MPEG System Streams in Best-Effort Networks

Michael Hemy[1], Urs Hengartner[2], Peter Steenkiste[1], and Thomas Gross[1,2]

[1]Department of Computer Science
School of Computer Science
Carnegie Mellon

[2]Departement Informatik
ETH Zürich
CH 8092 Zürich

Abstract

The MPEG format is widely used and therefore an attractive vehicle for the distribution of video and audio material over the Internet. However, the hierarchical structure of MPEG systems complicates the task of delivering continuous, synchronized streams of video and audio in a best-effort environment (today's Internet). If the network throws away packets on encountering congestion, the video and audio stream may loose synchronization for a number of frames. Therefore, adapting the resource demands of an MPEG system must be done by an entity that is knowledgeable of the MPEG system structure: an MPEG system filter. This paper describes the integration of such an MPEG system filter into a network. Our experience to date indicates that mid-range PCs can host such a filter, and that the filter succeeds in adapting the resource requirements of an MPEG system in response to changes in the network load.

1 Introduction

Distribution of a movie via a network is a topic of great interest but also poses a number of challenges. Since the movie is played as it is received, the transfer over the network must proceed at a specific rate to prevent buffer overflow or underflow at the player. If there is competing traffic in the network, however, there is always the risk of *congestion*, and consequently, packets may be dropped or delayed.

A number of researchers have suggested the use of reservations of network resources to avoid congestion. Reservations, however, are only supported by some networks and often carry prohibitive costs or high overheads. The overhead of setting up a reservation may be tolerable if we play a full-length (i.e., 90 minutes) movie. However, the advent of multi-media indexing and retrieval systems means that we will increasingly see the distribution of many short video clips instead of a small number of large ones. A multimedia database like the core of the Informedia system[8] developed at Carnegie Mellon contains a large number of short movie segments – video clips, sound bites from TV news, movie story boards that provide access to individual scenes and shots, commercials, etc. (Today the size of the data collection is about 1 TeraByte.) To allow remote access to such a data collection, we investigate transmission of movies over existing best-effort networks that make up the Internet. The key problem is to ensure that the player receives a continuous feed in the presence of variable congestion. Our goal is to maximize the transfer of comprehensible information given the available (varying) bandwidth.

There are many formats to encode movies. We focus on the MPEG format – it is a widely used international standard [1] that has been adopted by the Informedia project[8]. MPEG is supported also by the Java Multimedia Framework (JMF)[9], so we anticipate that MPEG will continue to grow in importance for Internet applications (e.g., JMF makes it easy to include MPEG movies in applets or HTML pages). However,

Effort sponsored by the Advanced Research Projects Agency and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-96-1-0287. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

the characteristics of MPEG systems provide various challenges when streaming MPEG over a best-effort network where random packet losses can result in a significant loss in quality. In this paper, we present a solution based on a filter that understands the MPEG format and that can adapt the bandwidth requirements of the video stream based on feedback about the network conditions while maintaining the MPEG system format. While we focus on one particular video format, the basic concepts presented here can also be applied to other standards since the key features of MPEG (e.g., presence of meta data for synchronization, inter-frame encoding) are likely to remain widely used.

In the next two sections, we briefly review the structure of MPEG systems and the challenges it poses for streaming over best-effort networks, and we provide an overview of related work. Then we present our architecture, describe the implementation, and present a preliminary performance evaluation.

2 Background

We give some background on MPEG-1 and describe the challenges associated with streaming MPEG-1 over best-effort networks.

2.1 MPEG-1

MPEG-1 was primarily designed for storing video data (moving pictures) and its associated audio data on digital storage media. As such the MPEG standard deals with two aspects of encoding: compression and synchronization. MPEG specifies an algorithm for compressing video pictures (ISO-11172-2) and audio (ISO-11172-3) and then provides the facility to synchronize multiple audio and multiple video bitstreams (ISO-11172-1) in an *MPEG system*. MPEG-1 is intended for intermediate data rates on the order of 1.5 Mbit/sec.

