Architectural Considerations for Playback of Quality Adaptive Video over the Internet *

Reza Rejaie
AT&T Labs - Research
75 Willow Road
Menlo Park, CA 94025
reza@research.att.com

Mark Handley
ACIRI at ICSI
1947 Center St.
Berkeley, CA 94704
mjh@aciri.org

Deborah Estrin
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292
estrin@isi.edu

Abstract

Lack of QoS support in the Internet has not prevented rapid growth of streaming applications. However many of these applications do not perform congestion control effectively. Thus 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.

This paper aims to provide some architectural insights on the design of video playback applications in the Internet. We present fundamental design principles for Internet applications and 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. 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.

* This work was carried out while Reza Rejaie was with USC/ISI. This work was supported at USC/ISI by DARPA under contract No. DABT63-95-C0095 and DABT63-96-C-0054 as part of SPT and VINT projects.

1. Introduction

The Internet has been recently witnessing rapid growth in the use of audio and video streaming. Although some streaming applications such as conferencing tools (e.g. [12]) have a synchronized multi-user nature, the majority of current applications in the Internet involve web-based audio and video playback [9, 13] 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. Thus the overall behavior of these applications is likely to have a large impact on Internet traffic.

The Internet is a shared environment and its stability largely depends on end systems obeying the congestion control algorithms. End-to-end congestion control improves inter-protocol fairness and results in higher overall utilization of resources. Unfortunately, many of the current streaming applications do not perform end-to-end congestion control 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.

Our target environment is a video server that plays back video streams for a large group of clients with heterogeneous network capacity and processing power through the Internet where the dominant competing traffic is TCP-based. The server maintains a large number of video

streams. Video clips are sufficiently large that their transmission time is longer than acceptable playback latency. Thus pre-fetching the entire clip before its playback is not an option. We believe that this scenario reasonably represents many of the current streaming applications in the Internet. The goal is to maximize the overall stable playback quality while obeying congestion control constraints. Furthermore neither the server nor the clients should require excessive processing power or storage space.

This paper aims to provide a high level architectural view of the design of playback video applications for the Internet. Addressing design principles for Internet applications lead us to identify *congestion control*, *quality adaptation* and *error control* as three key components for any video streaming application in section 2. In section 3, we explore the design space for each one of these components in the context of video playback applications for the Internet and suggest a mechanism for each component from its design space. Our main contribution is to compose the three key components into a coherent architecture and describe the interaction among these components in section 4. In section 5, we argue that the architecture can be viewed as a generic architecture for video playback applications as long as different modules are properly integrated. Finally, section 6 concludes the paper and addresses some of our future work.

2. Design Principles

In a shared best-effort network such as the Internet, there are several principles that must be followed in the design of any new application including streaming applications as follows:

2.1. Social Behavior

The Internet is a shared environment and does not micromanage utilization of its resources. Since flows are not isolated from each other, a mis-behaved flow can affect other co-existing flows. Thus end systems are expected to be cooperative and react to congestion properly and promptly[7]. The goal is to improve inter-protocol fairness and keep utilization of resources high while the network operates in a stable fashion.

The current Internet does not widely support any reservation mechanism or Quality of Service(QoS). Thus the available bandwidth is not known a priori and changes with time. This implies that 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[11]. When any indication of congestion is detected, they rapidly *back-off* their transmission rate. This process is known as end-to-end 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. 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[23] or Differentiated services[1], 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.

2.2. 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. Furthermore, performing effective congestion control could result in random and wide variations in available bandwidth. Applications must be able to cope with these variations and adaptively operate over a wide range of network conditions.

Streaming applications are able to adjust the quality of delivered stream (and consequently its consumption rate) with long-term changes in available bandwidth and operate in various network conditions. However this mechanism is application specific. We call this *quality adaptation*.

2.3. 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. To maintain the reasonable quality, streaming applications need a way to recover from most losses before their playout time. Such a loss recovery mechanism is usually known as *error control*. The effect of loss on playout quality is also application specific.

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

The most well understood algorithm for rate adaptation is Additive Increase, Multiplicative Decrease(AIMD)[5]

used in TCP[11], where transmission rate is linearly increased until a loss signals congestion and a multiplicative decrease is performed.

A dominant portion of today's Internet traffic consists of a variety of TCP-based flows[6]. Thus TCP-friendly behavior is an important requirement for new congestion control mechanisms in the Internet otherwise they may shut out the well-behaved TCP-based traffic. By TCP-friendly we mean that a new application that co-exists with a TCP flow along the same path should obtain the same average bandwidth during a session.

TCP itself is inappropriate for streaming applications with hard timing constraints because its in-order delivery could result in a long delay. Even modified version of TCP without retransmission [10] exhibits bursty behavior. SCP[4] and LDA[20] protocols target streaming applications. Their goal is to be TCP-friendly, however they were not examined against TCP over a wide range of network conditions. RAP[18] is a rate-based congestion control mechanism that deploys an AIMD rate adaptation algorithm. RAP is suited for streaming applications and exhibits TCP-friendly behavior over a wide range of network conditions. Another potential class of rate-based congestion control schemes is based on modeling TCP's long-term behavior[8]. We have adopted RAP for congestion control in our architecture.

