Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective

Lei Guo[1], Enhua Tan[1], Songqing Chen[2], Zhen Xiao[3], Oliver Spatscheck[4], and Xiaodong Zhang[1]

[1]Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210, USA
{lguo, etan, zhang}@cse.ohio-state.edu

[2]Department of Computer Science
George Mason University
Fairfax, VA 22030, USA
sqchen@cs.gmu.edu

[3]IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
xiaozhen@us.ibm.com

[4]AT&T Labs-Research
180 Park Ave.
Florham Park, NJ 07932, USA
spatsch@research.att.com

ABSTRACT

Modern Internet streaming services have utilized various techniques to improve the quality of streaming media delivery. Despite the characterization of media access patterns and user behaviors in many measurement studies, few studies have focused on the streaming techniques themselves, particularly on the quality of streaming experiences they offer end users and on the resources of the media systems that they consume. In order to gain insights into current streaming services and thus provide guidance on designing resource-efficient and high quality streaming media systems, we have collected a large streaming media workload from thousands of broadband home users and business users hosted by a major ISP, and analyzed the most commonly used streaming techniques such as automatic protocol switch, Fast Streaming, MBR encoding and rate adaptation. Our measurement and analysis results show that with these techniques, current streaming systems tend to over-utilize CPU and bandwidth resources to provide better services to end users, which may not be a desirable and effective way to improve the quality of streaming media delivery. Motivated by these results, we propose and evaluate a coordination mechanism that effectively takes advantage of both Fast Streaming and rate adaptation to better utilize the server and Internet resources for streaming quality improvement.

Categories and Subject Descriptors

C.2 [**Computer Communication Networks**]: Distributed Systems

General Terms

Measurement

IMC'06, October 25–27, 2006, Rio de Janeiro, Brazil.

Keywords

Traffic analysis, Multimedia streaming

1. INTRODUCTION

The Internet has witnessed the surge of multimedia content from many application areas such as education, medical research and practice, news media, and entertainment industries [13]. Although the majority of media traffic on the Internet is delivered via downloading, pseudo streaming, and P2P techniques, streaming service is superior in handling thousands of concurrent streams simultaneously, flexible responses to network congestion, efficient bandwidth utilization, and high quality performance [17]. Different from downloading or pseudo streaming small sized video clips from a Web site such as YouTube [9], streaming long duration and high quality media objects on the Internet has several unique challenges. First, streaming services usually require high and stable end-to-end bandwidth between a media server and its clients. Due to the lack of Quality of Service (QoS) guarantee on the packet switching based Internet, the quality of Internet media streaming may significantly degrade due to bandwidth fluctuations during a streaming session, especially for delivering high quality video such as HDTV. Second, the connection speed of Internet end users ranges from slow dial-up connections to T1 or high speed cable network services. Thus, a fixed encoding rate for a media object is not desirable for clients with diverse network connections. Third, streaming media users always expect a small startup latency. However, due to the dynamics on the media server load and network bandwidth, a client may experience a prolonged startup delay. In addition, the filling of the client play-out buffer, which is used to smooth jitter caused by network bandwidth fluctuations, further increases the user's waiting time. With these challenges, a lot of studies have been conducted on the effective utilization of server and Internet resources to deliver high quality streaming media.

Today, more than 90% of streaming media traffic on the Internet is delivered either through Windows media services or RealNetworks media services [17]. These commercial streaming services have adopted various techniques to ad-

dress the above challenges and to satisfy the ever-increasing quality demands of users, such as TCP and HTTP based streaming, Fast Streaming, multiple bit rate (MBR) encoding and rate adaptation. Due to the wide deployment of Network Address Translation (NAT) routers and firewalls that often prevent UDP packet transversal, TCP-based streaming has been widely used and now accounts for the majority of Internet streaming traffic [17, 26]. Fast Streaming [2] is a group of techniques supported by the Windows media service, which aggressively utilizes the Internet bandwidth by delivering a media object at a rate much higher than its encoding rate, in order to minimize the user perceived startup latency and guard against potential network bandwidth fluctuations. MBR encoding is a technique that encodes a media object with multiple bit rates so that the streaming server can deliver the same content with different quality to clients with different network connections. MBR encoding also enables dynamic stream switch among streams of different rates encoded in the object during a user session, in order to adapt to the current bandwidth, which is called *Intelligent Streaming* [4] in the Windows media service and *SureStream* [7] in the RealNetworks media service.

In spite of the wide deployment of these techniques, existing measurement studies of Internet streaming media mainly focus on the characterization of media access patterns and user behaviors, such as [10, 12, 13, 15, 31], which is helpful to the design of media delivery systems such as server clusters and media proxies [11, 30]. However, the mechanisms of the commonly and practically used streaming techniques themselves and their effects on improving Internet streaming quality have not yet been thoroughly studied, which is necessary to understand state-of-the-art of Internet streaming media and to provide guidance on future Internet streaming services. Despite several experimental studies in lab environments on the Windows and RealNetworks media systems [14, 22], to the best of our knowledge, to date, there is no comprehensive study on the delivery quality and resource utilization of these streaming techniques in the Internet environment. It is highly desirable for both streaming service providers and system designers to be guided with an insightful understanding of existing Internet streaming techniques.

In order to investigate Internet streaming quality and the efficiency of resource utilization with the deployment of these techniques, in this work, we have collected a 12-day streaming media workload from a large ISP in the United States. The workload covers thousands of broadband home users and hundreds of business users who access both on-demand and live streaming media. Through extensive analysis of the majority of TCP-based streaming traffic on the Internet, we have the following observations:

- We found that the overhead of protocol rollover plays an important role in user perceived startup latency, and thus may have affected the way that media is served by content providers. When UDP is not supported, the overhead of protocol rollover from UDP to TCP contributes a non-trivial delay to the client startup latency. More than 22% of protocol rollover is longer than 5 seconds.

- By aggressively utilizing the Internet network bandwidth, Fast Streaming shows both positive and negative features. Although Fast Streaming can help smooth re-buffering jitter, it over-supplies media data to end users by about 55%, and consumes more CPU resources, which leads to a longer server response time.

- MBR-encoding is widely used in media authoring, and nearly half of streaming video and audio objects on the Internet are MBR-encoded. However, the rate adaptation functionality of MBR is poorly utilized, particularly when Fast Streaming is used.

- Overall, on the Internet, about 13% of home and 40% of business streaming sessions suffer various quality degradations, such as rebuffering, thinning, or switching to a lower quality stream.

Our measurement and analysis results show that with these techniques, current streaming services tend to over-utilize CPU and bandwidth resources to provide better service to end users, which may not be a desirable and effective way to improve the quality of streaming media delivery. Furthermore, the Fast Streaming technique does not work with rate adaptation, resulting in even worse user experiences than normal TCP-based streaming upon long-term network congestion. Motivated by these results, we propose *Coordinated Streaming*, a mechanism that effectively coordinates caching and rate adaptation in order to improve streaming quality with an efficient utilization of the server and Internet resources. The potential of such a mechanism in streaming quality improvement is evaluated accordingly.

The remainder of this paper is organized as follows. Section 2 describes our trace collection and processing methodology. Section 3 presents an overview of our collected workload. The measurement and analysis of the delivering quality and resource utilization of streaming media services are performed in Sections 4, 5, and 6. The coordinating caching and rate adaptation mechanism is discussed in Section 7. Some related work is outlined in Section 8. Finally, we make concluding remarks in section 9.

2. TRACE COLLECTION AND PROCESSING METHODOLOGY

The prevailing streaming protocols on the Internet are RTSP [25] and MMS [5]. In RTSP streaming, the client and the server exchange streaming commands via RTSP, running on TCP. The media data packets and streaming control/feedback packets are delivered via RTP/RTCP [24] (such as Windows and QuickTime media services) or RDT [6] (RealNetworks media services), running on UDP or TCP. In MMS streaming, all streaming commands and control packets between a client and a server are exchanged via MMS in the same TCP connection, and the media data can be delivered over UDP or TCP. For both RTSP and MMS streaming, when TCP is used to deliver media data, the media and control packets are interleaved with RTSP or MMS commands in a single TCP connection, instead of using two separate TCP connections. In addition to RTSP and MMS, media can also be streamed through HTTP [3]. Different from HTTP downloading (also known as pseudo streaming [17]), HTTP streaming uses the HTTP protocol to deliver both RTSP commands and media data. In Microsoft HTTP streaming, the RTSP headers are embedded in the `Pragma` headers of HTTP messages. In RealNetworks and QuickTime HTTP streaming, the RTSP commands are embedded in HTTP message bodies with the base64 encoding format.

