Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 03-035

An Architecture for Proxy-Assisted Periodic Broadcast for Large Scale Video Streaming

Ewa Kusmierek, David Du, and Yingfei Dong

September 17, 2003

An Architecture of Proxy-Assisted Periodic Broadcast for Large-Scale Video Streaming

Ewa Kusmierek, David H.C. Du and Yingfei Dong
Digital Technology Center and
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
{kusmiere, du, dong}@cs.umn.edu

***Abstract*— Many multimedia applications rely on video streaming techniques. However, large scale video delivery is still very challenging since it requires a large amount of resources such as storage space, network bandwidth and I/O bandwidth. In this paper we propose a proxy-assisted periodic broadcast architecture for video delivery to a large number of clients over the Internet. Our video delivery technique is based on a combination of periodic broadcast by central server and proxy server caching. A proxy server caches either part or the whole video based on the video popularity. We assume that each proxy server may have different capability and that the video popularity in each community can be different and dynamically changing. A video stored in the central server is partitioned into two parts, a server prefix and a server suffix, based on the aggregated demand for the video from all communities. In principle, the server prefix is delivered by unicast and the server suffix is delivered by periodic broadcast. Such an approach allows to significantly reduce the required I/O bandwidth at a server. The combination of proxy prefix and server prefix defines a wide spectrum of different video delivery modes. The transmission of a video can be either partially unicast or partially period broadcast depending on the relationship between proxy prefix and server prefix. We further define and solve the optimization problems for proxy prefix selection and server prefix selection in order to minimize the total resource requirements. Performance of our system is evaluated through a number of tests.**

Keywords: System design, Mathematical programming/optimization,

I. INTRODUCTION

A number of multimedia applications such as distance learning, digital library, video-conferencing, and entertainment on-demand rely on the technique of streaming stored or real-time video. Video-on-Demand (VoD) enables users to select a video from a server and view it on a local device. Data is transferred over the communication network using streaming techniques, i.e., the client can start the playback of the video almost instantly and process the data as a continuous stream. This area has received quite a bit of research attention in the last ten years. However, it still remains very challenging to provide VoD service t millions of subscribers over Wide Area Networks (WANs).

In this paper we assume that a central server works together with a number of proxy servers, one per community, to deliver video to a large number of clients. The central server and proxy servers are connected via WAN and the clients in a community are connected to its associated proxy via Local Area Networks (LANs). An efficient architecture for such an environment is a cost-effective way to satisfy all clients' demand in each community based on its viewing profile by accessing videos either from its proxy server or a central video server. The cost involved includes the required WAN bandwidth, the capacity of the central video server as well as the capacity of each proxy server. It is also important to reduce the buffer requirements and the processing power of each client.

The goal of this paper is to propose such an architecture for large-scale video streaming based on two essential techniques: periodic broadcast (PB) [1] and proxy caching [2, 3]. The buffer space at each proxy is allocated to minimize the aggregate network bandwidth usage between a server and a client. The periodic broadcast is used by the central video server to save the server I/O bandwidth requirement for popular videos. Our main contribution is an efficient server transmission schedule as well as a proxy prefix selection algorithm for a given set of videos of different popularities for different proxies. In the proposed architecture, a video prefix can be cached by a proxy based on the local popularity of a video in the corresponding community. The proxy prefix is accessed

This work is partially supported by NSF Grant EIA-0224424

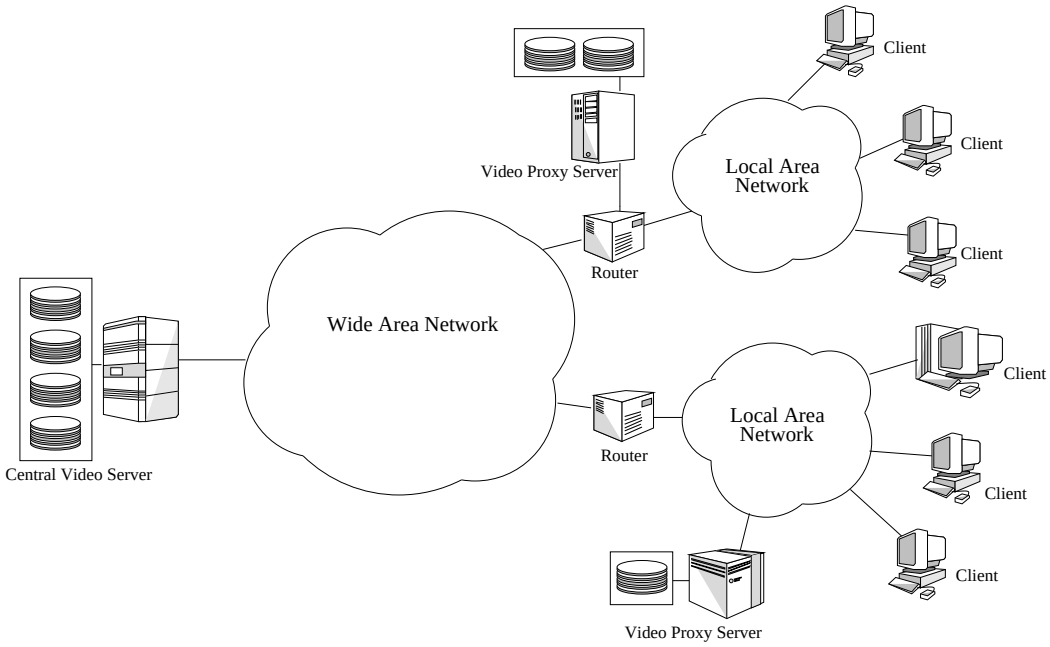


Fig. 1. Video delivery system architecture

via unicast by a client from the proxy and the rest of the video is accessed from the central server via either unicast or periodic broadcast depending on the global popularity (i.e., the aggregated demand from all communities) of the video.

In the proposed architecture the central server can choose not only between different transmission modes for any given video but can also partition a video into prefix and suffix delivered in different ways. By introducing a concept of server prefix in addition to proxy prefix, we can provide a wide spectrum of video transmission modes. Such flexibility allows us to choose a transmission schedule that results in optimal server I/O bandwidth usage. By combining an efficient caching scheme with an efficient server transmission mechanism we are able to optimize I/O bandwidth and network bandwidth requirements for large scale video delivery.

This paper is organized as follows. In Section II we present an overview of the proposed architecture. In Section III we formulate an optimization problem to determine the best prefix size for each proxy server and central video server. In Section IV we discuss issues related to periodic broadcast schemes. We present a heuristic approach to solving the problem in Section V. We evaluate the proposed solution in Section VI. Section VII contains a brief summary of the related work and in Section VIII we conclude the paper.

II. Proposed Two-Level Architecture

We first describe the proposed architecture and state our assumptions about its components. Next we address the issues of resource requirements in a *large-scale* system and describe the mechanism that allows us to improve the scalability and efficiency of such a system.

A. Architecture Components

Our proposed architecture consists of a central server providing access to a large set of videos, and a large number of clients from many communities spread over the Internet as illustrated in Figure 1. All clients from a community are connected to the same Local Area Network (LAN), the same cable head-end or the same Internet Service Provider. Each community has a proxy server to serve its clients and the network bandwidth between the proxy and the clients is considered adequate and inexpensive. We assume that the central server has an adequate storage, processing and I/O bandwidth capacity to satisfy the demand and the connections between the central server and proxies are through IP based WAN. The storage space and I/O bandwidth available at the proxy servers and the clients are limited. Our goal is to provide access to a set of Constant Bit Rate videos to a potentially large number of clients in such a way that the resource consumption, i.e., central server I/O bandwidth and WAN bandwidth, is minimized, subject to the resource availability at proxies and clients. We assume that the proxies may have different capabilities in terms of buffer space and I/O bandwidth (heterogeneous type of proxies). The communities that they serve may also be diverse in size (number of clients) and video popularity. The popularity varies not only from one community to another, but also over time

in a single community.

