Architectural Considerations for Playback of Quality Adaptive Video over the Internet

Reza Rejaie*, Mark Handley, Deborah Estrin
University of Southern California
Information Sciences Institute
Marina Del Rey, CA 90292
reza,mjh,estrin@isi.edu
November 3, 1998

Abstract

Lack of QoS support in the Internet has not prevented rapid growth of realtime streaming applications. Many such applications play back stored audio and video over the network. However most of these applications do not perform effective congestion control, and there is significant concern about the effects on co-existing well-behaved traffic and the potential for congestion collapse. In addition, most such applications are unable to perform quality adaptation on-the-fly as available bandwidth changes during a session, and so do not make effective use of additional bandwidth when it is available.

This paper aims to provide some architectural insights on the design of video playback applications in the Internet. We identify end-to-end congestion control, quality adaptation and error control as the three major building blocks for Internet video playback applications. We discuss the design space for each of these components, and within that space, present an end-to-end architecture suited for playback of layered-encoded stored video streams.

Our architecture reconciles congestion control and quality adaptation which occur on different timescales. It exhibits a TCP-friendly behavior by adopting the RAP protocol for end-to-end congestion control[RHE99]. Additionally, it uses a layered framework for quality adaptation with selective retransmission to maximize the quality of the delivered stream as available bandwidth changes. We argue that the architecture can be generalized by replacing the suggested mechanism for each component by another from the same design space as long as all components remain compatible. Quality adaptation is the most challenging component of the architecture that has not been sufficiently studied to date, and so we sketch some of the design issues and introduce one proposed solution. Results of a simulation-based evaluation of these mechanisms are presented.

Keywords: Architecture, Video Playback, Layered Transmission, Quality Adaptation, Internet

1 Introduction

The Internet has recently been experiencing explosive growth of the use of audio and video streaming. Although some streaming applications such as conferencing tools[vic, nv, lvs] have a synchronized multi-user nature, the majority of current applications in the Internet involve web-based audio and video playback [Real, Vxt, MSft] where a stored video is streamed from the server to a client upon request.

The increasing trend in deployment of streaming applications over the Internet is expected to continue, and such semi-realtime traffic will form a higher portion of the Internet load. The overall behavior of these applications is likely to have a large impact on Internet traffic.

Since the Internet is a shared environment and does not micro-manage utilization of its resources, end systems are expected to be cooperative by reacting to congestion properly and promptly[FF98, LLSZ96]. As a result overall utilization of the network remains high while each flow obtains a fair share of resources. Unfortunately, many of the current commercial streaming applications do not behave in a network-friendly fashion. This is mainly because stored video has an intrinsic transmission rate. These rate-based applications either transmit data with a near-constant rate or loosely adjust their transmission rate on long timescales since the required rate adaptation for being well-behaved is not compatible with their nature.

Ubiquitous deployment of these applications along with faster tail-circuits such as cable modems and ADSL could result in severe inter-protocol unfairness and possibly even congestion collapse. As a result, we believe it is crucial for streaming application developers to understand the importance of fundamental design principles for Internet applications and apply them to their applications.

This paper discusses these design principles and presents
a general architecture that may be used to make unicast streaming applications network friendly. The goal is to maximize playback quality while being network friendly and at the same time, not requiring excessive processing power at video servers. We will also briefly describe some specific solutions to parts of this problem space.

2 Design Guidelines

In a shared best effort network such as the Internet, there are several guidelines that must be followed in the design of streaming applications:

Social Behavior

Social behavior has to do with the impact of the stream on coexisting flows in the Internet. The current Internet does not widely support any reservation mechanism or QoS. Moreover the available bandwidth not only is not known a priori but also changes with time. Thus applications need to experiment to learn about network conditions. A common approach is that applications gradually increase their transmission rates to probe availability of bandwidth without severely congesting the network\cite{Jac88}. When any indication of congestion is detected, they back-off their transmission rate rapidly. This process is known as congestion control and is required for stability of the network. Because of the dynamics of the traffic, each flow continuously probes and backs off to adapt its transmission rate to the available bandwidth. If all flows perform rate adaptation correctly and effectively, each flow obtains a fair share of bandwidth and the overall utilization of the network remains high.