2.1.1 MPEG-1 Video

An MPEG video stream distinguishes between I-pictures, P-pictures, and B-pictures – these pictures differ in the coding scheme. The three types of coding provide three levels of compression by exploiting similarities within the picture or similarities to neighboring pictures.

2.1.2 MPEG-1 Audio

An MPEG-1 audio stream consists of audio coded using one of three algorithms, which offer different levels of complexity and subjective quality. These algorithms are referred as 'layers' in the coding standard. The coding algorithms use psycho-acoustic properties of the human hearing to compress the data (lossy compression).

2.1.3 MPEG-1 Systems

The MPEG system layer is responsible for combining one or more compressed audio and video bitstreams into a single bitstream. This is done by interleaving data from the video and audio streams, combined with meta data that provides the timing control and synchronization.

2.2 Best-effort networks

The infrastructure provided by today's IP-based networks provides access to a large number of nodes. Unfortunately, IPv4 (the current standard) provides no framework for resource reservation. Users are competing for bandwidth, and if a link becomes congested (demand for bandwidth is higher than the link capacity), packets are dropped. Since traffic conditions change continuously, congestion can start and disappear at any time. Note that in the current Internet, there is an assumption that it is the source's responsibility to reduce the data send rate when packet losses are observed to reduce congestion. For most applications, this reduction is done by TCP, the dominant Internet transport protocol, but if an applications takes control of managing the send rate, it should also abide by this rule.

Random packet loss can hurt MPEG-1 systems in two ways, besides the obvious fact that the information in the packet is lost. When we analyze random packet losses, we must take into account that network packets may not correspond to MPEG packets and that the latter are a layer completely separate from the video frames. The amount of impact that the loss of a particular packet will have depends on its location in the stream and on the robustness of the player in recovering from errors. In the worst case, we may loose a network packet that contains meta data of the whole MPEG stream (the MPEG system header), and players that rely solely on synchronization information found in the stream will significantly be impacted when such information is lost. In a typical scenario, it is most likely that a packet lost will contain some part of a video frame with meta data (video being the predominant stream).

In the context of the MPEG layers, a network loss translates into a disruption in the system pack layer and may result in concatenating parts from two different MPEG packets. This loss can induce corruption in the lower layers, e.g., corruption of the video or audio data. If video data has been affected, the frame is decoded incorrectly. An incorrect I-frame or P-frame propagates problems to all dependent frames and corrupts these as well. In the worst case, we may loose a whole group of pictures (GOP), typically equivalent to half a second of video. For various MPEG streams, our experiments have shown that a random loss of 1% of network packets can translate into as high as 10% damaged video frames. Similarly, Boyce et al. [4] noticed that packet loss rates as low as 3% translated into frame error rates as high as 30%.

3 Related work

There are several issues that must be dealt with when we adjust multimedia data so that they can be transmitted over a best-effort network, and these issues have been addressed in different ways by other researchers. Common to all approaches is the existence of a *filter* that removes data as needed. We now review some related work classified according to the following criteria: (i) location of the filter, (ii) type of filtering applied, (iii) error recovery scheme, and (iv) adaptation algorithm.

Location of filter A filter can be placed either in the network or in the end-system. Bhattacharjee et al. [3], Yeadon et al. [15], and Amir et al. [2] present filters for video data which are located in the network. Since different applications require different filtering strategies, such network nodes need some knowledge about the type of data being filtered. Also, strategies that allow a client to find out about the location of these nodes have to be developed. RTP [13] proposes "mixers" and "translaters", which are placed in the network. The former ones mix streams and perform conversion between encoding formats, the latter ones translate across transport protocols (e.g., tunneling of a multicast stream into several unicast streams). Berkeley's Continuous Media Player [11], OGI's distributed video player [5], and the Vosaic player [7] use the end-system to filter video data: Frames can be dropped either at the sender (in case of a shortage of network resources) or at the receiver (in case timely display is impossible).