Table 1: Home User Workload Overview

Content Type		Product Type	Number of Requests	Traffic (GB) TCP/UDP
on demand	audio	WM	28,210	5.86/0.89
		RM	9,139	0.79/2.26
		QT	244	0.00/0.082
	video	WM	67,002	151.21/20.64
		RM	12,117	6.25/17.31
		QT	113	0.01/0.34
live media	audio	WM	1,499	5.36/6.69
		RM	1,164	0.25/2.39
		QT	4	0.00/0.14
	video	WM	950	13.50/2.85
		RM	643	5.69/3.09
		QT	6	0.00/0.003

Table 2: Business User Workload Overview

Content Type		Product Type	Number of Requests	Traffic (GB) TCP/UDP
on demand	audio	WM	9,725	3.67/0.01
		RM	1,285	3.03/0.04
		QT	5	0.00/0.001
	video	WM	5,762	21.18/3.31
		RM	1,057	3.94/0.42
		QT	8	0.00/0.01
live media	audio	WM	493	5.56/0.01
		RM	350	4.46/1.05
		QT	–	–
	video	WM	50	0.65/0.00
		RM	7	0.20/0.00
		QT	–	–

In this study, we collected streaming media packets in a data center of a major ISP from 2005-04-29 15:00 (Friday) to 2005-05-10 20:30 (Tuesday), using the Gigascope appliance [16]. The data center hosts servers for thousands of business companies, and provides Internet access services for a large cable company. The Gigascope is running on a site close to the end users (broadband home users and business users). To collect streaming packets of RTSP/MMS requests, Gigascope captures all TCP packets from/to ports 554-555, 7070-7071, 9070, and 1755. According to a recent measurement study that collects RTSP/MMS packets based on keyword matching [17], our port number selection covers 97.3% of the RTSP/MMS streaming requests [1]. Meanwhile, we also collected UDP streaming traffic via ports 5004-5005, 6970-6980, and 7000-7010, which are the most popular ports for UDP streaming. For UDP streaming traffic over other port numbers due to network address translation (NAT), we calculate the traffic volume based on the summary information that a client reports to its server when a streaming session is terminated. Compared to existing studies that are based on server logs [12, 15, 26, 27, 31], in which only the summary information of streaming sessions is available, our study is conducted at the packet level, which facilitates more detailed analysis on the quality and the resource utilization of various Internet streaming techniques.

The initial trace processing is as follows. We first grouped TCP packets by TCP connections, based on the IP address, port number, and TCP SYN/FIN/RST flag. Then we extracted the RTSP/MMS commands from each streaming request. Based on the analysis of these commands, we identified and parsed media data and streaming control packets from the TCP or corresponding UDP streams, and dumped the corresponding RTP/RTCP, RDT, and MMS packet headers. Finally, we identified home users and business users in our traces based on IP prefix matching.

Our trace collection and processing methodology have been validated by extensive experiments on various media server and player products, including Windows Media Player and Windows Server 2003, Real Player and Helix Server, QuickTime Player and Darwin Server. All these products have extensions to the standard RTSP and RTP protocols. Due to the lack of documentation, we reverse-engineered proprietary protocols by capturing and analyzing media traffic under different environments, with the help of tools such as tcpdump/windump, ethereal, and NIST Net network emulator.

[1]There are about 2.6% RTSP/MMS/HTTP streaming requests using port 80 or 8080, which are hard to distinguish from regular HTTP downloading traffic. We exclude this traffic in this study.

3. TRAFFIC OVERVIEW

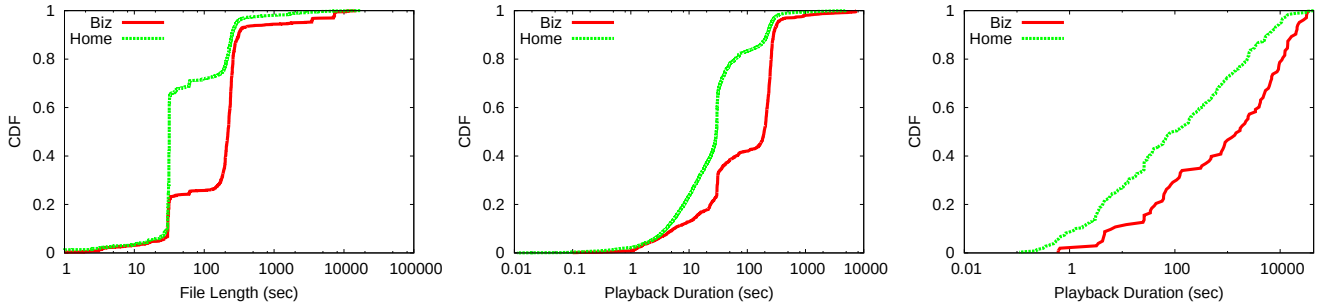
We have captured 126 GB of streaming data (compressed in gzip format) during the 12-day period. In our workloads, there are 7,591 home users accessing 1,898 servers in 121,091 requests, and there are 219 business users accessing 911 servers in 18,742 requests. Both users and servers are identified by their public IPs, and the real number of business users would be much larger due to the usage of NAT (a business IP may host up to 64 users as shown in Section 4.2).

3.1 Streaming traffic by user communities

Table 1 and 2 show the traffic breakdowns based on the content types (audio/video, live/on-demand) and media service products in the home user and business user workloads, respectively. In these tables, WM, RM, and QT denote Windows, RealNetworks, and QuickTime media services, respectively. In our workloads, most streaming traffic is delivered over Windows media services (80.7% and 85.5% of the requests in the home and business user workloads, respectively), and RealNetworks is next (19.0% and 14.4% of the requests in the home and business user workloads, respectively). Only a small fraction of streaming traffic is delivered over QuickTime. These tables also indicate that TCP is responsible for the majority of streaming traffic on the Internet, confirming previous studies such as [17, 26].

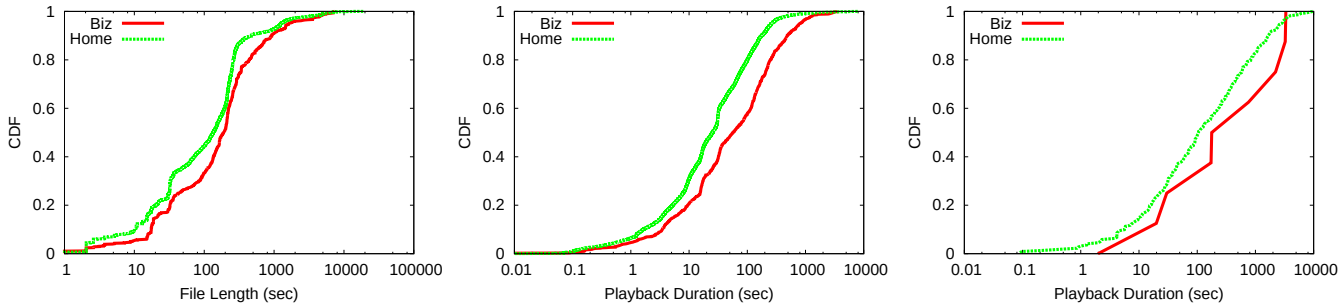
In the home user workload, 2.20% and 31.05% of the requests access live and on-demand audio objects, and 1.32% and 65.43% of the requests access live and on-demand video objects, respectively. Video is responsible for the majority of streaming media requested by home users. In contrast, in the business user workload, 4.50% and 58.77% of the requests access live and on-demand audio, while 0.30% and 36.43% of the requests access live and on-demand video, respectively. Audio is responsible for the majority of streaming media requested by business users. In addition, although the volume of live media traffic is far less than that of on-demand media traffic in both workloads, compared to home users, business users are more likely to access live media, most of which is audio.

Figure 1(a) and Figure 2(a) show the distribution of file length (in terms of playback duration) of on-demand audio and video objects in each streaming session requested by home and business users, respectively. These figures indicate that business users tend to request audio and video



(a) On-demand audio file length (b) On-demand audio playback duration (c) Live audio playback duration

Figure 1: On-demand and live audio distributions in the home and business user workloads



(a) On-demand video file length (b) On-demand video playback duration (c) Live video playback duration

Figure 2: On-demand and live video distributions in the home and business user workloads

objects with longer file lengths. Specifically, for audio objects, as shown in Figure 1(a), more than 70% of sessions in the business user workload request objects with a file length between 200–400 seconds, the typical duration of a pop song; in contrast, for home users, more than 50% of audio sessions request files with a length around 30 seconds, most of which are music preview samples.

Figure 1(b) and Figure 2(b) further compare user playback durations of on-demand audio/video sessions in the home and business user workloads. As shown in Figure 1(b), for more than half of on-demand audio sessions in the business user workload, the user playback duration is about 200–400 seconds, corresponding to the length of a typical pop song as in Figure 1(a). Figures 1(c) and 2(c) show the playback duration of live audio/video sessions in the home and business user workloads. From these figures, we can see that for both live and on-demand audio/video sessions, the playback duration of business users is much longer than that of home users (note that the x-axis is in log scale).

The above results indicate that business users tend to listen more audio on the Internet and tend to stick to the media content being played longer than home users. However, looking into the URLs and the `Referer` headers of RTSP commands in the trace, we found that the majority of streaming media accessed by home users and business users are both from news and entertainment sites. Thus, the different access pattern of business users is not due to the accesses of business related media content, but rather due to the working environments of business users—audio is more preferred possibly because it attracts less attention when they are working, and the long playback duration might be because their business work prevents them from frequently changing media programs.

