Aoxiang Xu and Jeremy Cooperstock,
Dept. of Electrical and Computer Engineering, McGill University, Montreal, Canada

Presented at
the 108th Convention
2000 February 19-22
Paris, France


AES

*This preprint has been reproduced from the author's advance manuscript, without editing, corrections or consideration by the Review Board. The AES takes no responsibility for the contents.*

*Additional preprints may be obtained by sending request and remittance to the Audio Engineering Society, 60 East 42nd St., New York, New York 10165-2520, USA.*

*All rights reserved. Reproduction of this preprint, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.*

AN AUDIO ENGINEERING SOCIETY PREPRINT

# Real-Time Streaming of Multichannel Audio Data over Internet

Aoxiang Xu and Jeremy R. Cooperstock
Centre for Intelligent Machines and
Department of Electrical and Computer Engineering
McGill University
Montreal, QC, H3A 2A7, Canada
+1-514-398-5992
{axu | jer}@cim.mcgill.ca

**Abstract**

On September 26, 1999, a musical performance, taking place at McGill University, was transmitted, in real time, to an audience at New York University, over the Internet. While Internet streaming audio technologies have been in use for several years, what made this event unique was that the audience in New York experienced high-fidelity, multichannel audio, without interruption. In order to achieve this result, a custom system was developed employing both TCP and UDP protocols, and providing its own buffering and retransmission algorithms. The motivation for this approach is explored, and experiments justifying the decisions made are explained.

## 1 Introduction

The growth of the Internet reshaped our work lives. In recent years, it has also begun to affect and redefine various areas of entertainment, in particular, that of music. Until the last decade, the idea of downloading music to one's home, at great convenience and essentially zero-cost, would have been unthinkable to many. Today, real-time Internet music streaming, such as RealAudio (www.real.com) and MP3 (www.mp3.com), is regularly enjoyed by millions of listeners.

The limiting factor in such transmissions has typically been the bandwidth of available networks. Therefore, Internet audio streaming technologies are invariably characterized by relatively low-bandwidth, low quality, stereo sound. Although this may satisfy the minimal requirements of casual audiences, it is by no means sufficient for a society of musicians and professional audio engineers [1], to whom high-fidelity, multichannel audio is a necessity. Fortunately, the recent emergence of advanced research networks provides an opportunity to explore the possibility of streaming high quality, high bandwidth, multichannel audio through the Internet, in real time.

This remainder of this paper describes our efforts in building a demonstration of such a system for the 108th Audio Engineering Society (AES) Convention.

# 2 The Demonstration

The demonstration, which took place on September 26, 1999, involved the transmission of an AC-3 audio signal accompanied by an MPEG-1 video stream. A live performance of the McGill University Swing Band, playing at McGill's Redpath Hall, was delivered to an audience at the Cantor Film Center of New York University, with a time delay of three seconds. At the Cantor Film Center, an NYU dancer performed with the music.

## 2.1 Hardware

Figure 1 illustrates the setup of the demonstration. The performance at McGill University was fed into a Sony PCM-800, which converted the six channels of analog input to PCM at 16 bits per channel, 48 kHz, which were in turn provided to a Dolby DP569 encoder via three AES/EBU streams. The resulting AC-3 data, encapsulated in an AES/EBU stream at 1.536 Mbps, was read by the *sender* program running on a Silicon Graphics Indy (R4600 with 64 Mbytes RAM). The sender was responsible for segmenting the audio data into discrete packets and transmitting them over the network, to another Indy R4600 with 150 Mbytes RAM), located at the Cantor Film Center. A *receiver* program, running on this machine, read the packets from the network, extracted the audio data, and provided it via the Indy's AES/EBU port, to a Dolby DP562 decoder, which decompressed the AC-3 data and delivered the audio to a playback system.

