frame per second 352x288 pixel resolution image, requiring a communication channel bandwidth of 4Mbit/s for successful transport.[2] This is a high data transmission rate that is nonetheless supportable on a modern LAN.

For a packet switching internetwork, the transmission latency is inherently non-deterministic, and when networks become congested, extreme latencies, many orders of magnitude greater than that of any average expected latency, can result. Figure 1 represents a set of images that comprise a timely delivered movie of a bouncing ball. Transporting this video stream across an internet with an unbuffered connection can undesirably vary the inter-frame temporal spacing, distorting the presentation of timed sequential data beyond what is acceptable to the viewer. (Figure 2)

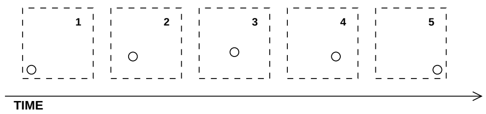

Figure 1: Frames in the complete video stream of a bouncing ball.

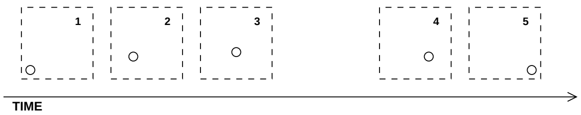

Figure 2: Frames in a video stream that doesn't preserve the linear time progression relationships.

The motion depicted by the images in figure 2 defies our kinematic sense about what happens when a ball bounces, and is different than figure 3, in which a frame is skipped, with the inter-frame timing relationships preserved.

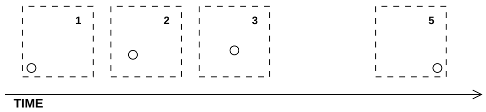

Figure 3: Frames in a damaged but linear time progression relationship preserving video stream of a bouncing ball.

Multimedia streaming was developed to attempt to overcome, or at least stave off in moderate cases, the effects of varying transmission delays. By buffering a sufficient amount of the data before presenting it to the user, multimedia streaming can allow an application to maintain the inter-frame timing in the face of variable delivery rates. The rate of data output is independent of the input rate, as long as there is enough data in the buffer to source the required amount of output. If the input rate lags behind the output rate, eventually there will not be enough data in the buffer to support the high output rate, resulting in the depletion of the multimedia reservoir, forcing the presented data stream to follow the variable rate network data supply.

Although this method of multimedia data presentation offers high-fidelity output at the user's end, it can tend to utilize quite a bit more of the available network bandwidth than is acceptable for a "nicely behaved application." If the streaming protocol is built on top of a fully reliable protocol such as TCP, segments of data that are lost due to congestion will be retransmitted until they have been acknowledged properly, adding to the congestion of the network. Reliable data transmission protocols such as TCP were not designed to handle the special requirements of real-time applications. They were designed for low bandwidth interactive applications such as `telnet` and potentially high bandwidth non-interactive applications such as electronic mail handling and `ftp`. The best-effort transport services inherent to UDP are more suited for delivering multimedia data payloads.

The User Datagram Protocol is becoming important in the realm of multimedia protocols. Because it is essentially an interface to the low-level Internet Protocol, and because it offers a speedy checksum and I/O multiplexing through Berkeley sockets, it is an ideal choice for applications that do not wish to be constrained to the flow control mechanism in TCP. However, operation without any flow control in place will quickly fill the local socket-level buffers and UDP datagrams will be discarded *before they even reach the physical network.*[3] In effect, protocols such as TCP, with a highly refined flow control mechanism, attempt to dynamically estimate the optimum transmission rate through the feedback loop formed by data transmission and subsequent data acknowledgment. Given this observation, operation without flow control is out of the question for high-bandwidth multimedia applications.

With earlier protocol efforts, all data was assumed to be important and therefore was delivered reliably; there was very little chance that a transferred file would be acceptable if corrupt or incomplete in any way. With real-time multimedia data, the primary concern should be preservation of the linear presentation of timed sequential data. Other aspects of the end product's quality should be allowed to suffer, as long as the remaining data is delivered in a way that does not disturb the linear time progression of the presentation. Our proposed protocol, XUDP is an internetworking, semi-reliable multimedia data transmission system with the ability to timestamp parcels of data with expiration dates. It is designed to preserve the linear progression of timed sequential data at the user's end in the face of network congestion related data loss and delay. It reduces network bandwidth utilization significantly by not attempting to transmit or retransmit data that has outlived its usefulness.