Type of filtering There are several ways to filter a video stream and to reduce its bandwidth: Frame-dropping [3] [15][7][11][5], low-pass filtering [15], color reduction [15], re-quantization [15][6], and transcoding [15][2] Another approach is hierarchical filtering: The layering coding scheme for MPEG presented by Li et al. [10] multicasts three video streams. Each receiver subscribes to the base stream consisting of the I-frames. Depending on its capabilities, a receiver can additionally subscribe to the stream transmitting the P-frames or even to the stream containing the B- frames.

Error recovery Lost data packets can be ignored, retransmitted, or recovered by a Forward Error Correction (FEC) scheme. The filter by Yeadon et al. [15] and OGI's player [5] ignore lost packets. The Continuous Media Player [11] pursues the second strategy by employing Cyclic UDP [14], which retransmits lost high priority data (i.e., I-frames in the case of MPEG video) to give them a better chance to reach the destination. In the Vosaic player [7] and in Columbia's VoD testbed [6], a client can demand retransmission of a lost frame. FEC is applied by Nonnenmacher et al. [12], where requests for retransmissions are handled by FEC transmissions.

Adaptation algorithms The Vosaic player [7] continually measures the rate of frames dropped by the receiver due to missing CPU power. If this rate exceeds 15% or falls below 5%, the server is instructed to lower, respectively to increase, the frame rate. To cope with network congestion, the rate of frames dropped by the network is also measured and fed back to the server every 30 frames.

Li et al.'s player [10] also uses two thresholds to decide whether a client subscribes to an additional multicast layer or whether it drops one of them. The decision is made after receiving a GOP. In addition to the packet loss ratio, the number of late frames is also taken into account.

In Columbia's VoD testbed [6], the occupancy of the sender buffer is measured over five or ten seconds intervals. In this way, momentary fluctuations due to the varying sizes of the different MPEG frame types can be overcome. The current occupancy is compared to the occupancy from the previous measurement. If necessary, the bitrate of the movie is adapted. Special care is taken to achieve convergence and to avoid oscillations around the desired rate.

In OGI's player [5], every component (server, network, client) can drop frames in case of missing ressources. Additionally, the (filtered) display frame rate at the receiver is compared to the sending frame rate. If the difference is large, the sending frame rate is (linearly) decreased. In case of a small difference, the rate is (linearly) increased.

All of the projects described so far deal either only with video data or they transmit video and audio in two separate streams, thus requiring additional synchronization information for their playback at the receiver. Transmitting video and audio data in one stream, as supported by the concept of MPEG systems, has not yet been reported; this approach is at the core of the system that is described in this paper.

4 Architecture

In a typical video streaming application, there are two elementary components: the client requesting the video and the server providing it. Typically, the server responds to requests from multiple clients. Our goal is providing the client with the best possible video stream under the current network conditions.

4.1 Overview

Since the video server may be too busy to handle the computation required to adapt the MPEG system to network conditions, we place the adaptation (filtering of data to meet a given resource bound) on other nodes in the network, as depicted in Figure 1.

Intermediate nodes labelled 'R' designate routers that have no knowledge of the application data and that may randomly throw packets away. The 'Filter' nodes implement filtering by watching a known port, intercepting the client's requests and adapting the stream sent by the server accordingly. This solution allows us to place the filter on the nodes that connect to network bottlenecks, and there can be multiple filters along the path from a server to a client.

The filter responsibilities can be summarized as follows:

1. receive video stream from server or previous filter;

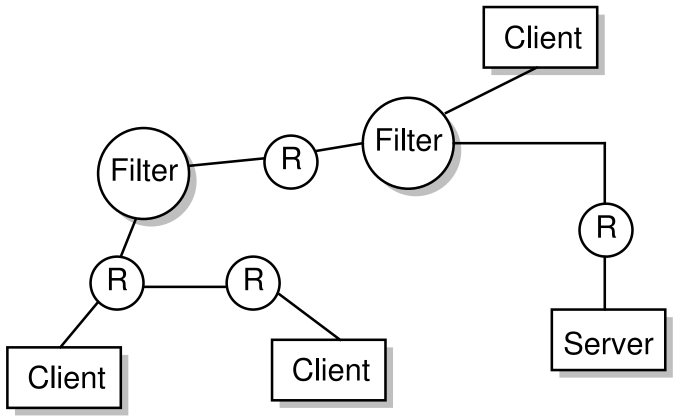

Figure 1: Filters in the network.