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 desired 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 encoder generates output with a near-constant bandwidth. In the context of video, this typically implies 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, performing TCP-friendly congestion control could result in unpredictable and potentially wide variations in transmission rate. The frequency and amplitude of these variations depends on the details of the rate adjustment algorithm and the behavior of 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 and stable quality. A common approach is to slightly delay the playback time and buffer some data at the client side to absorb the variations in transmission rate [16]. The more data is initially buffered, the wider are the variations that can be absorbed, but a higher startup playback latency is experienced by the client. The main reason that we target playback applications is because they can tolerate this buffering delay. If the transmission rate of a long-lived session varies widely and randomly, the client's buffer will either experience buffer overflow or underflow. Underflow causes an interruption in playback and is very undesirable. Although buffer overflow can be resolved by deploying a flow control mechanism it then means that the fair share of bandwidth is not fully utilized.

To tackle this problem, a complementary mechanism for buffering is required to adjust the quality(i.e. consumption rate) of streams with long term variations of available bandwidth. This is the essence of *quality adaptation*[17]. A combination of buffering and quality adaptation is able to cope with random variations of available bandwidth. Short term variations can be absorbed by buffering whereas long term changes in available bandwidth trigger the quality adaptation mechanism to adjust the delivered quality of the stream.

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[2, 21]. 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 [22], 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 drop some of the active layers. 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[19], it requires less storage at the server side and it provides an opportunity for selective retransmission of the more important information. The main challenge of a layered approach for quality adaptation is primarily in the design of an efficient add and drop mechanism that maximizes overall delivered quality while

minimizing disturbing changes in quality.

3.3. Error Control

Streaming applications are semi-reliable, i.e. they require quality instead of complete reliability. However, with most encoding schemes, packet loss beyond some threshold will degrade the perceived playback quality because good compression has removed temporal redundancy and image corruption thus becomes persistent. Therefore these applications must attempt to limit the loss rate below that threshold for a given encoding.

Techniques for repairing realtime streams are well known[15], and include retransmission[14], FEC[3], interleaving and redundant transmission. The appropriate repair mechanism is selected based on the level of reliability that is required by the application codec, the delay that can be tolerated before recovery, and the expected or measured loss pattern throughout the session.

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 more important than the higher layers.

Missing packets are retransmitted only if there is sufficient time for retransmission before playout. 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 more important for perceived quality.

4. An Architecture

In this section, we compose our design choices into a coherent end-to-end architecture and explain the interaction among different components. Figure 1 depicts the architecture. The three key components are labeled as 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 to the ACK stream to increase robustness against ACK loss. Moreover, each ACK packet carries the most recent playout time back to the server. This 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 QA 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 server's buffer manager that were acknowledged or their playout time have passed, or schedules retransmission of a lost packet. The EC module can selectively retransmit those packets that have high priority such as losses 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. The goal is to maximize the quality (taking into account packet losses of the final played-out stream) for the available bandwidth while minimizing any variations in quality. In general, retransmission has a higher priority than adding a new layer whenever extra bandwidth is available. These interactions among the RA, QA and EC modules are shown as part of the control path in Figure 1 with thicker arrows.

The data path which is followed by the actual multimedia 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 divided into packets by the server buffer manager just prior to the departure time of each packet. 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 whereas QA and EC modules control the content of each packet. The client's buffer manager receives the packets and rebuilds the layered encoded stream based on the playout 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 the decoder

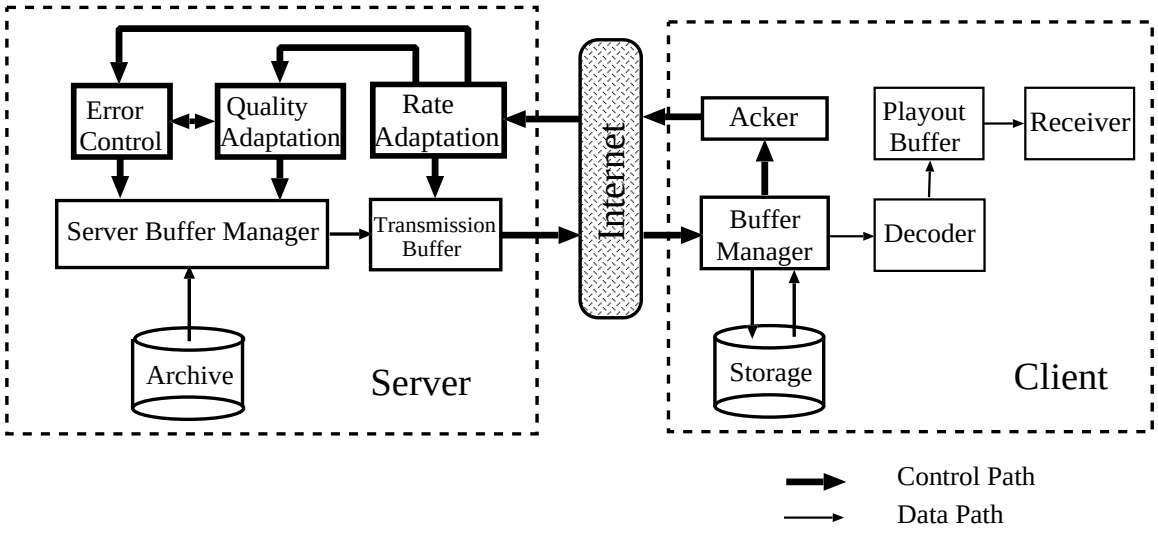

Figure 1. End-to-end architecture for playback streaming applications in the Internet