The core of our architecture is a two-level video delivery mechanism. It utilizes two techniques: *video caching* at proxy servers and *periodic broadcast* at the central server. Our mechanism combines these two techniques and introduces another level of scalability on top of them. Periodic broadcast schemes partition a video into a number of segments. The segments are repeatedly transmitted over a number of broadcast channels. Each client collects video segments from these channels and buffers them until their playback times. The total bandwidth required for broadcast transmission does not depend on the number of clients, but only on the segmentation of the video and the transmission rate of each channel. Therefore, periodic broadcast is an efficient way to save server I/O bandwidth to satisfy a large number of requests for popular videos. Video caching by a proxy further reduces network bandwidth consumption. A proxy caches a part of the video to decrease the amount of data that has to be delivered by the central server. We choose to utilize proxy resources by caching the initial portion of a video to reduce start-up delay. Therefore, a client in a given community receives proxy prefix from the proxy and the suffix from the central server.

The idea behind our two-level video delivery mechanism is based on two observations. First, in a large-scale system, video prefixes selected by proxy servers for caching may vary a lot from one community to another. Thus, it may not be efficient to have the same prefix cached by all proxies. Second, periodic broadcast reduces bandwidth requirements only for very popular videos. Only for such popular videos the bandwidth requirement of a broadcast transmission is lower than the sum of the requirements of individually unicasted transmissions. Therefore, popularity should determine the mode of transmission. Due to different proxy prefix sizes, the aggregate popularity may be different for different portions of the same video at the central server and the video should be partitioned into unicast-delivered and multicast-delivered parts.

We introduce the concept of *central server prefix* and utilize the concept of *proxy prefix*. A video prefix selected by a proxy determines which part of the video a client can obtain from a proxy in its own community and which part has to be requested from a central server. The server prefix size determines which part of video is transmitted in a unicast mode and which part in a periodic broadcast mode.

B. Limitations of Periodic Broadcast

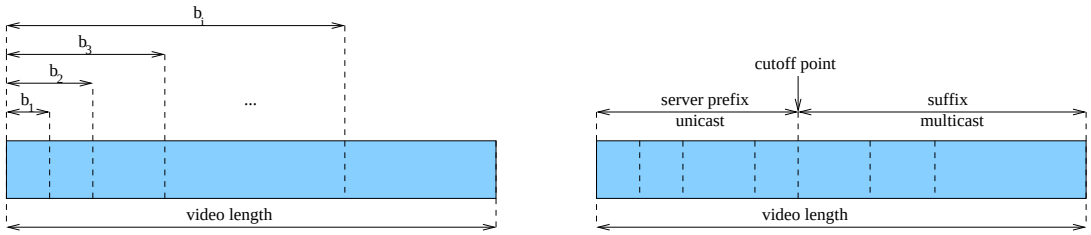
Video delivery through periodic broadcast in the Internet, although scalable, has some limitations. Periodic broadcast requires that a video is transmitted by the server all the time regardless of clients' requests. Thus, some part of the transmission may be utilized by none of the clients. The server I/O bandwidth is reduced significantly but PB may be quite demanding with respect to client's bandwidth and capability. We take the limited client capability into consideration by assuming that there is a limit on the number of broadcast channels that need to be received simultaneously. The number of concurrent accesses to each broadcast channel is considerably reduced compared to the schemes that require a client to receive many channels simultaneously, and the fraction of the broadcast transmission, which is actually received by a client is even smaller.

The network bandwidth required by continuous transmission can be reduced and the utilization level increased by giving the service a more *on-demand* character. Such a mode of transmission can be implemented using *multicast* [4]. We realize that multicast capability is not ubiquitous in the Internet yet. However, we expect that in the future it will become widely available. We address the issue of resource requirements at a server assuming that multicast is a way to deliver periodic broadcast of a video. However, since the network bandwidth saving due to multicast is very hard to predict, we do not include such saving in our problem formulation.

The network bandwidth consumption on the server depends on the actual multicast transmission, while the I/O bandwidth consumption depends on the underlying periodic broadcast schedule. The latter is caused by the fact that the server has to maintain the proper timing among channels for each video. Therefore, we use the term *broadcast* when talking about I/O bandwidth at the central server and *multicast* when talking about network bandwidth.

C. Proxy Prefix and Network Bandwidth Requirements

Proxy caching reduces the bandwidth requirement in the network as well as the I/O bandwidth requirement at the server. Ideally all videos would be cached close to the clients. However, since a proxy has limited I/O bandwidth and storage space (buffer size), such a solution is not feasible. Therefore, the goal is to utilize proxy resources to achieve the highest possible reduction of the WAN bandwidth requirement. We assume that from the proxy point of view, server I/O bandwidth usage is proportional to the network bandwidth usage.



(a) Global video popularity partition

(b) Server transmission modes

Fig. 2. Server prefix selection

We examine now the general guidelines for proxy prefix selection. Video popularity plays an important role in the process of selecting the prefix size for each video. Intuitively, the higher the popularity of the video, the larger reduction in bandwidth consumption can be achieved by caching a larger prefix. Therefore, the prefix sizes for popular videos should be larger than prefixes for the less popular videos. This is the case, when proxy buffer size is the bottleneck. For a popular video, an increase in the prefix size results in a higher reduction of the network bandwidth than for less popular videos. When the I/O bandwidth is the bottleneck, larger prefixes can be selected for less popular videos because for such videos the I/O bandwidth consumption increases slower with the prefix size than for the popular ones. Hence, by selecting larger prefixes for some less popular videos, we can make better use of the available buffer space without violating the I/O bandwidth constraint.

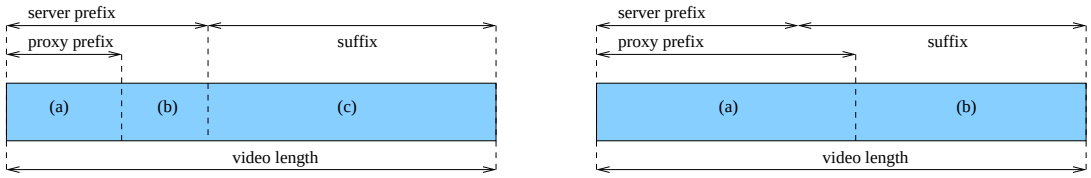
D. Server Prefix and I/O Bandwidth Requirements

The goal of periodic broadcast is to reduce the bandwidth requirement at a server. Such an approach is beneficial only if a video is very popular. For a less popular video, individually unicasted transmissions may consume less bandwidth. Hence, we can optimize the resource requirements by choosing one mode or the other depending on the video popularity. Video popularity defined in each community as a number of concurrent accesses to the video has only a "local" meaning. Global popularity expresses the number of concurrent access to the video *at a central server*. Recall that popularity at a server is not uniform for the whole video since different proxies cache potentially different-length prefixes. Therefore, we need to decide which *part* of a video is delivered in a unicast mode and which in a multicast mode. We first define the global popularity in a more formal way and then introduce a concept of *server prefix*.

Let us consider a single video and let $\{b_i\}$ be a set of proxy prefixes of this video in a set of communities. For the convenience of discussion, we assume that the set is arranged in the increasing order: $b_i \leq b_{i+1}$. Let α_i be the video popularity corresponding to prefix b_i (popularity in the community whose proxy selected prefix b_i). The global popularity of the part of the video equal to the minimum proxy prefix b_1 is 0 since none of the clients requests that part of the video from the central server. The part of video corresponding to the difference between prefix b_2 and b_1 has a global popularity equal to $\alpha_1 \frac{b_2 - b_1}{m}$, where m is the total length of a video. This part of the video is requested from the server only by the clients in the community whose proxy chose b_1 as the prefix size. In a general case, the part of the video corresponding to the difference between b_i and b_{i-1} has a global popularity equal to: $\sum_{k=1}^{i-1} \alpha_k \frac{b_i - b_{i-1}}{m}$. It is requested only by clients from the communities, whose proxies selected prefix sizes smaller than or equal to b_{i-1}. Thus, only the local popularities in these communities are taken into account. Figure 2(a) illustrates the video division into parts with different global popularities.