Since flows are not isolated from each other, a misbehaved flow can affect other coexisting flows. The rate adaptation for congestion control should result in a fair share among the coexisting flows. It is crucial to understand that congestion control is a network dependent mechanism and must be equally deployed by all applications.

Even if the Internet eventually supports reservation mechanisms\cite{ZDE93} or differentiated services\cite{BBC98}, it is likely to be on per-class rather than per-flow basis. Thus, flows are still expected to perform congestion control within their own class.

Being Adaptive

With the Internet’s best-effort service model there is neither an upper bound for delay nor a lower bound for available bandwidth. The quality of the service provided by the network changes with time. Applications must be able to cope with these variations and adaptively operate over a wide range of quality of service. We call such mechanisms quality adaptation. Generally, these variations in quality of service affect application performance. For example, a video server can adjust the quality of delivered video as the available bandwidth changes, or limit the level of interactivity of the session as delay increases. The relationship between quality of the network service and application performance is application specific.

Recovery From Loss

Packets are randomly lost in the network mainly due to congestion. Although streaming applications can tolerate some loss, it does degrade the delivered stream quality. Thus streaming applications need a way to be able to recover from most losses before their playout time to maintain the reasonable quality. This mechanism is usually known as error control. The effect of loss on playout quality is also application specific.

![Video Server](image1.png)

Figure 1: A typical scenario for a sever with heterogeneous clients

2.1 Applying These Guidelines

Our target environment is a video server that plays back video streams on demand for a large group of heterogeneous clients simultaneously through the Internet. Figure 1 illustrates such a scenario where different streams are played back for four clients with different bandwidth. We consider the following assumptions:

- Clients have heterogeneous network capacity and processing power.
• Large numbers of clients may access the server simultaneously.
• Users expect startup playback latency to be low.
• Video clips are sufficiently large that their transmission time is longer than acceptable playback latency, so that pre-fetching is not an option.
• The server maintains a large number of different video streams.
• The server and clients are connected through the Internet where the dominant competing traffic is TCP-based.

We believe that this scenario reasonably represents many of the current streaming applications in the Internet.

Our goals are:
1. To make streaming applications well-behaved in general and TCP-friendly in particular,
2. To utilize a fair share of bandwidth,
3. To maximize the overall quality of delivered stream to the client for a given fair share of bandwidth,
4. To minimize the storage requirement at the server and the client side,
5. To minimize the playback delay between the client and the server,
6. To minimize processing requirements at the server.

We aim to provide a high level architectural view of the design of playback video applications for the Internet. We identify congestion control, quality adaptation and error control as three key components for any video streaming application in order to satisfy the above design guidelines. Toward that goal, we explore the design space for each one of these components in the context of video playback applications for the Internet in the next section. We compose these three components into a coherent architecture and suggest a possible mechanism for each component from its design space paying attention to the interaction among these components in section 4. Section 5 provides an overview of the RAP protocol. We also sketch an overview of a quality adaptation mechanism and address the impacts of the other components on this key component of the architecture in section 6. In section 7, we present some representative simulation results on RAP and our quality adaptation mechanism. In section 8, we argue that the architecture can be viewed as a generic architecture for video playback applications as long as the different modules are properly integrated. Finally, section 9 concludes the paper and addresses some of our future work.

3 Design Space

Before we describe our proposed architecture, we explore the design space for the key components and specify our design choices.

3.1 Congestion Control

As we mentioned earlier, the best-effort Internet service model requires end-systems to actively perform congestion control by adapting their transmission rate to the available bandwidth. Moreover, the rate adaptation must result in a fair share of bandwidth. The most well understood algorithm for rate adaptation is Additive Increase/Exponential Decrease(AIMD)[CJ89] used in TCP[Jac88], where bandwidth is linearly increased until a loss occurs, when a multiplicative decrease is then performed.

