Analysis of Multimedia Workloads with Implications for Internet Streaming

Lei Guo[1], Songqing Chen[2], Zhen Xiao[3] and Xiaodong Zhang[1]

[1]Department of Computer Science
College of William and Mary
Williamsburg, VA 23187, USA
{lguo, zhang}@cs.wm.edu

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

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

ABSTRACT

In this paper, we study the media workload collected from a large number of commercial Web sites hosted by a major ISP and that collected from a large group of home users connected to the Internet via a well-known cable company. Some of our key findings are: (1) Surprisingly, the majority of media contents are still delivered via downloading from Web servers. (2) A substantial percentage of media downloading connections are aborted before completion due to the long waiting time. (3) A hybrid approach, *pseudo streaming*, is used by clients to imitate real streaming. (4) The mismatch between the downloading rate and the client playback speed in pseudo streaming is common, which either causes frequent playback delays to the clients, or unnecessary traffic to the Internet. (5) Compared with streaming, downloading and pseudo streaming are neither bandwidth efficient nor performance effective. To address this problem, we propose the design of AutoStream, an innovative system that can provide additional previewing and streaming services automatically for media objects hosted on standard Web sites in server farms at the client's will.

Categories and Subject Descriptors

C.2.5 [**Computer Communication Networks**]: Local and Wide Area Networks—*Internet*; C.4 [**Performance of Systems**]: Performance Attributes

General Terms

Measurement, Performance, Design

Keywords

Network measurements, Traffic analysis, Multimedia, Streaming, System design

1. INTRODUCTION

The past decade saw the evolution of Internet content from mostly text and images to increasingly more multimedia objects such as audio and video [11, 14]. Three methods are commonly used to deliver multimedia content on the Internet, namely, *downloading*, *pseudo streaming*, and *streaming*. Initially, multimedia objects were distributed in the same way as non-media objects: a client downloads an audio or a video clip from the Web server. This approach is easy to implement and requires no change to the Web server. The drawback is that the client has to finish downloading the entire object before it can start playing the media. This can incur a long startup latency for large media objects or for clients who have limited bandwidth to the Internet (e.g., dial-up clients). Moreover, if a client decides that the content of a large media object is not interesting after playing for a few seconds, most of the traffic in downloading this object is simply wasted.

Pseudo streaming is another delivery method for multimedia objects. It has the same nature of downloading, but provides an option on the client side to play the object while it is being downloaded. Most media players, such as Windows Media Player and Real Player, support the pseudo streaming mechanism. A major limit of pseudo streaming appears when the downloading connection is slow and cannot catch up with the playback speed. In this case, a client has to stop frequently to wait for new data.

To address the deficiency of downloading and pseudo streaming, researchers have developed streaming as the most efficient technique for delivering multimedia content. With streaming, the playback of a media object can start shortly after the client receives the initial portion of the object from the streaming server. In addition, streaming provides clients with a variety of controls during playback, such as pause, rewind, jump, etc. This allows a client to start or stop the media stream easily at any time. Compared with downloading and pseudo streaming, the amount of data transferred in streaming is closest to what the client really needs. Because of its many advantages, streaming is used in various Internet applications today.

There have been some previous studies on the characteristics of multimedia workloads. They typically either analyze the media trace of a small number of streaming/Web servers or study the behaviors of clients in educational [11, 6] or enterprise environments [10]. Few of them have focused on the overall distribution of multimedia traffic from both the server side and the client side. For example, it is not clear what percentage of Internet media traffic is delivered via downloading versus via streaming. As we discussed above, different delivery methods can have profound impact

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.

on playback quality and bandwidth efficiency. Similarly, it would be useful to quantify the benefits of a streaming service to a group of Web servers. Unfortunately, these kinds of questions have not been answered by the existing literature.

In this paper, we study the media workload collected from a large number of commercial Web sites hosted by a major ISP (a Web server farm) and the media workload from a large group of home users connected to the Internet via a well-known cable company (cable clients). These two workloads are independent to each other, which represent the workload of media requests from the whole Internet to our server farm and the workload of media requests from our cable network to the servers of the whole Internet, respectively. To the best of our knowledge, this is the first comprehensive study on the three media delivery methods from both the client side and the server side. In particular, we quantify the potential traffic waste in downloading and pseudo streaming by using the amount of data transferred in streaming as a benchmark. We also study the quality of service issues in pseudo streaming systematically. A detailed description of our findings will appear later in the paper, but here are some highlights:

- Surprisingly, the majority of media content is still delivered via downloading from Web servers.
- A substantial percentage of downloading connections for media objects were aborted before completion due to the long waiting time and low patience of clients, resulting up to 20% pure bandwidth waste.
- Compared with downloading, clients using pseudo streaming tend to abort more connections and abort earlier due to the early feedback of the media content, and result in less bandwidth waste.
- The mismatch between the downloading rate and the client playback speed in pseudo streaming is common, which either causes frequent playback delays to the clients, or unnecessary traffic to the Internet.
- Compared with streaming, downloading and pseudo streaming are neither bandwidth efficient nor performance effective.

Our findings indicate that inadequate streaming support has caused many clients to suffer from poor quality of service and has resulted in a significant waste of network bandwidth. To address this problem, we propose the design of AutoStream, an innovative system that can provide streaming service automatically for media objects hosted on standard Web servers. AutoStream also supports a *preview* function that allows a client to decide if an object is worth downloading after viewing its beginning portion. Trace driven simulations demonstrate that AutoStream has the potential of substantially improving the quality of service to the clients while reducing network bandwidth consumption significantly.

The rest of the paper is organized as follows. Section 2 presents our methodology for trace collection and processing. Sections 3 and 4 give first an overview and then a detailed analysis of the workloads. We propose and evaluate the design of the AutoStream system in section 5. Section 6 discusses related work and Section 7 concludes the paper.