3 Implementation

XUDP is currently implemented as a UNIX daemon, as shown in Figure 4. Utilizing a separate process for the protocol implementation allows the protocol flow control to function asynchronously from the connected multimedia application and preserves the layered programming paradigm; only those functions related to the application programmers interface (API) are implemented in the user's process. However, this separation comes at the cost of an additional connection to move the data between the protocol's daemon process and the user's application. Fortunately, the connection exists only between two processes on the same machine, so it has been implemented with UNIX streams - yielding much lower system load than a TCP/IP implementation.

Figure 4 also depicts XUDP's positioning in the protocol stack. Implemented in user space, XUDP rests on top of the User Datagram Protocol (UDP), making use of UDP's highly refined payload checksum and port multiplexer. Future implementations of XUDP may choose to use raw IP, skipping UDP and removing a layer of complexity from the protocol.

From the application programmer's perspective, XUDP is accessed through a series of high-level Berkeley sockets-like functions: `xudpSocket, xudpBind, xudpListen, xudpAccept, xudpConnect, xudpSend, xudpRecv` and `xudpClose.` It effectively preserves the conceptual protocol layering scheme found in the TCP/IP protocol suite. Figure 5 illustrates the similarity of this interface to the existing Berkeley sockets API (the diagram is purposely similar to Stevens' *Figure 6.2[7, p.261]*).

Shown below is an excerpt of a simple client application, showing the usage of `xudpSocket, xudpConnect, xudpSend` and `xudpClose` typically exhibited by clients. Important parameters are

