

Quantitative Analysis of Strategies for Streaming Media Distribution

Marisa A. Vasconcelos Leonardo C. da Rocha Juliano de C. Santos Jose Ismael P. Jr
Leonardo L. P. da Mata Jussara M. de Almeida Wagner Meira Jr Virgilio A. F. Almeida

Department of Computer Science, Federal University of Minas Gerais
Av. Antonio Carlos 6627, Belo Horizonte MG, 31270-010 , Brazil
{isa, lcrocha, juliano, ismaeljr, barroca, jussara, meira, virgilio}@dcc.ufmg.br

Abstract

Streaming media applications are becoming more popular on the late years, as for example, news transmitted live through the web, music, show, and films. Traditional client/server architectures have shown to be inefficient for distributing streaming media services. The total number of users is limited due to the demand of computational resources, which are higher than in traditional web applications. One of the proposals for an efficient distribution is the use of a P2P overlay network. Although there are some works that evaluate this proposal through simulation, there is a lack of quantitative analysis of the requirements for server and network resources (i.e., CPU, server and network bandwidth) in actual P2P systems, compared to traditional client/server systems. This work is to fill this gap by providing experimental results that quantify the savings in server and network resources if a P2P approach is used for distributing live streaming media instead of the traditional client/server approach. Towards this goal, we build an experimental testbed, in a controlled environment, to evaluate actual systems with varying number of clients during periods when the distribution structure is static. This experimental testbed is composed of one new efficient and scalable application called streaming servent, one streaming server application (Darwin) and a workload consisting three media files. In this work, we use simple but significative analytical formulas to evaluate the scalability of our servent application. The experimental results show a significative scalability potential of the P2P architecture. For example, for 24 clients it is necessary to dedicate, on average, 15 Mbits/s of its available network bandwidth to deliver the stream in a client/server system. In a P2P system, on the other hand, the required average server bandwidth for delivering the video clip would be just 9 Mbits/s for 24 clients, providing a 40% reduction on bandwidth consumption.

1. Introduction

The Internet has recently experienced a significant increase in streaming media traffic [18], which can be explained by several factors including improvement of network bandwidth in the Internet backbone and the availability of faster “last mile” connections such as cable modems and DSL as well as the development of more efficient video and audio compression methods such as divX [2] and MP3 [3]. The traditional approach to streaming media distribution is a client/server architecture where the distribution of popular media is done via unicast streaming from a server directly to the clients.

The high server and network bandwidth requirements of streaming media content distribution motivate the use of multicast distribution, which can be implemented either at the network level (IP multicast [9]) or at the application level [10, 6]. Whereas IP multicast has not been effectively deployed in the Internet, there have been several proposals for application level multicast, which can be implemented through overlay-router networks or peer-to-peer overlay networks [17].

In overlay-router networks, such as Vcast [10], a central server distributes the content to other servers, placed in different points in the network, which act as application level routers for the streams originated from the central server to the clients. In case of pre-stored content, these servers may also (partially) cache some of the streams so that future client requests may be served locally. The main disadvantage of this type of architecture is the extra costs incurred by the need for dedicated servers.

On the other hand, in a peer-to-peer (P2P) overlay network, the extra costs are avoided by implementing application level multicast through cooperation of clients and server. In other words, the application level multicast distribution tree is built among clients (i.e. peers), which request the stream either from the central server or from another client, and may forward the content to other clients further down in the distribution tree. Conceptually, the to-

tal load that would be imposed on the server in a traditional client/server architecture is distributed among server and clients. Thus, the system is expected to have better scalability with practically no extra cost. We consider scalability as the server's capacity to keep the quality of the service provided under high loads.

Despite the intuitive appeal of P2P overlay networks, little has been done to quantify its scalability. In other words, despite a number of related projects, which focus mainly on simulating alternative policies for dynamically building the P2P distribution tree [6, 8, 17, 13], there is a lack of a quantitative analysis of the requirements in terms of server and network resources (i.e., CPU, server and network bandwidth) in actual P2P systems, compared to traditional client/server systems.

The main goal of this paper is to fill this gap by providing experimental results that quantify the savings in server and network resources if a P2P approach is used for distributing live streaming media instead of the traditional client/server approach. Towards this goal, we build an experimental testbed, in a controlled environment, to evaluate actual systems with varying number of clients during periods when the distribution structure is static. In other words, we filter out the issues related to the dynamic behavior of peers, network nodes, and focus on the resource requirements for each architecture with a static distribution tree for a given number of clients.

The main contributions of this paper are:

- An experimental testbed for evaluating alternative streaming media distribution mechanisms. In addition to a commercial and publicly available streaming server (the Apple Darwin Streaming Server [1]), two key components of this testbed are: (1) a new efficient and scalable application called *streaming servant*, which can act as both client and server, forwarding packets to other servants in a P2P network and (2) a workload consisting of three media files with different duration, average streaming rate and variability in the bitrate that stress different aspects of the system.
- Quantitative measures, obtained experimentally, that show the scalability of client/server and P2P overlay architectures. In particular, we quantify the savings in terms of server and network resources as well as the improvement in the average loss rate and loss duration (i.e., bursts) if a P2P overlay-based distribution of streaming media is used instead of the traditional unicast client/server mechanism for varying number of clients.
- We use simple analytical formulas (i.e., Little's Result [11]) to evaluate the scalability of our servant application. These analytical results are validated by

the experimental measures collected. Furthermore, we also show how these simple formulas can be used to define the distribution tree of a P2P overlay network given the maximum amount of resources (CPU and I/O network bandwidth) a peer is willing to dedicate to content distribution.

The remaining of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the experimental testbed, the performance metrics used for comparing the alternative architectures and the workload. The experiments are analyzed in Section 4. Finally, Section 5 presents our conclusions and future work.

2. Related Work

Peer to peer overlay networks have been the focus of several recent studies. In [14, 12], the authors provide a characterization of the workload and client behavior of two existing P2P file sharing systems, namely Napster and Gnutella. Four other studies have investigated the issues related to implementing a P2P architecture for streaming media distribution [6, 13, 8, 19].