A dominant portion of today’s Internet traffic consists of a variety of TCP-based flows[CMT98]. It is necessary for new applications to be not only network friendly but also TCP-friendly otherwise they may shut out the well-behaved TCP-based traffic. By this we mean that a new applications that coexists with a TCP flow along the same path should obtain the same average bandwidth.

There are few congestion control mechanisms for streaming application that exhibit TCP-friendly behavior. TCP itself is inappropriate for applications with hard play-out constraints because it is totally reliable. Work in [JE97, CPW98] proposes modified versions of TCP that still inherit TCP’s bursty behavior. Moreover, neither this work nor LDA[SS98] was not examined against TCP over a wide range of network conditions. RAP[RHE99] is a rate-based congestion control mechanism based on an AIMD rate adaptation algorithm, but with slightly less bursty behavior than TCP. RAP is suited for streaming applications which need to exhibit TCP-friendly behavior over a wide range of network conditions. Another potential class of rate-based congestion control schemes are based on modeling TCP’s long-term behavior[MSMO97, PFTK98]. There is on-going work[HF] to evaluate stability of these mechanisms.

We have adopted RAP for congestion control in our architecture. We provide a brief overview of RAP below, but for a detailed description, refer to [RHE99].

3.2 Quality Adaptation

Streaming applications are rate-based. Once the desired quality is specified, the realtime stream is encoded and stored. The output rate of the encoder is a direct function of the requested quality, the encoding scheme and the content of the stream. Although the output rate of the encoder could vary with time, for simplicity we assume that that
encoder generates output with a near-constant bandwidth. With video, typically this means that the perceived quality is inversely proportional to the motion in the video. Remaining small variations in bandwidth are smoothed over a few video frames using playout buffering.

In contrast, AIMD rate adaptation results in a continuously variable transmission rate. The frequency and amplitude of these variations depends on the details of the rate adjustment algorithm and on competing background traffic during the life of the connection. The main challenge for streaming applications is to cope with variations in bandwidth while delivering the stream with an acceptable quality. A common approach is to slightly delay the playback time and buffer some data at the client side to absorb transmission rate variations[RKTS94]. The more data is initially buffered, the wider the variations that can be absorbed, but a higher startup playback latency will be experienced by the client. The main reason that we target playback applications is because they can tolerate this buffering delay. However we prefer the amount of buffered data to remain relatively low otherwise the available bandwidth is not utilized effectively. For a small amount of buffering, if the transmission rate varies widely and randomly, the client's buffer will either experience buffer overflow or underflow. Underflow causes interruption in playback and is very undesirable. Although buffer overflow can be resolved by deploying a flow control mechanism it then means the received quality is less than available bandwidth allows for.

A complementary mechanism for buffering is to adjust the quality of realtime streams with long term variations of available bandwidth. This is the essence of quality adaptation. There are several ways to adjust the quality of a pre-encoded stored stream, including adaptive encoding, switching between multiple encoded versions and hierarchical encoding.

One may adjust the resolution of encoding on-the-fly by re-quantization based on network feedback[OK95, TZ98, BT94]. However, since encoding is a CPU-intensive task, servers are unlikely to be able to perform on-the-fly encoding for large number of clients during busy hours. Furthermore, once the original data has been stored compressed, the output rate of most encoders can not be changed over a wide range.

In an alternative approach, the server keeps several versions of each stream with different qualities. As available bandwidth changes, the server switches playback streams and delivers data from a stream with higher or lower quality as appropriate.

With hierarchical encoding[MV95, McC96, VC94, LKK98], the server maintains a layered encoded version of each stream. As more bandwidth becomes available, more layers of the encoding are delivered. If the average bandwidth decreases, the server may then drop some of the layers being transmitted. Layered approaches usually have the decoding constraint that a particular enhancement layer can only be decoded if all the lower quality layers have been received.

