Measurement and Analysis of a Streaming-Media Workload

Maureen Chesire, Alec Wolman, Geoffrey M. Voelker†, and Henry M. Levy

Department of Computer Science and Engineering
University of Washington †*University of California, San Diego*

Abstract

The increasing availability of continuous-media data is provoking a significant change in Internet workloads. For example, video from news, sports, and entertainment sites, and audio from Internet broadcast radio, telephony, and peer-to-peer networks, are becoming commonplace. Compared with traditional Web workloads, multimedia objects can require significantly more storage and transmission bandwidth. As a result, performance optimizations such as streaming-media proxy caches and multicast delivery are attractive for minimizing the impact of streaming-media workloads on the Internet. However, because few studies of streaming-media workloads exist, the extent to which such mechanisms will improve performance is unclear.

This paper (1) presents and analyzes a client-based streaming-media workload generated by a large organization, (2) compares media workload characteristics to traditional Web-object workloads, and (3) explores the effectiveness of performance optimizations on streaming-media workloads. To perform the study, we collected traces of streaming-media sessions initiated by clients from a large university to servers in the Internet. In the week-long trace used for this paper, we monitored and analyzed RTSP sessions from 4,786 clients accessing 23,738 distinct streaming-media objects from 866 servers. Our analysis of this trace provides a detailed characterization of streaming-media for this workload.

1 Introduction

Today's Internet is increasingly used for transfer of continuous-media data, such as video from news, sports, and entertainment Web sites, and audio from Internet broadcast radio and telephony. As evidence, a large 1997 Web study from U.C. Berkeley [10] found no appreciable use of streaming media, but a study three years later at the University of Washington found that RealAudio and RealVideo had become a considerable component of Web-related traffic [30]. In addition, new peer-to-peer networks such as Napster have dramatically increased the use of digital audio over the Internet. A 2000 study of the IP traffic workload seen at the NASA Ames Internet Exchange found that traffic due to Napster rose from 2% to 4% over the course of 10 months [13], and a March 2000 study of Internet traffic at the University of Wisconsin-Madison found that 23% of its traffic was due to Napster [20].

Streaming-media content presents a number of new challenges to systems designers. First, compared to traditional Internet applications such as email and Web browsing, multimedia streams can require high data rates and consume significant bandwidth. Second, streaming data is often transmitted over UDP [14], placing responsibility for congestion control on the application or application-layer protocol. Third, the traffic generated tends to be bursty [14] and is highly sensitive to delay. Fourth, streaming-media objects require significantly more storage than traditional Web objects, potentially increasing the storage requirements of media servers and proxy caches, and motivating more complex cache replacement policies. Fifth, because media streams have long durations compared to the request/response nature of traditional Web traffic, multiple simultaneous requests to shared media objects introduce the opportunity for using multicast delivery techniques to reduce the network utilization for transmitting popular media objects. Unfortunately, despite these new characteristics and the challenges of a rapidly growing traffic component, few detailed studies of multimedia workloads exit.

This paper presents and analyzes a client-based streaming-media workload. To capture this workload, we extended an existing HTTP passive network monitor [30] to trace key events from multimedia sessions initiated inside the University of Washington to servers in the Internet. For this analysis, we use a week-long trace of RTSP sessions from 4,786 university clients to 23,738 distinct streaming-media objects from 866 servers in the Internet, which together consumed 56 GB of bandwidth.

The primary goal of our analysis is to characterize this streaming-media workload and compare it to well-studied HTTP Web workloads in terms of bandwidth utilization, server and object popularity, and sharing patterns. In particular, we wish to examine unique aspects of streaming-media workloads, such as session duration, session bit-rate, and the temporal locality and degree of overlap of multiple requests to the same media object. Finally, we wish to consider the effectiveness of perfor-

mance optimizations, such as proxy caching and multicast delivery, on streaming-media workloads.

Our analysis shows that, for our trace, most streaming-media objects accessed by clients are encoded at low bit-rates (< 56 Kb/s), are modest in size (< 1 MB), and tend to be short in duration (< 10 mins). However, a small percentage of the requests (3%) are responsible for almost half of the total bytes downloaded. We find that the distributions of client requests to multimedia servers and objects are somewhat less skewed towards the popular servers and objects than with traditional Web requests. We also find that the degree of multimedia object sharing is smaller than for traditional Web objects for the same client population. Shared multimedia objects do exhibit a high degree of temporal locality, with 20–40% of active sessions during peak loads sharing streams concurrently; this suggests that multicast delivery can potentially exploit the multimedia object sharing that exists.

The rest of the paper is organized as follows. In Section 2 we provide an overview of previous related work. Section 3 provides a high-level description of streaming-media protocols for background. Section 4 describes the trace collection methodology. Section 5 presents the basic workload characteristics, while Section 6 focuses on our cache simulation results. Section 7 presents our stream merging results. Finally, Section 8 concludes.

2 Related Work

While Web client workloads have been studied extensively [3, 5, 10, 8, 29], relatively little research has been done on multimedia traffic analysis. Acharya et al. [1] analyzed video files stored on Web servers to characterize non-streaming multimedia content on the Internet. Their study showed that these files had a median size of 1.1 MB and most of them contained short videos (< 45 seconds). However, since their study was based upon a static analysis of stored content, it is unclear how that content is actually accessed by Web clients.

Mena et al. [14] analyzed streaming audio traffic using traces collected from a set of servers at Broadcast.com, a major Internet audio site. The focus of their study was on network-level workload characteristics, such as packet size distributions and other packet flow characteristics. From their analyses, they derive heuristics for identifying and simulating audio flows from Internet servers. Their study showed that most of the streaming-media traffic (60–80%) was transmitted over UDP, and most clients received audio streams at low bit-rates (16–20 Kb/s). The most striking difference between results obtained from their trace and our analysis is that most of their streaming sessions were long-lived; 75% of the sessions analyzed lasted longer than one hour. We attribute this difference to the fact that they studied server-based audio traces from a single site, while we study a client-based trace of both audio and video streams to a large number of Internet servers.

Van der Merwe et al. [15] extended the functionality of tcpdump (a popular packet monitoring utility) to include support for monitoring multimedia traffic. Although the primary focus of their work was building the multimedia monitoring utility, they also reported preliminary results from traces collected from WorldNet, AT&T's commercial IP network. As in [14], their study of over 3,000 RTSP flows also focused on network-level characteristics, such as packet length distributions and packet arrival times. In addition, they characterized the popularity of object accesses in the trace and similarly found that they matched a Zipf-like distribution. In contrast to our university client trace, the WorldNet workload peaks later in the evening and has relatively larger weekend workloads. We attribute this difference to the time-of-day usage patterns of the two different client populations; WorldNet users are not active until after work, while the university users are active during their work day.