2. TRACE COLLECTION AND PROCESSING METHODOLOGY

We collected our server farm multimedia workload and cable client multimedia workload using the Gigascope appliance [13]. In the following context, we simply refer to these terms as the *server workload/trace* and the *client workload/trace*. Each workload in this study covers a 24-hour time period, from 2004-06-15 20:00:00 to 2004-06-16 20:00:00. Downloading and pseudo streaming workload are HTTP based. The most commonly used streaming protocols are RTP [22]/RTSP [23] and MMS (proprietary to Microsoft) [1] running on TCP for media control messages and UDP for data transmission (if UDP is disabled, TCP can be used, and thus the data and control message transmission can share a channel). In our trace collection, we collected the first IP packets of all HTTP requests and responses, and the first IP packets of all RTSP and MMS control messages. We also collected the amount of bytes transferred through each TCP/UDP connection per second, in order to analyze the traffic flows in second level. All HTTP based peer-to-peer traffic was carefully filtered out. There are a total of 1,095,984 media requests in the server workload, and 579,693 media requests in the client workload. The total data size is about 100.36 GB before the processing in compressed format (gzip). There are a total of 4,498 unique server IPs and a total of 79,309 unique client IPs involved in the server workload, while there are a total of 13,110 unique server IPs and a total of 7,906 unique client IPs involved in the client workload.

2.1 Session and Traffic Matching

In both workloads, we first carefully removed the retransmissions of the first IP packets according to the sequence numbers in TCP headers. The processed workloads contain interleaved HTTP, RTSP, and MMS messages. We match messages for sessions as follows: for HTTP sessions, each session includes an HTTP request and response pair. We match all HTTP requests and responses based on the timestamp, source IP and port, and destination IP and port of each IP packet. We also consider the order of different messages to avoid mismatching. About 97.36% HTTP requests and responses can be matched in our client trace. For the server trace, about 56.72% HTTP requests and responses can be matched due to the different routing paths of requests and corresponding responses [7]. For RTSP and MMS sessions, we group RTSP/MMS messages according to the corresponding TCP connections first, and distinguish different sessions in the same TCP connection based on the RTSP session ID and MMS client ID, respectively.

Then we matched the real transferred traffic for each media access session. Since there may be multiple HTTP sessions in a single TCP connection, we matched the data traffic in each connection for all HTTP objects according to the timestamp and delivering order. For RTSP and MMS sessions, we parsed/decoded messages to get the protocol and port number for data transmission, and then matched the corresponding TCP/UDP traffic. Finally, we removed all unsuccessfully matched HTTP sessions, all non-downloading HTTP sessions such as sessions with a HTTP `304` response (`304` is a Web server response code indicating that the sta-

[1]There is also a HTTP based streaming protocol, but the amount of traffic is too trivial so we ignore it in this study.

tus of the being requested object is "Not Modified"), and all unsuccessfully matched RTSP/MMS sessions for our study.

2.2 Trace Processing

After traffic matching and exclusion, we divided each workload into the following three categories according to their accessing mechanisms.

2.2.1 Downloading Traffic

We identify media downloading sessions from all HTTP sessions as follows: the `User-Agent` field in the HTTP request is a Web browser, and the `Content-Type` field in the corresponding HTTP response is `audio` or `video`. Since there are other content types, such as `application` and `multipart`, which may actually represent a media object, we further use the file name suffix (extension) in the URL to identify whether it is a media access session. We have considered 34 most popular suffixes for different media file formats. After identifying a HTTP media session, we decoded the media information such as encoding rate and playback time from the file headers attached in the HTTP responses. We have decoded the most popular media formats, including Windows Media, Real Media, and QuickTime Media.

2.2.2 Pseudo Streaming Traffic

There are subtle differences between pseudo streaming and downloading, although they both use HTTP. Some pseudo streaming requests also try the HTTP based streaming first, then resort to pseudo streaming after failure. We have identified the pseudo streaming sessions as follows: the `User-Agent` in the request of a pseudo streaming session corresponds to a media player: `QuickTime` for QuickTime media player; `RMA` and `RealPlayer` for Real media player; `NSPlayer` and `Windows-Media-Player` for Windows media player; `Winamp` for Winamp mp3 media player, etc.. We decoded the media information in the same way as we did for downloaded media objects. Our experience through extensive trace analysis indicates that this is a reliable way to identify pseudo streaming traffic.

2.2.3 Streaming Traffic

Streaming sessions are based on RTSP/MMS protocol. For a streaming request, a sequence of control messages is exchanged between the client and the streaming server. For a standard RTSP based streaming session, the messages include *DESCRIBE*, *SETUP*, *PLAY*, and *TEARDOWN* in a normal playback. If there are client interactions, such as pause, fast forward and rewind, corresponding messages need to be exchanged on the spot. For MMS based streaming, the mechanism is similar, but the control messages are encoded in binary format. We sorted the control messages by timestamp in each RTSP/MMS session, and then parsed/decoded the control messages in each session. We extracted the URL, media encoding rate, and full playback time for the objects in each session, and matched the corresponding traffic in our trace.

3. TRAFFIC OVERVIEW

After initial trace processing, we have the requests for audio and video objects. In this section, we overview the prevalence of audio and video media formats, the media player prevalence for pseudo streaming, and the transport protocols used for real streaming.

Table 1: Audio Objects Overview in Two Workloads

	Audio Type	Request Number (%)	Requested Traffic (%)	Transferred Traffic (%)
Server Workload	MP3	9,696 (16.56)	24.00 (51.21)	9.36 (65.14)
	WAV	44,133 (75.36)	4.95 (10.57)	2.27 (15.77)
	RM	868 (1.48)	3.07 (6.56)	0.86 (6.01)
	WM	1,043 (1.78)	4.36 (9.31)	0.85 (5.96)
	AU	1,118 (1.91)	8.71 (18.58)	0.07 (0.46)
	Other	1,706 (2.91)	1.77 (3.77)	0.96 (6.66)
Client Workload	MP3	7,555 (24.28)	9.89 (17.15)	4.77 (43.04)
	WAV	1,662 (5.34)	0.71 (1.23)	0.21 (1.90)
	RM	1,186 (3.81)	8.23 (14.28)	1.32 (11.93)
	WM	4,507 (14.48)	17.76 (30.81)	3.28 (29.66)
	AU	13,726 (44.11)	19.82 (34.38)	0.39 (3.56)
	Other	2,484 (7.98)	1.24 (2.15)	1.10 (9.91)

3.1 Media Type Prevalence

We first summarize the properties and prevalence of different types of audio and video objects in our workloads. In both workloads, we classify different types of audio and video files based on the file and CODEC formats according to `Content-Type` in the HTTP responses and the object name suffix in the URLs.