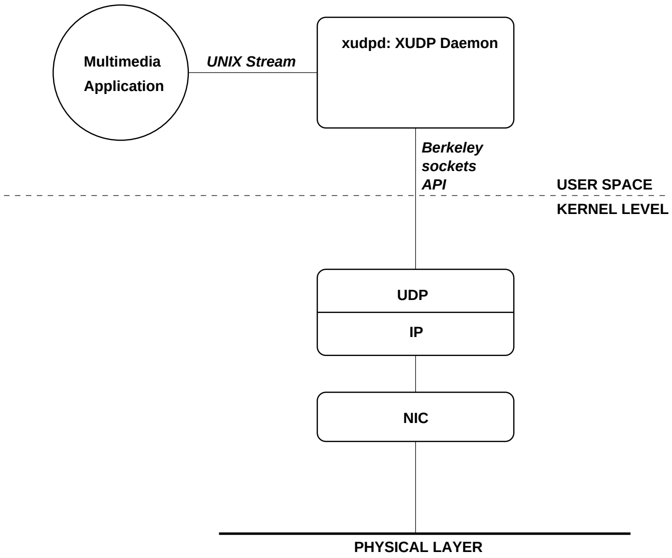

Figure 4: Protocol Layers of XUDP

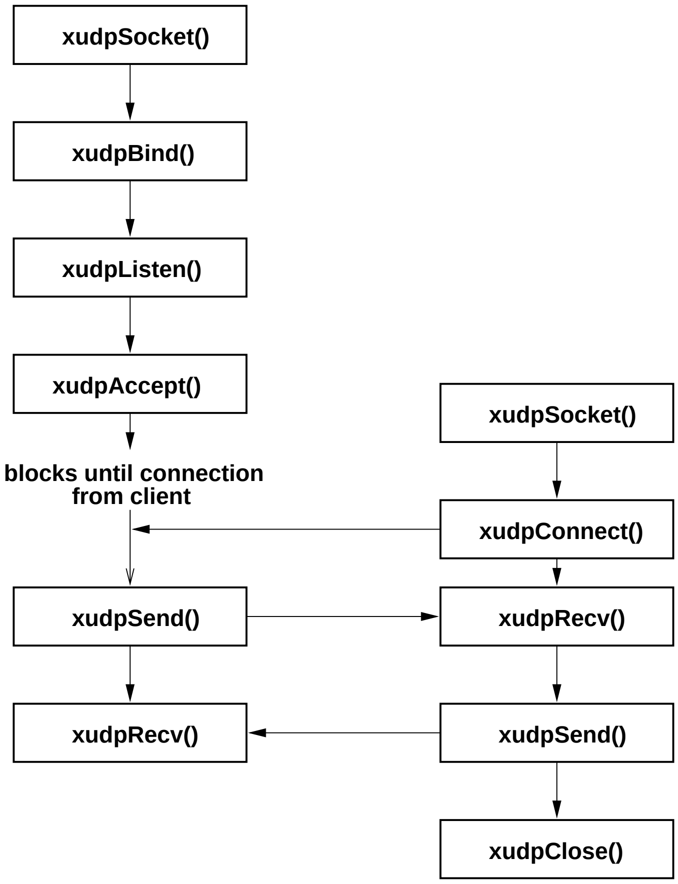

Figure 5: XUDP Socket API Calls

as follows: `argv[1]` contains the hostname of the remote application, `buffer` is a character buffer of length 10240 bytes and `timed_ob` is equal to the time until obsolescence in milliseconds for this parcel of data.

```
xudphost=xudpSocket();
xudpConnect(xudphost, argv[1], 0);
if (xudpSend(xudphost, buffer, 10240, timed_ob)<=0)
       printf("Error!");

sleep(2);
xudpClose(xudphost);
```

Excerpt from a simple client application.

Much of XUDP's functionality lies in the adoption of a new unit of internetworking data transfer named *the parcel.* A parcel is a single unit of inseparable data that is defined by the user's application, such as a frame in a video, a one second burst of audio or any number of imaginable constructs that require delivery as a whole. The size of each parcel is not fixed, and is set at the time that the parcel is sent from the application.

XUDP is capable of sending the parcel in one of three modes:

1. **Full reliability:** parcels are guaranteed to be received at the remote application in a timely manner. XUDP's ability to retransmit packets guarantees that all of the data in a parcel will be received by the application. (`timed_ob = MAXINT`)
2. **Semi-reliability (timed obsolescence:)** parcels are stamped with a time-to-obsolescence (sell-by time or expiration time). If the XUDP server detects that the entire parcel will not be received before the time expires, all data related to the parcel, whether acknowledged or unacknowledged, is removed. This allows XUDP to conserve network bandwidth for the purpose of sending data that is still *useful.* (`timed_ob =` time in milliseconds)
3. **Best-effort, one shot delivery attempt:** Similar to the service provided by UDP, this delivery method offers no chance of retransmission. (`timed_ob = 0`)

Unlike TCP, which transmits *segments* of a stream, parcels are not necessarily part of a continuous stream of data, and thus preserve record boundaries. A single parcel, comprising some indivisible cell of data such as a video frame, may be fragmented by XUDP; if this is the case, the receiver guarantees complete, in-order reassembly of data comprising a single parcel, or no delivery due to activation of the timed obsolescence mechanism. In addition, preservation of the timed obsolescence concept through the entire protocol stack necessitates selective acknowledgment and retransmission at the packet level. Although XUDP operates with a sliding window scheme, windowing is based on the *typically MTU-sized packet* as the fundamental unit of transmission, rather than the byte. XUDP can fill the network with as many packets as the window allows, waiting until the retransmission time out timer expires or an acknowledgment is received for either a packet or an entire parcel. Parcels are never delivered incomplete.

Many of XUDP's flow control algorithms have been modeled after enhancements to TCP implementations, documented in [1][4] and [5]. These flow control mechanisms include:

1. A slow-start on connection initiation that ramps up the number of packets in the data window. [5]

2. An exponential retransmit timer backoff on successive retransmits. [5]

3. Packet round trip time variance estimation and per-packet time-stamping is used to aid in retransmission timer timeout prediction. [4]

4. Congestion anticipation is implemented by scaling the size of the window through close monitoring of RTT's and comparisons with recent RTT's. [5] and [1]

To manage selective acknowledgment and retransmission, as well as the timed obsolescence mechanism, XUDP organizes data as doubly-linked lists at various points in the transmission process. The queues are presented graphically in figure 6.