5. Generalizing the Architecture

We outlined a sample architecture for a video playback server and its client to describe the interactions among different components in the previous section. This can be viewed as a generic architecture for a class of Internet streaming playback applications. The selected 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 of lost packets from the base layer. Instead the base layer requires a 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 that is customized to the design choices for the other components of the architecture.

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 are the following:

1. Select a TCP-friendly congestion control scheme.
2. Select an error control scheme that satisfies the application requirement given the expected or measured characteristics of the channel.
3. Design an effective quality adaptation mechanism and customize it such that it maximizes the perceptual quality of the delivered video for a given encoding, rate adaptation algorithm and the choice of error control mechanism.

Congestion control is a network specific issue and it has been extensively studied. However work on congestion control for streaming 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 application specific quality adaptation mechanisms that 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. We believe that quality adaptation is a key component of the architecture that requires more investigation.

6. 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: (1) End-to-end congestion control, (2) Quality adaptation, and (3) 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 presented 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 attempted to generalize the architecture by providing guidelines on design choice for each component from its own design space, and highlighted the implications on the rest of the architecture. We presented the idea of layered transmission for quality adaptation within congestion controlled rate limit and argued that quality adaptation is the main challenging issue for such architectures.

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 are also working on a prototype where we integrate all these pieces and evaluate them, first in a controlled physical environment and subsequently over the Internet. Finally, quality adaptation provides a perfect opportunity for proxy caching of multimedia streams where the proxy caches a low-quality version of a stream during the first playback and gradually pre-fetches higher-quality layers in a demand-driven fashion.

7. Acknowledgments

We would like to thank Ted Faber, John Heidemann, Allison Mankin, Ahmed Helmy, Art Mena and Nirupama Bulusu for their thoughtful comments on drafts of this paper.

References

[1] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *Internet draft*, Oct. 1998.

[2] J. Bolot and T. Turletti. A rate control mechanism for packet video in the internet. *Proc. IEEE INFOCOM*, pages 1216–1223, June 1994.

[3] J.-C. Bolot and A. V. Garcia. The case for FEC-based error control for packet audio in the internet. *To apear in ACM Multimedia Systems*.

[4] S. Cen, C. Pu, and J. Walpole. Flow and congestion control for internet streaming applications. *Proc. Multimedia Computing and Networking*, Jan. 1998.

[5] D. Chiu and R. Jain. Analysis of the increase and decrease algorithm for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[6] K. Claffy, G. Miller, and K. Thompson. The nature of the beast: Recent traffic measurements from an internet backbon. *Proc. INET, Internet Society*, Dec. 1998.

[7] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE Transactions on Professional Communication*, 19999.

[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. *To appear in ACM SIGCOMM*, Sept. 2000.

[9] M. Inc". Netshow service, streaming media for business. http://www.microsoft.com/NTServer/Basics/NetShowServices.

[10] S. Jacobs and A. Eleftheriadis. Real-time dynamic rate shaping and control for internet video applications. *Workshop on Multimedia Signal Processing*, pages 23–25, June 1997.

[11] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM*, pages 314–329. ACM, Aug. 1988.

[12] V. Jacobson and S. McCanne. vic: a flexible framework for packet video. *Proc. of ACM Multimedia*, Nov. 1995.

[13] R. Networks. Http versus realaudio client-server streaming. http://www.realaudio.com/help/content/http-vs-ra.html.

[14] C. Papadopoulos and G. M. Parulkar. Retransmission-based error control for continuous media applications. *In Proc. NOSSDAV*, Apr. 1995.

[15] C. Perkins. Options for repair of streaming media. *Internet Draft*, Mar. 1998.

[16] R. Ramjee, J. F. Kurose, D. Towsley, and . H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. *Proc. IEEE INFOCOM*, 1994.

[17] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for congestion controlled video playback over the internet. *ACM SIGCOMM*, Sept. 1999.

[18] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. *Proc. IEEE INFOCOM*, Mar. 1999.

[19] R. Rejaie, H. Yu, M. Handley, and D. Estrin. Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet. *Proc. IEEE INFOCOM*, Mar. 2000.

[20] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. *In Proc. NOSSDAV*, 1998.

[21] W. Tan and A. Zakhor. Error resilient packet video for the internet. *Proc. of the IEEE International Conference on Image Processing)*, Oct. 1998.

[22] M. Vishwanath and P. Chou. An efficient algorithm for hierarchical compression of video. *Proc. of the IEEE International Conference of Image Processing*, Nov. 1994.

[23] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: a new resource reservation protocol. *IEEE Network*, 7:8–18, Sept. 1993.