The server makes a decision whether a given part of the video should be transmitted in either unicast or multicast mode based on the global popularity. The higher the global popularity is the more likely that the part is transmitted in the multicast mode. Notice that global popularity increases as we move from the beginning toward the end of the video. Hence, there is always a cutoff point. All video segments from the beginning to the cutoff point are transmitted in the unicast mode, and the ones after the cutoff point transmitted in the multicast mode. The cutoff point may be at the beginning as well as at the end of the video. It is possible that the whole video is transmitted in multicast mode if the cutoff is at the beginning or unicast mode if it is at the end. All video segments transmitted in the unicast mode constitute *server prefix* as illustrated in Figure 2(b). The dashed lines in the figure separate the video into different parts of different global popularities.

Video popularity varies not only among communities, but also varies over time in a single community. There-



(a) proxy prefix $<$ server prefix

(b) proxy prefix $\geq$ server prefix

Fig. 3. Video transmission modes

fore, in order to minimize the resource requirements, both proxy and server prefix sizes need to be adjusted dynamically. Notice that the change in a server prefix size affects the periodic broadcast scheme used for the suffix delivery. In addition, the division of the video into a prefix and a suffix delivered using different methods (unicast vs. multicast) also imposes additional requirements on the broadcast scheme. We address this issue in section IV.

E. Video Transmission Modes

The video delivery mechanism based on proxy prefix and server prefix combined has a high degree of flexibility. We show that a wide variety of scenarios for video streaming are enabled by this technique. Its flexibility allows us to fully exploit the potential reduction of resource requirements.

We discuss the possible video modes by examining the relation between proxy prefix and server prefix for a video. The length of both prefixes can be a value from 0 to the length of the whole video. A proxy prefix can be smaller as well as larger than a server prefix. If a given proxy prefix is *smaller* than the server prefix, the client receives the proxy prefix from the proxy, the part of video in the server prefix but not in the proxy prefix from the server through a unicast transmission and the suffix of the server partition in the multicast mode from the server (Figure 3(a)). If the proxy prefix is *larger* than or equal to the server prefix, then the client obtains proxy prefix from the proxy and the remaining part of the video through multicast from the server (Figure 3(b)). Notice that our formulation covers all possible scenarios. Table I summarizes all possible combinations, where m denotes the video length, a and b are the lengths of server and proxy prefix, respectively.

Popularity is one of the factors that determine which combination is selected for the video transmission. If the video is very popular in most communities, most proxies will choose a large prefix. In the extreme case the whole video will be cached by most proxies. The resulting global popularity of (all parts of) the video at the server will be low. Hence, the server will choose a small (potentially zero-size) suffix for the multicast transmission. A medium popularity video will have some part of it cached by the proxies and the global popularity of the initial segments will be low. The server will choose unicast transmission for the initial part of the video and the multicast for the remaining part. A low-popularity video may not be cached at all by the proxies. The resulting global popularity may be still low so that the server chooses unicast transmission for the whole video. In the case when the local popularity is not uniform and varies a lot from one community to another, the server will likely choose some part of the video for the unicast transmission and the remaining part for the multicast transmission.

TABLE I
VIDEO TRANSMISSION MODES

proxy prefix	server prefix	mode of transmission
$b = 0$	$a = 0$	periodic broadcast
$b = 0$	$a = m$	unicast from server
$b = m$	$a = m$	unicast from proxy
$0 < b <$ m	$b < a <$ m	(a) prefix unicast from proxy, (b) prefix unicast from server, (c) suffix periodic broadcast (Figure 3(a))
$0 < b < m$	$0 < a \leq b$	(a) prefix unicast from proxy, (b) suffix periodic broadcast (Figure 3(b))

III. PROBLEM FORMULATION

Having defined an architecture for proxy assisted video delivery we proceed now to define the problem of selecting the proxy prefix size for each video in each community and the server prefix size for each video. We first assume that popularity does not change and make prefix selection that results in the optimal resource usage. We then address the issue of dynamic popularity and prefix

TABLE II
NOTATION

m^j	length of video j
α_i^j	popularity of video j in community i
b_i^j	prefix size of video j at proxy i
a^j	server prefix size of video j
n^j	number of suffix segments of video j
r_u^j	unicast transmission rate of video j
r_k^j	transmission rate of the kth segment of video j
U_i^j	portion of video j delivered to community i in unicast
C_i^j	portion of video j delivered to community i by server
S_k^j	size of ith segment of jth video
$\|x\|$	number of segments in x
B_i	buffer size in proxy i
W_i	I/O bandwidth available for proxy i

size adjustment. Notation used throughout the paper is summarized in Table II.

The lengths of the proxy prefix and server prefix determine the mode of video delivery to a client. We first examine the dependence of the resource requirements upon the choice of both types of prefixes and use the results of this investigation to choose the optimal delivery mode. Since the resource availability in the server is much higher than that in the proxy, our approach is to minimize server resource consumption while respecting resource constraint at the proxy. We define the consumption of the I/O bandwidth at a server and the WAN bandwidth as functions of server prefix size a^j, proxy prefix sizes b_i^j and video popularity α_i^j for video j at proxy i. Each video j is further characterized by the following set of parameters: length m^j and transmission rate r_u^j. The popularity of a video is expressed as the expected number of concurrent accesses to that video. In other words, it is an average number of clients in a given community receiving the video at any instant of time. Notice that a longer video with a given access probability has higher average number of concurrent accesses than a shorter one with the same probability.

A. WAN Bandwidth

The network bandwidth requirement is determined by the amount of data that has to be transmitted by a central server, i.e., all segments of the video that are not cached by the proxy. The network bandwidth is defined in the following way:

$$R(b_i^j, \alpha_i^j) = \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_i^j C_i^j \quad (1)$$

where M is the number of proxies (communities), N is the number of videos, and C_i^j is the relative size of the part of the video that is delivered either through a unicast or multicast transmission by the central server:

$$C_i^j = \frac{m^j - b_i^j}{m^j} \quad (2)$$

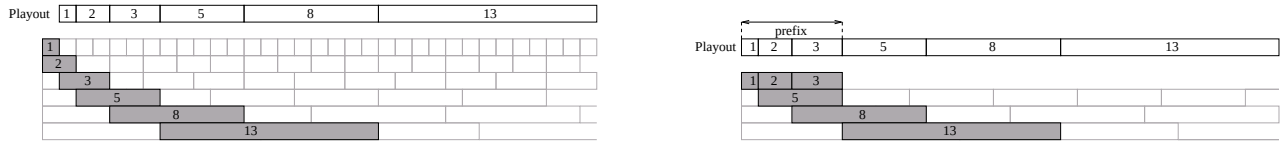
The network bandwidth requirement is independent of the server prefix size selection, which affects only the mode of transmission but not the amount transmitted by the server. Hence, the network bandwidth is minimized by choosing an optimal proxy prefix for each video in each proxy. The choice is limited by the following two constraints. Each proxy has a limited buffer size B_i and limited I/O bandwidth W_i. The buffer size constraint is formulated as: $\sum_{j=1}^{N} b_i^j \leq B_i$ and the I/O bandwidth constraint is formulated as: $\sum_{j=1}^{N} \alpha_i^j \frac{b_i^j}{m^j} r_u^j \leq W_i$. Notice that $R(b_i^j, \alpha_i^j) = \sum_{i=1}^{M}(\sum_{j=1}^{N} \alpha_i^j - \sum_{j=1}^{N} \alpha_i^j \frac{b_i^j}{m^j}) \geq \sum_{i=1}^{M}(\sum_{j=1}^{N} \alpha_i^j - W_i)$. Hence, the I/O bandwidth constraint defines the lower limit on the network bandwidth. It is intuitive since increasing a proxy prefix of a video decreases the network bandwidth consumption but increases the I/O bandwidth consumption.