Table 3: Traffic by different hosting services

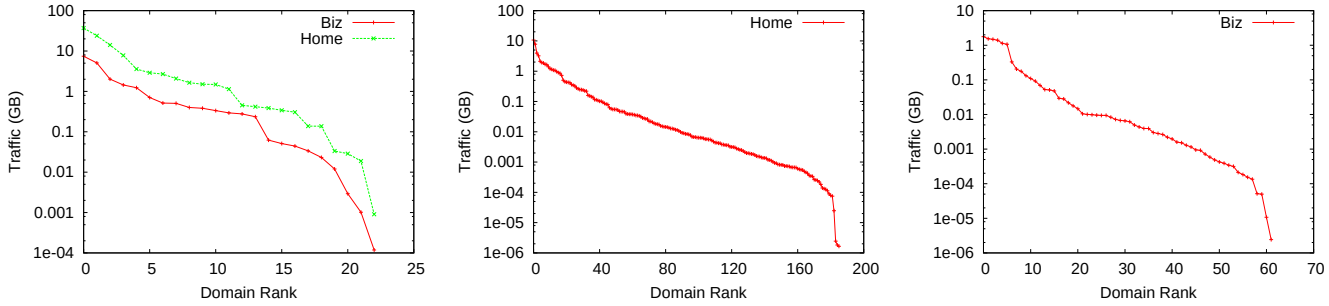
Hosting Service	Content Type	Home User		Business User	
		Vol. (GB)	Req.	Vol. (GB)	Req.
Third Party	audio	13.82	27,896	10.19	9,810
	video	126.24	54,136	13.71	3,820
Self Hosting	audio	11.41	12,188	7.64	2,056
	video	95.33	26,939	16.00	3,085

3.2 Streaming traffic by media hosting services

In general, there are two approaches that a content provider can use to deliver its streaming media. The first is that a content provider can host the streaming server itself. We call this *self-hosting*. The second is that a content provider can ask help from a third party, such as a commercial content delivery network (CDN) or media delivery network (MDN), to avoid service management and hardware/software investments. We call this *third-party hosting*. For both self-hosting and third-party hosting, the portals of streaming media are usually on the Web sites of their content providers.

We identified service providers in our workloads by their host names and IP addresses. The host names are extracted from the URLs of media objects encoded in RTSP/MMS packets. We have identified 24 third-party media services, including 22 CDNs/MDNs and 2 media services hosted by ISPs (we anonymize the company names due to customer privacy concerns).

Table 3 shows the volume and number of requests of streaming traffic served by third-party media services and self-hosting media services in the home and business user workloads, respectively. In the home user workload, third-party media services serve 56.8% of the traffic and 67.7%



(a) Third-party media services (b) Self-hosting media services in the home user workload (c) Self-hosting media services in the business user workload

Figure 3: Service providers ranked by traffic volume

of the requests. In the business user workload, third-party media services serve 50.3% of the traffic and 72.6% of the requests. The percentages of audio traffic served by third-party services are similar for home and business users, but business users request more video from self-hosting media services. Our further investigation finds that a substantial amount of such video traffic comes from a news site and a sports site outside United States, which might be due to the foreign employees in these business companies. In general, more than half streaming traffic is served by third-party hosting services in both workloads.

Figure 3(a) and Figures 3(b), 3(c) further show the rank of traffic volume served by different service providers in the home and business user workloads, for third-party hosting services and self-hosting services, respectively. Most streaming traffic is served by the top five CDNs/MDNs and the top two self-hosting commercial sites (one video site and one well-known search engine site).

4. PROTOCOL ROLLOVER AND USER STARTUP LATENCY

Although traditionally UDP is the default transport protocol for streaming media data transmission, in practice, UDP is often shielded at the client side. Therefore, today streaming media data are often delivered over TCP or even HTTP. Among the three options, generally UDP is tried first upon a client request. If UDP is not supported due to either server side or client side reasons, TCP can be used instead. If TCP is not supported either, HTTP will be used. Such a procedure is called *protocol rollover*, which is conducted automatically by the media player.

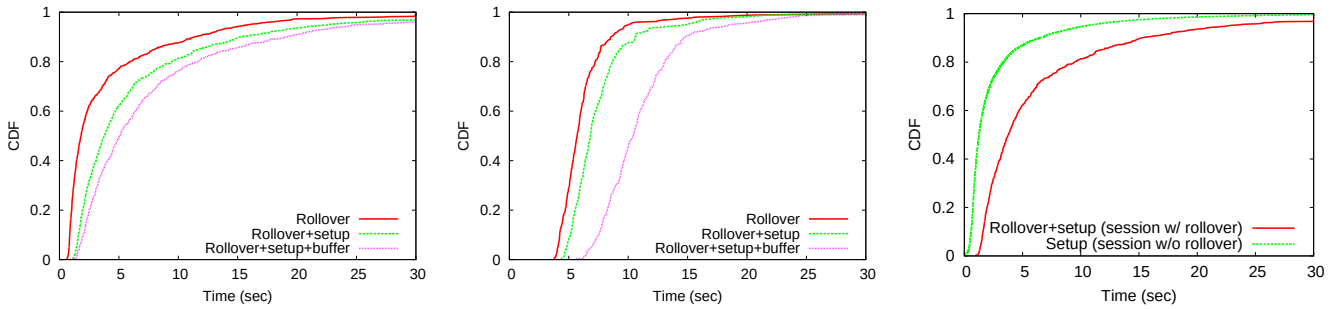
Due to the wide deployment of NAT routers/firewalls in home user networks and small business networks, protocol rollover plays an important role in the user perceived startup latency and may have affected the way that media is served by content providers. In this section, we first analyze the impact of protocol rollover on the user perceived startup latency, then investigate rollover avoidance in these streaming media services.

4.1 Startup latency due to protocol rollover

Protocol rollover in RTSP works as follows (protocol rollover in MMS is similar). Upon a client request, the media player sends a `SETUP` command with a `Transport` header, specifying the transport protocol it prefers. If UDP is supported, the port numbers for receiving data and sending feedback are also specified. Then in the `Transport` header of the `SETUP` reply message, the streaming server returns the protocol it selects. If it selects UDP, the port numbers for sending data and receiving feedback are also returned. If the player requests UDP but the server does not support it, the server responds with the protocol it supports (i.e., TCP) directly, and the protocol switches without additional overhead. However, if the server supports UDP but the player is shielded by a NAT router/firewall, the incoming UDP traffic may not be able to go through the router. After a timeout, the player has to terminate the current RTSP connection and sends a new RTSP request in a new TCP connection, specifying TCP as the transport protocol in the `SETUP` command. As a result, such a negotiation procedure for protocol rollover takes a non-trivial time.

Thus, the *startup latency* of a user session can be further decomposed into three parts: (1) *protocol rollover time* is the duration from the beginning of the first RTSP/MMS request to the beginning of the last RTSP/MMS request in the user session; (2) *transport setup time* is the duration from the the beginning of the last RTSP/MMS request (or the first request if no protocol rollover) to the time when the transport protocol setup succeeds; (3) *startup buffering time* is the time to fill the play-out buffer of the media player, starting from the transport setup success time to the playback start time. In our workloads, we have found that although most user sessions with protocol rollover try UDP only once, some sessions may try UDP up to 3 times before switching to TCP. As the negotiation process of protocol rollover may take a non-trivial time, the protocol rollover increases the startup latency a user perceives.

Assuming a five-second play-out buffer [1] is used, Figure 4(a) and Figure 4(b) show the distribution of startup latency in Windows and RealNetworks media services, respectively, for RTSP sessions with protocol rollover in the home user workload (the distribution for business user workload is similar). In Windows media services, more than 22% of the streaming sessions have a rollover time longer than 5 seconds, in addition to the normal startup latency due to the transport setup and the initial buffering. Figure 4(b) shows that RealNetworks media services have an even longer rollover time—more than 67% of the streaming sessions have a rollover time longer than 5 seconds. We also observe that in general the Windows Media service has a much shorter buffering time than the RealNetworks Media service. This is probably due to the higher buffering rate of Windows media streaming (see *Fast Start* in Section 5), since the object



(a) Startup time of rollover sessions in Windows media services

(b) Startup time of rollover sessions in RealNetworks media services

(c) Startup comparison of sessions with and without rollover in Windows media services

Figure 4: Protocol rollover increases startup latency of streaming sessions

encoding rates of Windows and RealNetworks media in our workloads are comparable. Figure 4(c) further compares the delay from the session beginning time to the transport setup completion time for sessions with and without protocol rollover in Windows media services in the home user workload. As shown in the figure, about 37% of the sessions with protocol rollover have a delay longer than 5 seconds. In contrast, only about 13% of the sessions without protocol rollover have a delay longer than 5 seconds.

4.2 Protocol selection and rollover avoidance

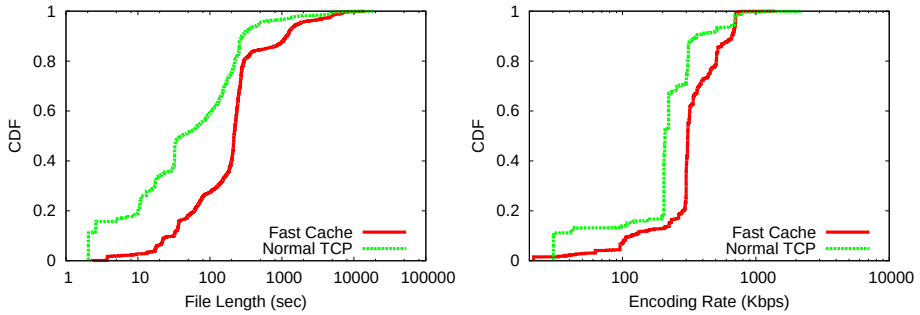
In most client side media players, although the transport protocol of streaming media can be specified by a user manually, the default protocol is usually UDP. In Section 3, we have also found that the majority of streaming traffic in our workloads is delivered over TCP. Thus, it is reasonable to expect that a large portion of the user sessions experience protocol rollover.