There has been significant commercial activity recently (by companies such as FastForward Networks, Inktomi, and Akamai) on building caching and multicast infrastructure for the delivery of both on-demand and live multimedia content. However, little has been published about these efforts.

Our paper builds upon this previous work in a number of significant ways. First, we study a trace with an order of magnitude more sessions. Second, we focus on application-level characteristics of streaming-media workloads – such as session duration and sizes, server and object popularity, sharing patterns, and temporal locality – and compare and contrast those characteristics with those of non-streaming Web workloads. Lastly, we explore the potential benefits of performance optimizations such as proxy caching and multicast delivery on streaming-media workloads.

3 Streaming Media Background

This section defines a number of terms and concepts that are used throughout the paper. We use the term streaming media to refer to the transfer of live or stored multimedia data that the media player can render as soon as it is received (rather than waiting for the full download to complete before rendering begins). Although streaming techniques are typically used to transfer audio and video streams, they are sometimes used to deliver traditional media (such as streaming text or still images). A wide variety of streaming-media applications are in use today on the Internet. They employ a wide variety of protocols and algorithms that can generally be classified into five categories:

1. **Stream control protocols** enable users to interactively control the media stream, e.g., pausing, rewinding, forwarding or stopping stream playback. Examples include RTSP[26], PNA[21], MMS[16] and XDMA[31]. These protocols typically rely on TCP as the underlying transport protocol.

2. **Media packet protocols** support real-time data delivery and facilitate the synchronization of multiple streams. These protocols define how a media server encapsulates a media stream into data packets, and how a media player decodes the received data. Most media packet protocols rely on UDP to transport the packets. Examples include RDT[22], RTP[25], PNA[21], MMSU and MMST[16].

3. **Encoding formats** dictate how a digitized media stream is represented in a compact form suitable for streaming. Examples of encoding schemes commonly used include WMA[16], MP3[19], MPEG-2[18], RealAudio G2 and RealVideo G2 [23].

4. **Storage formats** define how encoded media streams are stored in "container" files, which hold one or more streams. Headers in the container files can be used to specify the properties of a stream such as the encoding bit-rate, the object duration, the media content type, and the object name. ASF[9] and RMFF[2] are examples of container file formats.

5. **Metafile formats** provide primitives that can be used to identify the components (URLs) in a media presentation and define their temporal and spatial attributes. SDP[11], SMIL[28] and ASX[17] are examples of metafile formats.

4 Methodology

We collected a continuous trace of RTSP traffic flowing across the border routers serving the University of Washington campus over a one week period between April 18th and April 25th, 2000. In addition to monitoring RTSP streams, the trace tool also maintained connection counts for other popular stream control protocols: PNA[21] used by Real Networks' servers, MMS[16] used by Microsoft Windows Media servers, and XDMA[31], used by Xing's StreamWorks servers. Our trace data was collected using a DEC Alpha workstation connected to four Ethernet switches at the University network border. The software used to monitor and log streaming-media packets is described in Section 4.2.

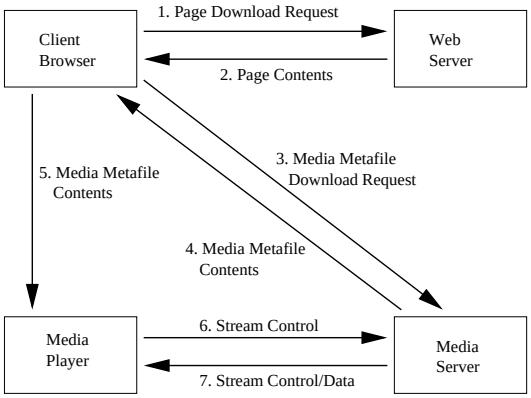

Figure 1: *Streaming media communication.*

4.1 Protocol Processing

Capturing streaming-media traffic is challenging because applications may use a variety of protocols. Moreover, while efforts have been made to develop common standardized protocols, many commercial applications continue to use proprietary protocols. Given the diversity of protocols, we decided to focus initially on the standardized and well documented RTSP protocol [26]. Widely used media players that support RTSP include Real Networks' RealPlayer and Apple's QuickTime Player. A high level description of the RTSP protocol is presented below.

4.1.1 RTSP Overview

The RTSP protocol is used by media players and streaming servers to control the transmission of media streams. RTSP is a request-response protocol that uses a MIME header format, similar to HTTP. Unlike HTTP, the RTSP protocol is stateful and requests may be initiated by either clients or servers. In practice, the RTSP control traffic is always sent over TCP. The media data is often sent over UDP, but it may also be interleaved with the control traffic on the same TCP connection. RTSP uses sequence numbers to match requests and responses. Media objects are identified by an RTSP URL, with the prefix "rtsp:".

Figure 1 illustrates a common streaming media usage scenario. First, a user downloads a Web page that contains a link to a media presentation. This link points to a metafile hosted by the media server. The Web browser then downloads the metafile that contains RTSP URLs for all the multimedia objects in the presentation (e.g., a music clip and streaming text associated with the audio). Next, the browser launches the appropriate media player and passes the metafile contents to the player. The media player in turn parses the metafile and initiates an RTSP connection to the media server.

Many of our analysis results will refer to an RTSP "session." An RTSP session is similar to an HTTP "GET" request, in that typically there will be one session for each access to the object. A session begins when the media player first accesses the object, and it ends when the media player sends a TEARDOWN message, though there may be a number of intervening PAUSE and PLAY events. There is not a one-to-one mapping between sessions and RTSP control connections; instead, the protocol relies on session identifiers to distinguish among different streams. In order to make our results easier to understand, when a single RTSP session accesses multiple objects, we consider it to be multiple sessions – one for each object.

4.2 Trace Collection and Analysis Software

We extended our existing HTTP passive network monitor [30] to support monitoring the RTSP streaming-media protocol. Our trace collection application has three main components: a packet capture and analysis module that extracts packet payloads and reconstructs TCP flows and media sessions; a protocol parsing module that parses RTSP headers; and a logging module that writes the parsed data to disk. After the traces are produced, we analyze the collected trace data in an off-line process. We now provide a brief overview of the operation of the trace collection software.

