

Leveraging Broadband Access for True On-Demand Delivery of Internet Videos

Ying Cai Zhan Chen Johnny Wong

Department of Computer Science
Iowa State University
Ames, Iowa 50010
U.S.A.

Email: {yingcai, zchen, wong}@cs.iastate.edu

Abstract

We consider in this paper how the increasingly popular broadband access can be leveraged for scalable and cost-effective video service. Although much work has been done for such service, the existing techniques either are not designed for on-demand video delivery, or do not take into account the vastly improved client receiving capability. In this paper, we address this problem and develop three novel video delivery techniques with two ultimate goals: minimizing service latency and maximize data sharing. With the new techniques, a client can be served as soon as server resource becomes available and meanwhile, all its receiving bandwidth can be used for data receiving. In particular, in the proposed dynamic scheduling algorithm, the efficiency of data sharing is further improved by also exploring server bandwidth for early delivery of video segments. This innovation is contradicted to the intuition that each video segment should be delivered as late as possible in order to maximize data sharing. We evaluate the performance of our new techniques using simulation and our study convincingly shows that with the new schemes, zero service latency can indeed be achieved with very minimal system resource.

KEYWORDS: *video service, broadband, multicast, server bandwidth, service latency.*

1. Introduction

Not long ago, the only means for home residents to access Internet is using dial-up modems, typically running no faster than 56.6 Kbps. This situation has changed dramatically as a result of recent advance in network communication technology. Today, many new techniques are available for "last-mile" connections and they all come with signifi-

cant bandwidth improvement. For instances,

- ADSL (*Asymmetric Digital Subscriber Line*) provides a new broadband point-to-point Internet connection. It currently achieves 8 Mbps in one direction, and eventually can be as high as 50 Mbps.
- Cable modem is another highly popular way for broadband access. By replacing the current 300- to 450-MHz coax cables with 750-MHz coax cable, this scheme can achieve a total of 2 Gbps of new bandwidth for a group of users.
- In addition to ADSL and cable modem, PON (*Passive Optical Network*) is emerging as a new robust network with broadband access for home residents. This strategy consists of three parts, fiber-to-the-curb, fiber-to-the-loop, and fiber-to-the-home, trying to bring fiber as close to customers as possible. It supports switched wavelength services with bandwidth to customer at rates up to 622 Mbps downstream and 155 Mbps upstream, about 100 times faster than dedicated T-1 copper services.

While these new techniques offer tremendous bandwidth advantage, some of them have become cheap and are widely available now. Even at global economic downturn, the past few years have seen a significant growth of broadband market. In Japan, NTT communications activated its broadband communication services simultaneously in all of Japan's 47 prefectures on April 1, 2002 [18]. Its broadband content distribution services support on-demand streaming up to 6 Mbps. In U.S., the number of people who have broadband access jumped to 30 million in March 2003, up 50 percent from the same period in the previous year and more than double the number who had a high-speed connection at home at the end of 2001 [2]. According to a report from Multimedia Research Group, Inc (MRG) [1], the number

of worldwide broadband Internet subscribers will reach 54 millions in 2004.

The bandwidth advantage and the wide deployment of these techniques are spurring many important applications, making the Internet a more significant and powerful part of our lives. In this paper, we investigate how this vastly improved “last-mile” connection can be leveraged for efficient delivery of *Internet videos*, such as commercial advertisements, news clips, and the likes. In our research, we focus on *true* on-demand delivery of such videos, i.e., a video request is admitted for service as soon as the server resource becomes sufficient. We note that for many Internet videos, the timely response to their requests is particularly critical. As an example, many realtors now advertise their property on Internet with some video segments allowing so-called virtual home tours. When a client clicks on such a video, the video should be delivered immediately to the client for instant playback. Any delay to such service requests has the potential to lose some major business opportunity.