2. send video to client or next filter;
3. receive requests from client or next filter;
4. act upon requests or forward them to previous filter.

4.2 Server-client interaction

Requests for changes in the quality of the transmitted movie must originate at the client side. Ultimately the person(s) watching the video determine(s) the quality of the video, and a user (viewer) could be given the capability to interact with the client (player), i.e., requests to increase or reduce the stream are made by the user. Unfortunately, perception-based models are computationally expensive. Since the client is already busy with decoding and rendering an MPEG stream, we need a non-intrusive method that is also computationally cheap to determine when and how the MPEG stream should be filtered. We choose to monitor the network traffic while receiving the network packets and devised an algorithm that adjusts the degree of filtering to the current packet loss rate. This algorithm is described in Section 5.3.4.

4.3 Filters

When modifying an MPEG system, an important requirement is that the modifications are done on the fly. Encoding MPEG systems requires typically two passes. We need the filter to be able to remove a selection of video frames, such that the reduced stream is still an MPEG stream, without fully decoding and re-encoding the incoming stream. The algorithms we devised are described in Section 5.2.2. The filtering of each level of the MPEG system is handled by a different algorithm, to decouple the frame-dropping mechanisms as much as possible from the module that selects the policy for reduction or increase of the data rate. The client sends, as mentioned in Section 4.2, to the filter a request to either increase or decrease the bandwidth, and the filter picks the appropriate level of filtering based on the current level of filtering and the direction of the change request (increase or decrease of data).

Since the filter must analyze the MPEG stream, we make the filter responsible for pacing the packets according to the required movie bandwidth.

To correctly analyze and filter the MPEG system, the filter needs a correct stream at its input. While the filter should be able to recover from errors since it can be cascadable to other filters, having the first filter analyze an uncorrupted MPEG stream improves reliability. Hence the connection between the server and the first filter must be reliable. To satisfy this condition, it may be necessary to place a filter in proximity to the server. The actual placement, however, will depend on the network typical traffic.

Figure 2: Channels of communication in the network.

There are three main components of our architecture: the client, the server, and at least one instance of a filter, and we maintain two channels between them, one for control and the other for data, as indicated by Figure 2.

The same channels exist between any pair of filters. The control channel is bidirectional, realized by TCP connections. The requirement that filters are chainable fits well into the current WWW browser environment that uses Java. Java applets are restricted to talk only to the host from where the applet originates. Our setup allows the client to communicate with only one host (the last filter), which will either forward the requests or act upon them.

5 Implementation

We now describe the two major parts of our implementation. The filter reshapes the MPEG system and is described in Section 5.2. The overall protocol is described in Section 5.3.

5.1 Client

The client is based on the Java Media Framework (JMF), a package that supports the replay of audio and video streams in a browser. JMF supports a wide range of video and audio formats, including MPEG-1 systems, and promises to be a widely used package to display multimedia material.

JMF consists of two main components, a player and a data source. The player is responsible for replaying the audio and/or video stream and it is typically optimized for a particular platform, e.g., it may use optimized native methods to deal with specific devices. The source is responsible for retrieving the data. Data sources exist that retrieve data from disk, or retrieve data over the network using a variety of protocols. The transport protocol described in Section 5.3 is implemented as a new data source. The system we describe has been used with JMF implementations from Intel and Sun.