In [6], the authors propose the SpreadIt architecture for P2P distribution of live media streams. By using experimentally collected measures, such as packet delay and packet loss, as parameters for a simulator of the system, the authors evaluate the performance of different strategies for dynamically rebuilding the P2P distribution tree as clients join and/or leave the system.

The CoopNet architecture [13], proposed for live and on-demand streams, relies on the independent distribution of multiple descriptions of each stream to improve the media quality received by the clients. The system was compared against the traditional client/server approach, using trace driven simulation and the access log of September 11, 2001 of the MSNBC website as input. The authors found that the requirements for server bandwidth are significantly reduced without placing an unreasonable burden at the clients. Different from the CoopNet study, our work analyzes not only server bandwidth but also server CPU utilization and packet losses for an actual architecture.

In [8], the authors introduce a P2P media streaming model for on-demand streams and propose algorithms for disseminating media data into the system and searching algorithms for locating peers with required objects. The architecture was simulated and evaluated with several performance measures under different rate arrivals, including constant, flash crowd, and Poisson arrivals. An algorithm for assigning media data to each supplying peer taking into account its out-bound bandwidth is presented in [19]. Like the other studies, this algorithm was only evaluated by means of simulation.

Our approach differs from these previous studies in three key aspects: (1) we experiment with and evaluate a real P2P architecture instead of a simulator, (2) we evaluate server and network resources (CPU, bandwidth) consumption as well as packet losses as a function of the number of clients in the system, and (3) we evaluate the system during periods when the client set and the distribution tree is fixed, so that we are able to determine the maximum scalability achievable.

3. Evaluation Methodology

This section describes the methodology used to experimentally analyze client/server and P2P overlay architectures for live media distribution. Section 3.1 describes the experimental setup, the media server (Section 3.1.1) and the new server application (Section 3.1.2). The performance metrics used to evaluate the two architectures are discussed in Section 3.2. Finally, Section 3.3 presents the main characteristics of the workload used in our experiments.

3.1. Experimental Setup

In a client/server architecture, the traditional mechanism for streaming media distribution in the Internet, an origin server transmits the content directly to each client, typically via unicast streaming. This mechanism creates a single level distribution tree rooted at the origin, as illustrated in Figure 1. The server resources, especially server bandwidth, limit the scalability of this centralized approach.

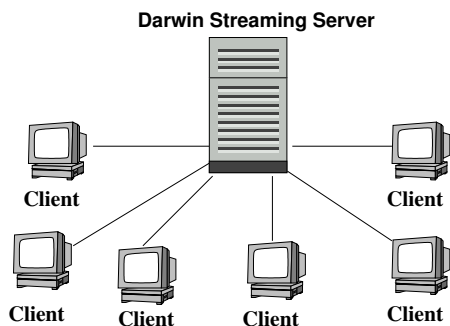


Figure 1. Client/server Architecture

In a peer-to-peer overlay architecture, on the other hand, the distribution tree, also rooted at the origin, may have multiple levels as clients may receive the requested content either directly from the origin or from another client located at a higher level in the tree. Figure 2 illustrates a P2P distribution tree with 2 levels. In this decentralized architecture, the server bandwidth requirements are lowered and the system is expected to have better scalability.

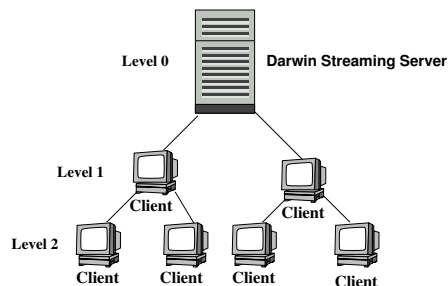


Figure 2. Peer to Peer Overlay Architecture

In this paper, we build an experimental testbed to quantify the resource consumption and the scalability of these two architectures for live media distribution. This testbed consists of seven PCs running Linux Red Hat 7.3 (kernel 2.4.18-3), each with one 750 MHz AMD Duron processor, 256 MB of RAM, and 514 MB of swap disk, connected through a dedicated switched Fast Ethernet LAN. In this testbed, the publicly available Darwin streaming server, briefly described in Section 3.1.1, is used as the origin server, running in one of the seven machines. A new client application, which has also the capability of forwarding packets to other clients in a P2P network, is built to generate requests to the server. A number of this new application, called *streaming server* or simply *server* and described in Section 3.1.2, runs in the other six machines. Throughout the rest of this paper, the terms client and server will be used interchangeably.

We evaluate the client/server and P2P overlay architectures for a varying number of clients. For a given number of clients, a set of three experiments is performed. In each experiment, the client (i.e., server) processes are homogeneously distributed among all six client machines and simultaneously request a given file from Darwin. The experiment lasts until all data packets have been transmitted to all clients. For the P2P architecture, we consider the two-level distribution tree shown in Figure 2, which, for any given number of clients, is statically built at the beginning of each experiment. In determining the number of clients that are served by each server at the intermediate level, we make the reasonable assumption that clients of a P2P network are willing to dedicate only a small fraction of its resources to forward content to other clients. Thus, unless otherwise noted, we assume each server process, running on one of the client machines in the intermediate level of the P2P distribution tree, forwards packets to only two clients in the lower level.

Note that as the number of levels increase, the savings in server resource consumption are expected to increase, but so is the packet loss observed by the clients at the lower levels. Thus, we make the compromise of using only two levels in the distribution tree.

3.1.1. Server: Darwin The Darwin Streaming Server [1] is an open source event-driven server developed by Apple. It runs as a set of processes which perform the standard RTSP, RTP, and RTCP streaming protocols [15, 16]. Darwin operates either by broadcasting the content to all clients or by serving it on demand. In the former, the number of packets sent to each client depends on the time, relative to the beginning of the broadcast, when the client established the connection with the server, whereas in the latter every client request is served with a complete transmission of the file. In each of our experiments, all client processes are initiated at (approximately) the same time. However, the moment on which each client actually starts receiving data depends on how fast Darwin is able to process all connection requests. Thus, as we observed in some previous experiments, the number of packets sent to each client may vary significantly. Determining this number, which is needed to compute the number of packets lost in the streams sent to the clients, a key measure of the scalability of the system would require instrumentation of Darwin. In order to avoid the possible overhead introduced by this instrumentation, we chose to run Darwin in the on-demand mode.