## 2.2 Audio and Video Format

The AC-3 audio data (encoded Dolby 5.1) was encapsulated in a stereo AES/EBU (48 kHz, 16 bit) stream provided by the Dolby Encoder. AC-3 carries five full-range channels and one subwoofer [2], but because of compression, does not require the full bandwidth of AES/EBU. While it is therefore possible to reduce the AC-3 bandwidth demands by discarding the padded zeros of the AES/EBU stream, we chose not to do so, as our research goal involves even higher bandwidth applications.

In addition to the audio stream, we utilized Cisco's IP/TV system for the encoding, transmission, and decoding of an MPEG-1 video. As IP/TV does not presently provide support for external synchronization, the alignment of audio and video streams was performed manually, prior to the demonstration. The MPEG-1 stream required approximately 1.5 Mbps, thus, the average bandwidth for the demo was in the order of 3.0 Mbps. However, as will be discussed below, additional bandwidth was required following a period of network congestion.

### 2.3 Network Route

The audio and video streams were transmitted over the high-performance networks managed by Canarie (Canada) and Internet2 (US) corporations. Although these networks have relatively high capacity, there were several bottlenecks between Montreal and New York that limited the bandwidth available to us (see Figure 2).

These bottlenecks may not appear significant, as our bandwidth demands were only a fraction of the available capacity. However, transient network congestion induced by other high bandwidth applications would prove fatal to a simple audio transmission protocol. We were challenged to ensure robustness, despite the lack of bandwidth guarantees and the unavailability of any bandwidth reservation protocols on the network. Just as on a public Internet, we had to compete with all the other users. The only concession made was for the first half of the demonstration, during which the network news feeds (Usenet) coming into both McGill and NYU were temporarily disabled.

## 3 The Audio Transmission System

The design of our transmission protocol was the most critical component of the system. Since the audio data was to be streamed and played back in real time, the protocol had to ensure delivery of data to the receiver with low latency, low error and loss rate, and most importantly, without breaks in continuity. It is this final point that ultimately determined the success of the demonstration, as any interruption of the output stream would cause an immediately noticeable break in the music.

Our design was based in part on the Adaptive File Distribution System (AFDP) [3], which provides efficient and reliable delivery of a file to multiple hosts on an Internet. The fundamental difference between AFDP and our application is that the former is not bound by real-time constraints. For a streaming audio application, we are willing to trade a small amount of reliability in exchange for real-time performance.

### 3.1 Network Protocols

TCP [4] and UDP [5] are the dominant network protocols used for computer communication. TCP, a connection-oriented, reliable, and full-duplex protocol, uses an acknowledgement and retransmission scheme to guarantee delivery of data. If a segment is not acknowledged by the receiver, the sender will continue to retransmit until the data is received successfully. Furthermore, TCP ensures an ordered reception of packets by delivering packet $p_n$ only after all previous packets, $p_i$, $i < n$, have been received.

While these mechanisms are well suited to applications such as *ftp*, *telnet*, and *http*, they are unsuitable for the transmission of real-time media. This unsuitability is related to the characteristics of Internet communication, typified by widely fluctuating transmission delays. For example, although the average round trip time (RTT) for data sent between McGill and NYU was in the order of 50 ms, it was not unusual to observe RTTs of several hundred milliseconds during periods of high network traffic. Such delays can cause TCP to activate its congestion avoidance mechanism [6][7] or lead to timeouts at the sender, substantially degrading performance [7][8].

From the perspective of someone listening to an audio broadcast, if one packet fails to arrive before its audio content is scheduled to be played, it is imperative that successive packets are not delayed unnecessarily, otherwise, a discernable break in the music will occur.

For these reasons, real-time applications tend to favor UDP, a connectionless, unreliable protocol with no flow control mechanism. However, UDP by itself is clearly insufficient, and requires additional layers to ensure efficient transport and robustness.

## 3.2 Program Overview

Figure 3 presents the structure of the sender and receiver programs and illustrates the interactions between them. There are three network sockets created at each host for communication between the two programs:

1. A TCP connection initiated by the receiver, used to request transmission of audio data from the sender.

2. A UDP channel, used for the transmission of most of the audio data.

3. A TCP connection initiated by the receiver, used for the exchange of control packets between the two programs and for the occasional retransmission of missing audio data, as will be explained below.