There is a duality between adding or dropping of layers in the layered approach and switching streams with the multiply-encoded approach. The layered approach has several advantages though: it is more suitable for caching by a proxy for heterogeneous clients, it requires less storage at the server side and it provides an opportunity for selective retransmission of the more important information. The design space of a layered approach for quality adaptation is primarily in the design of an efficient add and drop mechanism that maximizes quality while minimizing the probability of base-layer buffer underflow.

A combination of buffering and quality adaptation is able to cope with variations of available bandwidth. Short term variations of transmission rate can be absorbed by buffering without adding or dropping layers whereas long term changes in available bandwidth trigger the quality adaptation mechanism to adjust the delivered quality of the stream by changing the number of layers transmitted.

### 3.3 Error Control

Streaming applications are semi-reliable, i.e. they do not need complete reliability. However, with most encoding schemes, packet loss will degrade the perceived playback quality because good compression has removed temporal redundancy and image corruption thus becomes persistent. These applications must therefore attempt to limit the loss at playback.

Techniques for repairing realtime streams are well known[Per98], and include retransmission[PP95], FEC[BG], inter-leaving and redundant transmission. In the context of unicast delivery of playback video, retransmission is a natural choice. The only disadvantage of retransmission-based approach is the retransmission delay, but in the context of non-interactive playback applications, client buffering provides sufficient delay to perform retransmission. Moreover retransmission can be performed selectively, which nicely matches our layered framework for quality adaptation where the lower layers are most important.

Only if there is sufficient time for retransmission before playout, missing packets are retransmitted. With a layered codec, retransmission of packets from layer \( i \) have priority over both new packets from layer \( i \) and over all packets from layer \( i + 1 \). This is because immediate data is more important than future data, and the lower layers are most important for perceived quality.
4 The Architecture

In this section, we compose our design choices into the coherent end-to-end architecture illustrated in figure 4, and show how this architecture follows the guidelines. The three key components of this architecture are rate adaptation, quality adaptation and error control.

End-to-end congestion control is performed by the rate adaptation (RA) and acker modules at the server and client respectively. The RA module continuously monitors the connection and regulates the server’s transmission rate by controlling the inter-packet gaps. The acker module acknowledges each packet, providing end-to-end feedback for monitoring the connection. The acker may add some redundancy\(^1\) to the ACK stream to increase robustness against ACK loss. Moreover, each ACK packet carries the most recent playout time back to the server, which allows the server to estimate the client buffer occupancy and perform quality adaptation and error control more effectively.

The quality of the transmitted stream is adjusted by the quality adaptation (QA) module at the server. This module periodically obtains information about the available bandwidth and the most recent playout time from the RA module. Combining this information with the average retransmission rate provided by the error control module, the quality adaptation module adjusts the quality of the transmitted stream by adding or dropping layers accordingly.

Error control is performed by the error control (EC) module at the server. It receives information about available bandwidth, loss rate and recent playout time from the RA module. Based on this information, it either flushes packets from the servers buffer manager that were acknowledged or are now too late for playout, or schedules retransmission of a lost packet if it has sufficiently high priority. The error control module can selectively retransmit those packets that have higher priority such as those from the base layer.

Since both quality adaptation and retransmission must be performed within the rate specified by the RA module, the EC and QA modules need to interact closely to share the available bandwidth effectively. They do this so that the quality (taking into account packet loss of the final played-out stream) is maximized for the available bandwidth while the quality has minimum variations. Retransmission has a higher priority than adding a new layer in general if there is extra bandwidth available.

These interactions among the RA, QA and EC modules are shown as part of the control path in figure 4 with thick arrows. The data path which is followed by the actual multi-media data, is specified separately with thinner arrows. The server maintains an archive of streams in local mass storage. A requested stream is pre-fetched and chopped into packets\(^2\) by the server buffer manager before the departure time of each packet arrives. The resolution (i.e. number of layers) of the pre-fetched stream is controlled by the QA module. Moreover, the QA and EC modules cooperatively arrange the order of packets for upcoming transmission. In summary, the RA module regulates the transmission rate and QA and EC modules control the content of each packet.