Our research is based on two outstanding features shared by most Internet videos. First, they are usually made with small display dimensions (e.g., 320x240 or 240x160, etc.) for Internet users. As a result, the playback rate of such videos is relatively low, say, from 128Kbps to 512Kbps. With the broadband access provided by the aforementioned techniques, a client is clearly capable of receiving data from many of such video streams simultaneously. Second, such videos are normally very short in nature, from a few seconds to a few minutes. Therefore, it is reasonable to assume that they can be buffered entirely at client sites, given the fact that one can hardly find a hard drive less than a few gigabytes in today’s storage market. We note that although many bandwidth-sharing techniques have been proposed over the years, they either are not designed for on-demand video delivery, or are just for the clients with very constrained receiving and/or buffering capability. To the best of our knowledge, the technique we present in this paper is the first one that is designed to leverage broadband access for true video-on-demand service.

The remainder of this paper is organized as follows. We review some related works and discuss their limitations in Section 2. In Section 3, we address these problems by presenting three novel video scheduling techniques. The performance study is presented in Section 4, and in Section 5, we give our concluding remarks.

2. Related Work

For more than one decade, video-on-demand has been an active research area. To support continuous video playback, the server resource is typically organized into many *video channels*, each sufficient to sustain one video stream. In general, a video stream flows through server disk I/O,

over backbone networks, to a remote client site. The communication interface between the video server and the backbone network determines the total number of video channels available to the server. This number is normally very limited because each video stream requires a significant amount of communication bandwidth. To provide scalable and affordable video service, each channel must be able to serve many clients simultaneously. This can be achieved by allowing many users to share channels using multicast. Many new techniques have been proposed:

- **Non-Periodic Multicast:** The server can make a video request wait for more peer requests to come and serve them together using one multicast. This is referred to as *batching* in [8]. The server can also dynamically accelerate or slow down the playback rate of on-going video streams and merge them together once they are at the same video frame. This *Piggybacking* scheme was proposed in [13, 19]. Some other non-periodic multicast schemes are proposed in [16, 7, 12, 22, 5, 14, 10, 11]. These techniques allow a client to join some existing multicasts and the server needs to send the client only the missing portion of the video.
- **Periodic Broadcast:** A video can be broadcasted periodically, i.e., a new stream is started every t minutes for each video. Some efficient techniques (e.g., Pyramid Broadcasting[23], Skyscraper Broadcasting[17], Pagoda Broadcasting[20, 21], etc.) have been developed for this purpose. In these techniques, each video file is partitioned into K fragments of increasing sizes, each repeatedly broadcast on a dedicated channel. On the receiving ends, the playback and download mechanisms are carefully arranged so that before the current fragment is consumed, the next fragment is always accessible to the client. Since the size of the first fragment can be made small, low service latencies can be achieved.

Traditionally, the disk buffer and in particular, the receiving bandwidth at client sites, are both very constrained; and it is under this assumption that most existing techniques were designed. Therefore, they have to compromise on various aspects. For instances, the client requirements under both Batching and Piggybacking are minimal: no disk buffer is required and each client needs to have only one-channel receiving capability. As a trade off, Batching puts most clients on waiting while Piggybacking needs to adjust the video playback rate on the fly, which is not trivial in practice and usually requires some specialized hardware.

Exploring broadband access for large-scaled video services was first studied in [15]. The proposed CCA (*Client-Centric Approach*) is a periodic broadcast technique designed for disseminating popular videos. Unlike other periodic broadcast techniques (e.g., Pyramid Broadcasting

[23, 4], Skyscraper Broadcasting [17], etc.), CCA takes both server broadcast bandwidth and client receiving capability into consideration. Specifically, given the same server broadcast bandwidth, CCA is able to achieve less broadcast latency with improved client receiving capability. In the case that each client can download data simultaneously from all broadcast channels, CCA can reduce the broadcast latency exponentially with respect to the broadcast bandwidth: if n channels are used to broadcast a video, then the worst service latency can be minimized to $\frac{1}{2^n - 1}$ of the video length.