5.2 Filter

The filter needs to reduce the bandwidth required by the MPEG stream. Since MPEG is already a compressed representation of a movie (video + audio), reducing the data directly impacts the quality. Since most of the data in the MPEG stream is used to encode video, our goal is to develop a filter that can reduce the video from an MPEG system while maintaining the audio and synchronization information. Additional requirements are that the filter formats the streams as valid MPEG-1 systems so we can use a standard player, that the video stream remains smooth, and that multiple levels of reduction are possible so we can better adapt to the available bandwidth. In this section, we describe a software filter that meets these goals and is efficient enough that it can be executed on a PC.

5.2.1 Filter operation

The idea behind the filter is to *partially* decode the MPEG system so we can identify the different video frame types, the audio information, and the synchronization information. We then drop some of the video frames, as described in the next section, and we reassemble a new MPEG stream, making sure that the audio and video are appropriately synchronized. Data reduction is done on the fly: the filter decodes data as it arrives and newly encoded data is immediately forwarded to the receiver.

The partial decoding of the incoming stream is based on a state machine. The states are obtained by identifying the MPEG unique start codes. The state machine maintains state across all the MPEG layers. This fact is important since the video sequence is broken into MPEG packets without considering frame boundaries. This state machine tracks all the audio and video streams simultaneously. When a frame of a particular video stream is detected, the filter checks whether it should be dropped or forwarded. Then the filter searches for the next start code, which is either in a higher layer (GOP, sequence header, packet or pack) or in the same layer (frame). With this method, there can be empty MPEG packets, and even empty GOPs, but keeping these empty packets provides the important benefit that synchronization is maintained, and the client can continue to decode and render correctly.

5.2.2 Selecting the frames to drop

The next question is what frames should be dropped. While each displayed frame gets the same amount of playing time and is thus equally important, there is a big difference in both the size and information contents of each of the encoded frames. These differences are a consequence of the inter-frame encoding. I-pictures are the largest and the "richest", while B-pictures are the smallest and contain the least information. While the relative sizes suggest that we should drop I-pictures first, the information contents and inter-frame dependencies make this of course impractical. The filter must first drop B-pictures. If that action is not sufficient, the filter drops P-pictures as well, and in the worst case, it will drop all B- and P-pictures as well as some I-pictures. Note that the first levels of reduction do not reduce the data bit rate significantly since B-pictures contain the least amount of physical data. E.g., if we look at a video sequence that has 15 frames in a GOP

$$IBBPBBPBBPBBPBB$$

then removing the B-frames leaves us with sequence

$$IPPPP$$

This transformation reduces the frame rate by a factor of three (66%), but typically removes only 15%-25% of the data.

In addition to the frame rate, the filter must consider the smoothness of the video stream, i.e., we want to prevent "stop and go" jerkiness. This goal can be achieved by distributing the dropped frames as evenly as possible, although the precise placement of the different frame types restricts what frames can be dropped at each reduction level. E.g., if the current reduced video sequence is IPPPP, the only additional frame that can be dropped without affecting others is the last P-frame.

The resulting algorithms are as follows: Based on the information in the beginning of the MPEG stream, the filter can deduce the pattern of I, P, and B frames. The drop patterns for different levels are evaluated to conform with the dependencies and to provide smoothness. Since B-frames do not have dependencies, the first algorithm removes the middle B-frame from each group of contiguous B-frames. The second algorithm removes two B-frames by selecting equally spaced frames. The third algorithm removes all the B-frames. E.g., in a GOP consisting of the following sequence:

$$IB(B)BPB(B)BPB(B)B$$

the frames in parenthesis represent the ones dropped for the first level. The same GOP in the second level will result in the following frames dropped:

$$I(B)B(B)P(B)B(B)P(B)B(B)$$

When all B-frames are being dropped, the bandwidth is further reduced by dropping P-frames. The next P-frame to be dropped is always the last one before the next I-frame. When all B- and P-frames are gone, we start dropping I-frames. At this point, the quality of the video typically degrades significantly. If we remove one of two I-frames (above all B- and P-frames), we remain with a video rate of 1 frame per second. The algorithms can remove further I-frames, distancing them apart as much as needed to achieve the required bandwidth. The resulting stream looks like a slideshow at this point. The maximum reduction will leave an audio-only stream. (In practice, we found out that most players are not able to play an MPEG system stream that is defined to have both audio and video when the video stream is competely removed.)

