Proceedings of SPIE Image and Video Communications and Processing, January 2000

Signal Processing for Internet Video Streaming: A Review

Jian Lu
Apple Computer, Inc.

ABSTRACT

Despite the commercial success, video streaming remains a black art owing to its root in proprietary commercial development. As such, many challenging technological issues that need to be addressed are not even well understood. The purpose of this paper is to review several important signal processing issues related to video streaming, and put them in the context of a client-server based media streaming architecture on the Internet. Such a context is critical, as we shall see that a number of solutions proposed by signal processing researchers are simply unrealistic for real-world video streaming on the Internet. We identify a family of viable solutions and evaluate their pros and cons. We further identify areas of research that have received less attention and point to the problems to which a better solution is eagerly sought by the industry.

Keywords: video streaming, video compression, Internet video, rate adaptation, packet loss, error control, error recovery.

1. INTRODUCTION

Video streaming is one of the most exciting applications on the Internet. It already created a new business known as Internet broadcast, or Intercast/Webcast. Although Internet broadcast is still in its infancy, we get a glimpse of the future of broadcasting where people can enjoy not only scheduled live or pre-recorded broadcast but also on-demand and personalized programs. Stirring more excitement is the idea that anyone will be able to set up an Internet TV station and broadcast his or her video programs on the Internet just like we publish our “homepage” on the Web today. While video streaming as a core technology behind a new business is a success story, it remains very much a black art owing to its root in proprietary commercial development. Indeed, to the author’s knowledge, there has been no published research literature to date that offers a comprehensive review of video streaming architecture and system, and related design issues.

Even the definition of streaming is sketchy; it is often described as a process. Generally speaking, streaming involves sending media (e.g., audio and video) from a server to a client over a packet-based network such as the Internet. The server breaks the media into packets that are routed and delivered over the network. At the receiving end, the packets are reassembled by the client and the media is played as it comes in. A series of time-stamped packets is called a *stream*. From a user’s perspective, streaming differs from simple file transfer in that the client plays the media as it comes in over the network, rather than waiting for the entire media to be downloaded before it can be played. In fact, a client may never actually download a streaming video; it may simply reassemble the video’s packets and play the video as the packets come in, and then discard them. To a user the essence of streaming video may lie in an expectation that the video stream the user has requested should come in continuously without interruption. This is indeed one of the most important goals for which a video streaming system is designed.

Over the last decade, IP multicast and particularly the MBone have received considerable attention from the research community. IP multicast provides a simple and elegant architecture and service model that are particularly suitable for real-time multimedia communications. The emergence of the MBone, an experimental virtual multicast backbone, has offered a testbed for researchers around the world to develop and test a suite of applications known as MBone tools for real-time multimedia communications and collaborations. See, for example, a recent article by McCanne[1] for the evolution of the MBone and its applications. Many of these applications have been designed for audio and video conferencing and broadcasting using IP multicast. On November 18, 1994, the MBone reached a festive milestone amid great fanfare when Rolling Stones broadcast the first major rock band concert on the MBone.

Despite the promise demonstrated by the MBone and a logical reasoning that IP multicast would serve as a foundation for video streaming on the Internet, commercial deployment of multicast applications and services on the public Internet has been painfully slow by the Internet standard. The bulk of the streaming media traffic generated by three major commercial

Correspondence: One Infinite Loop, MS: 302-3MT, Cupertino, CA 95014, USA; Email: jian@computer.org.

streaming systems (i.e., QuickTime, RealSystem, and Windows Media) on today's Internet consists of largely unicast streams. It is out of this paper's scope to discuss the unresolved issues that have slowed down the deployment of multicast applications and services; interested readers are referred to a recent white paper published by the IP Multicast Initiatives.[2] However, we want to note that even when multicast is fully deployed, a very large volume of streaming media traffic will remain to be unicast streams. This is because a large portion of the streaming media content is archived and played on-demand by users on the Internet. Multicast is of little help to on-demand streaming unless a large number of users from the same neighborhood of the Internet would request the same content simultaneously.