B. Server I/O Bandwidth

The central server I/O bandwidth consumption is affected by the proxy prefix size of each video, the broadcast scheme used for the suffix, and the video popularity determining the frequency of the unicast transmissions of the part of prefix not cached by the proxies. It is defined in the following way:

$$W(a^j, b_i^j, \alpha_i^j) = \sum_{j=1}^{N} \sum_{k=1}^{n^j} r_k^j + \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_i^j U_i^j r_u^j \quad (3)$$

where r_k^j is the transmission rate of the kth segment of video j in the broadcast scheme and n^j is the number of segments (channels) for the suffix portion of the video to be accessed using a periodic broadcast scheme. The first part of the summation represents the I/O bandwidth consumption due to the broadcast of all suffix segments of video j. The second component of the summation represents the I/O bandwidth consumption due to the unicast transmission of a part of the video prefix, which is not cached by some proxies. Since each proxy may choose a different prefix for video j, we calculate the unicast I/O bandwidth requirement for each community i and each video j:



(a) Fibonacci PB

(b) Fibonacci PB with prefix

Fig. 4. Fibonacci periodic broadcast reception schedules

$$U_i^j = \begin{cases} 0 & \text{if } b_i^j \geq a^j \\ \frac{a^j - b_i^j}{m^j} & \text{otherwise} \end{cases} \quad (4)$$

U_i^j is the relative size of the part of video j that is delivered through a unicast transmission by the server to community i. Its value depends on the difference between server prefix size and proxy prefix size relative to the length of the video. The product $\alpha_i^j U_i^j$ is interpreted as an average number of concurrent accesses to this part of the video. The total I/O bandwidth consumption depends on both server and proxy prefix sizes as well as the popularity of each video and thus is denoted by $W(a^j, b_i^j, \alpha_i^j)$. For a fixed popularity and proxy prefix size, decreasing the server prefix increases I/O bandwidth due to periodic broadcast, but decreases I/O bandwidth due to unicast transmissions. There is a trade-off between these two quantities and the goal is to find the optimal point.

To summarize, the objective is to choose proxy prefix in such a way that the network bandwidth requirement is minimized, and to choose server prefix size in such a way that the I/O bandwidth requirement at the server is minimized. The problem is constrained by the buffer size and I/O bandwidth available at each proxy.

IV. Periodic Broadcast with Prefix

Before we proceed with a solution for the resource requirement optimization problem, we examine the conditions a periodic broadcast scheme has to satisfy for the two-prefix level video delivery to be feasible. Next we describe a scheme that satisfies these requirements.

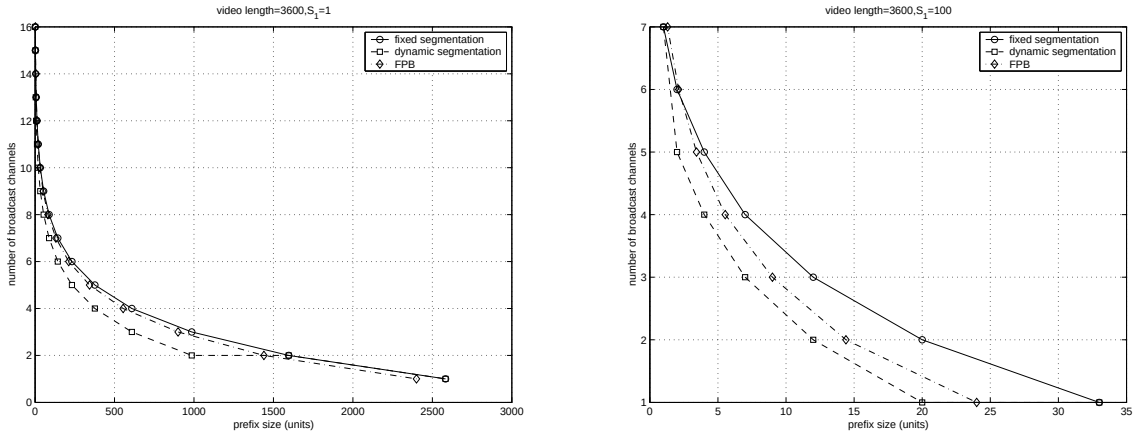
Periodic broadcast scheme has to satisfy the following three requirements. The prefix transmission must be synchronized with the multicast transmission of the suffix in a way that always results in a feasible reception schedule. The periodic broadcast schemes that minimize the server bandwidth usually have high network and client bandwidth requirements. In order to limit these requirements we assume that the number of broadcast channels received simultaneously by the client is limited to 2. The broadcast scheme has to be able to handle prefix size changes. The desirable feature is, therefore, a smooth transition property: the change of prefix size should not disrupt the ongoing transmissions and should not increase the resource usage significantly during the transition period. We first address the feasibility of reception schedule and the limited client's bandwidth. Then, we examine ways to implement the dynamic adjustments of prefix size.

A. Fibonacci Periodic Broadcast

A scheme that minimizes server bandwidth requirements and respects the constraint on the number of channels received simultaneously by a client uses Fibonacci series as a segment size progression [1]. Each segment is broadcast on a separate channel at the playback rate. A client can start reception of a video at any time and the start-up delay is equal to the reception time of the first segment. Each segment has to be received before its playback starts in order to ensure a feasible reception schedule as illustrated in Figure 4(a). We use this scheme as a base for the server suffix delivery. In order to achieve the synchronization between prefix and suffix reception, we overlap the reception of the first server suffix segment with the possible reception of the prefix. A client receives all segments in the maximum of (proxy prefix, server prefix) sequentially either from the proxy or from the central server. Each of these segments can be played immediately, which reduces the start-up delay. The reception of the first remaining suffix segment is aligned with the prefix reception in such a way that it is completely received by the end of the prefix playback. Figure 4(b) shows the reception schedule for a prefix consisting of three segments.

B. Prefix Size Transition

Due to a change in video popularity, the server may have to adjust its prefix size for a given video and consequently the broadcast transmission of the suffix. There are two ways to deal with the dynamic changes in server prefix size. One of them is to adjust video segmentation for the suffix whenever a change occur. Another is to use a fixed segmentation of the video and make prefix consists



(a) First segment size = 1s

(b) First segment size = 100s

Fig. 5. Comparison between fixed and dynamic segmentation schemes

of a number of initial segments. We examine both possibilities in terms of their resource requirements and the complexity of transition to a different prefix size.

1) Dynamic Segmentation: Assume first that prefix can have an arbitrary length. Once its size is decided, the suffix is partitioned into a number of segments. We still use the Fibonacci series as a segment size progression to satisfy client bandwidth constraint. The size of the first suffix segment is equal to the server prefix size and both are received simultaneously. Since video segmentation is redesigned after each prefix change, we expect an optimal I/O bandwidth consumption from this scheme, i.e., a minimal number of broadcast channels (segments). This scheme does not exhibit smooth transition property. Switching from one broadcast schedule to another requires transmission of a set of channels for both, the old and the new, segmentation schemes. The I/O bandwidth consumption during the transition period is equal to the sum of the I/O bandwidth requirements of both schemes.

2) Fixed Segmentation: Assume now that a video is partition into segments following Fibonacci series *prior* to the server or proxy prefix choice. The prefix must now consist of a number of initial segments. Its size is adjusted by including or excluding some segments. With such a strategy the transmission of segments that are not being moved between prefix and suffix is not affected by the prefix size adjustment. The transition is more smooth, but the server I/O bandwidth requirement may be higher.

3) FPB: In order to provide a fair comparison, we consider one more scheme proposed in [5]. In this scheme Fibonacci-based periodic broadcast (FPB) is designed to adjusts the video segmentation according to the demand, but the design does exhibit a smooth transition property. The I/O bandwidth requirement during the transition period is equal to the maximum of two requirements: the old and the new schemes. However, this scheme is much more complex than the scheme proposed in this paper.

All three schemes use Fibonacci series for suffix segmentation. We refer to the first scheme that can choose prefix of an arbitrary length as *dynamic segmentation* scheme, the second scheme offering a set of discrete values for prefix size as the *fixed segmentation* scheme, and the scheme proposed in [5] as the *FPB* scheme.