The sender process maintains an audio queue containing pointers to the data packets being transmitted. This process has two threads. One thread continuously reads the output of the Dolby encoder, stores this data in 128 packets per second, and adds these packets to the audio queue. The second thread is responsible for the transmission and retransmission of audio data to the receiver.

The receiver process consists of a single thread, which continuously checks the UDP and TCP sockets for incoming data and stores these received packets in an audio queue. At every iteration through the loop, a periodic timer is checked. Whenever the timer expires, audio data is dequeued and sent to the playback device and the remainder of the queue is checked for missing data. If necessary, a retransmission request is then issued to the sender.

All the data transmission by our system takes place in the form of user-defined data and control packets. Data packets consist of an identifying header and a payload, which carries audio data. The control packets are used primarily for retransmission requests. TCP is used to ensure delivery of the high-priority control packets, while data packets are usually transmitted by UDP.

### 3.3 Packet Recovery Mechanism

Since packet loss is inevitable, we require a lightweight recovery protocol in our program. This protocol must be reliable, subject to best-effort limitations, and fast, such that the receiver recovers the missing data in time for playback. Our ability to satisfy these demands is determined jointly by the network architecture, link capacity, competing traffic, and the algorithms employed. It is only this last variable over which we have any control.

As the stop-and-wait and go-back-N recovery mechanisms of TCP are inappropriate for real-time media, we employ a selective-repeat technique instead. Each data packet is labeled with a unique sequence number by the sender. The receiver detects missing data by periodically checking for a gap in the sequence numbers of received packets. When such a gap is found, the receiver issues a negative acknowledgement (NAK) to the sender as a retransmission request [3].

We illustrate the packet recovery protocol, by intentionally dropping 30 seconds of audio data in the middle of a transmission, as shown in Figure 4. At the onset of simulated packet loss, the receiving rate drops to zero. The receiver then issues a retransmission request and shortly thereafter, the retransmitted packets begin to arrive.

Since the packet loss in this case is not caused by network congestion, the simulation is, of course, completely artificial, and serves only to demonstrate the high-level operation of the program. During periods of network congestion, retransmission of lost data cannot be guaranteed to arrive in a timely manner. We address this point in the following section.

## 4 Protocol Issues

Consistent with our observations, it has been demonstrated previously that packet losses are typically bursty [9][10]. Under conditions of packet loss caused by network congestion, it is therefore likely that the congestion is short-lived, in which case, retransmitted data can arrive without difficulty. However, it is also possible that the retransmission will face similar or worse congestion than that which caused the original packet loss, especially if the resend occurs in close proximity to the initial transmission. By introducing a receive buffer that is longer than the expected burst duration, retransmission congestion problems can be reduced significantly.

recovery mechanism, TCP is preferable under modest network loads. This should come as no surprise, since TCP has been optimized for such environments.

However, when network congestion increases, TCP does not cope as well. In fairness, TCP was designed to be *well behaved* to the network under such conditions, whereas our needs are somewhat greedy. As can be seen in the figure, for our particular experimental configuration, UDP is better able to recover from network congestion brought on by competing traffic in excess of 30 Mbps. Although TCP will eventually deliver 100% of the packets, they do not arrive in time for playback. Thus, the effective recovery rate of TCP is significantly lower than our UDP protocol.

Interestingly, we note that despite the quantitative superiority of UDP (70% recovery vs. 17% for TCP) under severe congestion, there is little observable difference in terms of the audio quality. Once there is a gap in the playback, it is immaterial to the listener whether that gap lasts for 80 ms or 800 ms. In fact, it may be preferable to mute for several seconds rather than play a segment corrupted by missing packets. As discussed in Section 6, we are presently working on a variation of the protocol in which a small amount of loss will be unnoticeable.