The purpose of this paper is to review several important issues in Internet video streaming from a signal processing perspective. For the reasons we have just mentioned above, our review will focus on unicast streaming while considering multicast as a possible extension. Drawing from the author's experience in developing one of the major commercial media streaming systems, we attempt to identify the most challenging problems in video streaming and evaluate a number of signal processing solutions that have been proposed; some of these solutions have already been implemented in experimental and/or commercial video streaming systems. To make the review useful to both researchers and application developers, we would like to clearly identify among the proposed solutions: what works in real world, what doesn't, and why? We want to note that although to a Web surfer streaming video commonly implies synchronized audio as well, video and audio are actually delivered in independent streams in a streaming session. We will not discuss streaming audio in this paper.

The rest of this paper is organized as follows. Section 2 introduces streaming system and architecture and describes what we believe to be the most challenging problems in video streaming. We also examine some important characteristics of the streaming system and the constraints that are imposed on any prospective signal processing solutions. Section 3 reviews a number of signal processing solutions to the video streaming problems in great detail. We first comment on some proposed solutions that are simply unrealistic for real-world video streaming on the Internet. Then, we evaluate a family of viable solutions, identify areas of research that have received less attention, and point to the problems to which a better solution is eagerly sought by the industry. Section 4 concludes the paper.

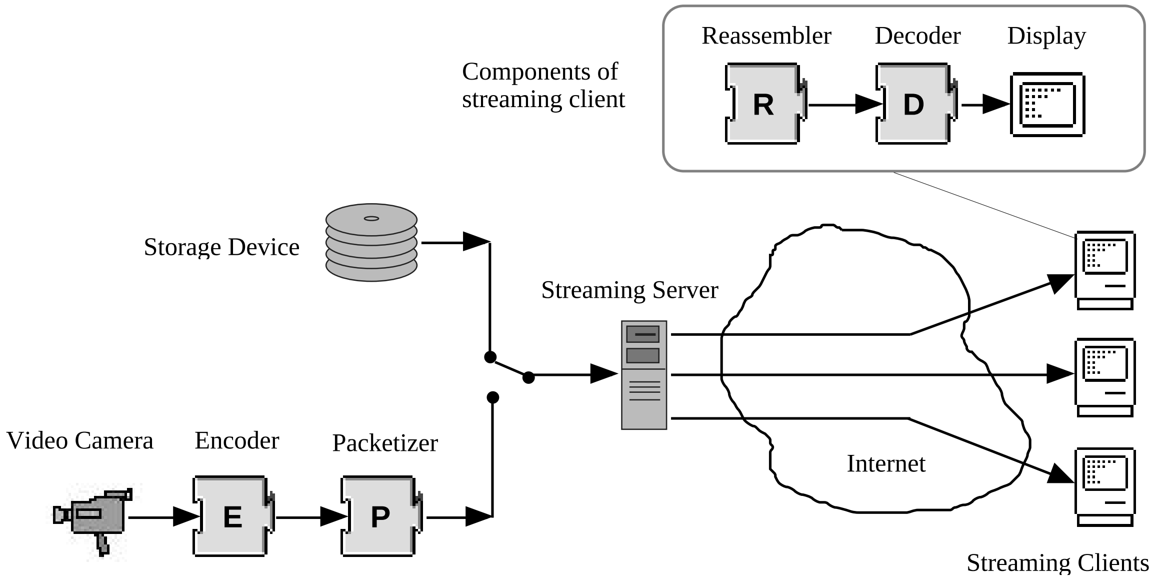
2. VIDEO STREAMING SYSTEM AND ARCHITECTURE