Although CCA can effectively leverage broadband access for more efficient video delivery, it is a periodic broadcast technique and can only be used for popular videos. To support videos with more diversity, a new technique called *Full-Sharing* was proposed in [9]. Full-Sharing is designed specifically for broadband cable networks, but its concept can be applied in general. This scheme assumes each client can buffer a video in its entirety and has K -channel receiving capability, where K is the number of channels used by the server to deliver each video. In Full-Sharing, a video is partitioned into c clips, each has the same playback duration, say, T time units. Accordingly, the global time is divided into intervals with a fixed length of T time units and at the beginning of each interval, the server attempts to schedule service for pending requests, if any, as follows. It first scans all channels and find out the clips that have been scheduled for delivery. Since these video clips are *shareable* to the new client, the server needs to send only the missing clips. Given a missing clip, Full-Sharing tries to deliver it as late as possible, hopefully just before its playback time, to maximize sharing for future requests. It is possible, however, a clip cannot be scheduled for on-time delivery because all channels have been occupied. When this happens, the server either delays the service to the next scheduling period, or tries to adjust the existing delivery schedule to accommodate the new clip. If all missing clips can be successfully scheduled, the client is notified with a *transmission schedule*, containing the information about where and when to download each video clip.

Unlike those periodic broadcast techniques, Full-Sharing does not have to start a new video stream unless some video request arrives in the previous batching period (i.e., T time units). In addition, it significantly reduces the cost of serving a new client by assuming it is able to buffer an entire video. This technique, however, is not designed for the applications that require on-demand video delivery. Because of its batching nature, each client has to wait until the end of the current batching period. Even with an unlimited bandwidth, the average service latency in this scheme is constant to be $\frac{T}{2}$ time units.

3. Proposed Techniques

In this section, we present our new techniques. Without loss of generality, we consider only one video. If the system has n videos, we can divide the server bandwidth for n virtual servers, each serves one video. We assume the client receiving capability is K channels and each client has enough disk space to accommodate the entire video. At the server site, most resource is dedicated for video delivery while a small amount of bandwidth is reserved for control messages (e.g., service notifications, etc.). In our design, a video request is scheduled for service immediately if the server has sufficient resource. Initially, the server allocates a set of K channels to serve video requests. If any request cannot be served immediately because of high request rate, the server allocates another set of K channels and so forth until all channels are exhausted. Since the overall system performance is determined by the throughput of each set of K channels, our research focuses on providing on-demand video service for as many clients as possible with a single set of channels. Therefore, we will assume the server has only K channels in the remainder of this paper.

3.1 Client Design

When a client submits a video request, it immediately starts to download data from all K channels. The received data are temporarily saved to the client's local disk. The client starts the video playback after it receives the first time unit of the video data. Our server design guarantees the client playback continuity by ensuring that all remaining data will arrive before their individual playback time.

In most cases, a client will receive the first time unit of the video data as soon as it submits its video request and therefore, can start the playback immediately. However, when the server is overloaded, the client may receive the first time unit of the video data after some time period, i.e., service latency. In our design, a client starts its download activity as soon as it submits its video request, even though it may not start playback until the first time unit of the video data arrives. The video data received during the service latency are buffered and used for future playback. As we will see shortly, this innovation allows us to exploit both server and client resource more efficiently. Under existing techniques, a client either simply waits until its service time arrives, or discards all data received before its service time.

We also note that many techniques (e.g., Patching, Full-Sharing, etc.) require each client to have specific receiving schedule. As a result, if there is any change on its delivery schedule, the server must find out all affected clients and needs to update them with the new schedule. In contrast, our design allows a client to download and playback video segments regardless of the server schedule. Therefore, our

technique incurs less management overhead and has lower implementation cost.

3.2 Server Design

At the server side, each video channel is associated with a workload, which is a list of item $\{V[i, j], t\}$, saying that this channel will be used to deliver video segment $V[i, j]$, starting at time t . In the rest of this paper, we will use $V[i, j]$ to denote the video segment in between the playback time i and j of the video, where $0 \leq i < j$. If a channel is free of workload from time t_a to t_b , where $0 \leq t_a < t_b$, then the time period is a *vacant slot*, denoted as $[t_a, t_b]$. Given a vacant slot $[t_a, t_b]$, we will call t_a and t_b as the *left* and *right* boundary of the slot, respectively.