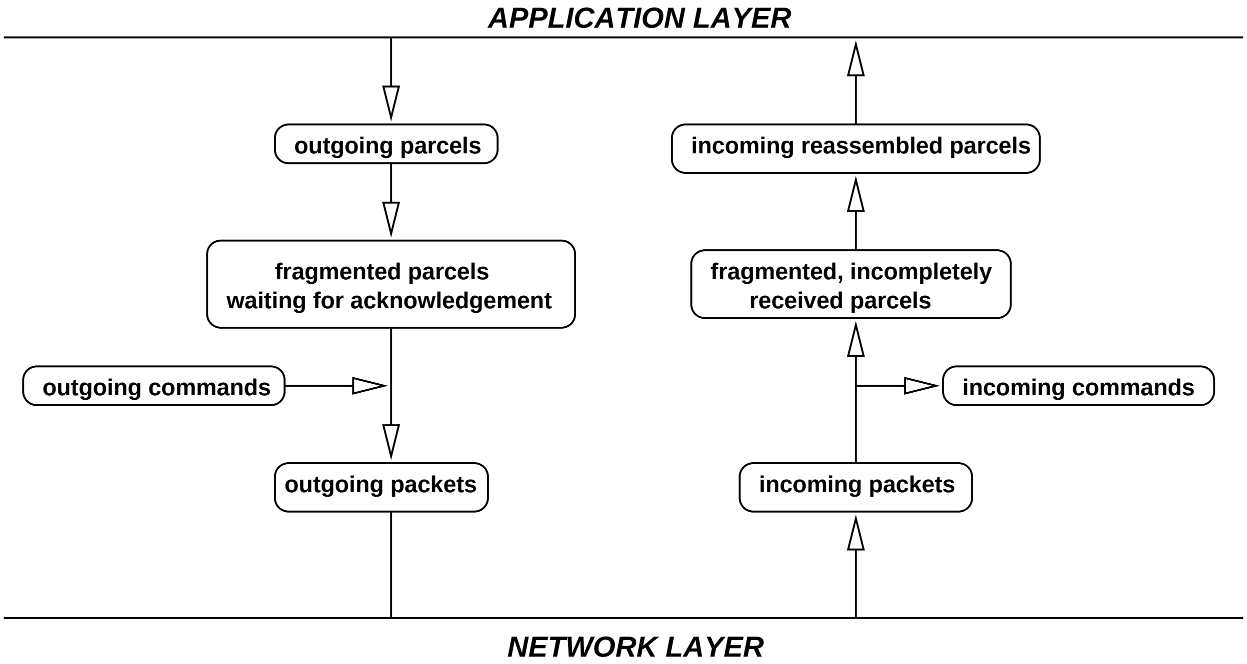

Figure 6: XUDP's Queues

As data isn't presented in a stream, and parcel boundaries need to be preserved, intermediate buffers (*fragmented parcels waiting for acknowledgment* and *fragmented, incompletely received parcels*) are required to hold incomplete parcels. Upon activation of the timed obsolescence mechanism, the data comprising the obsolete parcel(s) is removed from these queues.

4 Conclusions

The eXtended User Datagram Protocol provides a unique method of delivering real-time multimedia data in a manner that can minimize both bandwidth utilization and distortion in the timing of sequential data in minimally buffered multimedia stream delivery applications. By enforcing a policy of timed obsolescence, deliberately discarding data that has outlived its usefulness, XUDP saves network bandwidth and presents a timely, variable quality communication channel to the application layer. It shoulders the responsibility of dynamic flow control, simplifying multimedia application development.

The following graph illustrates the functionality of XUDP's timed-obsolescence algorithm on a network suffering from severe load, with the load removed one-third of the way to the end of the test. In a 30 second test of a simulated 20fps, 160x120 uncompressed 6-bit videoconferencing application, XUDP's transmission is finished in just over 32 seconds, while TCP's stream isn't finished until the 42 second mark - *a ten second difference.* During the period of severe loading, the XUDP transmission is reduced to a minimum one frame per second, while TCP stretches the time-wise pacing of data. An XUDP application could have presented the video stream unbuffered with satisfactory results, while the TCP's unbuffered stream would, on average, get presented 25% slower. The two "extra" seconds are equal to the *time until obsolescence* for all parcels, and could be smoothed by a two second video stream buffer.