The packet capture module uses the Berkeley packet filter [12] to extract packets from the kernel and then performs TCP reassembly for each connection. Simple predicates are used on the first data bytes of each connection to classify TCP connections as RTSP control connections. This module also maintains a session state table that is used to map control messages to client streaming sessions; each table entry models the state machine for a client's multimedia session. In addition to maintaining the session table, this module provides support for: handling responses that arrive earlier than the corresponding requests due to asymmetric routing; merging control messages that cross packet boundaries; and extracting messages from connections transmitting control data interleaved with media stream data.

Data in the control connections is used to determine which UDP datagrams to look at. We record timing and size information about the UDP data transfers, but we do not attempt to process the contents of media stream packets because almost all commonly used encoding formats and packet protocols are proprietary.

The protocol parsing module extracts pertinent information from RTSP headers such as media stream object names, transport parameters and stream play ranges. All sensitive information extracted by the parser, such as IP addresses and URLs, is anonymized to protect user privacy. Finally, the logging module converts the data to a compact binary representation and then saves it to stable storage.

Attribute	Values
Connection counts	58808 (RTSP); 44878 (MMS); 35230 (PNA); 3930 (XDMA)
RTSP Servers	866 (External)
RTSP Total Bytes	56 GB (Continuous media)
RTSP Clients	4786 (UW clients)
RTSP Sessions	40070 (Continuous media)
RTSP Objects	23738 (Continuous media); 3760 (Other)

Table 1: Trace statistics.

5 Workload Characterization

This section analyzes the basic characteristics of our streaming-media workload; when appropriate we compare these characteristics to those of standard Web object workloads. The analysis ignores non-continuous media data (e.g., streaming text and still images). Since we were interested in the access patterns of the UW client population, we ignored sessions initiated by clients external to UW that accessed servers inside the campus network.

Table 1 summarizes the high-level characteristics of the trace. During this one-week period, 4,786 UW clients accessed 23,738 distinct RTSP objects from 866 servers, transferring 56 GB of streaming media data. Using the connection counts from Table 1, we estimate that RTSP accounts for approximately 40% of all streaming media usage by UW clients.

The detailed analyses in the following sections examine various attributes of the traffic workload, such as popularity distributions, object size distributions, sharing patterns, bandwidth utilization, and temporal locality. Overall, our analysis shows that:

- Most of the streaming data accessed by clients is transmitted at low bit-rates: 81% of the accesses are transmitted at a bit-rate less that 56 Kb/s.
- Most of the media streams accessed have a short duration ($<$ 10 minutes) and a modest size ($<$ 1 MB).
- A small percentage of the sessions (3%) are responsible for almost half of the bytes downloaded.
- The distribution of client requests to objects is Zipf-like, with an α parameter of 0.47.
- While clients do share streaming-media objects, the degree of object sharing is not as high as that observed in Web traffic traces [5, 30].

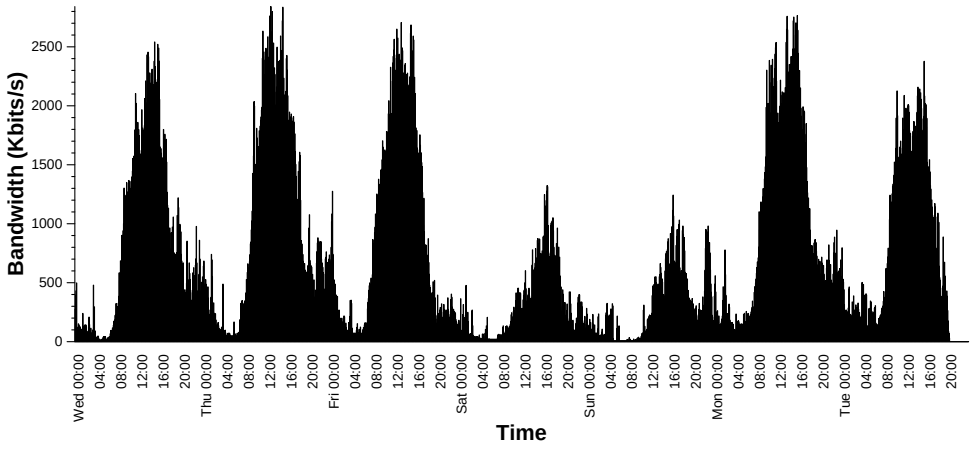

Figure 2: *Bandwidth utilization over time (in Kbits/sec).*

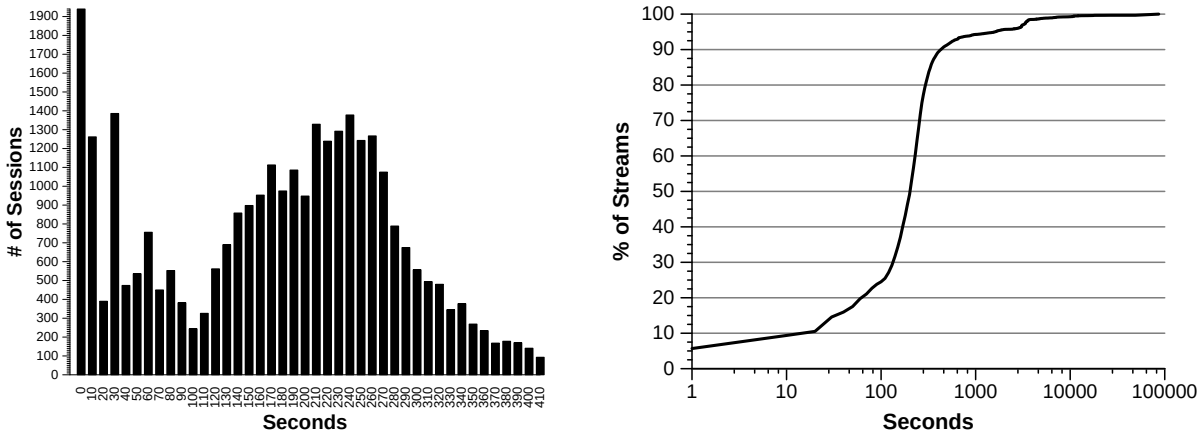

Figure 3: *Advertised stream length. (a) Normal and (b) CDF.*

- There is a high degree of temporal locality in client requests to repeatedly accessed objects.

5.1 Bandwidth Utilization

Figure 2 shows a time-series plot of the aggregate bandwidth consumed by clients streaming animations, audio, and video data. We see that the offered load on the network follows a diurnal cycle, with peaks generally between 11 AM and 4 PM. The volume of traffic is significantly lower during weekends; peak bandwidth over a five-minute period was 2.8 Mb/s during weekdays, compared to 1.3 Mb/s on weekends. We found that, on average, clients received streaming content at the rate of 66 Kb/s. This bit-rate is much lower than the capacity of the high-bandwidth links of the clients and the UW ISP links. We conclude from the prevalence of these low-bit-rate sessions that the sites that clients are accessing encode streaming content at modem bit-rates, the lowest common denominator.