The client’s buffer manager receives packets and rebuilds the layered encoded stream based on the playout

\(^1\)For example, the acker for the RAP protocol reports the last hole in received sequence number as well as the most recent packet received.

\(^2\)In fact it may have been stored as packets to reduce the work the server needs to perform on playback.
time of each packet before they are fed into the decoder. The playout time of the base layer is used as reference for layer reorganization. The buffered data at the client side is usually kept in memory, but if the client does not have enough buffer space, the data could be temporarily stored on a disk before it is sent to decoder. Figure 3 shows a sample rebuilt stream in client’s buffer manager. Note that the packets do not need to arrive in playout order. The decoder consumes data from the client’s buffer manager with a near-constant rate which depends on the number of active layers at that point of time for a given encoding scheme.

We briefly describe the RAP protocol in the next section, then we sketch some of the design goals and trade offs for quality adaptation mechanisms to identify the open issues that require further investigation. This helps us to illustrate the implications of the design choice for one module on the other modules and in particular the implications of the congestion control mechanism on the design of a quality adaptation mechanism.

![Figure 3: Rebuilding the stream in client buffer manager](image)

### 5 Rate Adaptation Protocol (RAP)

RAP is an end-to-end rate-based congestion control mechanism suited for realtime applications in best effort networks. It performs congestion control in a network-friendly fashion by adjusting the transmission rate based on an additive-increase, multiplicative-decrease algorithm. Figure 4 shows the instantaneous transmission rate of a single RAP flow through a bottleneck.

![Figure 4: Transmission rate of a single RAP flow without fine grain rate adaptation](image)

We have also devised a fine-grain rate adaptation mechanism to fine tune the transmission rate according to short term variation of network load. The fine grain adaptation tends to emulate congestion avoidance behavior of TCP, and makes RAP fairer to TCP in lightly loaded networks. Figure 5 illustrates the impact of fine grain adaptation on the instantaneous transmission rate of a single RAP flow. Detailed description of the RAP protocol and simulation results can be found in [RHE99].

![Figure 5: Transmission rate of a single RAP flow with fine grain rate adaptation](image)

### 6 Quality Adaptation

The main challenge of the layered approach for quality adaptation is to design an add and drop scheme that adaptively changes the number of layers in response to long-term changes in available bandwidth indicated by the congestion control module. The add and drop scheme must...
achieve the following goals:

1. It must continuously deliver the maximum number of layers that can be fit in the average transmission rate.
2. It must rapidly adjust the number of layers with any long-term change in bandwidth.
3. Changes in quality (i.e. number of layers) must be as smooth as possible and not rapidly increase and decrease.
4. It must give higher priority to lower layers since the quality of the stream is more dependent to them.

One may design a generic add and drop mechanism for a general case. However, customizing the add and drop mechanism based on properties of the rate adaptation mechanism and loss patterns improves the effectiveness of the algorithm. To demonstrate the impact of rate adaptation and error control schemes on the quality adaptation mechanism, we will sketch an add and drop mechanism based on additive increase, multiplicative rate adaptation mechanism. We assume that the encoding implies that all layers have the same bandwidth for the sake of simplicity, but this need not hold in the general case although it would add additional complexity.

6.1 Add and Drop Scheme

The basic idea behind quality adaptation is to use data buffered at the receiver to absorb short-term variations in available bandwidth. During each ramp up, the receiver fills up its buffers while the available bandwidth exceeds the aggregate consumption rate. This period is called a filling phase of the stream. After each back off, buffers start to drain whenever the available bandwidth drops below the aggregate consumption rate. This period is called a draining phase of the stream. Figure 6 shows the transmission rate of a server with \( n_a \) active layers, assuming that the available bandwidth fluctuates around the consumption rate (i.e. \( n_a C \)) in the steady state.