Figure 5 presents the comparison of the number of broadcast channels between the three schemes. The video length is set to 3600s (seconds). The size of the first segment S_1 in the fixed segmentation scheme is equal to 1s in the first test and to 100s in the second test. Since the prefix size for the fixed segmentation scheme is limited to a set of discrete values, we use only these values for comparison. Each point on the fixed segmentation curve corresponds to a single prefix size expressed in seconds. For each value, a new segmentation is calculated for the dynamic segmentation scheme and the corresponding number of broadcast channels is shown on the dynamic segmentation curves. The prefix in this case always consists of one segment of a variable size. The difference between the number of broadcast channels for the fixed segmentation scheme and the dynamic segmentation scheme does not exceed one channel. The same result holds for various lengths of first segment S_1. The number of channels in FPB is close to the number of channels used by the fixed segmentation scheme for small values of S_1. For larger values of S_1, the number of segments used by FPB is smaller than the fixed segmentation scheme but larger than that of the dynamic segmentation scheme. This result

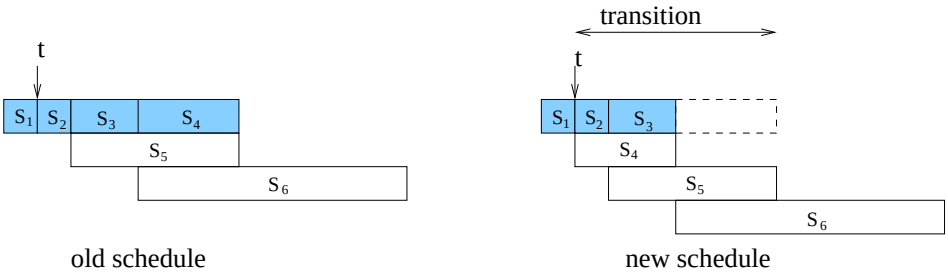


Fig. 6. Server prefix size change

is expected since FPB pays the price for the smooth transition property by having worse than optimal performance during non-transition periods. It performs better than the fixed segmentation scheme since it adjusts the segmentation.

We conclude that the performance gain of the dynamic segmentation scheme is not significant enough to compensate for the lack of smooth transition. On the other hand the complexity of FPB scheme is quite high. Our proposed fixed segmentation scheme has another advantage over the other two schemes. The prefix change at the server does not enforce similar change at the proxy. Both can be done independently if needed. The dynamic segmentation may face the situation, where proxy prefix does not consist of an *integer* number of segments. Receiving a partial segment from the multicast channel poses a much more difficult challenge in design of reception schedule. For more details on implementing prefix change see [6].

C. Prefix Change Implementation

We now present how the prefix size change is implemented by both the proxy and the server with the fixed segmentation scheme. We examine the length and the cost of transition period.

1) Proxy Prefix Change: The prefix size is adjusted by the proxy by including or excluding a video segment from the prefix. If the prefix change is made for more than one video, we have to make sure that the cumulative change in buffer occupancy does not cause the buffer overflow at the proxy. The change has to be made gradually, i.e., a segment of one video is gradually replaced by a segment of another video.

a) Segment Removal: For each segment that is to be removed from the buffer, we check what part of the segment has already been received by the client, which requested the segment most recently. This part is removed from the buffer immediately. The rest of the segment is removed at the rate, at which it is transmitted. Based on this schedule we can find how the buffer occupancy increases with time.

b) Segment Addition: Next, we design the transmission schedule for each new segment that has to be received from the server, by choosing the start time of the transmission. There is a trade-off between the length of the transition period and the network bandwidth needed to transmit all new segments. The length of transition period is defined as the time need to complete the prefix change, i.e., the removal of old prefix segments or transmission of new ones, whichever takes longer.

If the length of the transition period is the main concern, each of the new segments should be received from the server as soon as possible without overflowing the buffer. A simple heuristic can be used for that purpose. The new segments are sorted by the time needed for their transmission in the increasing order. Next we find the earliest feasible transmission time for the video with the longest time. The procedure is repeated for each of the remaining segments. If a prefix size is extended by more than one segment, we treat all new segments as one with the size equal to the sum of segment sizes.

2) Server Prefix Change: The prefix size change at the server is implemented by adding or removing a channel from the broadcast transmission schedule.

a) Segment removal: A removal of segment from prefix is implemented by starting a broadcast transmission of this segment. All ongoing unicast transmission of the segment have to be completed. In addition, all requests received during the next segment transmission period need to be served through a unicast transmission. Otherwise, there may not be a feasible reception schedule.

Figure 6 shows an example where the 4th segment of the video is removed from the prefix. According to the new schedule the client should start receiving the 4th segment at time t, i.e., simultaneously with segments 2 and 3. However, if the change occurs at time t^+, the client is not able to receive the whole segment before the end of the prefix playback. Hence, the client should continue the reception according to the old schedule. We may see a temporary increase in the I/O bandwidth consumption while both unicast and broadcast transmissions are in progress. The two modes of transmission overlap during the time

period equal to twice the time of the segment transmission. The transition cost is defined as an I/O bandwidth consumption due to unicast transmission of all segments removed from respective prefixes during transition period.

b) Segment Addition: An addition of a segment to the prefix causes removal of the corresponding broadcast channel. The broadcast transmission has to continue for a period of time equal to the segment transmission period. All new requests received during this time as well as all subsequent requests for the segment are satisfied through unicast transmissions. We observe again an increase in the I/O bandwidth consumption, as broadcast and unicast transmissions overlap for the period of the segment transmission time. The cost of the transition is evaluated in this case, as the I/O bandwidth consumed by the broadcast of all segments added to the respective prefixes during transition time.

V. Proposed Solution

The solution for the optimization problem formulated in Section III must specify the proxy prefix length for each video at each proxy and the server prefix length for each video. The problem of proxy prefix selection is NP-hard. It can be shown that it is a variant of knapsack problem. Therefore, we concentrate on a heuristic solution for this step. We design a low-complexity algorithm for server prefix selection.

A. Proxy Prefix Selection

Since the network bandwidth Equation (1) does not depend on the server prefix size, we select the proxy prefix for each video at each proxy first and then select the server prefix given the selected proxy prefixes for each community. The prefix selected by a single proxy does not depend on other proxies' choices either. Therefore, we can minimize the network bandwidth separately for each community:

$$R_i = \sum_{j=1}^{N} \alpha_i^j C_i^j = \sum_{j=1}^{N} \alpha_i^j (1 - \frac{b_i^j}{m^j}), \quad i = 1, ..., M$$

If the number of concurrent accesses for a video is 0, then the prefix size for this video is automatically set to 0. Let V_i denote the set of videos whose popularity is non-zero in community i: $V_i = \{j : \alpha_i^j > 0\}$, where j is a video index. In the following we consider only videos from this set. Given a fixed segmentation of each video, the problem becomes an integer (binary) programming problem. We introduce a set of variables x_k^j such that $x_k^j = 0$ if the kth segment of video j does not belong to its prefix, and $x_k^j = 1$ otherwise:

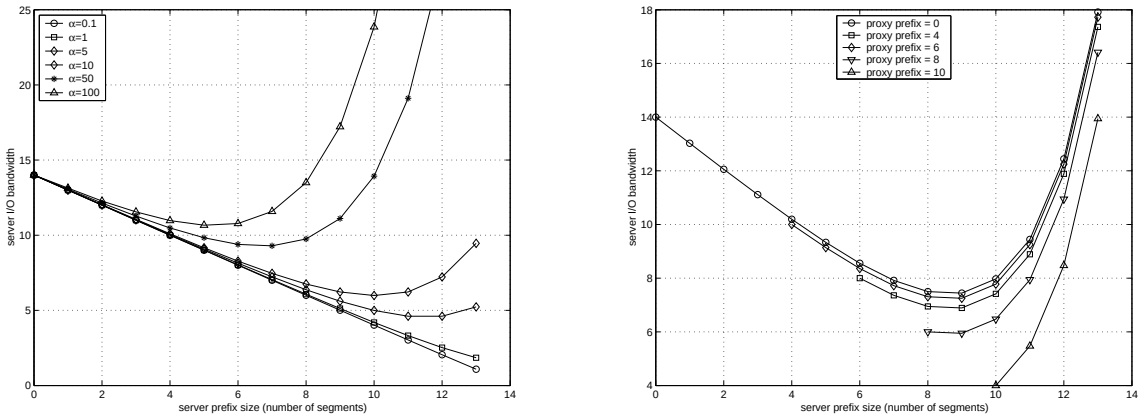
$$\begin{array}{ll} \text{maximize} & \sum_{j \in V_i} \frac{\alpha_i^j}{m^j} \sum_{k=1}^{|m^j|} S_k^j x_k^j \\ \text{subject to} & 1) \sum_{j \in V_i} \sum_{k=1}^{|m^j|} S_k^j x_k^j \leq B_i \\ & 2) \sum_{j \in V_i} \frac{\alpha_i^j}{m^j} \sum_{k=1}^{|m^j|} S_k^j x_k^j \leq W_i \\ & 3) x_{k+1}^j - x_k^j \leq 0, \quad k = 1, ..., |m^j| - 1, j \in V_i \\ & 4) x_k^j \in \{0, 1\}, \quad k = 1, ..., |m^j|, j \in V_i \end{array} \tag{5}$$