### 4.3 TCP Tuning

The results above encourage us to employ TCP for recovery provided packet loss is reasonably low. Although TCP is not an ideal protocol for bulk data transmission, various modifications can be made to improve its throughput [11]. For example, the size of the sliding window advertised by the receiver limits the amount of data that the sender can transmit pending an explicit acknowledgement [12]. To compensate, we increase the TCP buffer size on both sender and receiver, thereby improving throughput [11][13].

The selective acknowledgement protocol, along with various other extensions [8][14][15], is designed to improve the performance of TCP in the event of multiple missing segments within a window. Unfortunately, these modifications cannot change the fact that TCP, over a lossy network, is limited by its congestion control algorithm. Thus, a TCP-based retransmission approach may be ill-advised unless the receiver employs a large delay between packet receipt and playback. For interactive music applications [1], however, minimal latency is imperative so TCP is inappropriate.

## 5 Results and Conclusions

Although considerable efforts went into the design and development of the system, the authors were surprised by how well it worked. Given the unreliability of network communications and the unavailability of bandwidth reservations, at least a few glitches were expected. Indeed, during development and tuning, various problems with the network configuration were noted, each of which could have proved fatal to the demonstration.

Following a cautious initial demonstration employing a twenty second buffer, the delay was reduced to a mere three seconds. When this failed to introduce interruptions to the audio stream, the network news feed into NYU was resumed, thereby generating considerable competing traffic at the receiver end. At this point, several interruptions to the Cisco IP/TV video stream (requiring similar bandwidth to our audio system and utilizing the same delay) were observed. However, at no point during the demonstration did the audio break from a continuous stream.

In terms of lessons learned, our experiments highlighted some of the strengths and weaknesses of TCP and UDP as network protocols for real-time transmissions. Clearly, each has its place, but neither, alone, is sufficient for applications of this nature. It is our hope that the work started here will motivate the development of improved protocols that are better suited for high-fidelity, high-bandwidth, and low-latency communications over the Internet.

## 6 Ongoing Work

We are currently extending our system for a more demanding task: the transport of six channels of uncompressed audio at 96kHz, 24bits/sample, requiring over 13 Mbps of bandwidth. The demonstration of this system is not only a challenge to the designers, but is expected to push the limits of the underlying networks to their capacity. As such, some amount of packet loss will be unrecoverable within the delay limits imposed by a real-time demonstration. Therefore, our protocol will be modified to transmit selective bit slices of the audio samples, dynamically, based on available bandwidth. For example, under periods of severe congestion, we will make best-effort delivery attempts on the most significant bits in order to provide at least a low-fidelity playback. As congestion decreases, we can increase the number of bits to be transmitted, thereby improving the audio quality.

In another vein, we are developing a loss recovery protocol on top of UDP, which provides many of the desirable features of TCP, in particular, flow control and reliability, while offering a less restrictive congestion avoidance mechanism. For example, to maximize performance, the slow start algorithm [12] will not be implemented. However, some congestion control is still required, in particular under severe packet loss, as best-effort (i.e. greedy) delivery over the Internet is highly problematic [16].

## Acknowledgements

The authors wish to extend their gratitude to all of the participants who helped specify, test, and tune the system, in particular, Wieslaw Woszczyk, Zack Settel, and Bruce Pennycook, who determined the requirements of the demonstration and supported its ongoing development, and Quan Nguyen, who provided constant assistance at the network and administration level. Thanks are also due to the many individuals who took part in the actual performance, to the network providers, and to Dolby Laboratories and Cisco Systems, each of which lent considerable equipment and offered countless hours of support to the demonstration.

## References

1. AES White Paper. Technology Report TC-NAS 98/1: Networking audio and music using Internet2 and next-generation Internet capabilities. http://www.aes.org.

2. ATSC A/52. Digital audio compression (AC-3) standard. United States Advanced Television Systems Committee. http://www.atsc.org/Standards/A52.

3. J. R. Cooperstock and S. Kotospoulos. "Why Use a Fishing Line When You Have a Net? An Adaptive Multicast Data Distribution Protocol." Proc. of USENIX '96. San Diego, 1996.