Table 1 shows the prevalence of audio objects in terms of *Request Number*, *Requested Traffic*, and *Transferred Traffic*. *Requested Traffic* is computed according to the `Content-Length` field in the HTTP response or media length and encoding rate extracted from RTSP/MMS messages. *Transferred Traffic* shows the amount of data that was actually transferred. In this table, *MP3* represents the MPEG Audio Stream Layer III format files (with a `.mp3` suffix), WAV represents the Waveform Audio format files (with a `.wav` suffix). *RM* includes all audio files in the Real Media formats (with a suffix of `.ra`, `.rm`), while *WM* includes all audio files in the Window Media formats (with a suffix of `.wma, .wm, .asf`). *AU* represents UNIX-generated sound files (with a suffix `.au`). Note there are two numbers in each cell in the table. The first number is the absolute request number or the traffic amount in GB. The number in the parentheses represents its corresponding percentage in the workload.

As shown in Table 1, in terms of *Transferred Traffic*, *MP3* audio is the most popular in both workloads. In the server workload, *WAV* is the second popular audio while in the client workload, *WM* is the second popular audio. It is notable that in the client workload, a substantial number of small *AU* files exist, which only takes 3.56% of the *Transferred Traffic*. In both workloads, the number of *WM* requests is more than that of *RM*, indicating client access preferences.

Table 2 shows the video object properties in the two workloads. In this table, *MPEG* represents all MPEG-series video files (with a suffix of `.mpeg`, `.mpg`, `.mp`, `.mpeg2`, `.mp4`). *WM* indicates the video files of Windows Media formats (with a suffix of `.wmv`, `.wm`, `.asf`, `.svt`). *QT* represents the QuickTime format files (with a suffix of `.qt`, `.mov`), while *RM* represents Real Media video format files (with a suffix of `.rm`, `.rmvb`). *AVI* represents Audio Video Interleave File format files (with a suffix `.avi`).

Unlike the audio files, generally the *WM* and *MPEG* videos are the most popular in terms of transferred traffic in both workloads. *WM* takes the leading position in the

Table 2: Video Objects Overview in Two Workloads

	Video Type	Request Number (%)	Requested Traffic (%)	Transferred Traffic (%)
Server Workload	WM	5,620 (54.46)	32.95 (58.69)	8.78 (51.08)
	MPEG	2,862 (27.74)	12.19 (21.70)	4.34 (25.21)
	QT	1,063 (10.3)	6.32 (11.26)	2.80 (16.29)
	RM	309 (2.99)	2.39 (4.26)	0.55 (3.18)
	AVI	458 (4.44)	2.25 (4.01)	0.72 (4.17)
	Other	7 (0.07)	0.05 (0.08)	0.01 (0.06)
Client Workload	WM	20,316 (34.63)	93.11 (48.63)	27.36 (40.01)
	MPEG	33,229 (56.65)	61.97 (32.37)	27.78 (40.63)
	QT	2,046 (3.49)	12.31 (6.43)	7.05 (10.31)
	RM	1,923 (3.28)	17.42 (9.10)	3.37 (4.94)
	AVI	1,035 (1.76)	6.28 (3.28)	2.53 (3.70)
	Other	111 (0.19)	0.36 (0.19)	0.28 (0.41)

Table 3: Media player prevalence in pseudo streaming workloads

	Media Player	Request Number (%)	Requested Traffic (%)	Transferred Traffic (%)
Server Workload	WM	4,700 (70.81)	17.83 (57.88)	7.17 (81.57)
	RM	1,765 (26.59)	12.18 (39.53)	1.19 (13.61)
	QT	58 (0.87)	0.47 (1.55)	0.30 (3.43)
	Other	114 (1.73)	0.31 (1.04)	0.12 (1.39)
Client Workload	WM	20,179 (90.60)	43.44 (60.32)	14.11 (76.50)
	RM	1,483 (6.65)	25.65 (35.61)	1.97 (10.70)
	QT	498 (2.23)	2.76 (3.83)	2.24 (12.16)
	Other	112 (0.50)	0.16 (0.22)	0.11 (0.62)

server workload, while they are close in the client workload. QT follows them. These three are more popular than the others.

The above study sketches the prevalence of different audio and video types available on the Internet.

3.2 Media Players Supporting Pseudo Streaming

Pseudo streaming is supported by media players without streaming servers. Table 3 indicates the different prevalence of three major media players we identified through pseudo streaming in the server and client workloads, respectively. In the table, *WM* represents the *Windows-Media-Player*. *RM* represents *Real-Player*. *QT* represents the *QuickTime-Player*. *Winamp* and other players are in the *Other* category.

Table 3 shows that *WM* is the most popular media player to support pseudo streaming, in both the server workload and the client workload. The result is consistent with the audio and video file format prevalence study in Table 1 and Table 2, where a large portion of *WM* format media files are found.

3.3 Transport Protocols for Streaming

On the Internet, most real streaming is based on MMS or RTSP. The former is proprietary to Microsoft, while the latter is available in the public domain. Besides MMS, Microsoft also has begun to support RTSP.

Table 4 shows that in the server workload, although in terms of *Session Number* and *Requested Traffic*, MMS has a smaller share, the *Transferred Traffic* amount on MMS is more than that of RTSP. In the client workload, RTSP and MMS have similar shares in terms of *Session Number* and

Table 4: Streaming protocols overview

	Protocol	Session Number (%)	Requested Traffic (%)	Transferred Traffic (%)
Server Workload	RTSP	1,136 (62.04)	10.08 (58.74)	1.16 (40.41)
	MMS	695 (37.95)	7.08 (41.25)	1.71 (59.58)
Client Workload	RTSP	4,858 (51.56)	50.75 (60.64)	7.55 (50.97)
	MMS	4,564 (48.43)	32.94 (39.35)	7.26 (49.02)

Table 5: Overview of Multimedia Delivery in Two Workloads

(a) Server Workload

Delivery Method	Request Number (%)	Requested Traffic (%)	Transferred Traffic (%)
Downloading	60,415 (87.7)	55.02 (53.4)	19.89 (63.0)
Pseudo Streaming	6,637 (9.6)	30.81 (29.9)	8.79 (27.9)
Streaming	1,831 (2.7)	17.17 (16.7)	2.88 (9.1)

(b) Client Workload

Delivery Method	Request Number (%)	Requested Traffic (%)	Transferred Traffic (%)
Downloading	58,086 (64.7)	93.37 (37.5)	46.18 (58.1)
Pseudo Streaming	22,272 (24.8)	72.02 (28.9)	18.44 (23.2)
Streaming	9,422 (10.5)	83.69 (33.6)	14.81 (18.6)

Transferred Traffic. We also study the UDP based streaming traffic percentages. In the server workload, 10.4% and 23.5% RTSP and MMS streaming traffic are based on UDP. While in the client workload, 26.8% and 21.5% are based on UDP.