where S_k^j is the size of the kth segment of video j. The first two conditions represents the buffer space and I/O bandwidth constraints. Without loss of generality and for simplicity of presentation we assume that $r_u^j = 1$. Consequently, the value on the left side of the second condition must be multiplied by the unit of transmission rate to achieve a proper unit conversion. The third constraint ensures that the prefix is continuous, i.e., segment k can be included in the prefix only if all earlier $(1, ..., k-1)$ segments are already in the prefix.

A problem formulated in this way is a variant of knapsack problem [7] with some additional constraints, and as such is an NP-hard problem. The exact solution can be obtained using a number of methods such as dynamic programming or branch-and-bound [8]. The implicit enumeration, which is a variant of the branch-and-bound, is promising. The fact that $x_k^j = 0$ implies that all variables x_k^j, $k = i+1, ..., |m^j|$ for video j are also equal to 0, allows to eliminate a lot of possible combinations of values from evaluation. The complexity is reduced in this way to $\mathcal{O}(\prod_{j=1}^{N} |m^j|)$. Nonetheless, the procedure may still be quite costly. Dynamic programming can be used to obtain solutions to the problem with only buffer size constraint for the following two reasons. First, considering both constraints, buffer size and I/O bandwidth, increases the complexity considerably. Second, the I/O bandwidth requirement cannot be easily expressed as an integer. We consider this approach for evaluation purposes.

In order to obtain an approximate solution of the optimization problem we first ignore the integrality restriction and solve the associated linear programming problem. The solution obtained gives a lower bound on the value of network bandwidth for a given buffer size and I/O bandwidth constraints. Next, we adjust the solution to enforce the integrality restriction. The number of variables in the associated linear programming problem can be reduced by using the following formulation:

$$\begin{array}{ll} \text{maximize} & \sum_{j \in V_i} \alpha_i^j \frac{b_i^j}{m^j} \\ \text{subject to} & 1) \sum_{j \in V_i} b_i^j \leq B_i \\ & 2) \sum_{j \in V_i} \alpha_i^j \frac{b_i^j}{m^j} \leq W_i \\ & 3) 0 \leq b_i^j \leq m^j, j \in V_i \end{array} \tag{6}$$



(a) Popularity influence

(b) Proxy prefix influence

Fig. 7. Server prefix dependence on popularity and proxy prefix

The relation between variables in Equations (5) and (6) is given by : $b_i^j = \sum_{k=1}^{|m^j|} S_k^j x_k^j$. We adjust the obtained solution in order to enforce the integrality restriction. More precisely, we find a number of initial segments such that the sum of their sizes is as close to the prefix size selected by linear programming as possible: $k_i^j = \arg\min_{0 \le k \le |m^j|} abs(b_i^j - \sum_{l=1}^{k} S_l^j)$. Then $x_k^j = 1$ if $k \le k_i^j$, and $x_k^j = 0$ otherwise. Notice that such a solution may not be feasible. The first two constraints in Equation (5) may be violated. It is also possible that the solution is feasible but quite different from the optimal one. In either case we use a greedy approach to either lower the buffer space and the I/O bandwidth requirements (first case) or to utilize the leftover buffer space and/or leftover I/O bandwidth (second case).

B. Server Prefix Selection

Given a proxy prefix size for each video at each proxy, we choose the server prefix that minimizes the server I/O bandwidth requirement. Recall that proxy prefix selection determines the amount of data that has to be delivered by the central server. Thus, server prefix selection cannot change the network bandwidth requirements as defined in (1). Our problem formulation decouples the required network bandwidth from the required server I/O bandwidth such that the complexity of the problem can be reduced. The server prefix selection affects the I/O bandwidth requirements only.

The server prefix size for each video can be selected independently. The required I/O bandwidth of a server is given as :

$$W = \sum_{j=1}^{N} \sum_{k=1}^{|m^j - a^j|} r_k^j + \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_i^j U_i^j r_u^j$$

where $|m^j - a^j|$ is the number of segments in a server suffix of video j. Given the fixed segmentation scheme and assuming that $r_k^j = r_u^j = 1$ for simplicity, the I/O bandwidth function can be rewritten as follows:

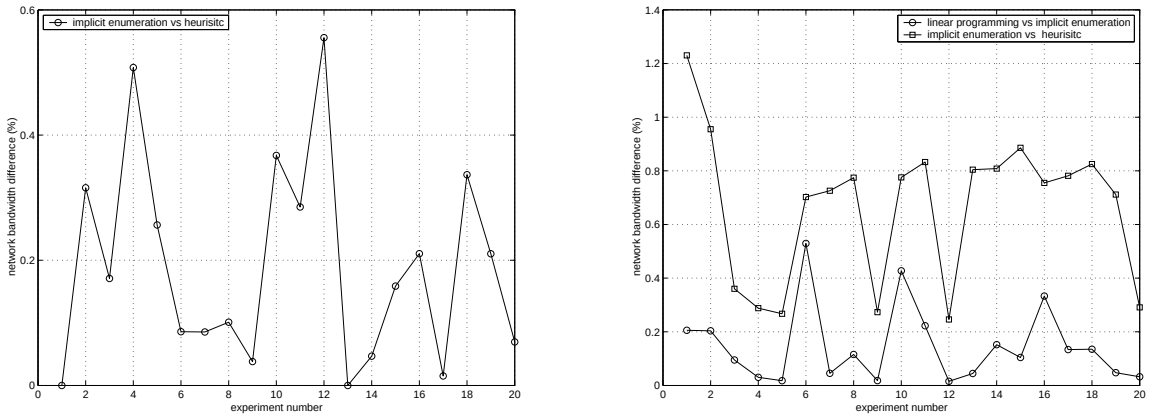
$$W = \sum_{j=1}^{N} W^j = \sum_{j=1}^{N} \left(|m^j - a^j| + \sum_{i=1}^{M} \alpha_i^j U_i^j \right)$$

where:

$$U_i^j = \begin{cases} 0 & \text{if } b_i^j \ge a^j \\ \dfrac{\sum_{k=|a^j|+1}^{|m^j|} S_k^j}{\sum_{k=1}^{|m^j|} S_k^j} & \text{otherwise} \end{cases}$$

Let $\hat{b}^j$ denote the minimum number of segments in any proxy prefix selected for video j: $\hat{b}^j = \min_{1 \le i \le M} |b_i^j|$. Then $|a^j| \in \{\hat{b}^j, \hat{b}^j + 1, ..., |m^j|\}$. Out of all possible values for a^j, the optimal one is found through an iterative approach. The value of I/O function is calculated for every $|a^j| \in \{\hat{b}^j, \hat{b}^j + 1, ..., |m^j|\}$. The one that yields the minimum value of W^j is selected as a server prefix. The complexity of such an approach is $\mathcal{O}(N|m^j|)$. It scales with the number of videos and the number of segments in each video.