5.2 Advertised Stream Length

In this section, we provide a detailed analysis of the advertised duration of continuous media streams referenced during the trace. Note that the advertised duration of a stream is different from the length of time that the client actually downloads the stream (e.g., if the user hits the stop button before the stream is finished). Since media servers generally do not advertise the duration of live streams, we limit our analysis to on-demand (stored) media streams. Sessions accessing these on-demand streams account for 85% of all sessions.

Figure 3a is a histogram of all streams lasting less than seven minutes, and Figure 3b plots the cumulative distribution of all stream lengths advertised by media servers. The peaks in the histogram in Figure 3a indicate that many streams are extremely short (less than a minute), but the most common stream lengths are between 2.5 and 4.5 minutes. These results suggest that clients have a stronger preference for viewing short multimedia streams. One important observation from Fig-

ure 3b is that the stream-length distribution has a long tail. Although the vast majority of the streams (93%) have a duration of less than 10 minutes, the remaining 7% of the objects have advertised lengths that range between 10 minutes and 6 weeks.

5.3 Session Characteristics

In this section, we examine two closely related properties of sessions: the amount of time that a client spends accessing a media stream, and the number of bytes downloaded during a stream access. In Figure 4 we present the relationship between the duration of a streaming-media session and the number of bytes transferred. In Figure 5 we look at the distinguishing characteristics between sessions accessing shared objects and sessions accessing unshared objects. Finally, in Figure 6 we compare the size and duration characteristics of sessions from clients in the campus modem pool to sessions from clients connected by high-speed department LANs.

A number of important trends emerge from these graphs. First, we see that client sessions tend to be short. From Figure 5a we see that 85% of all sessions (the solid black line) lasted less than 5 minutes, and the median session duration was 2.2 minutes. The bandwidth consumed by most sessions was also relatively modest. From Figure 5b we see that 84% of the sessions transferred less than 1 MB of data and only 5% accessed more than 5 MB. In terms of bytes downloaded, the median session size was 0.5 MB. Both the session duration and the session size distributions have long tails: 20 sessions accessed more than 100 MB of data each, while 57 sessions remained active for at least 6 hours, and one session was active for 4 days. Although the long-lived sessions (> 1 hour) account for only 3% of all client sessions, these sessions account for about half of the bandwidth consumed by the workload. From Figure 4 we see that these long sessions account for 47% of all bytes downloaded.

Most of the media objects accessed are downloaded at relatively low bit-rates despite the fact that most of the clients are connected by high-speed links. Using the raw data from Figure 4, we calculated that 81% of the streams are downloaded at bit-rates less than 56 Kb/s (the peak bandwidth supported by most modems today). In Figure 6, we separate all the trace sessions into those made from clients in the modem pool and those made from LAN clients. Although it does appear that the duration of modem sessions is shorter than the duration of LAN sessions (Figure 6a), the difference is not large. On the other hand, the difference in bytes downloaded between modem sessions and LAN sessions (Figure 6b) appears to be much more pronounced. For modem users, the median session size is just 97 KB, whereas for LAN users it is more than 500 KB.

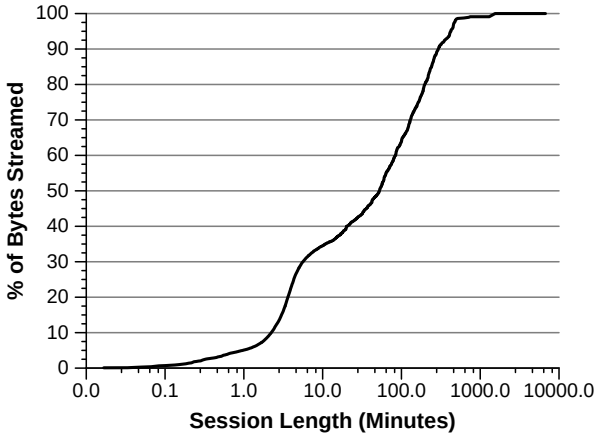

Figure 4: *Session duration vs. session size.*

Figures 5a and 5b also distinguish between accesses to shared objects (the dashed lines) and accesses to unshared objects (the grey lines). A shared object is one that is accessed by more than one client in the trace; an unshared object is accessed by only one client, although it may be accessed multiple times. Overall, sessions that request shared objects tend to be shorter than sessions accessing unshared objects. For example, 46% of shared sessions lasted less than one minute, compared with only 30% of the unshared sessions. Furthermore, we found that most of the sessions accessing shared objects transferred less data than unshared sessions. For example, 44% of shared sessions transferred less than 200 KB of data compared to only 24% of unshared sessions. However, Figure 5 shows that the situation changes for sessions on the tails of both curves, where the sessions accessing shared objects are somewhat longer and larger than sessions accessing unshared objects.

5.4 Server Popularity

In this section we examine the popularity of media servers and objects. Figure 7 plots (a) the cumulative distribution of continuous media objects across the 866 distinct servers referenced, as well as (b) the cumulative distribution of requests to these servers. These graphs show that client load is heavily skewed towards the popular servers. For example, 80% of the streaming-media sessions were served by the top 58 (7%) media servers, and 80% of the streaming-media objects originated from the 33 (4%) most popular servers. This skew to popular servers is *slightly less pronounced* than for requests to non-streaming Web objects. From a May 1999 trace of the same client population, 80% of the requests to non-streaming Web objects were served by the top 3% of Web servers [30].