When the server receives a video request, it scans the workload of each channel and finds out all video segments that have been scheduled for delivery. Since the new client can also download these video segments, the server needs to deliver only the missing segments. Assume n segments are missing: $V[0, y_1], V[x_2, y_2], \dots$, and $V[x_n, y_n]$, where $0 < y_1 < x_2 < y_2 < \dots < x_n < y_n$. These segments should be delivered with the following two principles:

- The delivery of $V[0, y_1]$ must start as early as possible. This is to minimize service latency.
- The remaining segments must be delivered before their playback time. That is, if the delivery of $V[0, y_1]$ starts at time t_0 , then the delivery of segment $V[x_i, y_i]$, where $2 \leq i \leq n$, must start no later than time $t_0 + x_i$. This is to guarantee the client playback continuity.

In our design, the server schedules the delivery of the remaining segments sequentially in the order of $V[x_2, y_2], V[x_3, y_3], \dots$, and $V[x_n, y_n]$. Given a segment $V[x_i, y_i]$, the server can easily find out all vacant slots on each channel between time t_0 and $t_0 + y_i$. The challenge is how to use these vacant slots to deliver $V[x_i, y_i]$. In following subsections, we address this problem and present three scheduling algorithms. We say a schedule *fails* if any segment cannot be scheduled before its playback time. When this happens, the server needs to find the next earliest delivery time for $V[0, y_1]$ and reschedules the remaining segments accordingly. This process is repeated until all segments are scheduled successfully.

3.2.1 Segment-Based Scheduling (SBS)

The design of this scheme is based on the fact that a slot $[t_a, t_b]$ can be used to deliver $V[x_i, y_i]$ if the slot is not smaller than $y_i - x_i$ (i.e., $y_i - x_i \leq t_b - t_a$). To schedule the delivery of $V[x_i, y_i]$, this scheme finds out all vacant slots between time t_0 and $t_0 + y_i$ and large enough to accommodate $V[x_i, y_i]$. If no such slots exist, the schedule fails.

Otherwise, among these slots, the server finds out the one whose right boundary is closest to time $t_0 + y_i$. This slot, say, $[t_a, t_b]$, will be used to deliver $V[x_i, y_i]$ starting from time $t_b - (y_i - x_i)$. This scheduling algorithm is illustrated in Figure 1 using an example. We call this scheme *segment-based* because it schedules the delivery of each segment in its entirety.

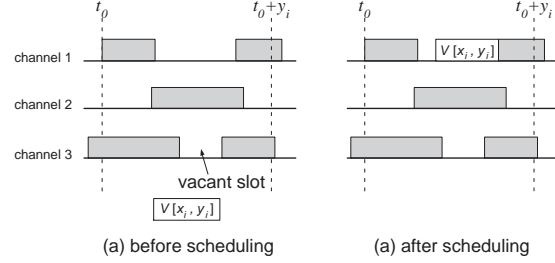


Figure 1. Segment-Based Scheduling

3.2.2 Channel-Based Scheduling (CBS)

In the previous approach, the vacant slot used to deliver a segment must be large enough to hold the entire segment. If all vacant slots are very small, the schedule will fail and the server has to delay the delivery of $V[0, y_1]$, increasing the service latency. It is possible, however, we can split a segment into several small segments so that they can be delivered individually using small vacant slots. Figure 2 illustrates such a scenario. Suppose the delivery of $V[x_i, y_i]$ must be started in between time t_0 and $t_0 + x_i$. According to the segment-based approach, the schedule will fail because none of the vacant slots in between time t_0 and $t_0 + y_i$ is large enough for this segment. However, if we partition $V[x_i, y_i]$ into two segments, $V[x_i, m]$ and $V[m, y_i]$, then each can be accommodated by one vacant slot. Since $V[x_i, m]$ is delivered ahead of its playback time while $V[m, y_i]$ is transmitted on time, the client playback continuity is guaranteed.