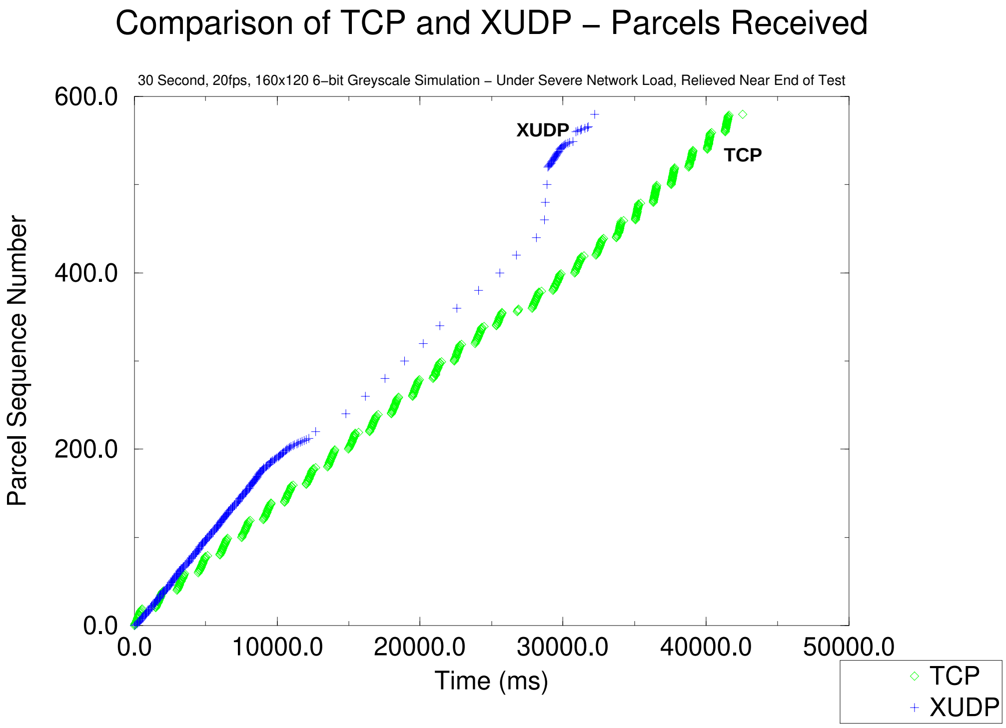

Figure 7: Comparison of TCP vs. XUDP Parcels Received (Tests 18:44 and 18:54 3/17/97)

Additionally, a variable bit and frame rate internetworked videoconferencing application that used XUDP as it's communication mechanism was developed as part of an undergraduate thesis at WPI[9]. Adding only a simple wrapper in the code to account for the possibility of discarded data through timed obsolescence, the communications handling routines in the application layer were simpler than what was required for an earlier TCP implementation. Performance was acceptable from the user's perspective, and network load was reduced slightly from the equivalent TCP-based version of the application.

XUDP is still under development. The current implementation is written in ANSI C for a UNIX environment and has been tested under Ultrix 4.5, OSF/1 V3.2 and Linux 2.0.28.

Acknowledgments

The authors wish to thank Andrew Thiessen for his work on developing a videoconferencing application designed around XUDP's functionality; and Brian Hazzard for his advice and knowledge of the workings of TCP/IP.

References

[1] Brakmo, L. S., O'Malley, S. W., and Peterson, L. L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. *Computer Communication Review 24*, 4 (Oct 1994), 24–35. ftp://ftp.cs.arizona.edu/xkernel/Papers/vegas.ps.

[2] Haskell, B. G., Puri, A., and Netravali, A. N. *Digital Video: An Introduction to MPEG-2.* Chapman and Hall, 1997.

[3] Hazzard, B. K. Evaluation of Optimal Socket Buffer Sizes for the Internet Protocol Suite (draft). Tech. rep., Worcester Polytechnic Institute, 1996.

[4] Jacobson, V., Braden, R. T., and Borman, D. A. RFC 1323: TCP Extensions for High Performance. ftp://ds.internic.net/rfc/rfc1323.txt, May 1992.

[5] Jacobson, V., and Karels, M. Congestion avoidance and control. In *SIGCOMM '88* (Palo Alto, CA, Aug 1988). ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[6] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. Rtp: A transport protocol for real-time applications. ftp://ds.internic.net/rfc/rfc1889.txt, January 1996.

[7] Stevens, W. R. *UNIX Network Programming.* Prentice-Hall, Englewood Cliffs, NJ, 1990. ISBN 0-13-949876-1.

[8] Strayer, W. T., Dempsey, B. J., and Weaver, A. C. *XTP: The Xpress Tranbsfer Protocol.* Addison-Wesley, 1992.

[9] Thiessen, A. P. *Real-Time Video Teleconferencing Over the Internet.* Major Qualifying Project Report, ECE Dept., Worcester Polytechnic Institute, 1997.