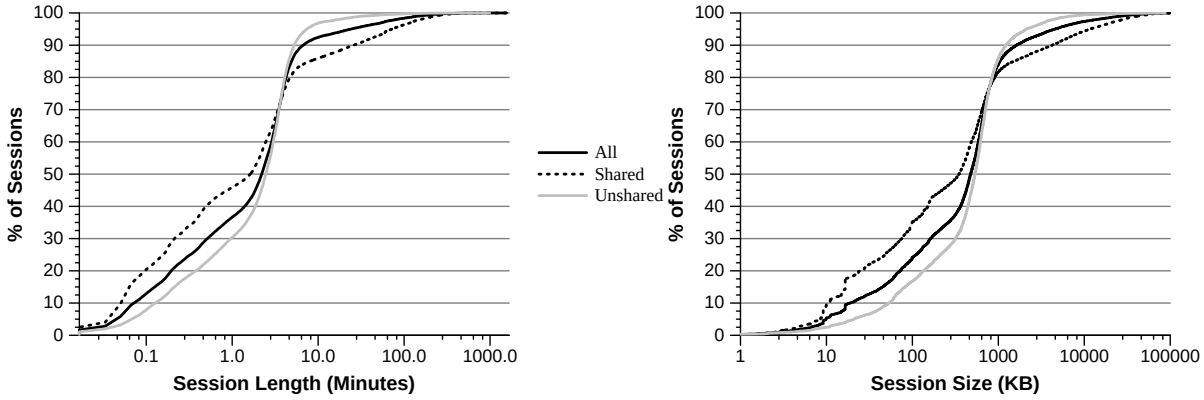

Figure 5: *Shared and unshared session characteristics. (a) Time and (b) Size.*

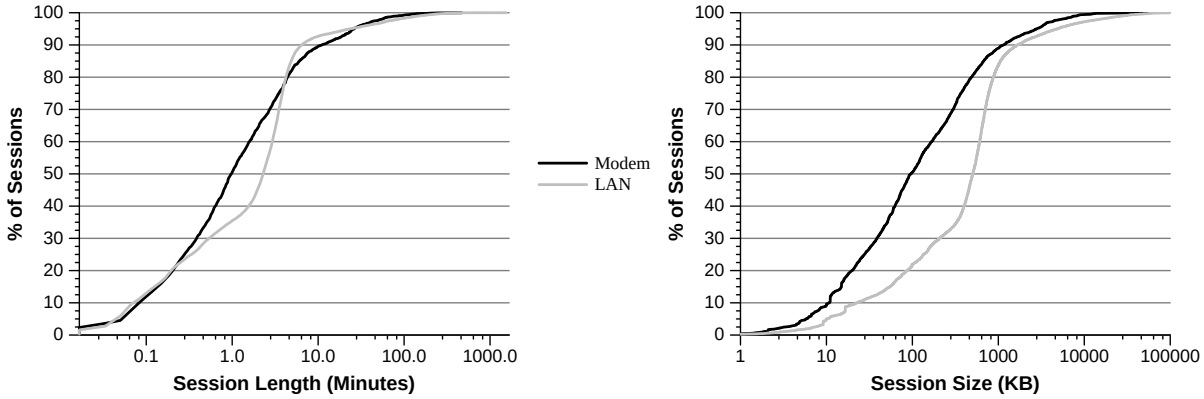

Figure 6: *Modem and LAN session characteristics. (a) Time and (b) Size.*

5.5 Object Popularity

One of the goals of our analysis was to understand how client requests were distributed over the set of multimedia streams accessed during the trace period. To determine this, we ranked multimedia objects by popularity (based on the number of accesses to each stream) and plotted the results on the log-scale graph shown in Figure 8. Our analysis found that of the 23,738 media objects referenced, 78% were accessed only once. Only 1% of the objects were accessed by ten or more sessions, and the 12 most popular objects were accessed more than 100 times each. From Figure 8, one can see that the popularity distribution fits a straight line fairly well, which implies that the distribution is Zipf-like [4]. Using the least squares method, we calculated the α parameter to be 0.47. In contrast, the α parameters reported in [4] for HTTP proxies ranged from 0.64 to 0.83. The implication is that accesses to streaming-media objects are somewhat less concentrated on the popular objects in comparison with previously reported Web object popularity distributions.

5.6 Sharing patterns

In this section we explore the sharing patterns of streaming-media objects among clients. We first examine the most popular objects to determine whether the repeated accesses come from a single client, or whether those popular objects are widely shared. In Figure 9, we compute the number of unique clients that access each of the 200 most popular streaming-media objects. This figure shows that the most popular streams are widely shared, and that as the popularity declines, so does the number of unique clients that access the stream.

Figure 10 presents per-object sharing statistics. Of the streaming-media objects requested, only 1.6% were accessed by five or more clients, while 84% were viewed by only one client. Only 16% of the objects were shared (i.e., accessed by two or more clients), yet requests for these shared objects account for 40% of all sessions recorded. From this data, we conclude that the shared objects are also more frequently accessed and can therefore benefit from caching. Note, however, that the degree of object sharing is low compared to the sharing rate for web documents [5, 30]. Consequently, multi-