3.1.2. Streaming Servent In order to implement and experiment with a P2P system, we need an application that can act as both client and server, forwarding packets to other clients. A few experiments with a previously developed Python application [6], which has this capability, revealed a surprisingly high packet loss when even a single client requested a video from Darwin. We also evaluated previously developed proxy servers, such as QuickTime Streaming Proxy. However, the high complexity of such systems, unnecessary to a P2P client application, and some previously reported performance problems [7] motivated us to build a new application, the streaming servent, from scratch. The development of the servent was driven by efficiency and simplicity. In particular, since our current goal is to experiment with distribution of live media content, the servent was built to act as a packet forwarder and does not store any content locally. However, the application is highly modular facilitating future extensions (e.g. caching module).

The servents communicate with each other and with Darwin using the RTSP, RTP, and RTCP streaming protocols. Figure 3 shows the messages that are exchanged between a servent and Darwin for establishing a connection before the transmission of a new stream is initiated. The same messages may be exchanged between two servents. The process is initiated with an interaction of RTSP messages: the servent sends a RTSP request to the server, which sends back a stream descriptor (SDP) with information about the new stream created to deliver the requested content. After the descriptor is received, servent and server exchange SETUP messages in order to allocate RTP/RTCP ports. Finally, the

servent sends a PLAY message to the server, which, then, starts streaming the content. Content and control packets are exchanged between both entities using the RTP and RTCP protocols.

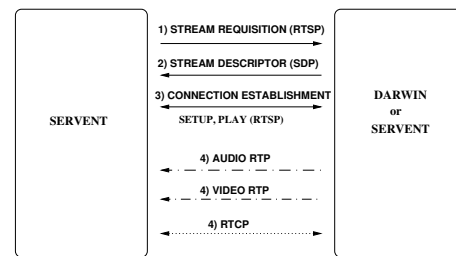


Figure 3. Messages Exchanged between Servent and Darwin (or another servent)

As input, each servent process receives the IP address and the RTSP port of the server, the name of the file that it should request from the server and the RTP ports. The servent runs as three separate threads, implemented using the C pthread library. The *main* thread is responsible for handling new RTSP requests from other servents. Two other threads are created for receiving packets from the server and forwarding them to other servents. The display of the packets received at each servent is done by redirecting them to a media player, such as the QuickTime player [4].

3.2. Performance Metrics

The performance metrics used to quantitatively analyze client/server and P2P overlay architectures can be grouped into two categories: server-side and servent-side metrics. The server-side metrics, used to evaluate the resource requirements at the server machine, are:

- **CPU utilization:** the time spent by the CPU in either system or user modes, as a fraction of the total elapsed time. Both average CPU utilization (i.e., measured for the whole experiment) and instantaneous CPU utilization (i.e., measured at each one second interval) are considered.
- **Server bandwidth:** instantaneous and average number of bytes (or packets) transmitted per second.

The servent-side metrics, collected at each servent machine, are used to assess not only the requirements for servent resources (i.e, CPU and bandwidth) but also the media quality perceived by each servent. The media quality perceived by a servent is directly related to the number of packets received in a stream, compared to the total number of packets. Conversely, media quality is also directly related to the number of packets lost in a transmission as well

as to the number of consecutive packets that were lost in each loss event, or in other words, the “duration” of a loss event. Thus, perceived media quality is assessed indirectly, by measuring the quantities described above. The server-side metrics are:

- **CPU utilization:** measured both at each one-second interval and aggregated for the whole experiment.
- **Server bandwidth:** instantaneous and average number of bytes (or packets) received and transmitted per second.
- **Observed stream bitrate :** instantaneous and average number of bytes (or packets) received per second.
- **Packet loss:** instantaneous and average number of RTP packets lost measured as a fraction of the total number of packets the client(s) should have received in a transmission.
- **Packet loss duration:** distribution of the “duration” of loss events, measured as the number of consecutive packets lost in each event.

Statistics related to these metrics are gathered by two collector tools. One tool runs in each (server and client) machine and periodically collects data related to CPU utilization and I/O network bandwidth from the /proc file system. At each one-second interval, the tool extracts data from the /proc/stat and /proc/net/dev log files and generates new statistics. A second collector module runs as part of the server application and stores a description of each packet received and sent by the server in a log file. This description includes a sequence number, size, and timestamp. The log file is post-processed to compute the number of packets lost and the distribution of packet loss duration observed by each server, during each experiment.

3.3. Workload

The workload used in our experiments, summarized in Table 1, consists of two VBR video files (a video clip and a trailer) and one audio file (a music), all of them in *MOV* format. These files have characteristics that span a wide range of sizes, durations, average bitrate and, as will become clear in Section 4, variability in the instantaneous bitrate. It is important to note that we have carefully chosen our workload to represent different classes of media objects that are currently popular in the Internet. Furthermore, the diversity of the characteristics of the selected files stress different aspects of the system operation. Thus, experimenting with these files allows us to explore different scenarios that occur in real systems.

Type	Frame rate	Size (MB)	Size (min)	Bitrate (kbytes/s)	# of RTP packets
video clip	15	60	04:38,16	217,6	44772
trailer	30	22	03:31,17	101,9	18311
music	-	8	09:54,13	12,5	6193

Table 1. Workload Description

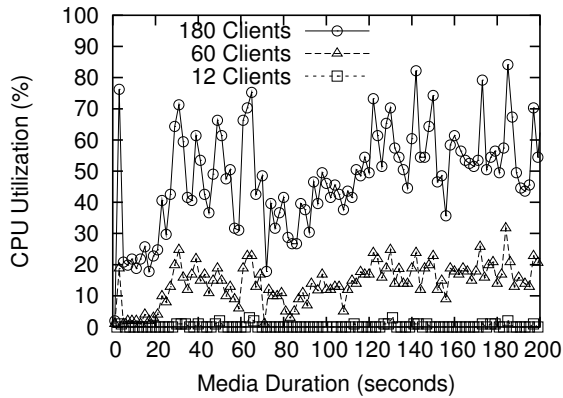
4. Results

This section provides a quantitative evaluation of the scalability of P2P overlay and client/server networks for live media distribution, measured both in terms of the requirements for server resources (Section 4.1) and in terms of packet loss, which directly impacts the quality of media received by each client (Section 4.2). Section 4.3 presents experimentally collected measures of the server requirements for CPU and bandwidth as a function of the number of clients to which they forward packets (i.e., the server fan-out). Finally, Section 4.4 shows that simple analytical formulas can be used to assess the scalability of the server and to determine the maximum fan-out of a server given the amount of resources it is willing to dedicate for forwarding packets to other clients. The results shown below are average of three experiments.