4. J. Postel. Transmission control protocol. RFC 793, USC/Information Sciences Institute, Sep. 1981.

5. J. Postel. User datagram protocol. RFC 768, USC/Information Sciences Institute, Aug. 1980.

6. V. Jacobson. Congestion avoidance and control. Proceedings of SIGCOMM'88, 1988.

7. M. Allman, V. Paxson, W. Stevens. TCP congestion control. RFC 2581, USC/Information Sciences Institute, April, 1999.

8. K. Fall, and S. Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. Computer Communication Review, V.26, No. 3, July 1996.

9. M. Borella, D. Swider, S. Uludag, G. Brewster. Internet packet loss: measurement and implications for end-to-end QoS. Proceedings of the 1998 ICPP Workshops on Architectural and OS Support for Multimedia Applications/Flexible Communication Systems/Wireless Networks and Mobile Computing, 1998.

10. V. Paxson. End-to-end Internet packet dynamics. IEEE/ACM Transactions on Networking, Vol.7, No.3, June, 1999.

11. R. Cohen, S. Ramanathan. TCP for high performance in hybrid fiber coaxial broad-band access networks. IEEE/ACM Transactions on Networking, Vol.6, No.1, Feb, 1998.

12. R. Stevens. TCP/IP illustrated. Vol. 1, Addison-Wesley, Reading, Mass., 1994.

13. J.C. Mogul. IP network performance, in Internet System Handbook, eds, D.C. Lynch and M.T.Rose, Addison-Wesley, Reading , Mass., 1993.

14. M. Mathis, J. Mahdavi, S. Floyd, A. Romanow. TCP selective acknowledgement options. RFC 2018, USC/Information Sciences Institute, Oct. 1996.

15. M. Mathis, J. Mahdavi. Forward acknowledgement: refining TCP congestion control. Proceedings of SIGCOMM 96, August 1996.

16. S. Floyd. Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking, Vol. 7, No. 4, Aug. 1999.

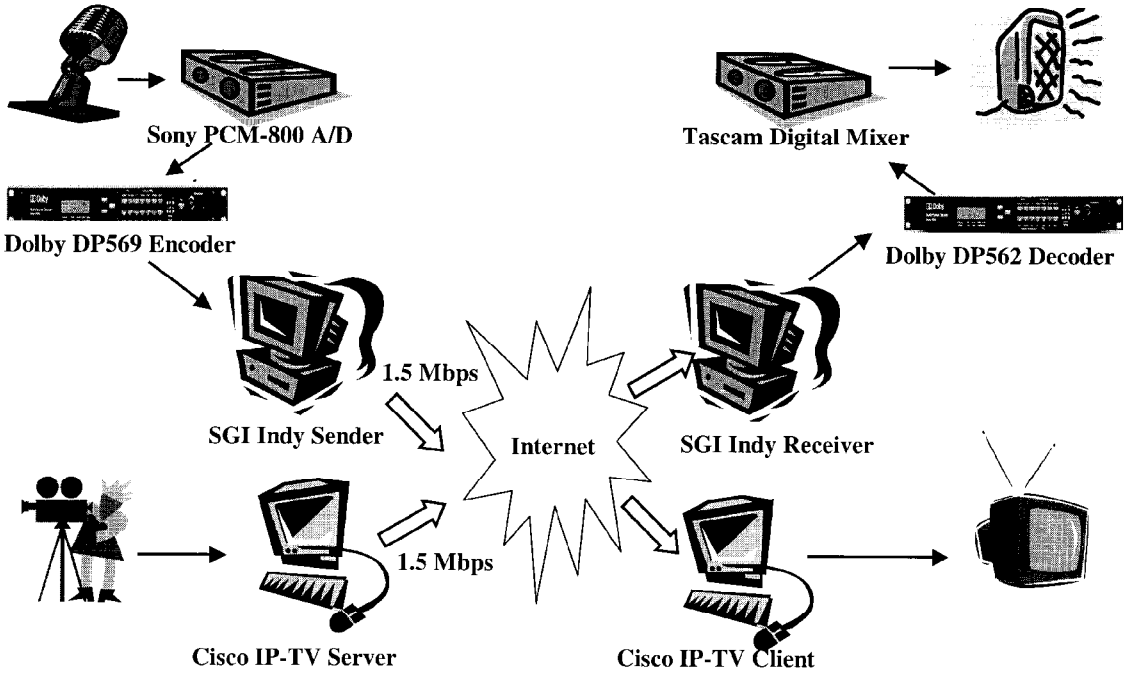


Figure 1. The hardware components involved in the demonstration.