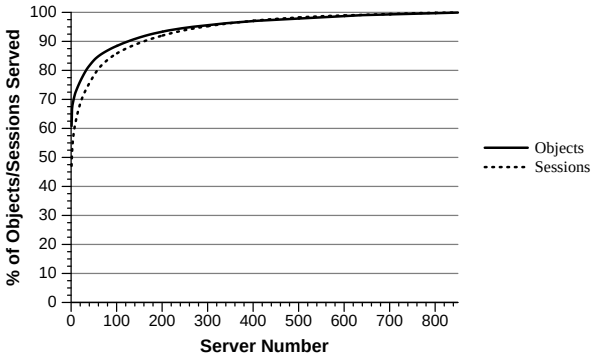

Figure 7: *Server popularity by object and session.*

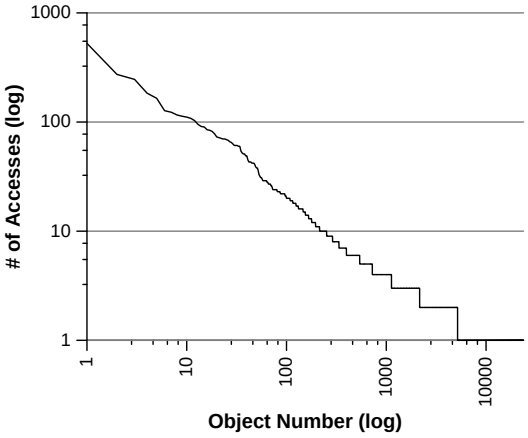

Figure 8: *Object popularity by number of sessions (note log scale).*

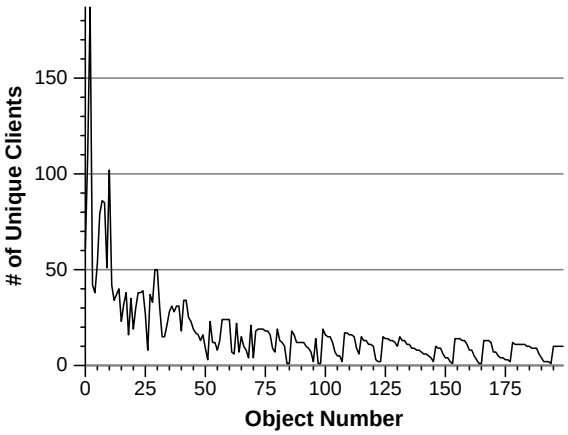

Figure 9: *Number of clients accessing popular objects.*

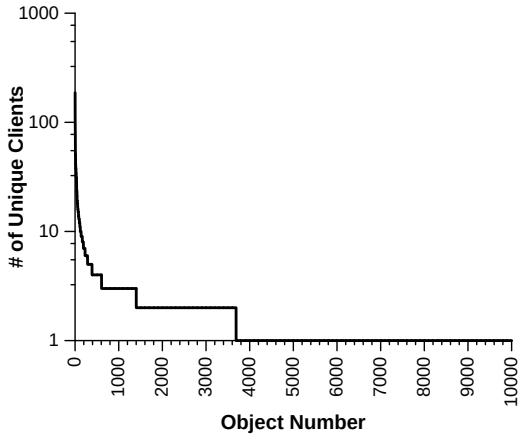

Figure 10: *Object sharing.*

media caching may not be as effective as Web caching in improving performance.

Figure 11 shows the overlap among accesses to the shared media objects by plotting the number of sessions that access unshared objects (black) and the number of sessions that access shared objects (grey) over time for the entire trace. During peak load periods between 11 AM and 4 PM (weekdays), we see that 20%–40% of the active sessions share streams concurrently. This temporal locality suggests that (1) caching will work best when it is needed the most (during peak loads), and that (2) multicast delivery has the opportunity to exploit temporal locality and considerably reduce network utilization.

6 Caching

Caching is an important performance optimization for standard Web objects. It has been used effectively to reduce average download latency, network utilization, and server load. Given the large sizes of streaming-media objects and the significant network bandwidth that they can consume, caching will also be important for improving the performance of streaming-media objects. In this section, we study the potential benefits of proxy caching for streaming-media objects. In particular, we determine cache hit rates and bandwidth savings for our workload, explore the tradeoff of cache storage and hit rate, and examine the sensitivity of hit rate to eviction timeouts.

We use a simulator to model a streaming media caching system for our analyses. The simulator caches the entire portion of any on-demand stream retrieved by a client, making the simplifying assumption that it is allowed to cache all stored media objects. For live streams, the simulator assumes that the cache can effectively merge multiple client accesses to the same live stream by caching a fixed-size sliding interval of bytes from that stream [7]. The simulator assumes unlimited cache capacity, and it uses a timeout-based cache eviction policy to expire cached objects. For Figures 12 through 14, an object is removed from the cache two hours after the end of the most recent access.

The results of the simulation are presented in the set of graphs below. Figure 12 is a time-series plot showing cache size growth over time, while Figure 13 shows

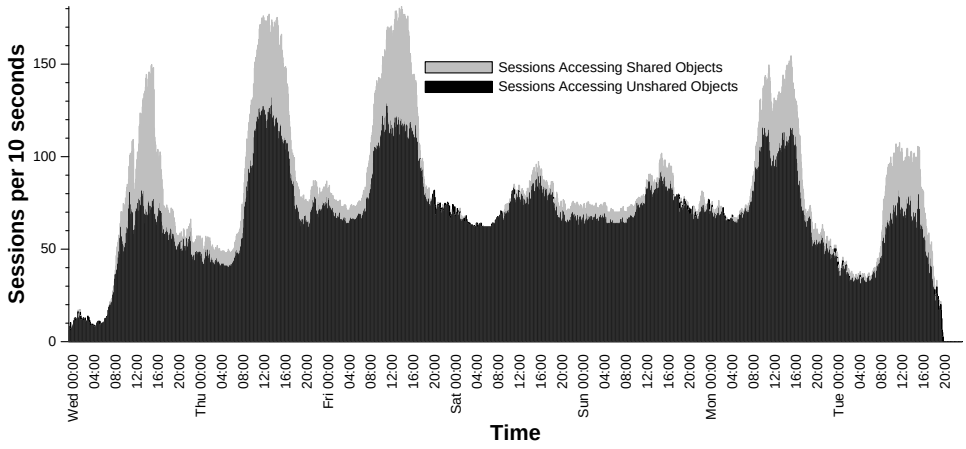

Figure 11: *Concurrent sharing over time.*

potential bandwidth savings due to caching. The total height of each bar in the stacked bar graph in Figure 14 reflects the total number of client accesses started within a one-hour time window. The lightest area of the graph shows the number of accesses that requested fully-cached objects; the medium-grey section represents the number of accesses that resulted in partial cache hits. Partial cache hits are recorded when a later request retrieves a larger portion of the media stream than was previously accessed. The height of the darkest part of the graph represents the number of accesses that resulted in cache misses.

Since streaming objects are comparatively large in size, the replacement policy for streaming proxy caches may be an important design decision. Many proposed designs for streaming proxy caches assume that multimedia streams are too large to be cached in their entirety [24, 27]. As a result, specialized caches are designed to cache only selected portions of a media stream; uncached portions of the stream have to be retrieved from the server (or neighboring caches) to satisfy client requests.

To determine the need for these complex caching strategies, we explored the sensitivity of hit rate to cache replacement eviction policies by varying the timeout for cached objects. Using the default two hour expiration of our simulator, we found that the simulated cache achieved an aggregate request hit rate of 24% (including partial cache hits) and a byte hit rate of 24% using less than 940 MB of cache storage. Because the required cache size is relatively small (when compared to the total 56 GB of data transferred), it appears that conventional caching techniques and short expiration times might be just as effective as specialized streaming media caching schemes for client populations similar to this.

Figure 15 plots request and byte hit rates as object eviction time is increased from 5 minutes to 7 days (the entire trace duration). Notice that reducing the caching window to 5 minutes still yields reasonably high request hit rates (20%). By keeping objects in the cache for only two hours after the last access, we achieve 90% of the maximum possible byte hit rate for this workload while saving significant storage overhead. From this data, we can infer that requests to streaming-media objects that are accessed at least twice have a high degree of temporal locality.

7 Stream Merging

Stream merging is a recently developed technique that uses multicast to reduce the bandwidth requirements for delivering on-demand streaming media. The details of stream merging are covered extensively in [6, 7]. In this section we provide a high level overview of stream merging to motivate our measurements.