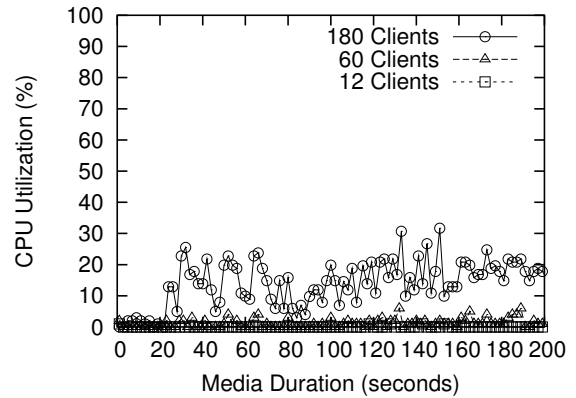
4.1. Server Load

This section provides results for the load imposed on the server, in terms of the requirements for CPU and I/O network bandwidth, as a function of the number of clients simultaneously requesting a stream. These results show that, compared to the traditional client/server solution, both CPU and network bandwidth utilizations are significantly reduced if a P2P approach is taken. Furthermore, they also show that the savings in the requirements for server resources consistently increase with the number of clients. In other words, the relative cost to upgrade a media server in response to an increase in the load is lower if a P2P approach is adopted.

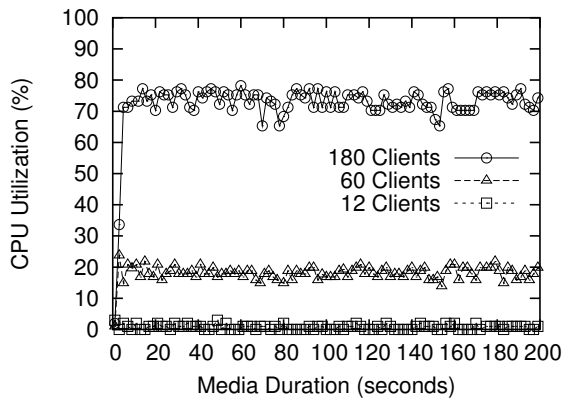
The instantaneous server CPU utilization measured at each one-second interval for the duration of each experiment is shown for the trailer in Figures 4(a) (client/server architecture) and 4(b) (P2P architecture). Figures 4(c) and 4(d) show the corresponding graphs for the video clip. Each graph shows the curves for three different numbers of clients. The savings in CPU utilization for the P2P architectures are clear, especially for large number of clients. In particular, for the video clip, the instantaneous CPU utilization drops roughly from 75% to 25% for 180 clients.



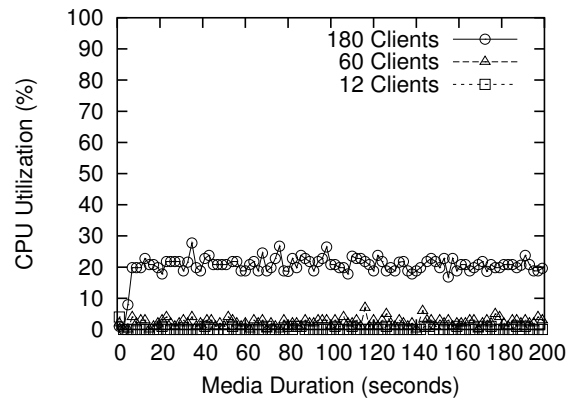
(a) Trailer, Client/Server Architecture



(b) Trailer, Peer to Peer Architecture



(c) Video Clip, Client/Server Architecture



(d) Video Clip, Peer to Peer Architecture

Figure 4. Instantaneous Server CPU Utilization

Figures 4(a) and 4(b) also show a great variability in the CPU utilization observed through time for the trailer, reflecting the variability in the instantaneous stream bitrate. This variability is due to a large number of scene changes, with action scenes followed by almost completely black screens. Note that the same behavior is observed for different number of clients and for both client/server and P2P systems, although it tends to be more significant as the load on the server increases. The video clip has a much less variable instantaneous CPU utilization. However, on average, the CPU utilization is much higher, because of the higher average bitrate (200kbytes/s). The music file, not shown in the Figures, has a roughly constant CPU utilization, with a maximum of around 10% for the client/server architecture with 180 clients. These results illustrate the diversity of our workload, as discussed in Section 3.3.

The average CPU utilization as a function of the num-

ber of clients is shown in Figure 5 for the video clip and the trailer. The curves for the music are omitted as they are close to 0. Similarly, the I/O network bandwidth, in Mbits/s, as a function of the total number of clients is shown in Figure 6 for the three files. All curves show an approximately linear increase in the requirements for server resources (CPU and network bandwidth) as the number of clients increases. However, for any given number of clients, the requirements for either CPU or network bandwidth are significantly lower in a P2P system than in a client/server system. Furthermore, for both metrics, the relative increase in requirements for server resources is less significant for the P2P systems than for the client/server systems. Another way of interpreting these results is that the cost of upgrading a server for handling a higher number of clients is lower if a P2P approach is used. As an example, in a client/server system, a server would have to dedicate, on average, 15 Mbits/s of its