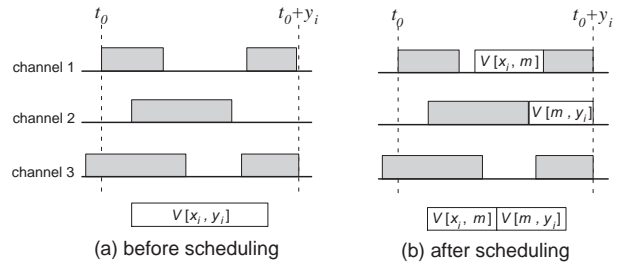


Figure 2. Channel-Based Scheduling

From the above motivation example, we develop a *channel-based* scheduling algorithm. To schedule the de-

livery of $V[x_i, y_i]$, this scheme finds out all vacant slots on each channel in between time t_0 and $t_0 + y_i$. If no vacant slots exist, the scheduling fails. Otherwise, the server finds out the slot whose right boundary is closest to time $t_0 + y_i$. Let this slot be $[t_a, t_b]$. If $y_i - x_i \leq t_b - t_a$, then $V[x_i, y_i]$ can be transmitted in its entirety without split and the scheduling of this segment is done. Otherwise, the server splits $V[x_i, y_i]$ into $V[x_i, x_i + t_b - t_a]$ and $V[x_i + t_b - t_a, y_i]$. Since $V[x_i + t_b - t_a, y_i]$ can be delivered using slot $[t_a, t_b]$, the server just needs to find other slots to deliver $V[x_i, x_i + t_b - t_a]$ before its playback time. This can be done by finding out all vacant slots between time t_0 and $t_0 + x_i + t_b - t_a$ and recursively applying the above process.

3.2.3 Dynamic Scheduling (DS)

Both of the above two algorithms try to deliver the missing data as late as possible but before their playback time. Intuitively, this strategy maximizes the data sharing for future requests. This, unfortunately, wastes the precious bandwidth in many cases. As an example, consider Figure 3. Initially, all channels are free. Suppose two clients arrive at time 0 and time 3, respectively. For the first client, the server uses the first channel to deliver $V[0, 9]$, starting at time 0. For the second client, the server uses the second channel to deliver $V[0, 2]$, starting at time 3. We observe that the vacant slot $[0, 2]$ on the second channel is wasted.

To address this problem, we can use vacant slot $[0, 2]$ to deliver the last three time units of the video, $V[9]$, $V[8]$, and $V[7]$. To serve the second client, the server delivers two segments, $V[0, 2]$ and $V[7, 9]$. It first seems that the server has to deliver $V[7, 9]$ again, but the delivery of this segment is actually delayed 3 time units, as compared to the previous approach. As a result, $V[7, 9]$ can be shared by the clients arriving in the next 3 time units.

The above example shows a scenario that delivering a video segment in an earlier time can actually improve the efficiency of data sharing, even though the same video segment may have to be delivered again in a later time. Based on this observation, we propose a *dynamic* scheduling algorithm. When a new request arrives, this scheme uses the same algorithm as the channel-based approach to schedule the delivery of each missing segment. The actual delivery of a segment, however, may be different from its initial schedule. Whenever a channel becomes free, the server finds out the segment that was scheduled for the last delivery. This segment is then delivered on the channel in a *reverse* order, i.e., the last frame is delivered first, then the second last frame, and so forth. If a new client arrives before the delivery of the entire segment is finished, then the unfinished portion will be delivered according to its original schedule. This scheme keeps all channels busy until all workloads are

finished.