Having obtained observations about these two workloads in media types, media players, and transport protocols, we dissect them in more detail in the next section.

4. ANALYSIS OF WORKLOADS

In this section, we study the two workloads described in the previous sections and discuss the implications of our findings.

4.1 Most Multimedia Traffic is Delivered via Downloading

Given the clear advantage of streaming over downloading, one might expect streaming service to become widely available and that most multimedia traffic nowadays to be delivered via streaming. Our analysis of the two workloads, however, has revealed some surprising results.

Table 5 shows the breakdown of the three multimedia delivery methods after our initial trace processing. *Requested Traffic* and *Transferred Traffic* are calculated as in the previous section. Surprisingly, the table indicates that the majority of multimedia traffic is delivered by downloading or pseudo streaming. The amount of streaming traffic is only a small percentage of the actual transferred traffic (9.1% for the server workload and 18.6% for the client workload). This implies that, *many years after its initial introduction, streaming service still has limited availability.* This is probably due to its associated expenses and technology availability for public usage.

As shown in this table, the average object size per request for streaming, pseudo streaming, and downloading decreases in this order. Figure 1 shows the distributions of sessions requesting different sizes of media objects and their corre-

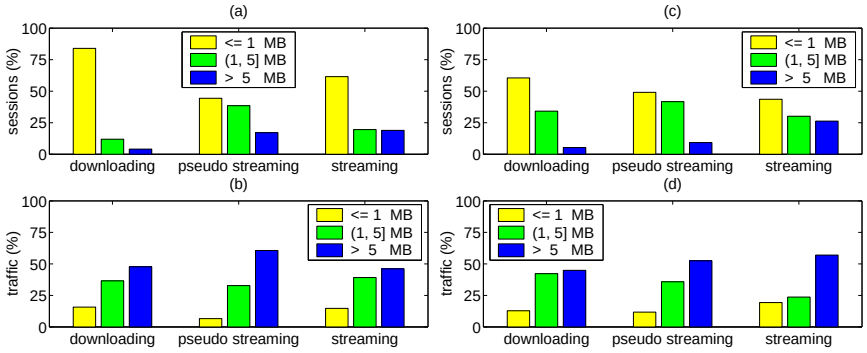

Figure 1: Media sessions and transferred traffic for different sizes of objects in the server workload (a, b) and the client workload (c, d)

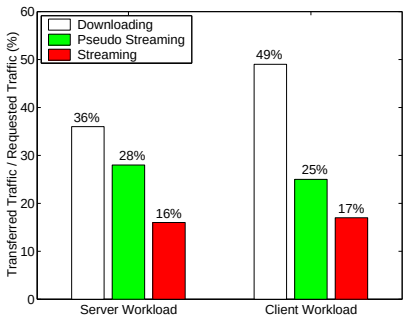

Figure 2: Bandwidth efficiency of 3 media delivery methods

sponding transferred traffic for the three mechanisms. Although there are many requests demanding small objects, the amount of their contributed traffic is small for each of the three mechanisms. In other words, media traffic is always dominated by large objects. Furthermore, as we shall see later, the downloading time of some large objects exceeds the client patience threshold, resulting in the early session termination before the downloading is completed.

4.2 Non-Streaming Delivery is Inefficient

Streaming reduces user perceived latency because a client can start playing a multimedia object shortly after it has received the initial portion of this object. Besides improving user experience, this kind of early feedback also allows the client to make a timely decision based on its interest to this object. If the object is not interesting, the client can terminate the session without consuming additional network bandwidth. In contrast, with downloading a client has to get the entire object before it can start the playback.

In this subsection, we quantify the bandwidth efficiency of the three multimedia delivery methods in Table 5 by computing the percentage of requested traffic that was actually transferred – the ratio of the last two columns in the table. The results are shown in Figure 2.

As can be seen from this figure, the percentage of traffic that was transferred is the highest for downloading and the lowest for streaming. Using the server workload as an example, downloading and pseudo streaming need to transfer 2.25 times and 1.75 times as much data as streaming, respectively. Given the large degree of control a client has during a streaming session, we believe that the amount of data transferred in streaming is a reasonable approximation of what is actually needed for a client playback. This implies that a large amount of traffic transferred via downloading and pseudo streaming is potentially unnecessary. For media objects smaller than 500 KB, the difference of bandwidth efficiency among the three mechanisms is relatively smaller due to the client side buffer for pseudo streaming and real streaming. A media player may buffer 0 - 30 seconds of media data to smooth the effect of end-to-end bandwidth fluctuation between the server and the client. For example, the default buffer size of Window Media Player is 5 seconds [2]. However, the amount of traffic affected by the client side buffer is small. This is because sessions requesting small media objects only account for a small amount of transfered traffic, even though the number of such sessions is

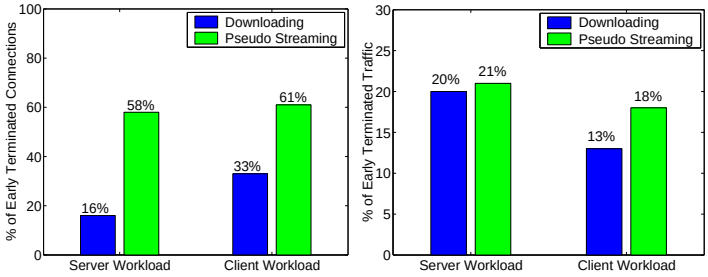

Figure 3: The percentage of early terminated connections (left) and the percentage of traffic contributed by early terminated connections (right)

large, as shown in Figure 1. In the following subsections, we only present media sessions requesting objects larger than 1 MB to minimize the effect of the client side buffer (the encoding rate of most streaming video is less than 250 Kbps. 250 Kbps $\times$ 30 sec = 937.5 KB $\approx$ 1 MB).