However, in the home user workload, we found that there are only about 7.37% of streaming sessions trying UDP first and then switching to TCP. In the business user workload, only about 7.95% streaming sessions switch from UDP to TCP. These results imply that TCP is directly used without protocol rollover in most streaming sessions, despite the default protocol setting in the player. We analyze this phenomenon as follows.

The Windows streaming service allows the specification of the transport protocol in the URL modifier at either the client side or the server side. In Windows media streaming, the URL passed to a media player is either input by a user (client side action) or exported from a media meta file stored on or dynamically generated by a Web server (server side action). For example, `rtspt` means using TCP as the transport protocol while `rtspu` means using UDP. We extracted URL modifiers from the summary of media playing information, which is sent by the client in a RTSP/MMS command when a session is terminated. In both home user and business user workloads, we found that for more than 70% of the Windows streaming sessions, TCP is specified as the transport protocol by content providers. This explains why TCP-based media streaming is so prevalent on the Internet. Study [26] suggests that the NAT and firewall deployment constrained the usage of UDP in streaming media applications. Our conjecture is that as content providers are generally aware of the wide deployment of NAT and firewalls, they actively use TCP to avoid any possible shielding or protocol rollover to end users. With such a configuration, even if UDP is supported at both the client side and the server side, the streaming media will still be delivered over TCP directly.

To validate our conjecture, we further investigate the NAT usage of home users and business users with the MMS streaming in our workloads. Different from RTSP, in MMS streaming, a client reports its local IP address to its server in clear text. Extracting this information from the MMS workload, we found that most MMS users in the home and business user workloads report private IPs (such as `192.168.1.100`), indicating that they access the Internet through NAT. In the home user workload, about 98.3% of the MMS requests are initiated from clients shielded by NAT, and about 99.5% of the MMS clients are shielded by NAT. These two numbers are 89.5% and 88.0% in the business user workload, respectively. A NAT router hosts up to 3 MMS clients in the home user workload, and up to 64 MMS clients in the business user workload. Thus, for these clients, the TCP transmission specified on the server side effectively avoids protocol rollover, and significantly reduces user perceived startup latency.

On the other hand, RealNetworks media services try to avoid protocol rollover by using NAT transversal techniques. By reverse-engineering of the protocol, we find that different from the Windows media service, in which a server sends UDP packets to its client first through the port that the client reports in the `SETUP` command, in the RealNetworks media service, a client sends UDP packets to its server first, so that the server can figure out the client's external UDP port number converted by NAT. To distinguish different user sessions shielded by the same NAT, the server uses different UDP port numbers to listen to UDP packets coming from different sessions, which are generated by the server dynamically and sent to its clients in the replies of `SETUP` commands. As a result, UDP accounts for the majority of streaming traffic in the RealNetworks media service, and protocol rollover is less frequent than that in the Windows media service. However, as indicated by Figure 4, once a protocol rollover happens, the rollover time in the RealNetworks media service is generally much longer than that in the Windows Media service. Furthermore, this solution somehow violates the standard RTSP specification because the UDP port number that a client reports to its server in the `SETUP` command is intentionally discarded.



(a) File length (b) Encoding rate

Figure 5: Features of media delivered by Fast Cache and normal TCP streaming

5. FAST STREAMING

In early streaming media services, a media object is streamed at its encoding rate and a small play-out buffer is used to smooth the streaming jitter. However, in practice, the play-out buffer may be exhausted, since the available bandwidth between a client and its server may fluctuate from time to time. This is particularly important for TCP-based streaming, in which the congestion control mechanism constrains the streaming rate. As we have shown in Section 4, content providers of Windows media services often use TCP-based streaming directly to avoid protocol rollover. In order to provide high quality streaming experience to end users, Windows Media services use *Fast Streaming* techniques [2], including *Fast Start*, *Fast Cache*, *Fast Recovery*, and *Fast Reconnect* [2]. Both Fast Start and Fast Cache transmit media data at a rate higher than the media encoding rate [3]. Fast Start can run over both TCP and UDP while Fast Cache always runs over TCP. Fast Start is enabled by almost all Windows media servers in order to reduce the startup buffering time for clients. Basically, Fast Start transmits data to the client as fast as possible until the play-out buffer is filled. After the Fast Start period, Fast Cache streams media data until the entire object is delivered or the session is terminated by the user. In order to smooth out network bandwidth fluctuations, Fast Cache transmits media data to a client at a speed usually up to 5 times the media encoding rate and the client maintains a growing buffer for the early arrived data.

Table 4 shows the total volume of streaming traffic delivered over Fast Cache (FC), normal TCP streaming excluding FC (TCP), and UDP streaming (UDP) in our workloads. As the table shows, Fast Cache is widely used by both third-party hosting services and self-hosting services, accounting for 50.1% and 21.0% of the streaming traffic in the home and business user workloads, respectively. There is less streaming traffic delivered over Fast Cache in the business user workload than in the home user workload, because the access pattern of business users is different from that of home users. Business users access more audio and live media than home users. Audio media objects have low bit rates and usually do not need Fast Cache, while live media objects cannot be streamed over Fast Cache at all. The on-demand video they access is different too.

[2]As Fast Recovery and Fast Reconnect events are rare in our workloads, we do not include them in this study.

[3]Similarly, RealNetworks media services can also stream a media object at a rate higher than its encoding rate. Due to page limits, we only present the analysis results of Windows streaming services.

Table 4: Streaming Traffic with Fast Cache

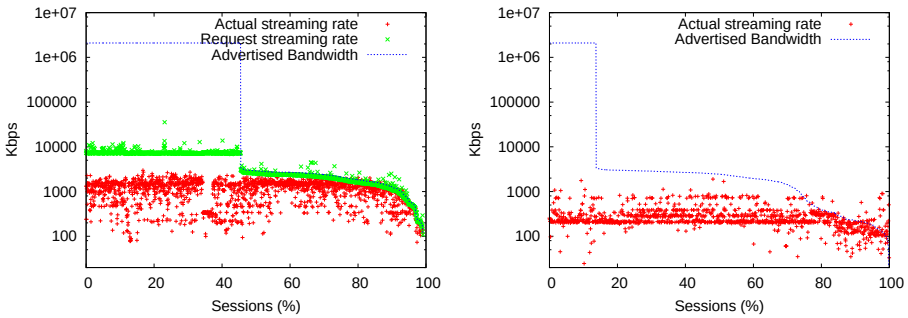
Delivery Method	home user		business user	
	third-party	self-hosting	third-party	self-hosting
FC	55.00 GB	68.72 GB	3.75 GB	6.24 GB
TCP	49.91 GB	15.27 GB	15.92 GB	16.79 GB
UDP	35.15 GB	22.74 GB	4.23 GB	0.62 GB

Figure 5 compares the distribution of file length and encoding rate of objects delivered over Fast Cache supported streaming and normal TCP streaming in on-demand Windows video sessions of the home user workload. As shown in the figure, Fast Cache is more widely used for objects with longer file lengths and higher encoding rates. This is reasonable because these objects are more sensitive to network bandwidth fluctuations and thus Fast Cache can help more.

Due to page limit, in the remainder of this section, we only present our analysis results of Windows media streaming for TCP-based on-demand video sessions with a playback duration longer than 30 seconds in the home user workload, if not specified particularly (the results for the business user workload are similar).

5.1 Fast Cache smoothes bandwidth fluctuation

As mentioned before, Fast Cache supported streaming smoothes the fluctuation of network bandwidth by maintaining a growing buffer for the early arrived data. To understand how Fast Cache utilizes the network bandwidth, we extract the `Bandwidth` header in a client's `PLAY` command, and the `Speed` header in this command and that in the server's reply. The `Bandwidth` header contains the client advertised bandwidth, which is either a default value set by a user in the media player, or measured in real time before media streaming with the so called "packet pair" technique. The `Speed` header in a client's `PLAY` command specifies the delivery speed that the client requests, in multiples of the media encoding rate, which is usually equivalent to the client available bandwidth. The speed that a server agrees to offer is specified in the `Speed` header of the reply to the `PLAY` command, which is usually not greater than 5 times of the media encoding rate. However, the actual streaming speed may be smaller than the speed that the server claims in the `PLAY` reply. Thus, we computed the average of actual streaming rate during each user session based on the packet level information in our workload (the startup buffering is excluded).



(a) Fast Cache supported streaming (b) Normal TCP streaming

Figure 6: The client advertised bandwidth, client requesting rate, and server streaming rate

Figure 6(a) shows the distribution of the client advertised bandwidth, the client requested streaming rate (i.e., the product of the `Speed` value and the media encoding rate), and the actual streaming rate for streaming sessions with Fast Cache support. Figure 6(b) shows the advertised bandwidth and the actual streaming rate for normal TCP streaming. Comparing Figures 6(a) and 6(b), we find that the actual streaming rate in Fast Cache supported streaming is much closer to the client advertised bandwidth than that in normal TCP streaming [4]. So Fast Cache exploits the unutilized network bandwidth under the constraint of TCP congestion control. In other words, for normal TCP streaming, it is possible to deliver the same media at a higher transmission rate, or to deliver the media with a higher encoding rate.