2.1. Video Streaming Model and Architecture

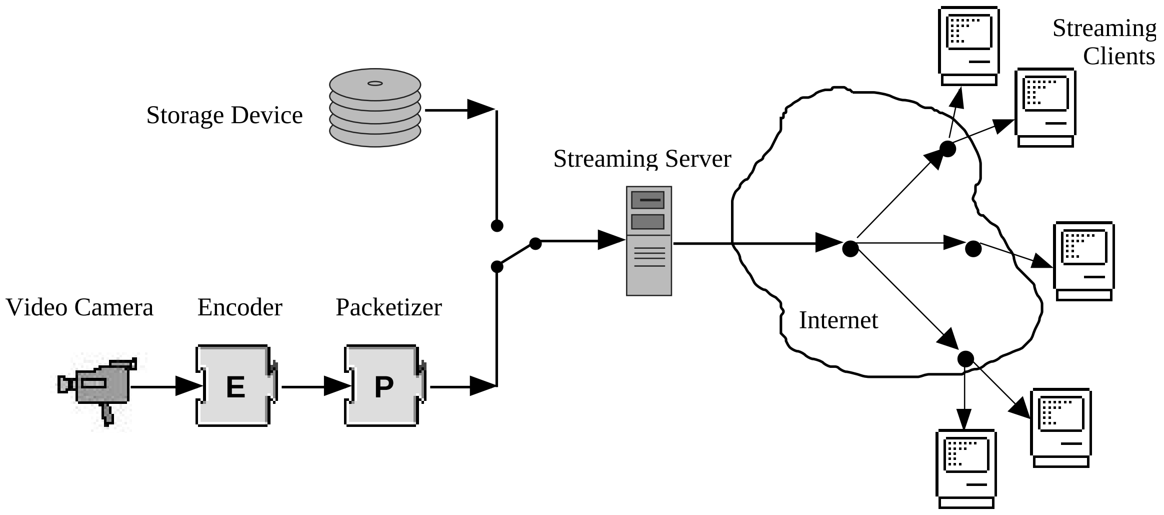
A schematic system diagram of Internet video streaming is shown in Figure 1 for both unicast and multicast scenarios. Figure 1(a) represents the unicast model for two types of video streaming services, *on-demand* or *live*. In on-demand streaming, pre-compressed, streamable video content is archived on a storage system such as a hard disk drive and served to a client on demand by the streaming server. In live streaming, a video stream coming out of a live source such as a video camera is compressed in real-time by the encoder. The compressed bitstream is packetized by the packetizer, and sent over the Internet by the streaming server. On the client side, the video bitstream is reassembled from the received packets by the reassembler, then decoded by the decoder, and finally rendered on a display. Noting that each client connecting to the server receives an individual stream, the disadvantages of the unicast streaming model can be explained. Clearly, the server load will increase proportionally to the number of clients connected to and served by the server. When a large number of clients try to connect to a server at the same time, the server gets overloaded, creating a "hot spot" on the Internet. Additionally, sending copies of the same data in individual streams to a large number of clients is obviously inefficient; it causes congestion on the network and deteriorates quality of services.

The multicast model of Figure 1(b) has many advantages over unicast in live video streaming (broadcasting) where a large audience could tune in for the broadcast at the same time. Instead of sending a stream to each client individually, the server sends out only one stream that is routed to one or more "group addresses". A client tunes in to a multicast group in its neighborhood to receive the stream. Since the server is decoupled from connecting to each client directly, server load does not increase with the number of receiving clients. In the case of delivering a stream to multiple groups, the stream on its route gets replicated and branched only at the fan-out points within the network. When combined with layered coding that will be discussed in Section 3, multicast provides a decentralized and highly efficient video streaming architecture.

Despite the clear advantages of multicast-based video streaming, it is employed in only a small fraction of today's Internet, primarily in Intranets within an institution or corporation. There are complex reasons for the slow deployment of multicast-based services in general.[2] Beyond deployment hurdles we note that a large volume of streaming video content is archived and streamed to users on demand. Multicast does not help very much in on-demand streaming where the service requests from users are random and highly desynchronized in terms of requested content and timing of these requests. Interestingly, recent development in the so-called “content delivery network” appears to have achieved some improvements in unicast-based on-demand streaming that are reminiscent of those we have seen with multicast-based live streaming. A content delivery network such as the one developed by Akamai Technologies[3] is essentially a load-balanced network of servers. The servers are deployed on the “edge” of the Internet, making them close to the end users. These “edge servers” communicate with a “source server” to update their content. A client requesting a streaming video is redirected dynamically to the closest edge server that is not overloaded. Like multicast-based video streaming, this decentralized service model can reduce “hot spots”. In principle, a content delivery network can also help unicast-based live streaming with a possibly increased latency.