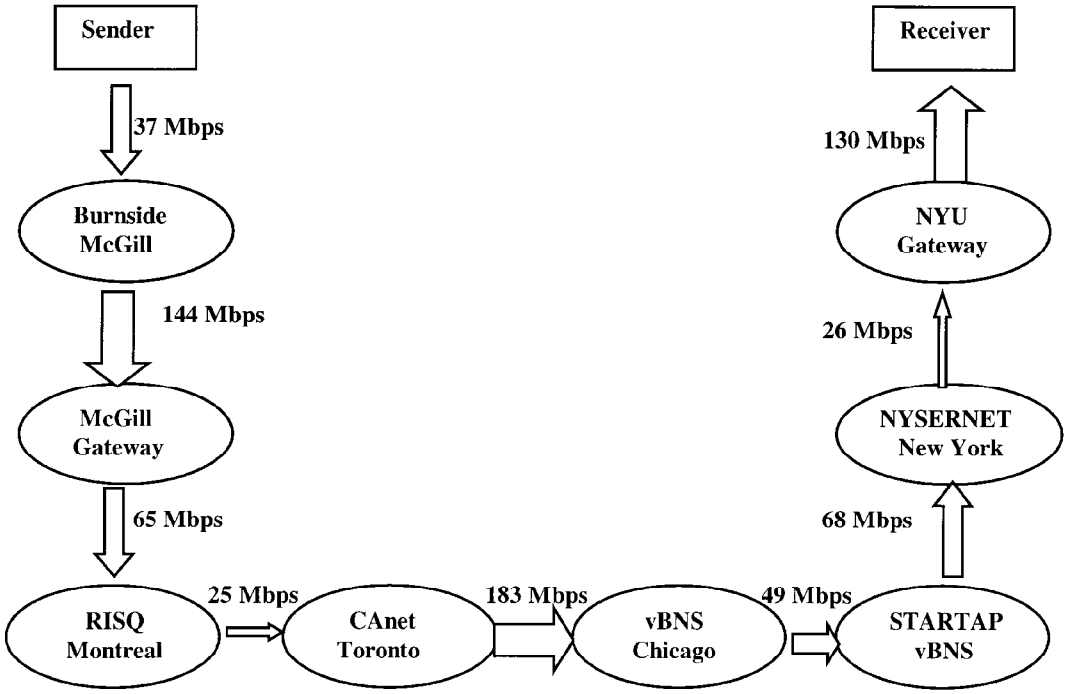


Figure 2. The network path between *sender* and *receiver* machines, illustrating the available capacity on each link, as measured by the *pathchar* utility (ftp.ee.lbl.gov/pathchar). Depending on the time of day and day of week, Internet traffic and the characteristics of the path can vary significantly.