In Windows media streaming, the default play-out buffer size accounts for five seconds media playback [1]. Upon network fluctuations, if the play-out buffer is empty, the client has to stop to buffer data, and a playback jitter occurs. With Fast Cache, the early buffered data could afford a smooth playback much longer before rebuffering is necessary. We define the *rebuffering ratio* of a streaming session as the total time for rebuffering over the total playback duration, which reflects the streaming quality that a user experiences. Figure 7(a) shows the rebuffering ratio of sessions streamed with and without Fast Cache support (i.e. Fast Cache and normal TCP streaming in the figure). To make a fair comparison, we only consider sessions requesting video objects with an encoding rate between 200–400 Kbps. About 15% of the normal TCP streaming sessions suffer rebuffering while only about 8.5% of the Fast Cache supported streaming sessions suffer rebuffering. Thus, Fast Cache can effectively eliminate rebuffering in streaming sessions. We also observe that for Fast Cache supported streaming, there are about 1.8% of the sessions with a rebuffering ratio larger than 50%, while for normal TCP streaming, there are only about 0.9% of the sessions with a rebuffering ratio larger than 50%. The reason is that when rebuffering happens, normal TCP streaming may switch to a stream of lower rate if the media object is MBR encoded (see Section 6.1), thus avoiding further rebuffering. In contrast, stream switch is disabled in Fast Cache supported streaming, thus the rebuffering may happen repeatedly, resulting a large rebuffering ratio (see Section 6.3).

Fast Cache supported streaming also decreases the possibility of encountering network congestion by reducing the data transmission time. Figure 7(b) shows the distribution of data transmission duration for Fast Cache supported streaming sessions and normal TCP streaming sessions, respectively. We can see that although in general the media objects streamed with Fast Cache have higher file lengths and encoding rates as shown in Figure 5, the transmission time of Fast Cache supported streaming is much shorter than that of normal TCP streaming (note that the x-axis is in log scale).

5.2 Fast Cache produces extra traffic

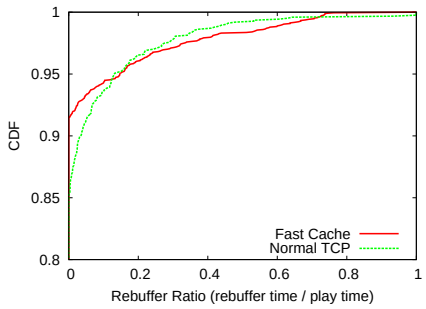
Fast Cache continuously delivers a media object at a rate higher than (usually up to five times) its encoding rate. In reality, the entire media object can be delivered completely to the user in the middle of the user's playback. If the user stops in the middle, the pre-arrived data for the remaining part are wasted. Considering the well known fact that most sessions only access the initial part of a video object [30], the extra delivered traffic by Fast Cache would be non-trivial, especially for large media objects such as movies.

In this study, we compute the over-supplied traffic as follows. The packet size and the time that a packet should be rendered can be extracted directly from the RTP packet header. Based on the timestamps of `PLAY`, `PAUSE`, and `TEARDOWN` commands, we get the user playback duration for each session. Assuming a default five-second play-out buffer [1] on the client side, we compute the extra traffic for each session. Figure 8 shows the extra traffic caused by Fast Cache supported streaming and normal TCP streaming, for *all Windows streaming sessions* in the home user workload. On average, Fast Cache over-supplies about 54.8% of media data to clients due to clients' early terminations, while the over-supplied traffic is only about 4.6% for normal TCP streaming sessions. Figure 9 shows the CDF of the average transmission speed (in multiples of the media encoding rate) of Fast Cache supported and normal TCP streaming, respectively. In the computation of the average tranmission speed, we only consider the data transmission after the Fast Start buffering period. The high data transmission speed in Fast Cache supported streaming indicates the reason for the over-supplied data.

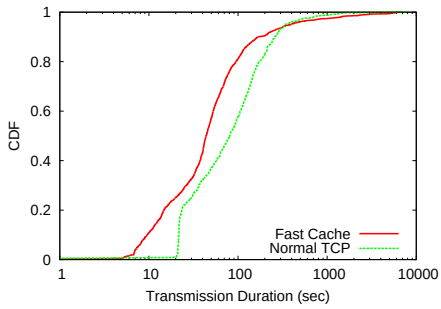
5.3 Server response time of Fast Cache

In addition to over-utilizing the network bandwidth, by transmitting media at a rate much higher than its encoding rate, a streaming server running Fast Cache may also consume more CPU, memory, disk I/O, and other resources

[4]The 2 Gbps client advertised bandwidth corresponds to the player's connection speed setting "10 Mbps and above".



(a) Rebuffering ratio



(b) Data transmission duration

Figure 7: Bandwidth fluctuation smoothed by Fast Cache

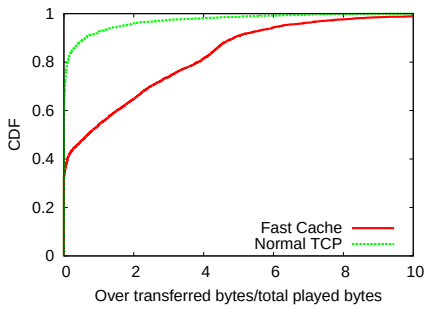


Figure 8: Extra traffic produced by Fast Cache

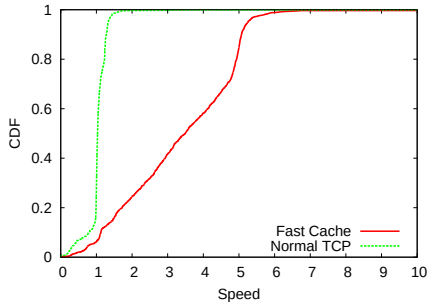
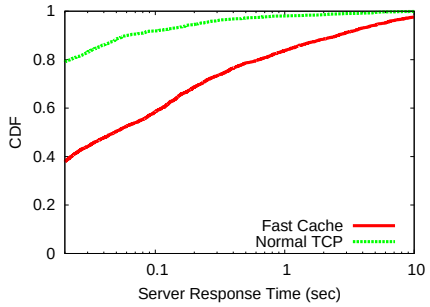
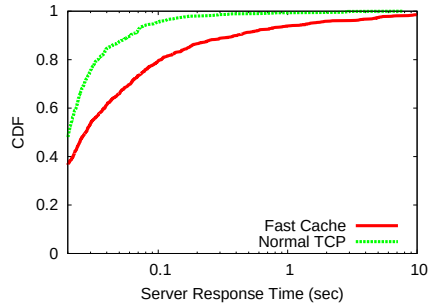


Figure 9: Average data transmission speed



(a) Third-party media service



(b) Self-hosting media service

Figure 10: The server response time to streaming requests

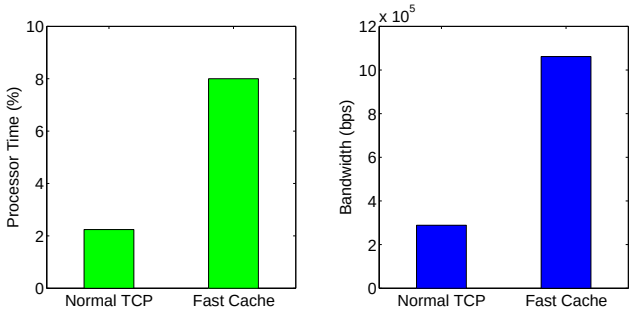
than a streaming server not running Fast Cache. As a result, a request may have to wait for a longer time to be served when a burst of requests arrive at the server. We define the *server response time* of a RTSP/MMS request to be the duration from the time instant when a server receives the first command from a client, to the time instant when the server sends its reply. Since our workloads are collected by Gigascope at a site very close to end users, the timestamp of a captured packet can be regarded as the time instant when the client sends or receives that packet. We use the timestamps of TCP handshake packets to estimate the packet round trip time (RTT), and then compute the server response time.

Figure 10(a) shows the distribution of the server response time for streaming requests served by third-party media services. We compare the response time of requests to servers with and without running Fast Cache. For servers running Fast Cache, about 43% of the requests have a response time longer than 0.1 second, while for servers not running Fast Cache, only about 9% of the requests have a response time longer than 0.1 second. Figure 10(b) shows the corresponding distribution of the server response time for streaming requests served by self-hosting media services. For servers running Fast Cache, about 21% of the requests have a response time longer than 0.1 second, while for servers not running Fast Cache, only about 5% of the requests have a response time longer than 0.1 second. These results indicate that the response time on servers running Fast Cache is statistically longer than that on servers not running Fast Cache. We also observe that the response time of the third-party hosting service is larger than that of the self-hosting service. As a commercial company, a third-party hosting service may want to fully utilize its server resources with many service subscribers. In contrast, self-hosting services are more dedicated and thus are often less heavily loaded.