Table 1 summarizes our notations in this section.

| \( D(t) \) | Aggregate volume of buffered data at time \( t \) |
| \( B(t) \) | Available bandwidth |
| \( B_{avg}(t) \) | Average bandwidth |
| \( S \) | Slope of linear increase |
| \( k \) | Back off factor |
| \( n_a \) | Number of active layers |
| \( C \) | Consumption/Generation rate of a layer |

Table 1

We can calculate variations of the aggregate buffered data during the filling and draining phases as follows:

Filling phase:
\[
B(t) \geq n_a C \\
D(t) - D(t_s) = \frac{1}{2S} (B(t) - n_a C)^2 \quad [1]
\]

Draining phase:
\[
B(t_k) \leq n_a C \\
D(t_k) - D(t) = \frac{1}{2S}(n_a C - B(t))^2 \quad [2]
\]

These equations relate the delivered quality of the stream measured in layers, \( n_a \), to the aggregate amount of buffered data \( D(t) \), available bandwidth \( B(t) \) at time \( t \), and network dynamics (i.e. \( S \)). Equation (2) suggests a mechanism for add and drop. The slope of increase in these equations is determined by the rate adaptation algorithm. For TCP-friendly congestion control mechanisms such as RAP, this slope is equal to one packet per RTT.

Drop Mechanism

A layer should be dropped after a back-off if the aggregate amount of buffered data does not suffice for a successful recovery with the current number of active layers. Solving equation (2) for \( n_a \), gives us this dropping mechanism:

\[
\text{WHILE}(n_a \leq \left\lfloor \sqrt{\frac{B(t_k) + \sqrt{2 \times S \times D(t_k)}}{2S}} \right\rfloor) \\
\text{DO } n_a = n_a - 1
\]

The dropping mechanism drops higher layers until the aggregate amount of buffered data would be enough to recover with the remaining layers.
Adding Mechanism

A new layer should only be added if it can be sustained after the upcoming back-off along with existing layers. Thus the server should ensure that sufficient resources in the form of buffered data and bandwidth exist before initiating a new layer. Accordingly, layer \( i \) can be added only if the following conditions are simultaneously satisfied at time \( t \):

1. The volume of aggregate buffer for the existing \((i-1)\) layers is sufficient to survive from a back off with \( i \) layers (i.e. all the existing layers and the new layer). This means \( D(t) \) satisfies the following equation:
   \[
   D(t) \geq \frac{1}{25}(B(t) - iC)^2
   \]

2. The average bandwidth is sufficient to add a new layer;
   \[
   B_{avg}(t) \geq iC
   \]

![Figure 7: Simple add and drop mechanism](image)

The first condition ensures that adding a new layer does not endanger the existing layers. We increase the amount of buffered data for the existing layers before sending any data for the new layer. The second condition prevents the server from adding a new layer when the average bandwidth is slightly higher than required bandwidth for existing layers. A new layer in these conditions would result in oscillation since the new layer can not be sustained. This is a conservative approach for adding. Intuitively, the conservative approach results in a smoother improvement in quality when available bandwidth suddenly increases and smoother variation in quality during normal operations.

Figure 7 illustrates these add and drop mechanisms using a simple inter-layer bandwidth sharing algorithm. The first graph shows the bandwidth allocation among the active layers during several filling and draining phases. All three layers have enough data buffered to recover from the back off at time \( t_1 \), and proceed to start their filling phase at time \( t_2 \). The filling phase for all three layers is completed at time \( t_3 \), however all layers continue to buffer more data to allow the adding of a new layer. At time \( t_4 \) the amount of buffered data suffices to recover with 4 layers for the given bandwidth. Thus each of the existing three layers drops its transmission rate to the consumption rate, \( C \). The left over bandwidth is allocated for a new layer. At time \( t_5 \), another back off occurs but all four layers have enough buffered data to be kept. All four layers start their recovery with an equal share of bandwidth at \( t_5 \). However, another back off right before the end of the new draining phase at time \( t_6 \) results in dropping the highest layer.