Stream merging occurs when a new client requests a stream that is already in transmission. In this case, the server begins sending the client two streams simultaneously: (1) a “patch stream” starting at the beginning of the client’s request, and (2) a multicast stream of the existing transmission in progress. The new client buffers the multicast stream while displaying the patch stream. When the patch stream is exhausted, the client displays the multicast stream from its buffer while it continues to receive and buffer the simultaneous multicast stream ahead of the display. At the merge point, only one stream, via multicast, is being transmitted to both clients. The cost of stream merging is that clients must be able to receive data faster than the required stream playback rate and must buffer the additional data.

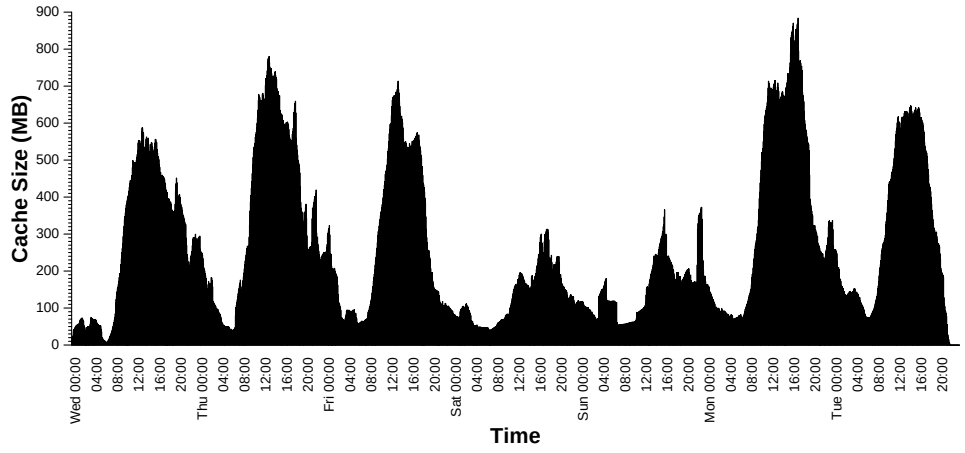

Figure 12: *Cache size growth over time.*

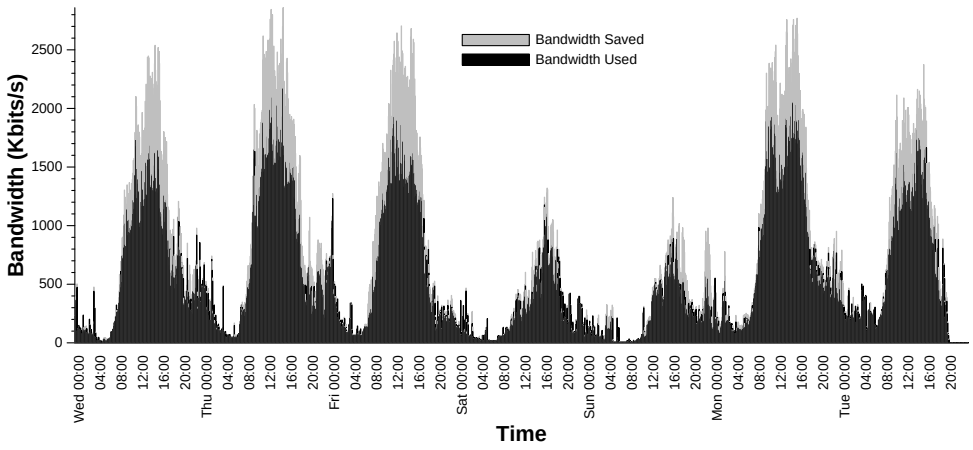

Figure 13: *Bandwidth saved over time due to caching.*

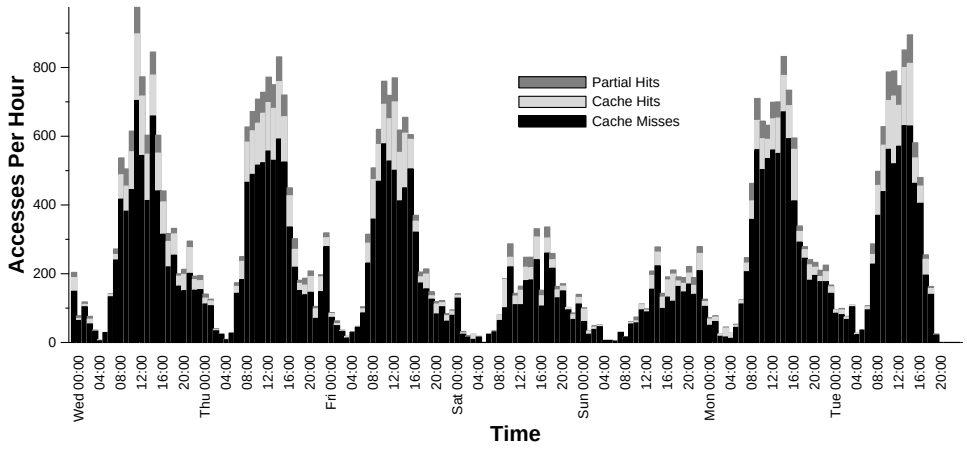

Figure 14: *Cache accesses: Hits, partial hits, and misses.*

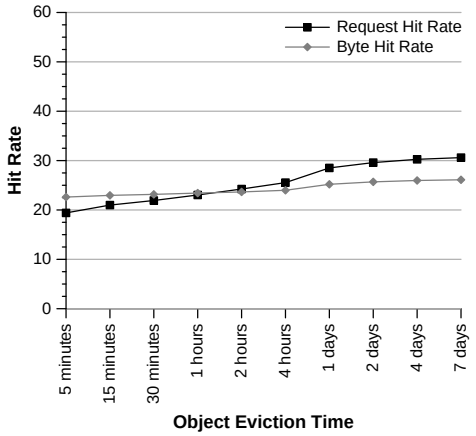

Figure 15: *Effect of eviction time on cache hit rates.*

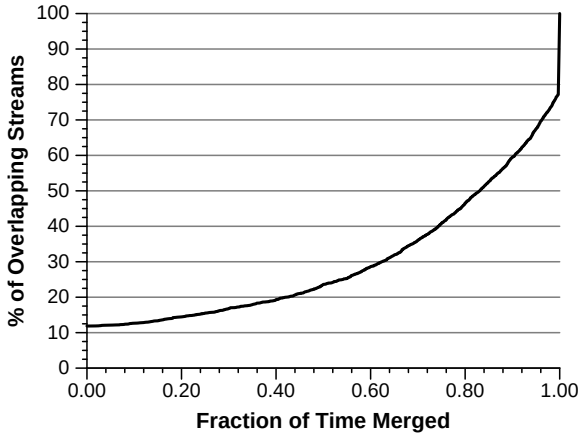

Figure 16: *Effectiveness of stream merging.*

To evaluate the effectiveness of stream merging for our workload, we consider consecutive overlapping accesses to each stream object in our trace and calculate the time it takes to reach the merge point based on [7]. Given the time of the merge point, we then calculate what percentage of the overlap period occurs after the merge point. This corresponds to the percentage of time that only one stream is being transmitted via multicast to both clients. The results of this analysis are shown as a cumulative distribution in Figure 16. Because this stream merging technique is only needed for on-demand streams, live streams are not included in Figure 16. This figure shows that stream merging is quite effective for this workload – for more than 50% of the overlapping stream accesses, shared multicast can be used for at least 80% of the overlap period. This result indicates strong temporal locality in our trace, which is consistent with our concurrent sharing and cache simulation results.