5.4 Server load of Fast Cache

To further investigate the system resources consumed by Fast Cache, we conducted experiments with the Windows server 2003 and the Windows media load simulator [8]. We ran Windows server 2003 on a machine with 2 GHz Pentium-4 CPU and 512 MB memory, and ran a Windows media load simulator on a Windows XP client machine. The server machine and the client machine are connected through a 100 Mbps fast Ethernet switch. We generated two streaming video files with the Windows media encoder, one is encoded with 282 Kbps and the other is encoded with 1.128 Mbps, both of which have a 20-minute playback duration. We duplicated each file with 50 copies, and saved each copy with a different name. We first ran 50 normal TCP streaming sessions for 5 minutes using the simulator, each of which requests a different copy of the 282 Kbps video file simultaneously. Since the simulator does not support Fast Cache, we ran 50 normal TCP streaming sessions requesting the 1.128 Mbps video using the simulator, in order to simulate the streaming with Fast Cache support for the 282 Kbps video (with 4 speed). This experiment was also conducted for 5 minutes, with each session requesting a different file copy simultaneously. In each experiment, the simulator recorded the CPU and memory usage reported by the server every second. The average bandwidth usage was recorded in the server logs. We repeated each experiment 10 times.

Figure 11 shows the average usage of CPU and bandwidth of the server over the entire duration of the simulation (the



(a) CPU consumption of all 50 sessions

(b) Average bandwidth consumption per session

Figure 11: Server load comparison between Fast Cache and normal TCP streaming

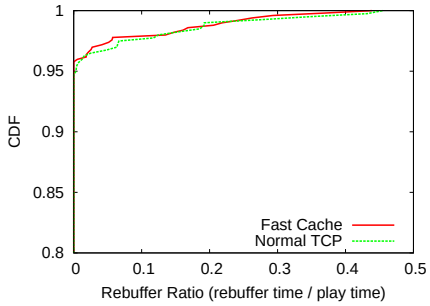


Figure 12: Effectiveness of resource over-utilization

memory usages of Fast Cache and normal TCP streaming are very close and thus are not presented). The bandwidth usage of Fast Cache is 3.67 times of that of the normal TCP streaming, while the CPU load of Fast Cache is 3.57 times of that of normal TCP streaming. This indicates that the CPU consumed by Fast Cache is approximately proportional to the streaming delivery rate. Given that Fast Cache could deliver a media object at a rate 5 times of its encoding rate, Fast Cache increases server load significantly, and thus limits the scalability of a streaming server. In our workloads, the Windows media servers in the second largest media delivery network and the largest self-hosting media service (a well known search engine site) do not support Fast Cache at all (we anonymize their domain names due to customer privacy concerns), which might be due to the concerns of high resource demands of Fast Cache.

5.5 Effectiveness of resource over-utilization

Fast Cache delivers a media object to a client faster than the playing speed by over-utilizing the bandwidth and CPU resources. However, streaming a media object at a rate higher than its encoding rate is only possible when the available bandwidth between a client and its server is large enough. Intuitively, when a media object is streamed at its encoding rate, the higher the average bandwidth between a client and its server over its encoding rate, the lower possibility at which performance degradation occurs during the playback. To understand whether Fast Cache performs better than normal TCP-based streaming when the average bandwidth between a client and its server is large enough, we plot the CDF of rebuffering ratio for Fast Cache based streaming sessions and normal TCP-based streaming sessions in the home user workload in which the media encoding rate of each stream is 200–320 Kbps and the client advertised bandwidth (extracted from the `Bandwidth` header) is at least 500 Kbps greater than the media encoding rate, as shown in Figure 12. Compared with Figure 7(a), the two curves in Figure 12 are very close, which means that, although temporary network congestion may occur from time to time, a small play-out buffer performs well enough to smooth out bandwidth fluctuation during streaming, when the average bandwidth is large enough. Thus, aggressively over-utilizing the server and Internet resources is neither performance-effective nor cost-efficient under a high bandwidth condition. The higher speed at which Fast Cache can stream a media object, the lower necessity is this speed for a client. Furthermore, even if no extra traffic generated (assume the media object is played completely in each session), the number of concurrent streams on a server is constrained by the streaming speed, and thus limits the server's capacity to service bursty requests.

6. RATE ADAPTATION

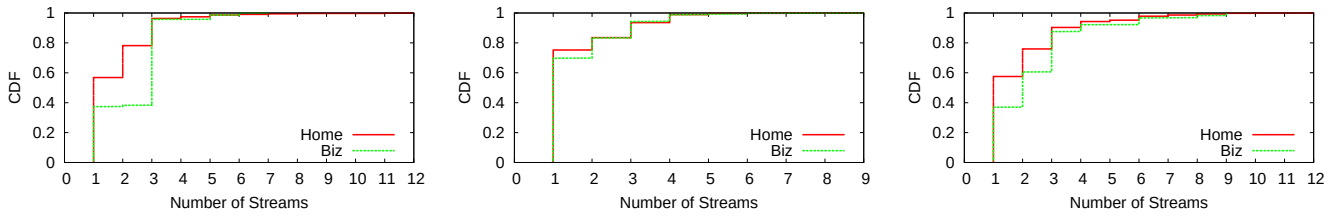
In order to adapt to bandwidth fluctuations, major media services such as Windows media and RealNetworks media support three kinds of techniques for rate adaptation. *Stream switch* enables a server to dynamically switch among streams with different encoding rates for the same object, based on the available network bandwidth. This technique is called *Intelligent Streaming* in the Windows media service [4] and *SureStream* in the RealNetworks media service [7]. *Stream thinning* enables a server to only send key frames to the client, when no lower bit rate stream is available. If the current bandwidth is not sufficient to transmit key frames, a server can only send audio to client, which is called *video cancellation*.

6.1 MBR encoding and stream switch

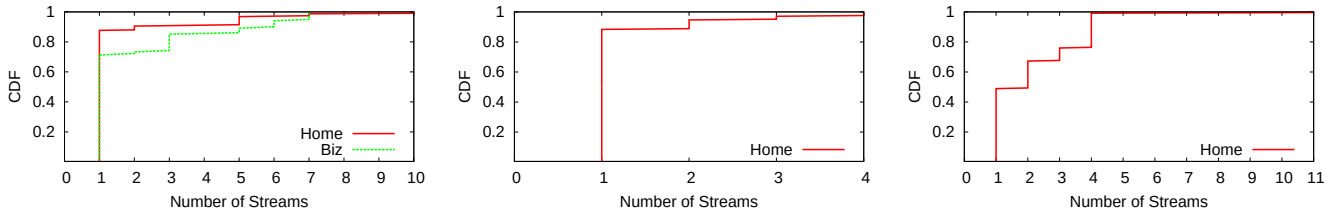
To enable stream switch, the media object must be encoded with *multiple bit rate*s (MBR): the encoder generates multiple streams with different bit rates for the same media content, and encapsulates all these streams together.

Figures 13(a), 13(b), 13(c) and Figures 13(d), 13(e), 13(f) show the distribution of the number of streams encoded in on-demand and live media objects in the home user and business user workloads, respectively. For video objects, we show the number of audio streams and video streams in a file separately (Figures 13(b), 13(c) for on-demand objects and Figures 13(e), 13(f) for live objects). Because there are only a small amount of live video objects in the business user workload, we do not present them in Figures 13(e), 13(f). These figures show that about 42% of the on-demand video objects in the home user workload are encoded with at least two video streams. The number of video streams in video objects is up to 12, and the number of video and audio streams together in a video object is up to 20. The number of streams in live audio objects is relatively small, but there are still 13% and 28% of the objects in home user and business user workloads encoded with at least two streams, respectively. These results indicate that the MBR encoding technique has been widely used in media authoring, which enables the rate adaptation—dynamically switching among streams based on the available bandwidth.

The stream switch in RTSP protocols works as follows (stream switch in MMS has a similar procedure). When a RTSP session is established upon a client request, the media

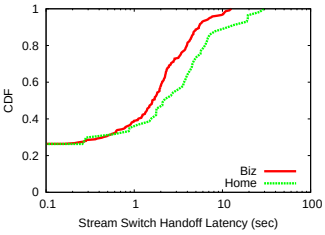


(a) On-demand audio objects (b) Audio in on-demand video objects (c) Video in on-demand video objects

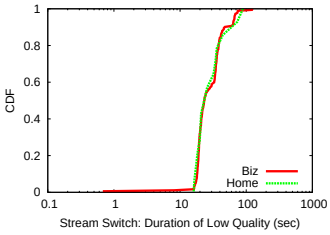


(d) Live audio objects (e) Audio in live video objects (f) Video in live video objects

Figure 13: MBR encoding in the home and business user workloads

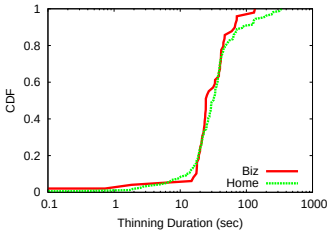


(a) Stream switch latency

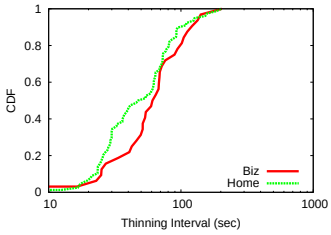


(b) Low quality duration

Figure 14: Stream switch



(a) Thinning duration



(b) Thinning interval

Figure 15: Stream thinning

player sends a `DESCRIBE` command to the server, asking for the description of the requested media object. In the reply to the `DESCRIBE`, the server sends the media description using SDP [19], including the description of each video/audio stream encapsulated in the media object. Then the client specifies the stream that it desires in the `SETUP` (Windows media service) or `SET_PARAMETER` (RealNetworks media service) command, based on its current available bandwidth. The server delivers the requested stream upon receiving the `PLAY` command from the client.