Hence, our conclusion is: *Streaming is the most efficient approach for multimedia delivery, while downloading is the least efficient one. Although pseudo streaming is an improvement over pure downloading, it can still generate a large amount of unnecessary network traffic.* We will revisit pseudo streaming later in this section.

4.3 Early Terminated Connections

Given the large percentage of multimedia traffic via downloading and pseudo streaming, we take a closer look at their characteristics in this subsection. In particular, we focus on connections that are terminated before downloading is completed. We call such connections *early terminated connections*. In downloading, the termination of a connection before the object is fully downloaded simply wastes network bandwidth: the partially downloaded object is not useful to the client. In contrast, in pseudo streaming, a client can play the media object while it is being downloaded. Hence, the traffic might not be totally wasted if the connection is terminated early.

We first study how many connections in our workloads are terminated early and the corresponding traffic contributed by these early terminated connections. Figure 3 shows the percentage of early terminated sessions and the percentage of corresponding traffic for the downloading workload and

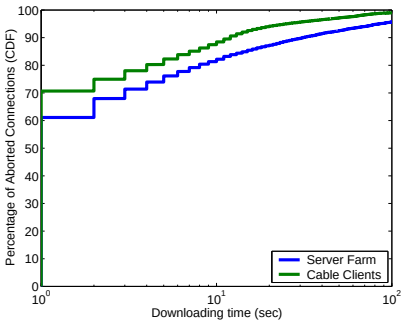

Figure 4: Downloading time in aborted connections

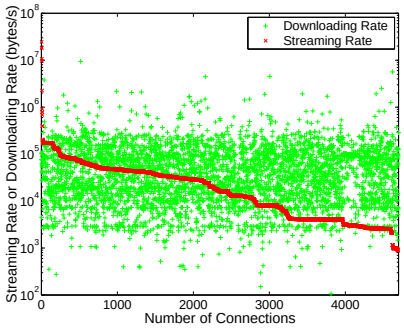

Figure 5: Comparison of average downloading and streaming rate in the server workload

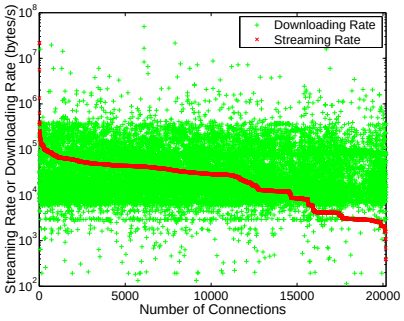

Figure 6: Comparison of average downloading and streaming rate in the client workload

pseudo streaming workload.

The left figure in Figure 3 shows that the early termination ratio for pseudo streaming is comparable between these two workloads and is significantly higher than that of downloading, indicating that by previewing the initial part of an object in pseudo streaming, a client can discard uninteresting objects earlier. This is also confirmed by the traffic figure on the right: the difference in traffic contributed by early terminated connections between downloading and pseudo streaming is much smaller than the difference in the percentage of such connections. Hence, if we amortize the traffic over the set of connections, clients using pseudo streaming clearly tend to abort more and cause less traffic.

Since aborted downloading connections waste up to 20% of overall bandwidth, it would be useful to find out the client abortion pattern so that a service provider can avoid providing inappropriately large files online for download. Figure 4 shows the downloading time of the aborted connections in our workloads.

In the server workload, 81% of the early terminated connections are terminated before 10 seconds, while only 8% wait more than 50 seconds before termination. In the client workload, 87% are terminated before 10 seconds, while 3% are terminated after 50 seconds. This suggests that downloading is efficient for small objects, but not appropriate for large objects, such as streaming media objects, as the client patience threshold is as small as 10 seconds.

From the server and client side downloading workload analysis, we have the following observations:

1. Most Internet clients are not patient. If a downloading cannot finish in 10 seconds, the possibility for the client to terminate the connection is very high.

2. In average, we observe that up to 20% downloading traffic comes from aborted connections, which wastes a significant amount of Internet bandwidth.

4.4 Delivery Quality of Pseudo Streaming is Inconsistent

As a special type of downloading, pseudo streaming can provide clients with limited streaming functionalities and is more efficient than downloading in network bandwidth. However, an inherent problem with pseudo streaming is how to match the downloading rate with the streaming rate required by the client for playback. In this subsection, we look into this issue in detail.

Figures 5 and 6 show a comparison of the average downloading rate with the streaming rate in the two workloads. The *downloading rate* is calculated by averaging the transferred bytes over the data transmission time. We extracted the object *encoding rate* for each media file and took it as the corresponding *streaming rate*. We can see roughly half of the sessions in the server workload do not have sufficient downloading bandwidth to keep up with the streaming rate. In contrast, more sessions in the client workload have sufficient bandwidth to do so. This is probably because the client workload consists of broadband users only, while the server workload has a mixture of broadband and dial-up users.

The mismatch between the streaming rate and the downloading rate can cause quality issues at the client side or unnecessary traffic on the Internet. When the downloading rate is smaller than the object streaming rate, the client perceives frequent playback delays until downloading is complete. When the object streaming rate is smaller than the downloading rate, part of the downloading traffic could be wasted if the client is not interested in the object and terminates the connection earlier (i.e. the later portion of the object may get transferred in vain). However, if a client stops playback after the entire object is downloaded, we do not know how much of the download traffic is actually used. Thus, we focus our study of rate mismatch only on early terminated connections.

Figure 7 shows the situation when the downloading rate is smaller than the streaming rate. We use the accumulated delay relative to the total downloading time as an indication of the playback quality. In the server workload, 71% early terminated sessions have an extra delay of more than 50% of their downloading times. This number is 72% for the client workload. Looking at the downloading time, we find for these early terminated connections in the server workload, 6%, 12% and 17% of them take more than 50, 20, and 10 seconds for downloading. In the client workload, a relative smaller percentages, 1.1%, 1.5% and 4.4%, take more than 50, 20, and 10 seconds for downloading, meaning cable clients tend to abort earlier.