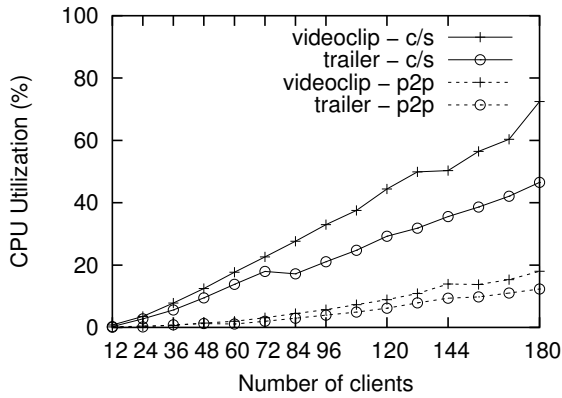


Figure 5. Average Server CPU utilization as a Function of the Number of Clients

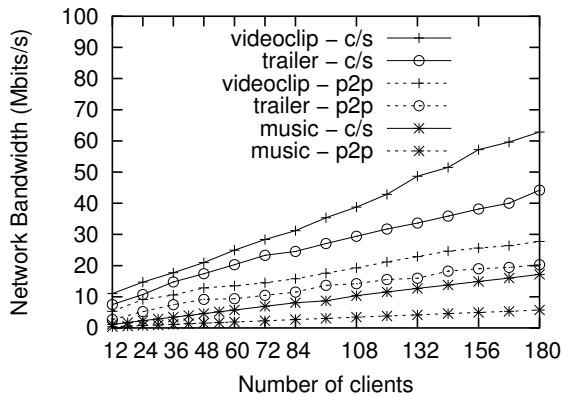


Figure 6. Average Server Network Bandwidth as a Function of the Number of Clients

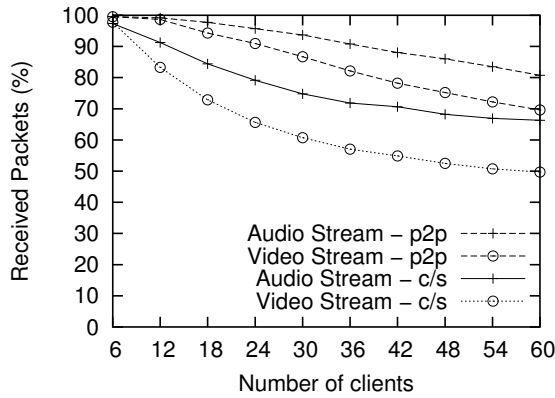
available network bandwidth to deliver the video clip to 24 clients. If the system has to handle requests 120 clients, the required average server bandwidth increases to 42 Mbits/s, 180% higher for an increase of a factor of five in the load. In a P2P system, on the other hand, the required average server bandwidth for delivering the video clip would be only 9Mbits/s for 24 clients and 21 Mbits/s (i.e., 130% higher) for 120 clients. Thus, in this hypothetical scenario it would be significantly cheaper to upgrade the server to handle the new load of 120 clients in a P2P system. Similar scenarios can be hypothesized for the two other files. These results quantitatively show the (intuitive) better scalability of the peer-to-peer approach.

4.2. Packet Loss

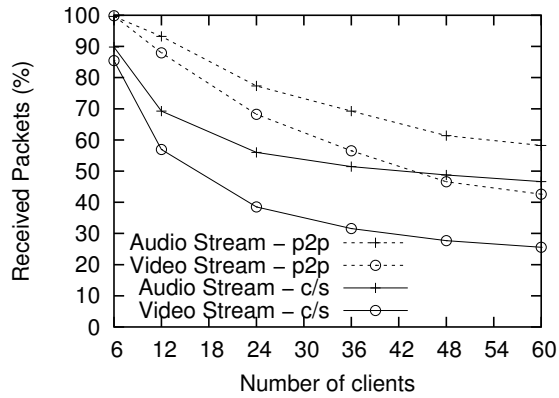
This Section presents results that show the scalability of client/server and P2P systems in terms of the quality of media received by the clients. As discussed in Section 3, we do not measure media quality directly. Instead, we use the number of packets received (or lost) in a stream as well as the distribution of the duration of packet loss events, measured as the number of consecutive packets lost in each loss event, as estimates of the quality of a stream. We point out that the duration of loss events needs special attention, as the impact of bursty losses on the quality of a stream can be quite significant. This Section shows only the results for the video clip and the trailer because we observed practically no packet loss during the transmission of the music file in the P2P systems and in the client/server systems with up to 84 clients.

Figure 7 shows the percentage of packets received, on average, by each client, as a function of the number of clients for client/server and P2P systems, for the trailer (Figure 7(a)) and for the video clip (Figure 7(b)). Since the transmission of each file actually occurs in two separate streams (an audio stream and a video stream), Figure 7 shows the percentage of packets received in each of these streams separately. As expected, the percentage of packets received, and consequently the media quality observed by the clients, decreases with the number of clients. In other words, the number of lost packets increases with the load on server and network. However, Figure 7 also shows that the P2P approach results in significantly higher percentages of received packets. As an example, considering the distribution of the video clip for 12 clients, the average percentage of packets received by each client in a P2P system is 88% (video packets) and 95% (audio packets) whereas the corresponding percentages for each client in a client/server system are only 56% and 70%, respectively. Furthermore, as the number of clients increases, the number of packets received in either video stream or audio stream decreases much more slowly in a P2P system than in a client/server system. These two results illustrate the better scalability of P2P systems.

It is also interesting to point out that, due to the higher average bitrate, the percentage of packets received is smaller for the video clip than for the trailer, for a fixed number of clients. Furthermore, note that, perhaps surprisingly, the number of lost packets is significant, especially for client/server systems, even for relatively small number of clients. We analyzed a series of experiments using *tcpdump* [5] and verified that, in those cases, many packets were not even being transmitted by the server. We conjecture that internal Darwin algorithms, such as the thinning algorithm, which drops packets if the packet transmission is running behind the schedule, are responsible for the large



(a) Trailer



(b) Videoclip

Figure 7. Percentage of Packets Received, on Average, by Each Client, as a Function of the Number of Clients

number of lost packets at least for the systems with small number of clients.