In summary, the core of quality adaptation mechanism is an add and drop scheme. This must address both inter-layer buffer and bandwidth sharing such that it maximizes the effectiveness of buffering while absorbing variations in available bandwidth and avoiding unnecessary fluctuations in delivered quality. There is a range of sharing mechanisms that defines the design space for quality adaptation mechanism. We are currently working on these mechanisms and have a simple near-optimal solution for the add and drop mechanism and inter-layer sharing[RHE].

7 Simulation

We have conducted extensive simulations to evaluate each component of the architecture. This section presents some representative results on RAP and our quality adaptation mechanism to show the feasibility of these schemes.

7.1 RAP Simulations

We have conducted a very large set of simulations to examine TCP-friendliness of the RAP protocol\(^5\) in a wide variety of scenarios by varying the number of coexisting

\(^5\)The detailed results have been published in [RHE99].
flows, RTT, bottleneck bandwidth, TCP flavor, and so forth. This is a crucial and challenging issue [PF97] in the evaluation phase because TCP is a moving target and its behavior changes in different situations. Unfortunately, this is usually underestimated and evaluation is only performed for few scenarios that are in favor of the new protocol. Due to space constraints, we only present results summarizing RAP’s behavior here.

One primary measure for RAP’s performance is the fairness ratio. The fairness ratio is the ratio of the average RAP bandwidth (calculated across all the RAP flows) to the average TCP bandwidth (calculated across all the TCP flows). Figure 8 shows the fairness ratio between RAP (without fine-grain adaptation) and traffic using the TCP SACK variant when they share the same bottleneck. This figure shows that RAP is TCP-friendly in networks with significant cross traffic.

Figure 9 shows the result of the same simulations when fine-grain adaptation is deployed for RAP flows. It clearly demonstrates that fine grain adaptation extends the range of scenarios where RAP exhibits TCP-friendly behavior, especially for scenarios with small RTT. This implies that congestion avoidance affects inter-protocol fairness in some scenarios.

7.2 Quality Adaptation

We have evaluated several add and drop mechanisms from the given design space through simulation. Each simulation is driven by an additive increase, multiplicative decrease pattern for transmission rate and a random loss pattern to model available bandwidth.

---

6fine-grain adaptation is a short term bandwidth modifier that attempts to mimic the congestion avoidance that TCP gets for free from ACK-clocking.

---

Figures 10 and 11 provide a detailed overview of the mechanisms in action for two cases where the layers are of different bandwidths. Four graphs are shown in each figure:

- The total send rate is shown, illustrating the saw-tooth output of RAP without fine-grain adaptation. Under this curve, we shade the proportion of bandwidth allocated to each layer. Overlaid over this graph is the exponentially weighted moving average of the bandwidth, \( B_{avg} \), which is used as a constrain for adding new layers.

- The send rate per layer is shown separately. Periods when a layer is being streamed above its consumption rate to build up receiver buffering are clearly visible as spikes in the bandwidth.

- The buffer drain rate per layer is also shown. Clearly visible are points where the buffers are used for playback because the total send rate is temporarily less than the total consumption rate.

- Finally the actual number of layers being played out and the accumulated buffering at the receiver for each of those layers is shown.

These graphs show that the short-term variations in bandwidth caused by congestion control can be effectively smoothed by receiver buffering. While doing this, playback quality is maximized without risking complete dropouts in the playback due to buffer underflow.

An explanation of the precise details of how bandwidth should be shared between the layers in these simulations and why the algorithm is optimal is too lengthy for this paper, but can be found in [RHE]. Although this same algorithm works for RAP with fine-grain adaptation, it is not
Figure 10: Quality adaptation with few layers
Figure 11: Quality adaptation with many layers
as optimal as without fine-grain adaptation. We are currently working on an appropriate modified variant, which is slightly more complex.