Further we examine the properties of the I/O bandwidth function. The value of the function depends on two factors: the global popularity of the video and the prefixes selected for the video by the proxies. We illustrate how each of these two factors affects the server prefix selection. We present two hypothetical cases. In each case we



(a) I/O bandwidth bottleneck

(b) Buffer space bottleneck

Fig. 8. Evaluation of proxy prefix heuristic algorithm

fix the value of one factor and show how the I/O bandwidth consumption changes as a function of the server prefix size for different values of the other factor. First, we assume that all proxies select a zero-size prefix. In this way we eliminate the influence of the proxy prefix selection and make the global popularity α the same for the whole video. Figure 7(a) shows the I/O bandwidth as a function of a^j for various values of global popularity α. The shape of the I/O bandwidth function depends on the value of the global popularity. For the small values of α, I/O bandwidth consumption decreases with increasing prefix size. Hence, the minimum is reached for a large prefix, potentially equal to the video length. For the medium values of α there is an optimal value for the prefix between the zero-size prefix and the video length. Finally, for the high value of α, the function increases with the prefix size and the minimum I/O bandwidth consumption is reached for a zero-size prefix.

Next we examine how the proxy prefix selection affects the server prefix selection. We assume that all proxies choose the same prefix size. Thus, the global popularity of the part of the video within the proxy prefix is 0. We fix the global popularity of the remaining part (suffix) at $\alpha = 10$ which corresponds to a middle range of the number of concurrent accesses to the video at the server. Figure 7(b) shows the I/O bandwidth as a function of the server prefix size for various values of the proxy prefix size. The function is clearly concave with a minimum reached for a certain prefix size. As the proxy prefix becomes larger, the I/O bandwidth consumption is reduced independent of the server prefix size. Thus, the proxy prefix selection sets the lower bound on the consumption of the I/O bandwidth at the server. It is also true that the smaller the popularity, the smaller the influence of a proxy prefix on the server prefix selection.

VI. Performance Evaluation

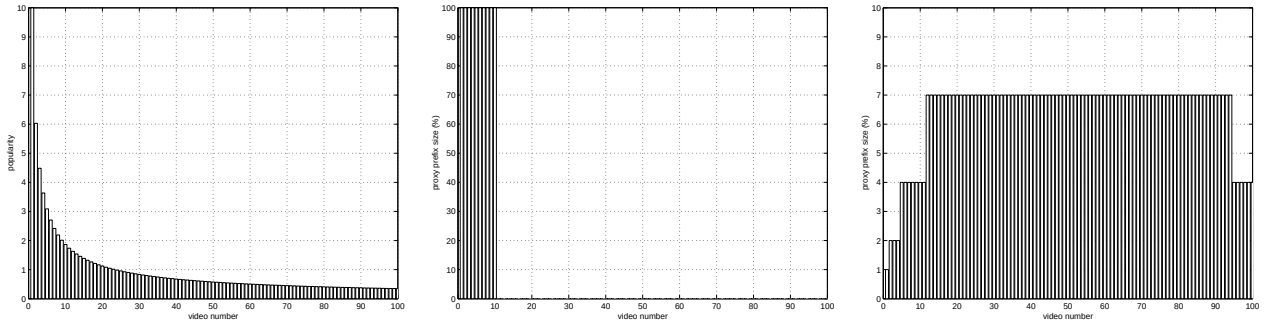
In order to to illustrate the behavior and evaluate the performance of our proposed architecture, we have conducted a number of tests. We concentrate on the proxy prefix selection first. We evaluate the performance of proposed heuristic algorithm and present several examples on how the prefix size is selected under various circumstances. Next we show how performance improvement can be achieved by applying the concept of server prefix. We discuss the performance with a set of homogeneous and heterogeneous proxies and communities.

A. Proxy Prefix Selection

1) Heuristic Algorithm Evaluation: We evaluate the performance of the heuristic algorithm for proxy prefix selection by comparing its network bandwidth requirements with an optimal value. The optimal value is obtained with an implicit enumeration method [1]. We also include the result of the linear programming in the comparison to illustrate the difference with the final heuristic solution. The video test set includes 10 videos of equal length but different popularities. We use a small set due to the complexity of implicit enumeration method.

In the first set of tests the I/O bandwidth is a bottleneck. The value of the network bandwidth obtained by solving associated linear programming and implicit enumeration are identical, although the prefixes selected by each of these two methods are different. The heuristic algorithm produces only slightly higher requirements. The maximum difference is less than 1% of the bandwidth needed

[1] Results were obtained using OPBDP [9]



(a) Popularity (b) Prefix size with buffer bottleneck (c) Prefix size with I/O bottleneck

Fig. 9. Proxy prefix dependence on I/O bandwidth and buffer size

for the exact solution. Figure 8(a) presents the comparison between the network bandwidth obtained for the optimal solution and the bandwidth obtained with the heuristic algorithm. The difference is expressed as a percentage of the optimal solution bandwidth.

In the second set of tests the buffer size is a bottleneck. In this case we compare the solution for associated linear programming, the optimal solution obtained by implicit enumeration and the solution produced by the heuristic algorithm. The optimal network bandwidth is higher than the bandwidth yielded by the solution of associated linear programming. Figure 8(b) shows the difference between the two requirements as a percentage of the linear programming solution bandwidth. It shows also the difference between network bandwidth given by the heuristic algorithm and the implicit enumeration expressed as a percentage of the implicit enumeration bandwidth. The difference is always less than 2 percents. Therefore, we conclude that the heuristic algorithm produces an acceptable solution while offering a significant reduction of complexity.

2) Prefix Selection Properties: The solution for the network bandwidth minimization problem depends on whether buffer space or I/O bandwidth is a bottleneck. Which one of the two factors is a bottleneck is determined by the buffer size and I/O bandwidth available at a proxy, but also on video popularity and, to a lesser degree, on video segmentation. When the overall video popularity is low, the buffer space tends to be a bottleneck. For the large value of α, the I/O bandwidth tends to be a bottleneck.

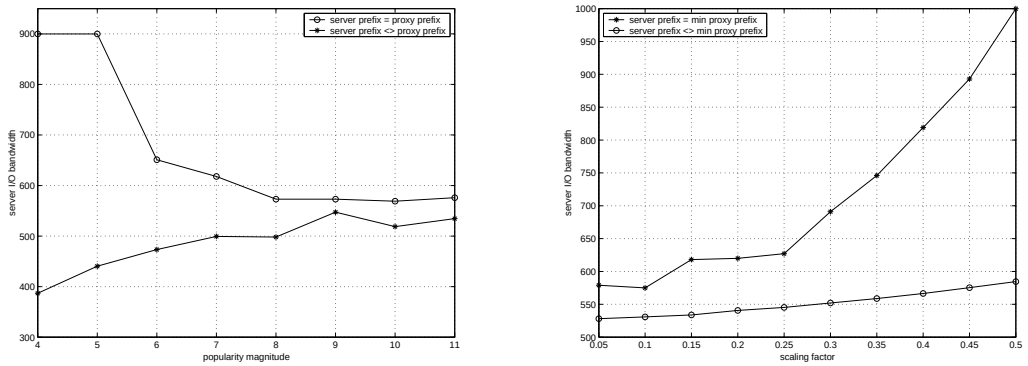
Figure 9(b) and Figure 9(c) show the proxy prefix sizes as a percentage of the video length with a buffer bottleneck and an I/O bandwidth bottleneck, respectively. All videos in this test have the same length but their popularities vary. The popularity for each video is drawn from a Zipf distribution with parameter 0.27 [10]. The distribution is illustrated in Figure 9(a). The buffer size is set to 10% of the storage space needed for caching the whole video set. The amount of I/O bandwidth changes from 100 units in the first case, to five units in the second case in order to shift the bottleneck from buffer space to I/O bandwidth.

In the first case we observe that only the most popular videos have prefixes of non-zero length. In fact, the prefix size of each one of them is equal to the video length. Recall that network bandwidth requirements for very popular videos decrease faster with the increase of prefix size than the bandwidth requirements for less popular videos. Therefore, we make a better usage of buffer space by caching prefixes of more popular videos. The buffer space utilization is 100% and the I/O bandwidth utilization is 38%. In the second case, the I/O bandwidth is the bottleneck. The buffer space utilization is reduced to 64%, while the I/O utilization is 99%. We observe that the most popular videos do not have the largest prefixes in this case. Notice also that a larger number of videos have prefixes of non-zero length. We have observed that as the amount of available I/O bandwidth decreases, the largest prefixes are "shifted" even more toward to less popular videos.