After removing frames, the network packets are built in a way to minimize the impact of network packet loss for the client. We require that each MPEG packet starts at the beginning of a network packet.

5.3 Transport Protocol

We developed a datagram protocol to distribute MPEG systems over an unreliable network. The filter receives feedback from the client and adapts its frame drop rate to the network condition between the filter and the client.

5.3.1 Transmission mechanism

We maintain two open connections between the client and the filter, as well as between the filter and server. The control connection is used exchange control information and it is always based on TCP. The data connection is used to transfer the MPEG data. Data is transferred between the filter and the client using UDP, as we described above. For the server-filter connection, we use TCP. The reason is that we expect the filter to be placed before the bottleneck, so there should be sufficient bandwidth available between the server and filter. Packet loss and timeouts should therefore not be a concern, and occasional packet loss can be handled without a problem by having enough buffered data. Moreover, this setup allows us to focus on the filter-client protocol.

5.3.2 Data streaming

At the filter, large MPEG systems packets generated by the filtering process are broken into smaller packets that fit into a network packet. Each packet gets a sequence number. Additionally, the current drop rate of the filter is also included in the packet header. This information is required by the adaptation algorithm.

The filter sends each packet at exactly the same time as the corresponding data packet from the non-filtered movie would have been sent. This rate is determined by the MPEG bitrate of the movie[1]. In our experiments, only movies with constant bitrates have been considered.

5.3.3 Control protocol

The control channels are used for a variety of purposes. First, they are used by the client to request video clips and other information. As explained above, the browser-based client can have only a single open connection so all requests are sent over the same channel. Requests for the server are forwarded by intermediate nodes. Second, the control channel is used to carry the feedback that is used in the adaptation algorithm from the client to the filter. Finally, the JMF player does not play movies that have losses in the first few Kbytes. For this reason, the filter transmits the first couple of GOPs (typically 24 or 30 frames) to the client over the control channel.

The control packets contain an opcode that identifies the request or information type, plus parameters. Note that there are clearly other ways of managing the control channel. Some of the control information could for example be sent using RTP [13].

[1] We assume that the filter and the client clocks are well synchronized.

5.3.4 Adaptation algorithm

The receiver can change the bandwidth of the MPEG stream by sending requests to the filter to increase or decrease the frame drop rate. To decide whether the filter should increase or decrease the bandwidth, the receiver continuously measures the current packet loss rate using a sliding window of length S packets. S is typically set to 500 in our experiments. If the packet loss rate is higher than a threshold α, the client will ask the filter to increase the frame drop rate. The value of α should be such that a packet loss rate of α still results in acceptable video quality. A second threshold β ($\beta < \alpha$) is used to determine when the frame drop rate should be reduced: if the packet drop rate is less than β, the receiver will ask the filter to lower the frame drop rate. The reason for using two thresholds is to allow the BW recovery to be less aggresive. This way the protocol id more friendly to competing traffic. Note that the waiting time before the client finally issues a request to reduce the frame drop rate is longer than the waiting before issuing a request to increase it. This behavior is somewhat similar to TCP's congestion control, that is, it reduces bandwidth more aggressively than it increases it. In our experiments, we typically set α to 5% and β to 1%.

After every request to increase or decrease the frame drop rate, the client temporarily suspends measuring the packet loss rate until it is notified by the filter that the drop rate change took place. This depends on the number of packets 'en-route' and the frame being processed by the filter when it receives the request to change the drop rate. To inform the client about the filter response, the header of a data packet includes the active drop rate.

6 Status and evaluation

At this point in time, we have experimented only with setups that include a single filter (however, the design and protocols support the chained operation described earlier). The filter and server are connected by a non-congested network.