Figure 8 shows the situation when the downloading rate is greater than the object streaming rate. To calculate the amount of wasted traffic, we multiply the rate difference between downloading and streaming by the transmission time of these early terminated sessions. This figure indicates that a large portion (72%) have more than 50% transferred traffic

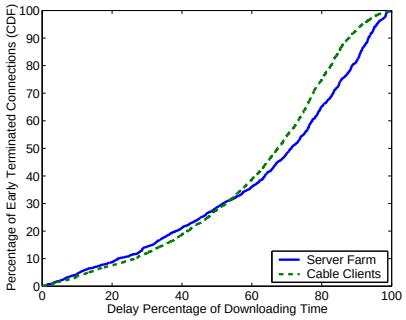

Figure 7: When transferring rate is smaller than streaming rate

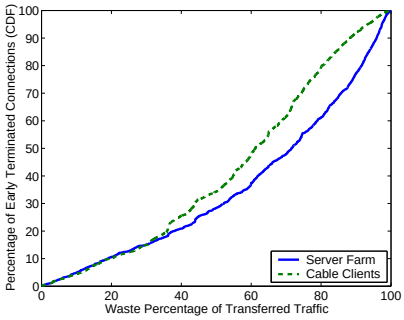

Figure 8: When transferring rate is larger than streaming rate

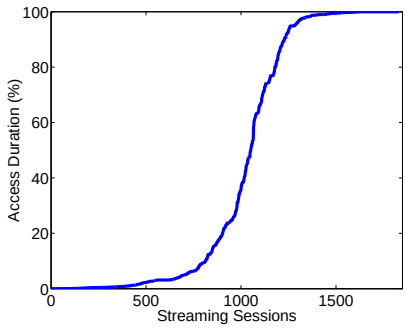

Figure 9: Client access duration in the server workload

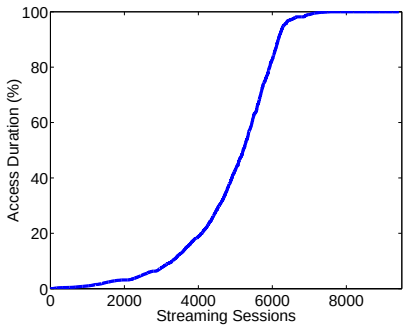

Figure 10: Client access duration in the client workload

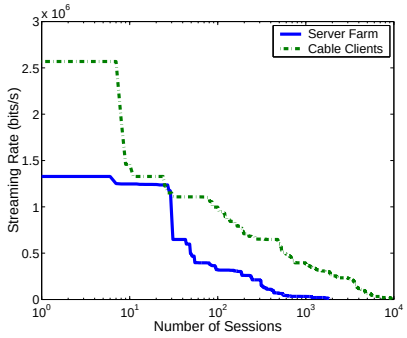

Figure 11: Client streaming rate distribution

wasted. Comparably, in the client workload, 66% of early terminated sessions have more than half of their transferred data wasted.

Hence, our conclusion is: *The rate mismatch in pseudo streaming is common, which either causes long delays to the clients, or unnecessary traffic on the Internet. Pseudo streaming is doomed to be a transient technique due to its performance drawbacks and its incapability of providing client interactive functions.*

4.5 Streaming Traffic Analysis

Having studied the drawbacks and implications of downloading and pseudo streaming, now we analyze the streaming traffic in the workloads.

From Table 5, we have learned that streaming is the most efficient by transferring only the necessary data. This indicates via streaming, there should be a lot of early terminated connections. Figures 9 and 10 show the client access duration relative to the full object playback time. In the figures, the y-axis is the percentage of a multimedia object accessed during its streaming session. The x-axis is the number of streaming sessions ranked by such percentages in an increasing order.

As shown in the figures, in the server workload, 44% of sessions are terminated before 10% of the object is played, while only 11% of sessions play the entire objects. Similarly, in the client workload, 35% of all sessions are terminated before 10% of the object is played, and only 20% of client sessions complete the whole object playback. Compared with downloading and pseudo streaming, clients using streaming are much more likely to terminate their access to an object earlier. This demonstrates that streaming allows a client to make a timely decision after viewing the initial portion of an object.

The streaming service is resource consuming. The client link bandwidth is always a bottleneck if it cannot catch up to the object streaming rate. Thus it is useful to know whether the current Internet service can provide a large enough bandwidth for streaming services.

Figure 11 shows the streaming rate distribution for the streaming objects in the server and the client workload (note that the x-axis is in log scale). This figure shows that in the server workload, the streaming rates range from 5 Kbps to 1.33 Mbps and the average rate is about 98.8 Kbps; in the client workload, the object streaming rate is relatively higher, up to 2.57 Mbps, and the average rate is about 169.5 Kbps. Today, broadband users can afford this rate.

Studying the streaming session duration and the streaming rate distribution, we have discovered the client access pattern and the bandwidth requirements for streaming on the current Internet. This information will be used in our next section.

5. AUTOSTREAM

In previous sections, we have studied the characteristics of media delivery via downloading, pseudo streaming, and streaming. Our analysis indicates that streaming is the most efficient approach, but has limited availability due to their associated expenses. Streaming needs dedicated system support, such as a streaming server, and thus requires a high cost. While big commercial media providers (e.g. Real-

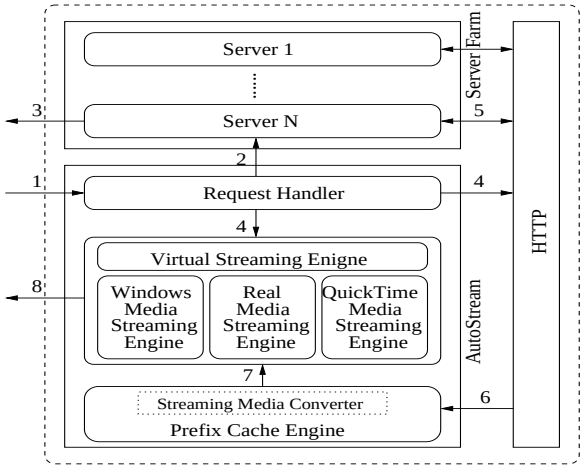

Figure 12: AutoStream for a server farm

Networks) may have the resources to provide such support for the large number of streaming objects they host, many medium scale content providers (such as those in our Web server farm) may have only a few streaming objects. It is not cost-effective and administratively easy for each of them to set up a streaming server individually. However, since the site population in a Web server farm is large, these sites can share a special server, capable of streaming functions, to amortize the cost for streaming service.