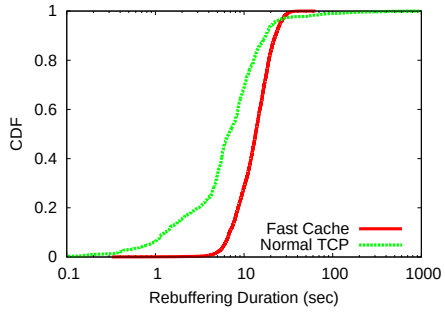
If during playback, the available bandwidth drops below the media encoding rate, the play-out buffer will be drained off. In this case, the media player may send a request to ask the server to switch to a lower rate stream. In Intelligent Streaming (Windows media service), the media player sends a `SET_PARAMETER` with a `SSEntry` message body via RTSP, specifying the current stream and the stream to switch to. In SureStream (RealNetworks media service), the client sends a `SET_PARAMETER` command with an `UnSubscribe` header to cancel the current stream and a `Subscribe` header to switch to the new stream.

We extracted all related information from RTSP/MMS commands, and analyzed these stream switches. To characterize the overhead and frequency of stream switches, we define the *switch latency* as the freezing duration between the end of the old stream and the beginning of the new stream, during which a user has to wait for buffering. We also define the *low quality duration* of a streaming session as the total playback time of streams with lower rates (relative to the highest encoding rate that the content is transmitted in this session).

Assuming a five-second play-out buffer [1], Figure 14(a) and Figure 14(b) show the distribution of stream switch latency and low quality duration in the home and business user workloads, respectively. As shown in Figure 14(a), about 30%–40% of the stream switches have a switch latency greater than 3 seconds, and about 10%–20% of the stream switches have a switch latency greater than 5 seconds, which is non-trivial for end users. In Figure 14(b), we observe that about 60% of the sessions have a low quality duration less than 30 seconds, and 85% of the low quality stream durations are shorter than 40 seconds.

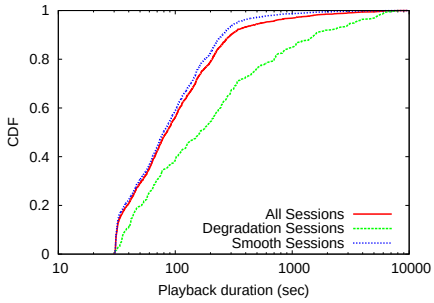
6.2 Stream thinning and video cancellation

Stream thinning works in a similar way as stream switch. To characterize the quality degradation due to stream thinning, we define the *thinning duration* as the time duration from the "thinning" command to the "un-thinning" command or the "stop playing" command, which reflects the quality degradation time that a user suffers. We also define the *thinning interval* as the interval between two consecutive stream thinning events, which reflects the frequency of such quality degradations. Figure 15(a) and Figure 15(b) show the thinning duration and the interval for video sessions longer than 30 seconds in the home and business user workloads, respectively. As shown in Figure 15(a), more than 70% of the thinning durations are shorter than 30 seconds. Figure 15(b) shows most (70% in the home user workload

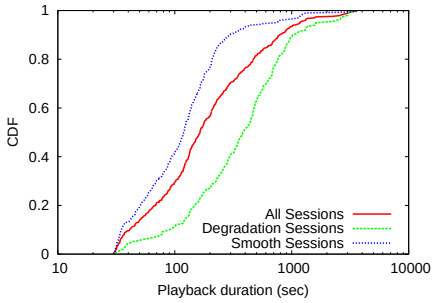


(Home user workload only)

Figure 16: Rebuffering duration for Fast Cache and normal TCP streaming sessions with rebuffering



(a) Home user workload



(b) Business user workload

Figure 17: Comparison of playback durations for sessions with and without quality degradations

Table 5: Summary of streaming quality

streaming quality	duration > 30 sec		duration > 300 sec	
	home	biz	home	biz
smooth playback	87.06%	59.96%	56.82%	19.73%
rebuffering only	8.65%	18.91%	32.87%	31.97%
stream switch	0.83%	16.30%	1.40%	37.42%
stream thinning	1.94%	3.42%	4.73%	6.80%
video cancellation	1.52%	1.41%	4.18%	4.08%

and 82% in the business user workload) thinning intervals are longer than 30 seconds.

When bandwidth is too low to transmit the key frame of video stream, the client may send a `TEARDOWN` command to cancel the video stream, then the server sends audio only. When the bandwidth increases, the client may set up and request the video stream again.

6.3 Summary of Internet streaming quality

According to our extensive trace analysis and real experiments, Fast Cache does not support rate adaptation in practice. In a streaming session with Fast Cache enabled, the client never requests to switch streams after the initial stream selection in the `SETUP` command, even if there is a more suitable stream matching the decreased/increased bandwidth during playback. Thinning and video cancellation are also disabled when Fast Cache is enabled. As a result, when the bandwidth drops below the encoding rate, Fast Cache supported streaming performs like *pseudo streaming* [17]: the player stops to buffer data for a while, then continues to play the media for about five seconds (the play-out buffer size), and this procedure repeats. With such a configuration, if a sudden network congestion happens and lasts for a long time, the streaming quality of Fast Cache supported streaming could be even worse than that of normal TCP streaming. Figure 16 shows that when rebuffering happens, the *rebuffering duration* of Fast Cache supported streaming is much longer than that of normal TCP streaming in the home user workload, because it cannot switch to a lower rate stream upon network congestion.

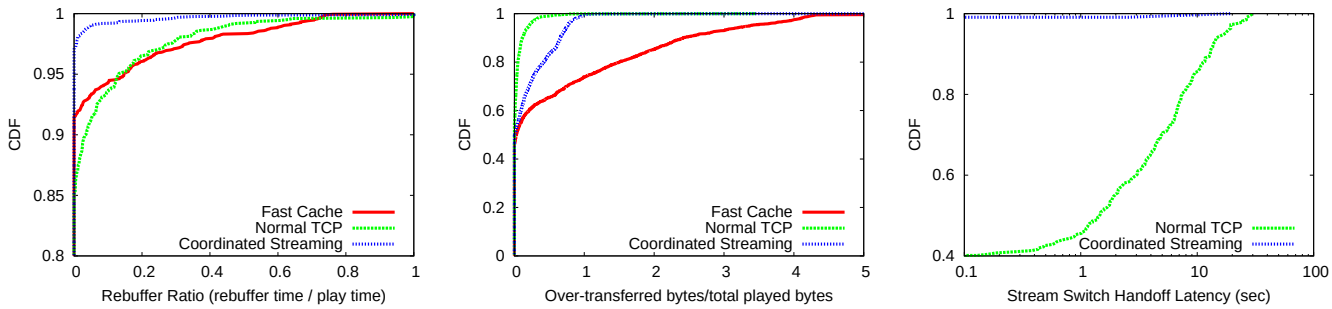
Figure 17(a) and Figure 17(b) show the CDF of playback duration of TCP-based video streaming sessions that are longer than 30 seconds in the home and business user workloads, respectively. The three curves in each figure denote all sessions, sessions without quality degradations, and sessions with quality degradations (including rebuffering, stream switch, stream thinning, and video cancellation), respectively. We can see that for sessions with longer durations, degradation happens with a higher probability. For example, in the business user workload, 88% of the sessions with quality degradations have a duration longer than 100 seconds, while 58% of the sessions without quality degradations have a duration longer than 100 seconds. Table 5 further shows the breakdowns of sessions with and without quality degradations for TCP-based video streaming sessions that are longer than 30 seconds and longer than 300 seconds, in the home and business user workloads, respectively. We can see that quality degradation happens less frequently in the home user workload than in the business user workload, which may be due to the longer playback duration of business users as shown in Figure 17. For sessions longer than 30 seconds, 13%–40% of the video sessions still have quality degradation due to the rebuffering, stream switch, stream thinning, and video cancellation. For sessions longer than 300 seconds, the quality is getting worse. Further investigation shows that in a significant amount of video sessions with rebuffering, the requested media objects are MBR encoded, and the lack of stream switch is largely due to the usage of Fast Cache, which disables rate adaptation.

In conclusion, the quality of media streaming on the Internet leaves much to be improved, especially for those sessions with longer durations.

7. DISCUSSION: COORDINATING CACHING AND RATE ADAPTATION

Fast Cache and rate adaptation are two commonly and practically used techniques that improve the experience of streaming media users from different perspectives. Fast Cache *aggressively* buffers media data in advance at a rate higher than the media encoding rate, aiming to absorb the streaming jitter due to network congestion. In contrast, rate adaptation *conservatively* switches to a lower bit rate stream upon network congestion. As shown in our analysis, both techniques have their merits and limits. Fast Cache has its problems such as increasing server load and producing extra traffic. On the other hand, the latency of stream switch is non-trivial in most sessions, due to the small size of play-out buffer.