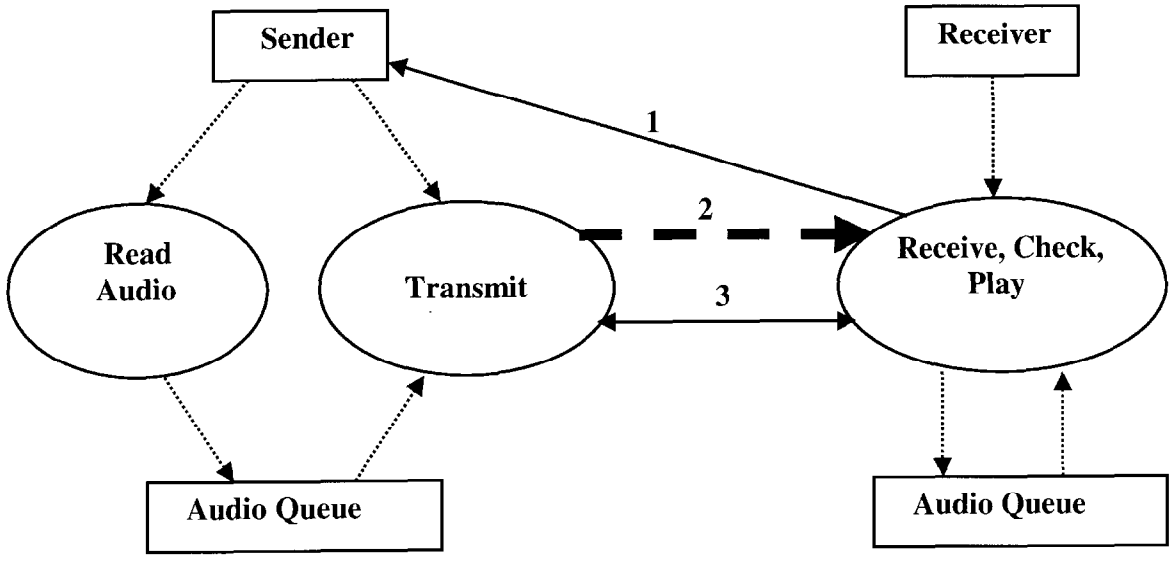


Figure 3. Communication Software Structure.