Figure 3 depicts how the filter adapts to network congestion for setup in a testbed. The server is a is 400 MHz Pentium II machine (with 264 MBbyte), the client a 200 MHz Pentium Pro (with 64 MByte); the filter executes on the server machine. These two systems are connected via a link that can be transparently "loaded" by a traffic generator to cause congestion for the server-client connection. Until receiving packet # 9084, no packets have been lost. During the receipt of packets 9084 .. 9214, 4.94% of the bytes have been lost. As can be seen by the overlay figure, which depicts the response of the filter (in dropping packets), the filter responds by successively removing more frames. Eventually the removal of frames is too aggressive and the filter responds by removing fewer frames, until it is possible to return to the "no-removal" mode.

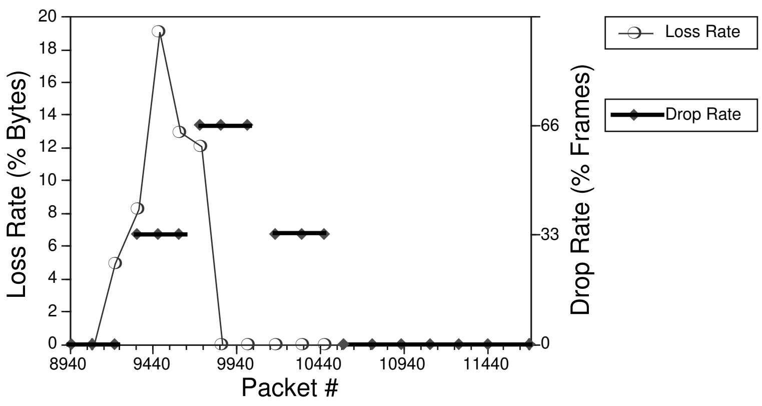

Figure 3: Adaptation in response to network losses.

Figure 4 shows the effectiveness of the filter in a real-life setup. The server and client have the same properties as described above. The two machines are connected via a network with three segments: a local area network connecting to a T1 line, the T1 line to the office of the local telephone company, and a synchronous DSL line from the phone company to a residence. The DSL line is the critical link, its maximum UDP throughput was measured to be 649 Mbit/sec. About 95% of this maximum bandwidth was available for the connection between filter (server) and client. For this experiment, several times an MPEG-1 movie with a bandwidth requirement of 1.07 Mbit/sec is transmitted from server to client. Figure 4 depicts the average transmission rate (measured at sender) and the receive rate (measured at the client). With adaptation, the sender stays close to the maximum bandwidth of the critical link. Without adaptation, the sender rate is unconstrained (the sender is connected to a local area network), but a large portion of the data is dropped along the way to the client. The randomly dropped packets without adaptation damage

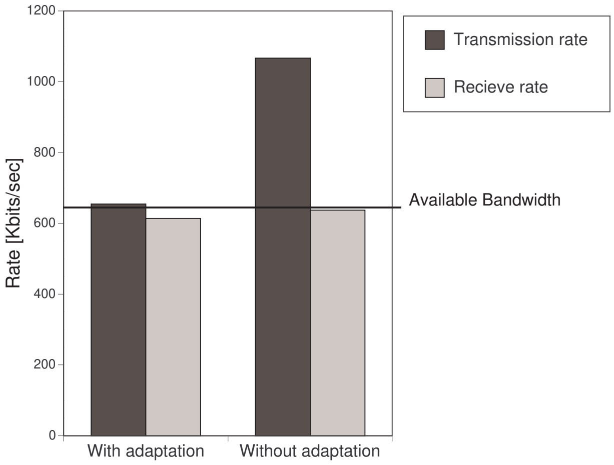

Figure 4: Effect of adaptation on network load.

more frames than are suppressed by adaptation resulting in a very low quality movie. Figure 4 shows that the receive rate is almost similar in both cases. However the filtered data received with adaptation contains much more usable information, where usable information is measured in terms of the amount of good frames that can be decoded at the client.