(a)

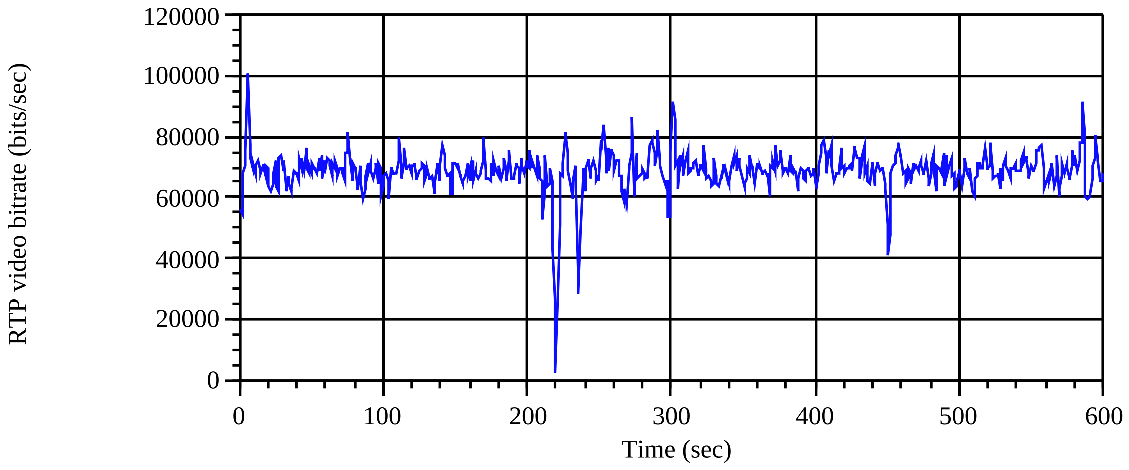


(b)

Figure 1. Video streaming system models. (a) Unicast video streaming, and (b) Multicast video streaming.

2.2. The Internet Challenge to Video Streaming

The Internet provides a great challenge to video streaming. From a communication point of view the Internet is a shared channel with a massive number of heterogeneous components. This channel is extremely volatile over space and time in terms of communication capacity and quality. To see what it means to video streaming let us look at some data that we have collected from a real-life video streaming session.



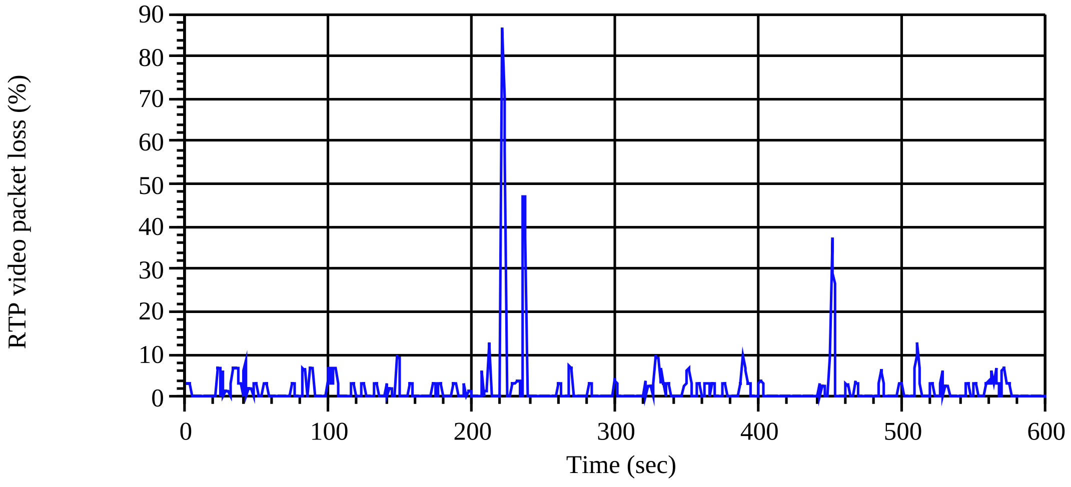


Figure 2. RTP video bitrate (top) and packet loss rate (bottom). The data was recorded at 10:08-10:18am on December 23, 1999, while watching BBC World live broadcast on QuickTimeTV from Sunnyvale, CA.

Figure 2 shows a 10-minute data log by a streaming client in Sunnyvale, California, during a session watching the BBC World channel on QuickTimeTV on December 23, 1999. The data shows the variations in bitrate and percentage of packet loss of the incoming RTP (Real-time Transport Protocol)[4] video stream. The client has signaled to the server that it had a dual ISDN connection with up to 112kbps in bandwidth. Knowing that the client's bandwidth was also shared by streaming audio, we observed that the bitrate of the incoming video fluctuated between 60 and 80kbps in general and the packet loss ranged from 0 to 10%. There are, however, a few exceptional instances where the bitrate dropped to as low as 2kbps and packet loss rate shot up as high as 85%. It is also interesting to observe from Figure 2 that there is a strong correlation between a sudden drop in bitrate and a burst of packet loss. This appears to agree with a common-sense logic about the relationship between congestion and packet loss.