Figure 8 shows the instantaneous stream bitrate received by each client, averaged over all clients and measured at one-second intervals for the duration of each experiment with the trailer. It shows curves for P2P and client/server systems with two different numbers of clients. It also shows the instantaneous stream bitrate for only one client, which corresponds to the instantaneous bitrate of the file, since no packet was lost during the experiments with one client. This graph provides another view of the better scalability of P2P systems. For any given number of clients, P2P systems can sustain a stream bitrate that is much closer to the original file bitrate, not only on average, as shown in Figure 7 but also throughout the transmission of the file.

Figure 8 also shows the high variability in the instantaneous bitrate of the trailer, consistently with the discussion in Section 4.1. In particular, note that after approximately 75 seconds from the start of the transmission, the file bitrate decreases significantly. Visually, this corresponds to a very dark screen with very few elements in the video and almost no sound. Both P2P and client/server systems are able to deliver almost all packets corresponding to that scene even for large number of clients. The file bitrate increases again after a few seconds.

Figure 9 shows the histograms of the “duration” of each packet loss event, observed, on average, by each client. Recall that the duration of a loss event is measured as the number of consecutive packets lost in each event. Figure 9 shows the histograms for the video clip (Figures 9(c) and 9(d)) and for the trailer (Figures 9(a) and 9(b)). For each

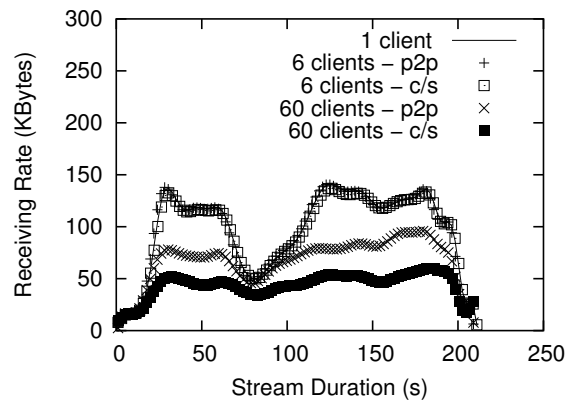
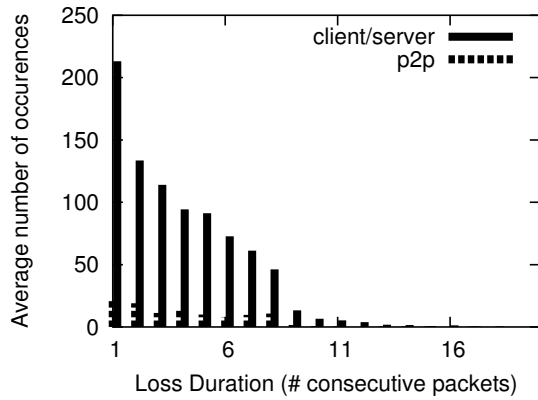
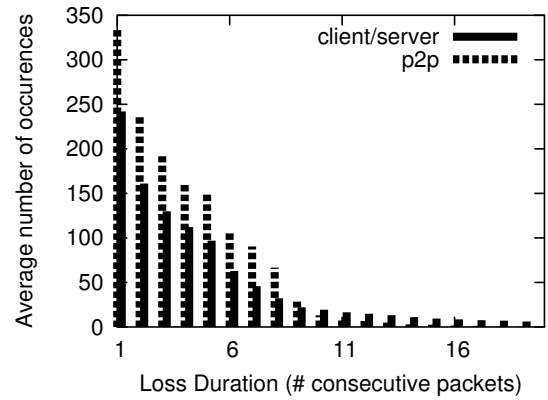


Figure 8. Instantaneous Stream Bitrate (KBytes/s) Received by Each Client (trailer)

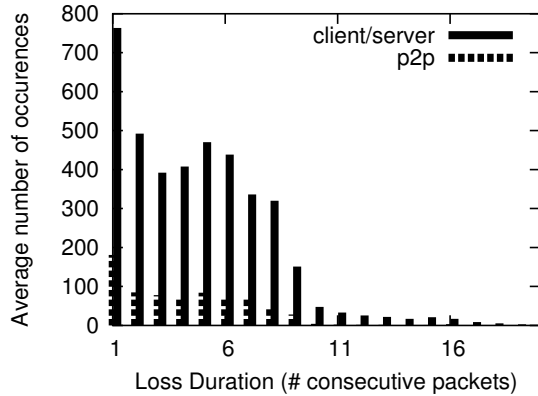
file, the histograms for 12 and 60 clients in both P2P and client/server architectures are provided. Note that the distribution of loss duration for the trailer, which has lower bitrate is more skewed towards a fewer number of consecutive packets. The graph shows that the number of occurrences of loss events of any given duration increases with the number of clients. Furthermore, in some cases, the distribution of loss durations for P2P systems seems more skewed towards smaller number of consecutive packets, suggesting a better expected media quality observed by the clients. As an example, in Figure 9(b), even though the number of occurrences of loss events with up to 9 consecutive packets is higher for P2P than for client/server systems, the average



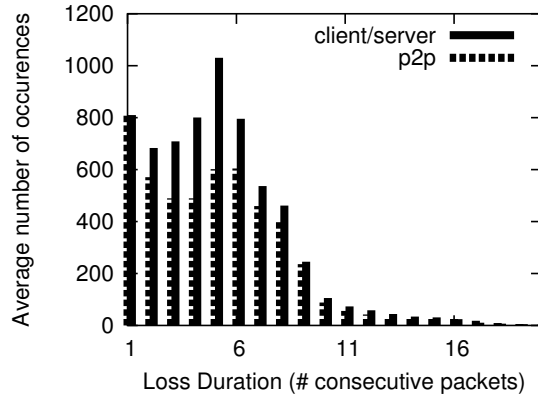
(a) Trailer, 12 clients



(b) Trailer, 60 clients



(c) Videoclip, 12 clients



(d) Videoclip, 60 clients

Figure 9. Histogram of Loss Duration

loss duration in the P2P system is just 3.8 consecutive packets, compared to 5.8 consecutive packets in the client/server system.

Table 2 summarizes the data in the histograms of Figure 9. It shows the total number of loss events and the average duration of loss events for each configuration. Note that, compared to the client/server approaches, either the average loss duration or the total number of loss events is shorter for the P2P systems with corresponding workloads. Thus, overall, the P2P systems are expected to deliver streams with significantly better quality to their clients.