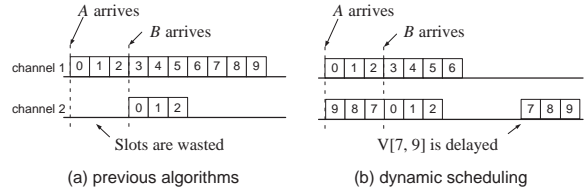


Figure 3. Dynamic Scheduling

4. Performance Study

In this section, we compare the performance of the proposed techniques with that of *Optimal Patching* [7, 6]¹. Without loss of generality, we assume in our study that the system has only one video. In addition, if a client has K -channel receiving capability, then the server will allocate only K channels to serve all requests. As we have discussed previously, the overall system performance is determined by the throughput of each set of K channels. Thus, the results reported in this section should fairly reflect the relative performance of these techniques in the systems with many channels.

The performance parameters are given in Table 1. We note that today's broadband service typically provides a bandwidth from 512Kbps to 2Mbps. Therefore, we vary the client receiving capability from 2 channels to 8 channels, assuming the video is compressed with a playback rate of 256Kbps. In our simulation, we assume the video request arrival follows a Poisson distribution and each simulation lasts 10 hours.

Parameter	default	variation
Client receiving bandwidth (channels)	4	2-8
Mean request interval $\frac{1}{\lambda}$ (seconds)	30	10 - 60
Video length $ v $ (minutes)	5	1 - 10

Table 1. Parameters

4.1 Effect of Request Inter-Arrival Time

In this study, we perform sensitive analysis with respect to the request inter-arrival time. The client receiving bandwidth is fixed at 4 channels and the video is assumed to be 5 minutes long. We vary the mean request inter-arrival interval from 10 seconds to 60 seconds. The results are plotted

¹Because of limited resource, we are not able to implement all existing techniques. With the assumption of broadband access and large buffer, a client in our techniques can share all video segments on all channels. Therefore, it is reasonable to believe that the proposed schemes will outperform the existing techniques to some degree.

in Figure 4(a). It shows that under all scenarios, Patching consistently incurs more service latency. In particular, the performance gap widens with a higher video request rate. We note that in Patching, a client can share data only from the channel that is used most recently to multicast the entire video. In contrast, a client with our new scheduling approaches can share all video segments on all channels, reducing the service cost for the client. As for the three new techniques, the performance of segment-based scheduling is always the worst. To deliver each missing segment in its entirety, this scheme needs to find a vacant slot to accommodate each segment and if no such slot exists, this scheme would have to delay the service. Channel-based scheme addresses this problem by splitting the segments according to the sizes of the vacant slots so that the scheduling can proceed even with small slots. Figure 4(a) shows that this strategy significantly improves the system performance. For example, when the request inter-arrival interval is 10 seconds, its service latency is about 30% of that of segment-based scheduling. The performance study also shows that by efficiently utilizing all available bandwidth, dynamic scheduling outperforms all other techniques and lifts the system performance to a new level. When the mean inter-arrival time is 25 seconds, it achieves close-to-zero average service latency. As a contrast, the service latency is about 2, 7, 16 seconds, respectively, for the other three techniques. This study confirms our argument that the “wasted” bandwidth can indeed be used to deliver some video data in an earlier time to maximize data sharing and improve system throughput.

4.2 Effect of Client Receiving Bandwidth

In this study, we fix the video length at 5 minutes and the mean inter-request arrival time at 30 seconds. The client receiving bandwidth is increased from 2 channels to 8 channels. The performance results are shown in Figure 4(b). Again, in all scenarios, the performance of Patching is the worst. Given two-channel receiving capability at the client sites, the average service latency under Patching is nearly 90 seconds. To the contrary, the three new techniques make clients wait no more than 40 seconds. The figure shows that the service latency under all four schemes is reduced dramatically as the number of channels increases. We note that for Patching, this is simply a result of that more bandwidth is allocated to serve the video requests, because clients under Patching can download data from a fixed number of channels. The design of the three new techniques, however, explicitly takes the broadband access into account. For these techniques, the increase of receiving bandwidth also means that a client can share data from more channels; and this is another key factor that contributes to their performance enhancement. As Figure 4(b), the service latency

under dynamic and channel-based scheduling is reduced to zero when the channel number is increased to five.