In this section, we present the design and evaluation of a shared streaming server, *AutoStream*, that can transform downloading and pseudo streaming delivery of media objects on the Web into streaming delivery automatically. The system integrates existing techniques, including prefix caching [24] and transcoding [4], to provide streaming service in server farms or content delivery networks with a very low cost. It also supports a *preview* function that allows a client to decide if an object is worth downloading after viewing its beginning portion. Recent studies [11, 10] suggest that many clients tend to watch only the beginning portions of media objects and then terminate. Hence, such a previewing function can not only reduce the client perceived start up latency, but also avoid traffic waste due to potentially unnecessary downloads.

5.1 Architecture Design

Figure 12 shows the design of AutoStream and its interactions with the Web servers in a server farm. By placing the AutoStream before the server farm, a client request for a media object is received in step 1. AutoStream allows clients to choose whether to continue the downloading/pseudo streaming access or to preview the object first. If the client continues with its original request, the request is forwarded to AutoStream in step 2 and is processed in step 3. If previewing is preferred, the *Request Handler* will take over the process. The Request Handler notifies the streaming engine and directs the communication between the server and the cache engine for the object prefix via HTTP when necessary (steps 4-6). After the data arrives in the cache engine, the streaming engine reads the data from cache (step 7) and streams it to the client in step 8. For downloading, the Request Handler will send a range request for the rest of the object for client to the server, allowing the client to save the prefix data as well. For streaming, steps 4-8 are performed. Note that the *Prefix Cache Engine* only cache the initial portions of media objects, since for most objects only those portions are viewed.

The *Streaming Media Converter* in the Prefix Cache Engine has two functions. First, it is responsible for converting non-streamable files (e.g. MPEG1) from Web servers to files in a streaming file format (e.g. Real Media format). This function is similar to Adobe Premiere [1] or Helix Producer [3]. Second, it is also responsible for adapting the encoding rate of a streaming media object to the connection speed of the corresponding client device. This function is similar to an online transcoding producer [4]. The converted prefixes of media objects are cached in the Prefix Cache Engine to serve subsequent requests from similar devices.

5.2 Evaluation

We conduct trace driven simulations using the media workloads we collected to evaluate the potential benefits of AutoStream. In these experiments, we assume that the cache space is large enough to hold the initial segments of all media objects, while no later segments are cached. The cache in the AutoStream is cleaned hourly to reflect the cache size limit in practical systems. We assume that the transmission latency between hosts in the server farm and AutoStream is negligible because they are typically located close to each other. When AutoStream converts non-streaming delivery of media objects into streaming delivery, we assume that the client access duration for these objects now follows the same pattern as the streaming media objects in our workload.

Figure 13 shows the potential traffic reduction in the server farm by AutoStream. The height of each bar represents the total downloading and pseudo streaming traffic in our server traces in each hour. The lower two parts of each bar represent the amount of streaming traffic converted from downloading and pseudo streaming traffic by AutoStream, respectively. The upper two parts of each bar thus represent the amount of reduced downloading and reduced pseudo streaming traffic, respectively.

The figure indicates that AutoStream achieves significant traffic reduction: on average, it reduces the downloading traffic by 78% and the pseudo streaming traffic by 72%. The hourly traffic reduction rate reaches the maximum (about 80%) in the traffic peak hour (20:00-21:00) for the server farm. Note that these achievements have considered the transcoding function in the Streaming Media Converter. When the end-to-end bandwidth between a client and the AutoStream is lower than the encoding rate of the media object, AutoStream transcodes the media into an appropriate, lower encoding rate. While this brings additional saving in bandwidth, it also degrades the quality of the media. If we do not consider transcoding, our evaluation shows that AutoStream still can reduce the downloading and the pseudo streaming traffic by 56% and 43%, respectively.

We also compute the number of sessions suffering delayed startup with and without the AutoStream system in the server farm, respectively, as shown on Figure 14. AutoStream cannot avoid a startup delay to the client when it needs to convert the media objects in non-streaming formats into streaming formats or transcode the objects encoded with high encoding rates into low encoding rates. On

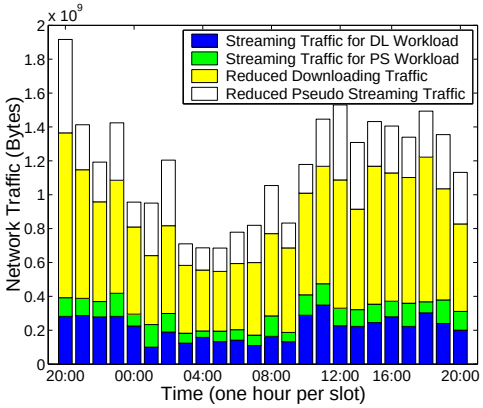

Figure 13: AutoStream on traffic reduction

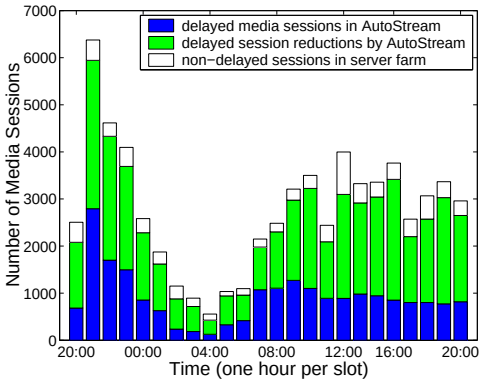

Figure 14: AutoStream on accumulated delay time

Figure 14, the height of each bar represents the total number of media sessions in our trace hourly, in which the lowest part represents the number of media sessions with startup delay with the AutoStream system, and the sum of the lowest and the middle part represents the number of media sessions with startup delays without the AutoStream system in the current server farm. AutoStream can reduce the startup delay for about 63.4% of the total sessions in the server farm without AutoStream.

Although AutoStream is designed for server farms, a similar idea can be applied to significantly reduce the traffic in the client workload (e.g., a CDN edge server with AutoStream functions). Our simulation based evaluation shows that the traffic reduction without considering the transcoding can be up to 65% for downloading and 32% for pseudo streaming in the client workload. We omitted the figures due to page limit.