4.3. Servent Load

Given the possibly high requirements for CPU and, especially, I/O network bandwidth for transmitting streaming media content, a client may not feel motivated to forward

content to other clients, jeopardizing the efficiency of P2P systems. In this section, we evaluate the load imposed on each servent as a function of the number of clients it forwards packets to, i.e., the number of clients that are directly connected to it in the P2P distribution tree. For this analysis, we run a new set of experiments with a single servent process running in one machine, receiving packets directly from Darwin and forwarding them to a varying number of clients, placed in the other five machines.

Figures 10 and 11 show the servent CPU utilization and output network bandwidth, respectively, as a function of the number of clients, respectively, for the three files. The curve for CPU utilization for the music file is omitted because it is very close to 0 across all number of clients considered. The “measured” curves contain the results of the new experiments. The curves labeled “analytical” will be discussed in the next section. The figures show that both CPU utiliza-

File	ct/s				p2p			
	12 clients		60 clients		12 clients		60 clients	
	Avg. # Events	Avg. Duration	Avg. # Events	Avg. Duration	Avg. # Events	Avg. Duration	Avg. # Events	Avg. Duration
trailer	862	3.80	1024	4.58	107	4.20	1412	3.80
video clip	3973	4.63	6467	4.95	721	4.12	5017	4.99

Table 2. Average Duration of Loss Events

tion and output network bandwidth grow (approximately) linearly with the number of clients and depend heavily on the bitrate of the file transmitted. CPU utilization remains below 10% for up to 10 clients for the video clip and 18 clients for the trailer. Thus only a tiny fraction of the server CPU time is required for forwarding packets to a small number of clients, say 2-5 clients, which is the server fan-out that we expect to find in real P2P streaming media systems. These results, combined with the results on packet loss discussed in section 4.2, show that if a client can dedicate enough network bandwidth to serve at least a couple of other clients, the expected benefits (i.e., less packet loss and, thus, better expected media quality) should serve as a strong incentive for it to participate in P2P distribution systems, because the impact on the client's CPU time is minimal.

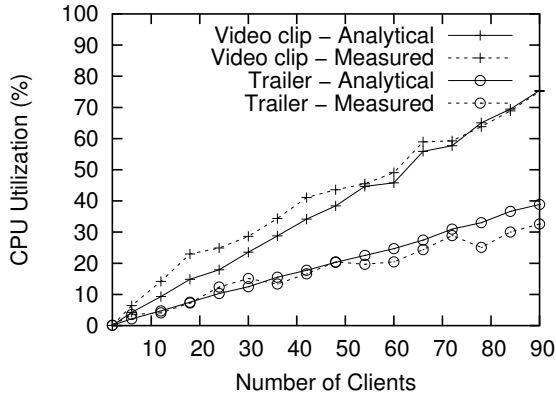


Figure 10. Servent CPU Utilization as a Function of the Number of Clients

4.4. Estimating Maximum Fan-out of a Servent

The curves labeled “analytical” in Figures 10 and 11 show the expected servent CPU utilization and output network bandwidth, calculated using simple analytical formulas. More precisely, we use Little's Result [11] to calculate the average servent CPU utilization. Since each file is trans-

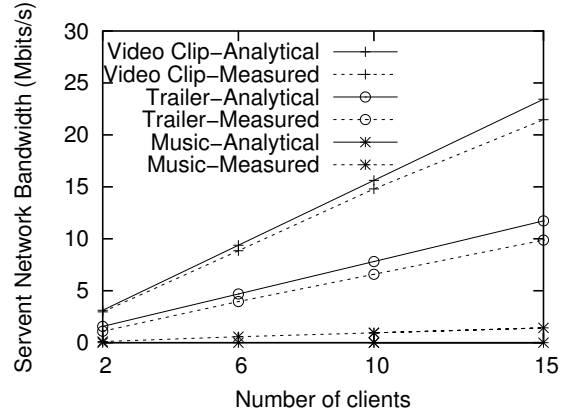


Figure 11. Servent Output Network Bandwidth as a Function of the Number of Clients

mitted in two separate streams (an audio stream and a video stream), which have different characteristics, in particular different packet size and packet rate, we apply Little's result separately to calculate the average CPU utilization required to process each stream, as follows:

$$\begin{aligned} U_{CPU,video} &= \lambda_{video} * S_{video}(n), \\ U_{CPU,audio} &= \lambda_{audio} * S_{audio}(n), \end{aligned} \quad (1)$$

where λ_{video} is the average file *video packet rate*, $S_{video}(n)$ the average CPU time spent by the servent processing each video packet, given that it has to forward packets to n clients and $U_{cpu,video}$ the average servent CPU utilization required. The values of λ_{audio} , $S_{audio}(n)$ and $U_{CPU,audio}$ are the corresponding measures for the audio stream. The average packet rates λ_{video} and λ_{audio} , specific of each file, and the values of $S_{video}(n)$ and $S_{audio}(n)$ are obtained from the data stored in the log file created by the servent during each experiment. The values of λ_{video} and λ_{audio} are 81.2 and 5.4 packets/second, respectively, for the trailer and 150.62 and 10.14 packets/second for the videoclip.

The overall average servent CPU utilization can be calculated as:

$$U_{CPU} = \frac{(\lambda_{video} * U_{CPU,video} + \lambda_{audio} * U_{cpu,audio})}{(\lambda_{video} + \lambda_{audio})}, \quad (2)$$

Note that the curves for CPU utilization, calculated by Equation (3), follow the behavior of the measured values very closely. Furthermore, the analytical curves are roughly straight lines, especially in the region where we expect a server to operate (i.e., small number of clients). Thus, Equations (1) can be rewritten as:

$$\begin{aligned} U_{CPU,video} &\approx \lambda_{video} * S_{video} * n, \\ U_{CPU,audio} &\approx \lambda_{audio} * S_{audio} * n, \end{aligned} \quad (3)$$

where S_{video} (S_{audio}) are the processing time to forward a video (audio) packet to a single client.