4.3 Effect of Video Length

The purpose of this study is to investigate how the video length affects the performance of the four techniques. We increase the video length from 1 to 10 minutes, while the number of channels is fixed at four and inter-request arrival time at 30 seconds. Figure 4(c) shows that as the video length increases, the Patching curve goes up sharply while the curves for the three new techniques are quite flat. For example, all four techniques achieve zero latency when the video length is 1 minute. When the video length is increased to 10 minutes, the service latency under Patching is 60 seconds. As a comparison, the worst segment-based scheduling makes the clients wait no more than 20 seconds on the average. This study indicates that Patching is highly sensitive to the video length and its performance deteriorates quickly with the increase of video length. Again, this is due to the fact that the three new techniques allow clients to share all segments in all video channels. As for the three new techniques, we observe again that the segment-based scheduling is less efficient than channel-based and dynamic scheduling. When the video length is very short, all of them achieve zero service latency. However, as the video length increases, the performance gaps become obvious. When the video length is 10 minutes, the segment-based scheduling incurs about 200% more service latency than dynamic scheduling.

5. Concluding Remarks and Future Works

Many people believe that true broadband access is the key to the next generation of communications and its widespread adoption will have a tremendous impact on our society. To achieve the full potential of Internet, policymakers in United States have been called to make broadband a national priority and to set a goal of making an affordable 100 Mbps broadband connection available to 100 million American homes and small business by 2010 [3]. The popularity of broadband access will enable many content-rich applications that are impossible with traditional dial-up connections. In this paper, we consider how broadband can be leveraged for large-scale on-demand video services. We presented three efficient video scheduling techniques, by which the vastly available client receiving bandwidth can be fully explored to maximize data sharing and minimize service latency at the same time. We are currently extending these techniques to make them adaptive to the heterogeneous networks, where different clients may have different connection bandwidth and even for the same client, its receiving capability could fluctuate vastly at different time.

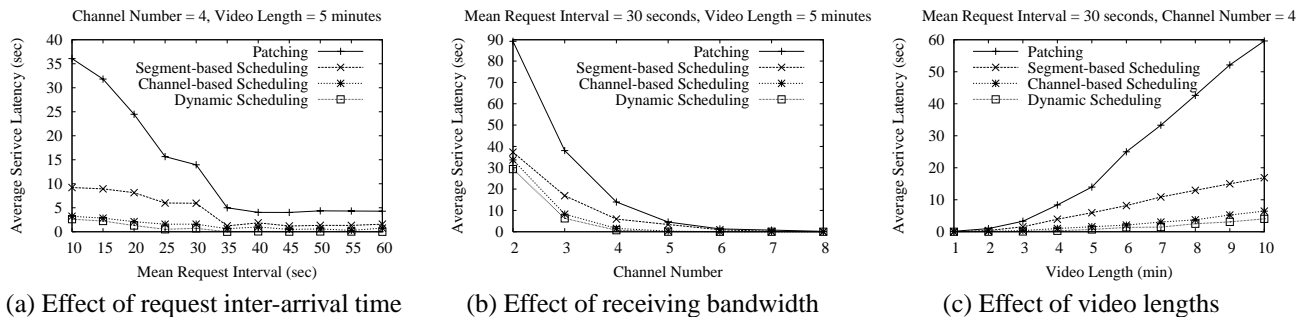


Figure 4. Performance Figures

Apparently, the work reported in this paper is just one step toward providing universal video services.