8 The Generic Architecture

Section 4 outlined a sample architecture for a video playback server and its client to describe the interactions among different components. This can be viewed as a generic architecture for a class of Internet video playback applications. The mechanisms we described for each component can be replaced by others from the corresponding design space as long as they are in harmony with other components. For example, one could deploy another technique such as FEC for error control on the base layer. Such a design choice would affect the buffer management scheme at the server and client side, and would change the interaction between QA and EC modules since there is no need to leave a portion of the available bandwidth for retransmission. Instead the base layer requires higher bandwidth. Another example would be to replace RAP with a congestion control mechanism based on modeling TCP’s long-term throughput. This implies that quality adaptation must be tuned based on the rate adaptation algorithm of the new congestion control mechanism. It is generally more effective to design a quality adaptation mechanism (i.e. add and drop scheme) that is customized to the design choices for the other components of the architecture. For example, knowing the rate adaptation algorithm allows us to devise a more optimal quality adaptation mechanism.

A key property of this architecture is to separate different functionalities and assign each of them to a different component. Given this generic architecture, the natural steps for designing an end-to-end scheme for video playback applications is the following:

1. Select a congestion control scheme that is network friendly.
2. Select an error control scheme based on the level of reliability that is required by the application codecs, the delay that can be tolerated before recovery, and the expected or measured loss pattern throughout the session.
3. Design an effective quality adaptation mechanism and customize it so that it maximizes the perceptual quality of the delivered video for a given encoding and rate adaptation algorithm. The design of the quality adaptation mechanism is also affected by the choice of error control mechanism.

We believe that quality adaptation is a key component of the architecture that requires more investigation. Congestion control is a network specific issue and it has been extensively studied. However work on congestion control for real time applications is more limited. Error control is a well understood issue and one can plug in one of the well known algorithms from the design space that suites the particular application. The remaining challenge is to design good quality adaptation mechanisms that must reconcile the constant-bit rate (or content-driven variable bit-rate) nature of video applications with the congestion-driven variable bandwidth channel. While doing this, it must interact appropriately with the error control mechanism toward the goal of maximizing the perceptual quality.

9 Conclusion and Future Work

This paper aimed to provide architectural insights into the design of Internet video playback applications. Toward that goal, we justified the need for three crucial components:

- End-to-end congestion control,
- Quality adaptation,
- Error control.

We believe that the majority of current Internet video playback applications are missing one or more of these components. Given the rapid increase in deployment of these applications and the severe consequences of ignoring these issues, it is important to understand these aspects and apply them.

We limited the design space for each of these components based on requirements that are imposed either by applications or by the network and indicate the natural choices for each one. Our main contribution is in combining these components into a coherent architecture and describing the interactions among them. As well as describing possible specific mechanisms for each component, we attempt to generalize the architecture by providing guidelines on design choice for each component from its own design space, and highlight the knock-on implications on the rest of the architecture.

We presented the idea of layered transmission for quality adaptation within the AIMD pattern for transmission rate and argue that quality adaptation is the main challenging open issue for such architectures. Hence, we elaborated on this issue by addressing some of the trade offs in design of an effective add and drop scheme that results in stable quality with smooth variations to motivate more work on this area. A few of our many simulation results were presented to illustrate that the proposed mechanisms are able to achieve the stated goals.

As part of our future work, we plan to study further the quality adaptation problem to find a near optimal solution.
for add and drop schemes with different congestion control schemes. We will validate our add and drop scheme along with the RAP protocol through simulations. We are also working on a prototype where we integrate all these pieces and evaluate them, first in a controlled physical environment such as CAIRN [Man98] and subsequently over the Internet. Since we have already evaluated TCP-friendliness of the RAP protocol, we only need to examine the add and drop scheme based on the smoothness and optimality of the delivered video quality. Finally, we plan to extend this architecture to a multi-client scenario, where the server plays back a video stream for a group of clients simultaneously using multicast.

References