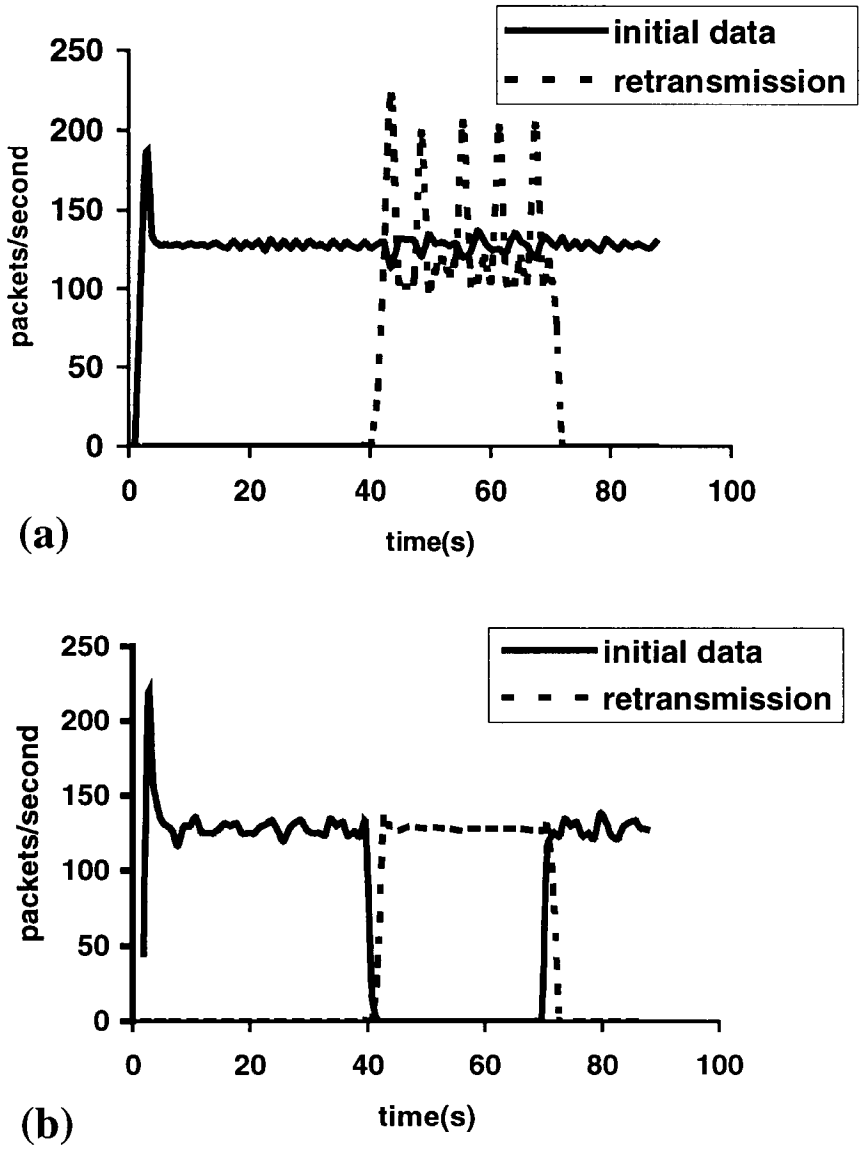


Figure 4. Demonstration of the loss recovery protocol, in which the receiver starts to drop packets at 40 seconds. (a) The sender begins transmitting packets normally, but in response to a resend request from the receiver, the retransmission protocol is activated at t=40s. (b) The receiver fails to receive any packets via the UDP socket so issues a retransmission request shortly after the onset of packet loss.

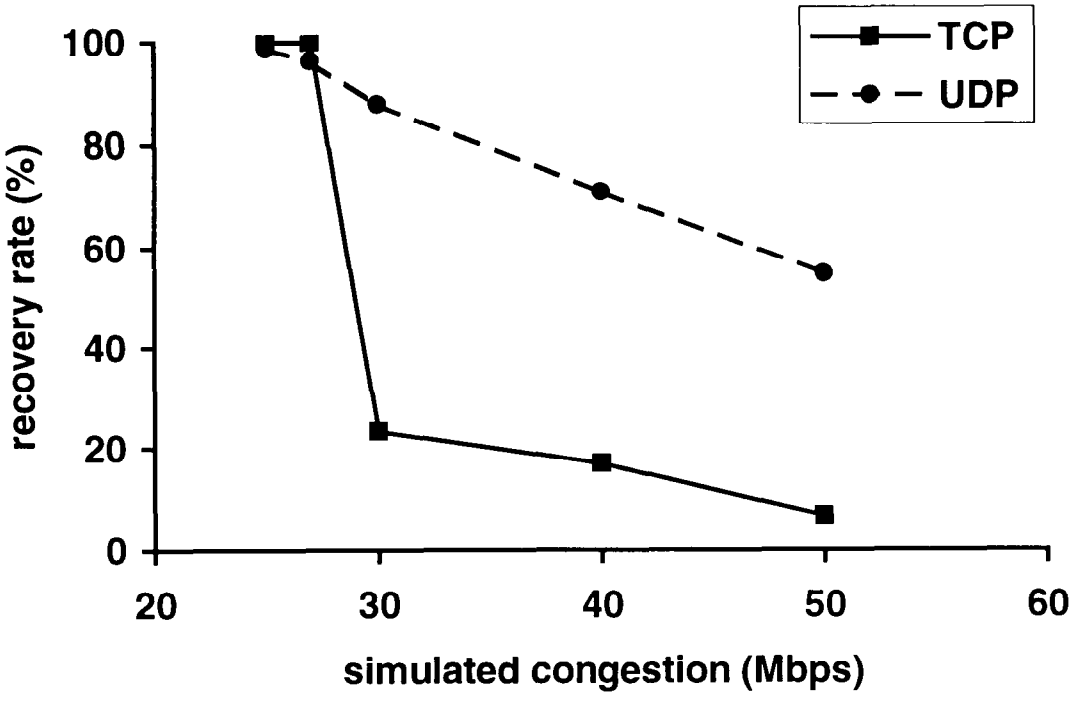


Figure 5. UDP- vs. TCP-based loss recovery protocols under network congestion. While competing throughput is under 30 Mbps, TCP is able to recover fully, but once congestion increases beyond this point, UDP is preferable for our application.