References

- [1] Web page at <http://www.mrgco.com/Reports.html>.
- [2] Broadband Adoption at Home: A Pew Internet Project Data Memo. Web page at <http://www.pewinternet.org/reports/toc.asp?Report=90>.
- [3] A national imperative: Universal availability of broadband by 2010. Web page at <http://www.technet.org/news/newsreleases/2002-01-15.64.pdf>.
- [4] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A Permutation-based Pyramid Broadcasting Scheme for Video-on-Demand Systems. In *Proc. of the IEEE Int'l Conf. on Multimedia Systems '96*, Hiroshima, Japan, June 1996.
- [5] Y. Cai and K. A. Hua. An Efficient Bandwidth-Sharing Technique for True Video on Demand Systems. In *Proc. of ACM Multimedia '99*, pages 211–214, Orlando, FL, USA, October 1999.
- [6] Y. Cai and K. A. Hua. Sharing Multicast Videos Using Patching Streams. *Multimedia Tools and Applications*, 21, 2003.
- [7] Y. Cai, K. A. Hua, and K. Vu. Optimizing Patching Performance. In *Proc. of SPIE's Conf. on Multimedia Computing and Networking (MMCN'99)*, pages 204–216, San Jose, CA, USA, January 1999.
- [8] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *Proc. of ACM Multimedia*, pages 15–23, San Francisco, California, October 1994.
- [9] Y. Dong, Z. L. Zhang, and D. H.-C. Du. Optimal Scheduling and Adaptation for VOD Service on Broadband Cable Networks. In *Packet Video 2003*, Nantes, France, April 2003.
- [10] D. Eager, M. Vernon, and J. Zahorjan. Optimal and Efficient Merging Schedules for Video-on-Demand Servers. In *Proc. ACM Multimedia '99*, pages 199–202, Orlando, FL, November 1999.
- [11] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand. In *Proc. of SPIE's Conf. on Multimedia Computing and Networking (MMCN'00)*, pages 206–215, San Jose, CA, January 2000.
- [12] L. Gao and D. Towsley. Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast. In *Proc. IEEE International Conference on Multimedia Computing and Systems*, pages 117–121, Florence, Italy, June 1999.
- [13] L. Golubchik, J. Lui, and R. Muntz. Adaptive Piggybacking: a Novel Technique for Data Sharing in Video-on-Demand Storage Servers. *ACM Multimedia Systems*, 4(3):140–155, 1996.
- [14] C. Griwodz, M. Liepert, M. Zink, and R. Steinmetz. Tune to Lambda Patching. *ACM Performance Evaluation Review*, 27(4):20–26, March 2000.
- [15] K. A. Hua, Y. Cai, and S. Sheu. Exploiting Client Bandwidth for More Efficient Video Broadcast. In *Proc. of Int'l Conference on Computer Communication and Networking*, pages 848–856, Louisiana, U.S.A., October 1998.
- [16] K. A. Hua, Y. Cai, and S. Sheu. Patching: A Multicast Technique for True Video-on-Demand Services. In *Proc. of ACM Multimedia*, pages 191–200, Bristol, U.K., September 1998.
- [17] K. A. Hua and S. Sheu. Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-On-Demand Systems. In *Proc. of the ACM SIGCOMM'97*, Cannes, France, September 1997.
- [18] E. Kuwana, T. Yahara, and Y. Hoshi. System Design Issues for Broadband Content Delivery Services. In *Proc. of Symposium on Applications and the Internet (SAINT'03)*, Orlando, FL, USA, January 2003.
- [19] S. Lau, J. Lui, and L. Golubchik. Merging Video Streams in a Multimedia Storage Server: Complexity and Heuristics. *ACM Multimedia Systems*, 6:29–42, 1998.
- [20] J. F. Paris, S. W. Carter, and D. D. E. Long. Efficient Broadcasting Protocols for Video on Demand. In *Proc. of SPIE's Conf. on Multimedia Computing and Networking (MMCN'99)*, pages 317–326, San Jose, CA, USA, January 1999.
- [21] J. F. Paris, D. D. E. Long, and P. E. Mantey. Zero-Delay Broadcasting Protocols for Video-on-Demand. In *Proc. of ACM Multimedia*, pages 189–197, Orlando, FL, USA, November 1999.
- [22] S. Sen, L. Gao, J. Rexford, and D. Towsley. Optimal Patching Schemes for Efficient Multimedia Streaming. In *Proc. IEEE NOSSDAV'99*, Basking Ridge, NJ, U.S.A., June 1999.
- [23] S. Viswanathan and T. Imielinski. Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting. *Multimedia systems*, 4(4):179–208, August 1996.