The server output network bandwidth can also be analytically estimated as:

$$BW = b * n, \quad (4)$$

where b is the file bitrate. As shown in Figure 11, also the server output network bandwidth can be analytically estimated with reasonable accuracy.

We can use Equations (2) and (4) to estimate the maximum fan-out F of a server (i.e., maximum number of clients directly connected to it in the P2P system), given the amount of resources it is willing to dedicate to forward packets of a given file. Given λ and b , characteristics of the file, S , the CPU time the server machine requires to process each packet and given that the server can only dedicate up to S_{CPU} of its CPU time and S_{bw} of its output network bandwidth, F can be estimated as:

$$F = \min(S_{CPU}/(\lambda * S), S_{bw}/b) \quad (5)$$

Equation (5) can be used to guide the construction and evaluate characteristics of a P2P distribution tree. For instance, assuming n homogeneous clients, i.e., clients with the same maximum fan-out F , calculated by (5), a lower-bound on the number L of levels in the P2P distribution tree built with these clients can be calculated as follows¹:

$$\begin{aligned} n &\leq \sum_{i=1}^L F^i \\ n &\leq (F^{L+1} - F)/(F - 1) \\ &= F/(F - 1) * (F^L - 1) \\ F^L &\geq n * (F - 1)/F + 1 \\ L &\geq \log_F (n * (F - 1)/F + 1) \end{aligned}$$

The number of levels in a distribution tree is key to assess the expected delay and loss observed by the clients, especially those in the lower levels. As an example, we can apply Equation (5) to determine the minimum number of levels a p2p distribution tree would have for distributing one of the three files used in this paper (say, the music) to 1000 clients, assuming that a client can dedicate up to 1 Mbps of

its output network bandwidth for content distribution. Figure 11 tells that the client has enough bandwidth to forward packets to up to approximately 10 clients. CPU utilization is minimal for the music file. Thus, the maximum client fan-out F is equal to 10. Applying $F=10$ and $n=1000$ in Equation (5) would give that the minimum number of levels in the p2p tree is 3. Thus, one could expect that some of the clients would observe a delay and loss equivalent to at least three hops in the tree.

We plan to further explore the application of Equation (5), especially together with algorithms for dynamically building the P2P distribution tree in the future.

5. Conclusions and Future Work

Scalability is arising as a key issue for effective distribution of live streaming media content. One strategy for improving the scalability is to distribute the stream employing a P2P overlay network, where each client may also act as a server (being called a server in this case), providing content to another clients. This work presents a experimental quantitative analysis of the aforementioned strategy and also models it analytically, comparing it to a traditional client/server approach where all clients request the stream to a single server.

The experimental results show that the P2P strategy results in significant reduction on server-side demands, namely computation resources and bandwidth. Further, it also reduces the demand increase rate as the number of clients increases, making the proposed approach more cost effective when providers evaluate upgrades and infrastructure enhancements. Significant gains are also observed when we evaluate the packet loss associated with each strategy, which is much smaller for the P2P-based strategy. On the server-side, the requirements in terms of computational resources are not significant, mainly when the number of clients served is small. Finally, the bandwidth requirements grow linearly with the number of clients, indicating the P2P strategy as being both scalable and cost-effective.

We also present an analytical model that allows us to derive the best fan-out for each server, showing that the P2P strategy is applicable to wide range of scenarios of computing platforms and connectivity.

We intend to extend this work by evaluating the benefits in actual network configurations and designing and evaluating resource discovery and allocation algorithms applied to the streaming distribution problem in P2P architectures.

References

- [1] Darwin Streaming Server.
<http://developer.apple.com/darwin/projects/streaming/>.

¹ For the sake of simplicity, we also assume F clients at the first level of the tree, i.e., F clients receiving content directly from Darwin.

- [2] Divx official site. <http://www.divx.com/>.
- [3] Mp3.com. <http://www.mp3.com/>.
- [4] Quicktime Player. <http://www.apple.com/quicktime/>.
- [5] Tcpdump. <http://www.tcpdump.org/>.
- [6] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming Live Media over a Peer-to-Peer Network. Technical report, Computer Science Department, Stanford University, Abril 2001.
- [7] D. Grimm. Quicktime Streaming Proxy Performance Improvements. <http://www.telecommunications.crc.org.au/content/ConfPapers/grimm.pdf>.
- [8] M. Hefeeda and B. Bhargava. On-demand Media Streaming Over the Internet. Technical report, Purdue University, 2002.
- [9] H. W. Holbrook and D. R. Cheriton. IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications. In *Proc. of the ACM SIGCOMM'99*, 1999.
- [10] K. A. Hua, D. A. Tran, and R. Villafane. Overlay Multicast for Video On Demand on the Internet. In *Proc. of ACM Symposium on Applied Computing*, 2003.
- [11] D. Menasce and V. Almeida. *Capacity Planning for Web Services: metrics, models and methods*, chapter Basic Performance Concepts. Prentice Hall, 2001.
- [12] D. Nogueira, L. Rocha, J. Santos, P. Araujo, V. Almeida, and W. Meira. A Methodology for Workload Characterization of File-sharing Peer-to-Peer Networks. In *Proc. of WWC-5: IEEE 5th Annual Workshop on Workload Characterization*, 2002.
- [13] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing Streaming Media Content using Cooperative Networking. In *Proc. of the 12th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2002)*, 2002.
- [14] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. of Multimedia Computing and Networking 2002 (MMCN'02)*, 2002.
- [15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, IETF, Janeiro 1996. <http://www.ietf.org/rfc/rfc1889.txt>.
- [16] H. Schulzrinne, A. Rao, and R. Lanphier. RTSP: Real-Time Streaming Protocol. RFC 2326, IETF, Abril 1998. <http://www.ietf.org/rfc/rfc2326.txt>.
- [17] D. Tran, K. Hua, and T. Do. ZIGZAG: An Efficient Peer-to-Peer Schema for Media Streaming. In *To appear at IEEE INFOCOM 2003*, 2003.
- [18] 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 2nd USENIX Conference on Internet Technologies and Systems (USITS '99)*, 1999.
- [19] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On Peer-to-Peer Media Streaming. In *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS 2002)*, 2002.