6. RELATED WORK

A number of studies have focused on characterizing Internet media contents recently. Pre-stored video objects on the Web servers were studied earlier to characterize the static attributes of video files [5]. In [11] and [6], the streaming media workloads from educational environments were collected. The client session duration, object popularities and sharing patterns were analyzed in [11], where the workload is collected from the client side, while the workloads in [6] were collected from the server side. In [10], streaming workloads from enterprise media server logs were studied for the locality, dynamics and evolution of the accesses to media objects along the time.

Besides the workload characterizations in different environments, a live streaming media workload was studied in [27], while client interactions were characterized for frequency of different types of client interactions and distributions of session on and off times for MANIC audio content system [21], the low-bitrate classroom 2000 systems [16], the educational eTech and BIBS media servers [6], and the educational internal server of a large international corporation [17]. In [12], four audio and video workloads from educational and entertainmental sites were studied for client interactions. Further, our recent work [15] measures the delays due to interactive operations in media streaming based on a large Internet workload, and proposes an effective interleaved segment caching solution to support interactive streaming. Authors in [26] analyzed a live streaming workload from a large content delivery network. The feasibility of supporting streaming applications through application overlays was studied in [25].

Targeting major streaming media content providers, measurement and analysis of RealNetwork audio [18] and RealNetwork video [28] and Windows media [20] workloads were performed. Tools for multimedia traffic monitoring, such as mmdump [19], which extends the functionalities of tcpdump, were also developed for media traffic study. A lot of strategies have also been proposed to deal with the Internet media objects. Streaming caching, such as prefix caching [24], exponential segmentation [29], Hyper-Proxy system [9, 8], have been proposed to cache the media objects in segments instead of their entirety in the proxy close to the client.

Different from previous studies, the media contents in our study contain a set of mixed workloads of various media types in all available delivery mechanisms collected from both the server side and the client side, which allowed us to compare their characteristics. We also explored the system implications by designing AutoStream system for server farms instead of for a particular streaming server as in most of the previous studies.

7. CONCLUSION

Media objects have played increasingly important roles in Internet content deliveries, and are becoming indispensable in many applications. However, our study shows that existing Internet infrastructure does not keep up with this high demand. For example, we surprisingly find that the majority of the Internet media contents are still delivered by downloading that causes substantial Internet bandwidth waste and low quality of delivery service when pseudo streaming is used. Motivated by our intensive workload measurements and analysis, we propose a design of an innovative streaming system for Web farms. This system can automatically and transparently transform the media downloading and pseudo streaming requests to high quality streaming with an amortized low cost. We have shown its effectiveness through simulations based on the large amount of traces we collected

from a Web farm. We plan to further enhance this study by implementing AutoStream for Web farms and other Web services.

8. ACKNOWLEDGMENTS

We would like to thank Oliver Spatscheck for many insightful discussions on this topic and for helping us understand Gigascope. We are also grateful to Michael Rabinovich, Geoffrey M. Voelker, William L. Bynum, Leeann Bent, and the anonymous reviewers for their constructive comments and suggestions. This work is partially supported by the National Science Foundation under grants CNS-0098055, CCF-0129883, and CNS-0405909.

9. REFERENCES

[1] Adobe Premiere. http://www.adobe.com/products/.

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

[3] Helix-producer project. https://helix-producer.helixcommunity.org/.

[4] IBM Research: Internet transcoding for universal access. http://www.research.ibm.com/.

[5] S. Acharya, B. Smith, and P. Parnes. Characterizing user access to videos on the world wide web. In *Proc. of ACM/SPIE Multimedia Computing and Networking (MMCN)*, January 1998.

[6] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon. Analysis of educational media server workloads. In *Proc. of ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, June 2001.

[7] L. Bent, M. Rabinovich, G. M. Voelker, and Z. Xiao. Characterization of a large web site population with implications for content delivery. In *Proc. of the International World Wide Web Conference*, May 2004.

[8] S. Chen. *Building Internet Caching Systems for Streaming Media Delivery*. PhD thesis, College of William and Mary, Williamsburg, VA, July 2004.

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

[10] L. Cherkasova and M. Gupta. Characterizing locality, evolution, and life span of accesses in enterprise media server workloads. In *Proc. of ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, May 2002.

[11] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming media workload. In *Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems*, March 2001.

[12] 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 the International World Wide Web Conference*, May 2004.

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

[14] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[15] L. Guo, S. Chen, Z. Xiao, and X. Zhang. DISC: Dynamic interleaved segment caching for interactive streaming. In *Proc. of IEEE International Conference on Distributed Computing Systems*, June 2005.

[16] H. Harel, V. Vellanki, A. Chervenak, G. Abowd, and U. Ramachandran. Workload of a media-enhanced classroom server. In *Proc. of 2nd Annual Workshop on Workload Characterization*, October 1999.

[17] L. He, J. Grudin, and A. Gupta. Designing presentations for on-demand viewing. In *Proc. of ACM Conference on Computer Supported Cooperative Work*, December 2000.

[18] A. Mena and J. Heidemann. An empirical study of real audio traffic. In *Proc. of IEEE INFOCOM*, March 2000.

[19] J. V. D. Merwe, R. Caceres, Y. H. Chu, and C. Sreenan. Mmdump - a tool for monitoring multimedia usage on the internet. In *Technical Report 00.2.1.*, AT&T Labs, February 2000.

[20] J. Nichols, M. Claypool, R. Kinicki, and M. Li. Measurements of the congestion responsiveness of windows streaming media. In *Proc. of ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, June 2004.

[21] J. Padhye and J. Kurose. An empirical study of client interactions with a continuous media courseware server. In *Proc. of ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, July 1998.

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

[23] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, April 1998.

[24] S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *Proc. of IEEE INFOCOM*, March 1999.

[25] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Proc. of ACM SIGCOMM*, Aug. 2004.

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

[28] Y. Wang, M. Claypool, and Z. Zuo. An empirical study of realvideo performance across the internet. In *Proc. of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, November 2001.

[29] K. Wu, P. S. Yu, and J. Wolf. Segment-based proxy caching of multimedia streams. In *Proc. of the International World Wide Web Conference*, May 2001.