Figure 2 shows us all but a typical video streaming session when everything runs normally. There are also extreme cases. Many of us may have experience of watching streaming video that stalled completely for several times during a streaming session. In such cases, the bitrate of the incoming video is nil and the packet loss rate is 100%, making it totally worthless to watch. Generally speaking, the experience of streaming video isn't always as bad (or good) as that in our memory or in Figure 2. A person who was watching the same streaming video from another location might have had a totally different experience. The point is that the Internet channel variations are so volatile and space- and time-dependent that the observation can be different by every client on the network at every second. The data shown in Figure 2 and our experience about video streaming challenge us with two fundamental signal processing problems whose solutions can improve the performance of Internet video streaming dramatically. The first problem is concerned with how to adapt to the constantly changing bandwidth and congestion condition on the Internet. The second calls for solutions to coping with packet loss due to the best-effort service model of the current Internet. We will, for short, refer to these two problems as "rate adaptation", and "error control and recovery", respectively. Before we attempt to address these problems and evaluate a number of proposed solutions, we need to examine the constraints that have been imposed on any prospective solutions. We go back to the video streaming model and architecture of Section 2.1 and make a few important observations.

2.3. Video Streaming System Characteristics and Constraints

Many signal processing researchers are familiar with the video conferencing model thanks to the popularity of ITU H.261 and H.263 standards. In the following we review the characteristics of a video streaming system and point out its difference from video conferencing. For the comparison purpose we provide in Figure 3 a functional diagram of an IP-based peer-to-peer video conferencing system. As we have mentioned, we will focus on analyzing the unicast video streaming system.

(a) *Latency*. One difference between streaming and conferencing is that the former does not require two-way user interaction. As such the requirement on latency is more relaxed in streaming than in conferencing. Latency requirement commonly translates into a decoder/client buffer of certain size; which is often simulated by the encoder/server for rate control. When a video streaming session is initiated, the client waits for a few seconds for its buffer to be filled by the server. This process is called pre-roll. After the pre-roll the client starts playing the video. Obviously, the client's buffer has to be kept from underflow in order for the video to continue to play without interruption. Generally speaking, streaming can afford to use a much larger buffer than that can be considered in conferencing to absorb network jitters. However, latency and smooth playback must be balanced to create a good user experience, which becomes more important in streaming live video. It is common in commercial video streaming systems to pre-roll a few seconds of data into the decoder/client buffer.

(b) *Client-Server Model*. This makes a major distinction between the video streaming system of Figure 1(a) and the peer-to-peer conferencing system of Figure 3. In the video conferencing system of Figure 3, there is no server involved. Additionally, most components including encoder, decoder, packetizer, and reassembler at either end often reside on one piece of hardware, which is a desktop computer in the so-called desktop conferencing environment. The video streaming system in Figure 1(a), however, operates in a client-server environment. As such, a streaming server may be required to serve thousands of clients simultaneously. It is therefore necessary for the encoder to be separate from the server hardware so that the server hardware can be dedicated to serving the clients. For archived, on-demand video streaming, encoder and server are separate by default. In live streaming, encoder/server separation enables a more efficient "encode once, serve multiple times" service model. This model is almost always adopted in practice. A common implementation for live video streaming is to combine the encoder and packetizer to reside on one piece of hardware that broadcasts a stream to the server. The server simply "reflects" the stream to its clients. The client-server model and the nature of encoder/server separation make it unsuitable for many signal processing techniques that have been developed for video conferencing to be used in video streaming. We will come back to this observation later.

(c) *Server Load Restrictions.* As we have pointed out that a streaming video server may be required to serve thousands of clients simultaneously. When the number of clients connected to a server increases, the server experiences an elevated load due to increased processing and I/O demands. Eventually the server can get overloaded and the quality of service drops. While it is relatively well understood that a good signal processing solution should not require the server to do much processing, the requirement for I/O is often overlooked. In archived, on-demand video streaming, when the number of clients increases, heavy I/O activity, particularly disk access, often becomes the bottleneck in the streaming system. This ought to be kept in mind in designing a signal processing solution for video streaming.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.