8 Conclusion

We have collected and analyzed a one-week trace of all RTSP client activity originating from a large university. In our analyses, we characterized our streaming multimedia workload and compared it to well-studied HTTP Web workloads in terms of bandwidth utilization, server and object popularity, and sharing patterns. In particular, we examined aspects unique to streaming-media workloads, such as session duration, session bitrate, temporal locality, and the degree of overlap of multiple requests to the same media object. We also explored the effectiveness of performance optimizations, such as proxy caching and multicast delivery, on streaming-media workloads.

We have observed a number of properties (e.g., degree of object sharing and lower Zipf α parameter for the object-popularity distribution) indicating that multimedia workloads may benefit less from proxy caching, on average, than traditional Web workloads. On the other hand, we have found that multimedia workloads exhibit stronger temporal locality than we expected, especially during peak hours. This suggests that multicast and stream-merging techniques may prove to be useful for these workloads.

Our results are fundamentally based upon the workload that we captured and observed. It is clear that usage of streaming media in our environment is still relatively small, and our results could change as the use of streaming media becomes more prevalent. Our one-week trace contains only 40,000 sessions from fewer than 5,000 clients. A one-week trace of Web activity for the same population about a year earlier showed more than 22,000 active clients, and more than 80 million web requests during one week. Shifts in technology use, such as the widespread use of DSL and cable modems, will likely increase the use of streaming media and change underlying session characteristics. As the use of streaming media matures in the Internet environment, further client workload studies will be required to update our understanding of the impact of streaming-media data.

9 Acknowledgments

We would like to thank David Richardson, Terry Gray, Art Dong, and others at the UW Computing and Communications organization for supporting our effort to collect traces. We would also like to thank the USITS referees for their comments and suggestions. This research was supported in part by DARPA Grant F30602-97-2-0226, National Science Foundation Grant EIA-9870740, Compaq's Systems Research Center, and a Microsoft Graduate Research Fellowship.

References

[1] S. Acharya and B. Smith. An experiment to characterize videos stored on the web. In *Proc. of ACM/SPIE Multimedia Computing and Networking 1998*, January 1998.

[2] R. Agarwal, J. Ayars, B. Hefta-Gaub, and D. Stammen. RealMedia File Format. Internet Draft: draft-heftagaub-rmff-00.txt, March 1998.

[3] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. Characterizing reference locality in the www. In *Proc. of IEEE Intl. Conference on Parallel and Distributed Information Systems '96*, Dec 1996.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of IEEE INFOCOM 1999*, March 1999.

[5] B. M. Duska, D. Marwood, and M. J. Feeley. The Measured Access Characteristics of World-Wide-Web Client Proxy Caches. In *Proc. of the 1st USENIX Symposium on Internet Technologies and Systems*, December 1997.

[6] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. In *Proc. of the 5th Int'l Workshop on Multimedia Information Systems*, October 1999.

[7] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth skimming: A technique for cost-effective video-on-demand. In *Proc. of ACM/SPIE Multimedia Computing and Networking 2000*, January 2000.

[8] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich. Performance of web proxy caching in heterogeneous bandwidth environments. In *Proc. of IEEE INFOCOM 1999*, March 1999.

[9] E. Fleischman. Advanced Streaming Format (ASF) Specification. Internet-Draft: draft-fleischman-asf-01.txt, February 1998.

[10] S. D. Gribble and E. A. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. In *Proc. of the 1st USENIX Symposium on Internet Technologies and Systems*, December 1997.

[11] M. Handley and V. Jacobson. RFC 2327: SDP: Session Description Protocol, April 1998.

[12] S. McCanne and V. Jacobson. The BSD Packet Filter: A new architecture for user-level packet capture. In *Proc. of the Winter 1993 USENIX Technical Conference*, 1993.

[13] S. McCreary and K. Claffy. Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange. http://www.caida.org/outreach/pa pers/AIX0005/, May 2000.

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

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

[16] Microsoft. Windows Media Development Center. http://msdn.microsoft.com/windowsmedia/.

[17] Microsoft. All About Windows Media Metafiles. http://msdn.microsoft.com/workshop/imedia/windows media/crcontent/asx.asp, April 2000.

[18] MPEG-2 Standard. ISO/IEC Document 13818-2. Generic Coding of Moving Pictures and Associated Audio Information, Part 2: Video, 1994.

[19] MPEG-2 Standard. ISO/IEC Document 13818-3. Generic Coding of Moving Pictures and Associated Audio Information, Part 3: Audio, 1994.

[20] D. Plonka. UW-Madison Napster Traffic Measurement. http://net.doit.wisc.edu/data/Napster, March 2000.

[21] RealNetworks. Firewall PNA Proxy Kit. http://www.service.real.com/firewall/pnaproxy.html.

[22] RealNetworks. RealNetworks Documentation Library. http://service.real.com/help/library/.

[23] RealNetworks. Realsystem production and authoring guides. http://service.real.com/help/library/encoders.html.

[24] R. Rejaie, M. Handley, H. Yu, and D. Estrin. Proxy caching mechanism for multimedia playback streams in the internet. In *Proc. of the Fourth Int. Web Caching Workshop*, March 1999.

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

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

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

[28] W3C. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. http://www.w3.org/TR/1998/REC-smil-19980615/, June 1998.

[29] C. E. Wills and M. Mikhailov. Towards a better understanding of web resources and server responses for improved caching. In *Proc. of the Eighth Int. World Wide Web Conference*, pages 153–165, May 1999.

[30] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy. Organization-based analysis of web-object sharing and caching. In *Proc. of the 2nd USENIX Symposium on Internet Technologies and Systems*, October 1999.

[31] Xingtech. Streamworks documentation. http://www.xingtech.com/support/docs/streamworks/.