7 Concluding remarks

The MPEG system format presents a number of challenges to a system that attempts to deliver MPEG streams over a best effort network. However, the complexity of the MPEG system exists for good reasons (to meet the requirements of many users of digital video) and given the amount of data available in this format, supporting the delivery of movies or newsclips in the MPEG system format is attractive.

One of the serious problems that must be addressed in a best-effort network is how to deal with congestion. The network layer is unable to understand the intricacies of the hierarchical MPEG system format, so we decided to implement an MPEG filter that transforms the MPEG streams to meet resource constraints. Such a filter processes a set of synchronized video and audio streams and removes frames as is necessary to deal with the network conditions. A careful design of the filter avoids expensive operations in the filter host and a current mid-range PC suffices as a host. Such a filter is well able to adjust the resource demands in a timely manner, and the overall success of this architecture has encouraged us to begin a limited wide-area deployment.

Although we have an operational system, many more tasks remain to be done. We are currently investigating the behavior in a inter-continental setting since past studies indicated that the trans-atlantic links are more congested that the intra-continental links. Furthermore, a detailed analysis of the response behavior of the filter is needed.

References

[1] ISO/IEC JTC 1/ SC 29/ N 071. *Coding of moving pictures and associated audio - for digital storage media at up to about 1.5 Mbits/s - Part 1: Systems, Part 2: video*, 1992. CD 11172.

[2] E. Amir, S. McCanne, and R. Katz. An Active Service Framework and its Application to Real-time Multimedia Transcoding. In *Proceedings of ACM SIGCOMM '98*, pages 178–189, Vancouver, Canada, September 1998.

[3] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura. On Active Networking and Congestion. Technical Report GIT-CC-96/02, Georgia Institute of Technology, 1996.

[4] J.M. Boyce and R.D. Gaglianello. Packet loss effects on MPEG video sent over the public Internet. In *Proceedings of ACM MULTIMEDIA '98*, pages 181–190, Bristol, England, Sept 1998.

[5] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A Distributed Real-Time MPEG Video Audio Player. In *Proceedings of NOSSDAV'95*, pages 18–21, Durham, New Hampshire, April 1995.

[6] S.-F. Chang, D. Eleftheriadis, D. Anastassiou, S. Jacobs, H. Kalva, and J. Zamora. Columbia's VOD and Multimedia Research Testbed With Heterogeneous Network Support. *Journal on Multimedia Tools and Applications*, 5(2):171–184, Sept 1997.

[7] Z. Chen, S.-M. Tan, R.H. Campbell, and Y. Li. Real Time Video and Audio in the World Wide Web. In *Proceedings of Fourth International World Wide Web Conference*, Boston, Massachusetts, Dec 1995.

[8] M. Christel, T. Kanade, M. Mauldin, R. Reddy, M. Sirbu, S. Stevens, and H. Wactlar. Informedia digital video library. *Comm. ACM*, 38(4):57–58, April 1995.

[9] Java Multimedia Framework.

[10] X. Li, S. Paul, P. Pancha, and M. Ammar. Layered Video Multicast with Retransmission (LVMR): Evaluation of Error Recovery Schemes. In *Proceedings of NOSSDAV'97*, St. Louis, Missouri, May 1997.

[11] K. Mayer-Patel and L.A. Rowe. Design and Performance of the Berkeley Continuous Media Toolkit. In *SPIE Proceedings Vol. 3020*, pages 194–2006, San Jose, California, Feb 1997.

[12] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. In *Proceedings of ACM SIGCOMM '97*, pages 298–300, Cannes, France, September 1997.

[13] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP: A Transport Protocol for Real-Time Applications, January 1996.

[14] B.C. Smith. *Implementation Techniques for Continuous Media Systems and Applications*. PhD thesis, University of California at Berkeley, 1994.

[15] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd. Filters: QoS Support Mechanisms for Multipeer Communications. *IEEE Journal on Selected Areas in Communications*, 14(7):1245–1262, Sept 1996.