Combining the merits of both techniques, in this section, we discuss *Coordinated Streaming*, a mechanism that coordinates caching and rate adaptation. In this scheme, an



(a) Rebuffering ratio (b) Over-supplied data (c) Switch handoff latency

Figure 18: Evaluation of Coordinated Streaming

upper bound and a lower bound are applied to the play-out buffer of the client player. The upper bound setting prevents aggressive data buffering while the lower bound setting eliminates the stream switch latency. When a streaming session starts, the server transmits data to the client as fast as possible until the lower bound is reached. Then the playback begins and the client continues to buffer data with the highest possible rate until the buffer reaches its upper bound. With a full buffer, the client buffers data at the media encoding rate, and the buffer is kept full. When network congestion occurs, the client may receive data at a rate lower than the object encoding rate, and the buffer is drained off. If the network bandwidth increases before the buffer drops below its lower bound, the client will request data at a higher rate to fill the buffer. Otherwise, the client will switch to a lower rate stream. The selection of the lower rate stream should be based on the following: in a typical bandwidth fluctuation period, the current bandwidth should be able to maintain normal playback of this lower rate stream and transmit extra data to fill the buffer to its upper bound. When the network bandwidth is increased, the client may switch to a higher encoding rate stream.

We conducted an ideal experiment as a proof of concept of this scheme. We set the lower bound of the buffer size as 5 seconds to cover the normal stream switch latency as well as the initial buffering duration, as the default play-out buffer size is 5 seconds, and the average stream switch latency is also about 5 seconds. The upper bound of the buffer size is set to 30 seconds, considering the typical network fluctuation periods that may affect streaming quality, such as low quality duration, thinning duration, and thinning interval (Figures 14(b), 15(a), and 15(b)). In a practical system, the lower and upper bound of buffer size should be adaptively tunable based on these quality degradation events. However, we will show that even with the above simple configuration, the streaming quality can be effectively improved and the over-supplied traffic can be significantly reduced.

We simulated the Coordinated Streaming scheme based on the packet level information of Fast Cache supported streaming sessions, and compared the quality and bandwidth usage of this scheme with that of Fast Cache supported streaming and normal TCP-based streaming. To have a fair comparison, we only consider video sessions that request objects with 200–400 Kbps encoding rates for a duration longer than 30 seconds in the home user workload. Figure 18(a) shows the rebuffering ratio in Fast Cache supported streaming, normal TCP streaming, and Coordinated Streaming. As shown in this figure, the rebuffering ratio of Coordinated Streaming is close to zero. The fraction of normal TCP streaming sessions with large rebuffering ratios is close to or even smaller than that of Fast Cache supported streaming sessions because many of them use rate adaptation to avoid rebuffering. Figure 18(b) shows the over-transfered data in the above three schemes, and we can see that Coordinated Streaming reduces 77% over-supplied traffic produced by Fast Cache, although not as good as normal TCP streaming. Figure 18(c) shows that the switch handoff latency of Coordinated Streaming is nearly zero, much less than that of normal TCP streaming. Furthermore, the number of stream switches in our scheme is only 33.4% of that in normal TCP-based streaming.

8. RELATED WORK

Existing measurement studies have analyzed the Internet streaming traffic in different environments and from different perspectives. Li et al. [20] characterized the streaming media stored on the Web, while Mena et al. [21] and Wang et al. [29] presented an empirical study of Real audio and Real video traffic on the Internet, respectively. These studies characterized the packet size, data rate, and frame rate patterns of streaming media objects. Almeida et al. [10] and Chesire et al. [13] studied the client session duration, object popularities and sharing patterns based on the workload collected from an educational media server and an university campus network, respectively. Cherkasova et al. [12] characterized the locality, evolution, and life span of accesses in enterprise media workloads. Yu et al. [31] studied the user behavior of large scale video-on-demand systems. Padhye et al. [23] and Costa et al. [15] characterized the client interactivity in educational and entertainment media sites, while Guo et al. [18] analyzed the delay of jump accesses for video playing on the Internet. Live streaming media workloads have also been studied in recent years. Veloso et al. [27] characterized a live streaming media workload in three increasingly granular levels, named clients, sessions, and transfers. Sripanidkulchai et al. [26] analyzed a live streaming workload in a large content delivery network.

However, these on-demand and live streaming media measurements mainly concentrated on the characterization of media content, access pattern, and user activities, etc. So far, few studies have focused on the mechanism, quality, and resource utilization of streaming media delivery on the Internet. Chung et al. [14] and Nichols et al. [22] conducted an experimental study in a lab environment on the responsiveness of RealNetworks and Windows streaming media, respectively. Wang et al. [28] proposed a model to study the

TCP-based streaming. In contrast to these studies, we analyzed the delivery quality and resource utilization of streaming techniques based on a large scale Internet streaming media workload.

9. CONCLUSION

In this study, we have collected a 12-day streaming media workload from a large ISP, including both live and on-demand streaming for both audio and video media. We have characterized the streaming traffic requested by different user communities (home users and business users), served by different hosting services (third-party hosting and self-hosting). We have further analyzed several commonly used techniques in modern streaming media services, including protocol rollover, Fast Streaming, MBR, and rate adaptation. Our analysis shows that with these techniques, current streaming services tend to over-utilize the CPU and bandwidth resources to provide better services to end users, which may not be a desirable and effective way to improve the quality of streaming media delivery. A coordination mechanism that combines the advantages of both Fast Streaming and rate adaptation techniques is proposed to effectively utilize the server and Internet resources for building a high quality streaming service. Our trace-driven simulation study demonstrates its effectiveness.

Acknowledgments

We thank the appreciations and constructive comments from the anonymous referees. William Bynum and Matti Hiltunen made helpful suggestions on an early draft of this paper. This work is partially supported by the National Science Foundation under grants CNS-0405909 and CNS-0509054/0509061.

10. REFERENCES

[1] Buffer settings in Windows media player. http://support.microsoft.com/?scid=kb;en-us;q257535.

[2] Fast Streaming with Windows Media 9 Series. http://www.microsoft.com/.

[3] HTTP streaming protocol. http://sdp.ppona.com.

[4] Microsoft Windows media - Intelligent Streaming. http://www.microsoft.com/.

[5] MMS streaming protocol. http://sdp.ppona.com.

[6] Real data transport (RDT). http://protocol.helixcommunity.org/.

[7] RealProducer 10 user guide. http://www.real.com/.

[8] Windows media load simulator. http://www.microsoft.com/.

[9] YouTube - broadcast yourself. http://www.youtube.com/.

[10] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon. Analysis of educational media server workloads. In *Proc. of ACM NOSSDAV*, June 2001.

[11] S. Chen, B. Shen, S. Wee, and X. Zhang. Designs of high quality streaming proxy systems. In *Proc. of IEEE INFOCOM*, Mar. 2004.

[12] L. Cherkasova and M. Gupta. Characterizing locality, evolution, and life span of accesses in enterprise media server workloads. In *Proc. of ACM NOSSDAV*, May 2002.

[13] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming media workload. In *Proc. of USITS*, Mar. 2001.

[14] J. Chung, M. Claypool, and Y. Zhu. Measurement of the congestion responsiveness of RealPlayer streaming video over UDP. In *Proc. of the Packet Video Workshop*, Apr. 2003.

[15] C. Costa, I. Cunha, A. Borges, C. Ramos, M. Rocha, J. Almeida, and B. Ribeiro-Neto. Analyzing client interactivity in streaming media. In *Proc. of WWW*, May 2004.

[16] C. Cranor, T. Johnson, and O. Spatscheck. Gigascope: a stream database for network applications. In *Proc. of ACM SIGMOD*, June 2003.

[17] L. Guo, S. Chen, Z. Xiao, and X. Zhang. Analysis of multimedia workloads with implications for Internet streaming. In *Proc. of WWW*, May 2005.

[18] L. Guo, S. Chen, Z. Xiao, and X. Zhang. DISC: Dynamic interleaved segment caching for interactive streaming. In *Proc. of IEEE ICDCS*, June 2005.

[19] M. Handley and V. Jacobsen. SDP: Session description protocol. RFC 2327, Apr. 1998.

[20] M. Li, M. Claypool, R. Kinicki, and J. Nichols. Characteristics of streaming media stored on the Web. Nov. 2005.

[21] A. Mena and J. Heidemann. An empirical study of Real audio traffic. In *Proc. of IEEE INFOCOM*, Mar. 2000.

[22] J. Nichols, M. Claypool, R. Kinicki, and M. Li. Measurements of the congestion responsiveness of Windows streaming media. In *Proc. of ACM NOSSDAV*, June 2004.

[23] J. Padhye and J. Kurose. An empirical study of client interactions with a continuous media courseware server. In *Proc. of ACM NOSSDAV*, July 1998.

[24] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 1889, Jan. 1996.

[25] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (RTSP). RFC 2326, Apr. 1998.

[26] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the Internet. In *Proc. of ACM SIGCOMM IMC*, Oct. 2004.

[27] E. Veloso, V. Almeida, W. Meira, A. Bestravos, and S. Jin. A hierarchical characterization of a live streaming media workload. *IEEE/ACM Transactions on Networking*, Sept. 2004.

[28] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. Multimedia streaming via TCP: An analytic performance study. In *Proc. of ACM Multimedia*, Oct. 2004.

[29] Y. Wang, M. Claypool, and Z. Zuo. An empirical study of RealVideo performance across the Internet. In *Proc. of the ACM SIGCOMM IMW*, Nov. 2001.

[30] K. Wu, P. S. Yu, and J. Wolf. Segment-based proxy caching of multimedia streams. In *Proc. of WWW*, May 2001.

[31] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large scale video-on -demand systems. In *Proc. of EuroSys*, Apr. 2006.