B. Server prefix

We illustrate next how we can improve the performance by using the concept of server prefix. The mode of transmission by the server is selected based on video popularity: periodic broadcast for popular videos and unicast for less popular videos. We show that by partitioning a video into unicast- and multicast-delivered sections, we can reduce I/O bandwidth usage at the server.

In the first set of tests we assume that all proxies are homogeneous. For a given video all proxies choose the same prefix. Also the popularity is the same for each video in each community. We compare the I/O bandwidth



(a) homogeneous proxy prefixes

(b) heterogeneous proxy prefixes

Fig. 10. Server I/O bandwidth

consumption at the server in two cases. In the first case, a server prefix is the same as the proxy prefix for each video. Broadcast is the only transmission mode for the server. In the second case server chooses its own prefix for each video and may transmit a portion of a video in unicast mode. We use a set of 100 videos and 20 proxies in our tests. The proxy prefix selection is made based on the buffer size equal to 10% of the aggregate video length and the I/O bandwidth such that the I/O bandwidth requirement is the bottleneck. The popularity for each video is drawn from a Zipf distribution as before.

Figure 10(a) presents the I/O bandwidth requirements in both cases as a function of video popularity. We increase popularity by multiplying its value for each video by a factor whose value is shown on the x-axis. We observe that even when all proxies choose the same prefix for a video, the server I/O bandwidth usage may be reduced by partitioning the video into different transmission mode sections. The server chooses large prefixes for low popularity videos exploring the trade-off between the I/O bandwidth consumption for the unicast and multicast transmissions. The larger the number of low popularity videos, the higher the potential reduction.

In the second set of tests we assume that the proxies are heterogeneous. Video popularity distribution is the same in every community and is fixed throughout all tests. We vary the amount of I/O bandwidth available at proxies and increase its variability in each test. In all cases, the I/O bandwidth is a bottleneck. As a result of increasing variability of I/O bandwidth, the variability of proxy prefix sizes also increases.

Figure 10(b) presents the I/O bandwidth usage when server prefix is equal to the minimum proxy prefix and when an optimal server prefix is selected for each video. We observe that as the variability of proxy prefix sizes increases, a larger reduction of I/O bandwidth usage can be obtained with the optimal server prefix selection. The reduction can be as high as 50%.

VII. Related Work

VoD has been an active research area for a decade and many approaches have been proposed to reduce service delays, increase the number of concurrent sessions, and provide certain quality for client playback. In the following, we briefly introduce two major research directions which are closely related to our work: video proxy servers and periodic broadcast.

Video proxy servers are one of the most common approaches for supporting VoD across WANs. Many proxy-assisted video delivery schemes have been developed [2, 11–15]. Caching at a proxy reduces the start-up delay as well as the network and server I/O bandwidth requirements. Due to the limited cache space and I/O capacity, a proxy is not able to cache all videos and support a large number of concurrent streams. These issues are addressed in several proxy-caching schemes including prefix caching [2], video staging [15], progressive caching [16].

Server-initialized broadcast addresses the issue of high network and I/O bandwidth requirements at a server. In Pyramid Broadcasting [17], Skyscraper Broadcasting [18], Fast Broadcasting [19], and Fibonacci Broadcasting [5], a video is divided into segments with increasing sizes and transmitted in logical channels of the same bandwidth. In order to address the issue of relatively large client-buffer requirements of pyramid-based broadcasting, Harmonic Broadcasting [20] divides a video into equal-size segments and transmits the segments in logical channels with decreasing bandwidth.

Proxy caching and periodic broadcast can be combined to reduce resource requirements. Video prefixes are

staged at proxy servers to reduce the startup delay and PB schemes are used to deliver video suffixes from a central server. To the best of our knowledge, only few studies have been done in this direction [21–24]. Their focus is on the reduction of WAN bandwidth consumption. In this paper, we address the issues in similar settings considering the I/O bandwidth limitation at a central server and proxy servers, the WAN bandwidth consumption, and the heterogeneity of proxy servers.

The prefix caching assisted periodic broadcast scheme [22] allocates proxy buffer space optimally for a set of popular videos and choses appropriate transmission schemes separately for video prefix and suffix. A greedy algorithm is proposed to determine the prefix size of each video to minimize the number of broadcast channels needed with a given proxy buffer capacity.

Our work differs from this approach in several important aspects. We consider not only the WAN bandwidth requirements but also the I/O bandwidth requirement at a central server and proxy servers. Furthermore, our scheme supports heterogeneous proxy servers in diverse communities. The video set under consideration includes popular videos as well less popular videos for which PB is not an optimal mode of delivery.

The assumption of multicast availability in WANs and LANs is another factor in system design. In Mcache [23] the assumption is made that the whole network path from a server to a client is multicast-enabled. A server uses a combination of patching and batching for transmission scheduling. A client receives a video prefix from a proxy and shares a multicast transmission of the suffix with other clients whose requests arrive within the duration of a prefix transmission.

In [24] the assumption about network multicast capability is different. The LAN between a proxy and clients is multicast capable while the WAN between a central server and a proxy supports only unicast. In this scheme a proxy delays suffix transmission to the first client as long as possible and batches all client requests arriving during the prefix transmission. Patching is used for the transmission of video suffixes. In addition, a proxy also multicasts a prefix from its cache and a suffix received from a central server to a number of clients. The goal is to minimize the aggregate network bandwidth by optimally utilizing proxy buffer space. We rely on the assumption that WAN between a server and a community is multicast capable. Therefor, a server can utilize periodic broadcast to exploit data sharing between clients.

VIII. Conclusions and Ongoing Work

We have designed an architecture for large-scale video delivery that allows a significant reduction of resource requirements. We use video popularity to choose the mode of transmission in such a way that the resource usage, namely central server I/O bandwidth and WAN network bandwidth, is minimized. The optimization is subject to resource availability at proxy servers and clients. The combination of the unicast/multicast transmission by the central server and prefix caching by the proxies allows us to provide a wide spectrum of possible transmission modes. The spectrum includes the special cases of the whole video cached at a proxy, the whole video delivered by a central server in either unicast or multicast mode, and anything in between. The mode of transmission is determined by the lengths of the server and proxy prefixes. We have formulated and proposed solutions for the problems of prefix selection. We have demonstrated through a number of tests that introducing the concept of server prefix in addition to proxy prefix leads to an I/O bandwidth requirement reduction at a server.

We are further exploring this area of research by including multiple geographically distributed servers in the architecture. We investigate how the number of servers affect the resource requirements. We plan also to extend our research in video delivery to wireless environment.

References

[1] A. Hu, "Video-on-demand broadcasting protocols: A comprehensive study," in *Proceedings INFOCOM*, 2001.

[2] S. Sen, J. Rexford, and D. F. Towsley, "Proxy prefix caching for multimedia streams," in *Proceedings of INFOCOM*, vol. 3, 1999, pp. 1310–1319.

[3] W. hsiu Ma and D. Du, "Reducing bandwidth requirement for delivering video over wide area networks with proxy server," *IEEE Transactions on Multimedia*, vol. 4, pp. 539–550, Dec 2002.

[4] R. Janakiraman, M. Waldvogel, and L. Xu, "Fuzzycast: Efficient video-on-demand over multicast," in *Proceedings of INFOCOM*, 2002.

[5] Y. Guo, L.Gao, D. Towsley, and S. Sen, "Seamless workload adaptive broadcast," in *12th International Packet Video Workshop*, 2000.

[6] E. Kusmierek, D. Du, and Y. Dong, "Proxy-assisted periodic broadcast architecture for large-scale video streaming," University of Minnesota, Tech. Rep., 2003.

[7] D. Pisinger, "Algorithms for knapsack problems," Ph.D. dissertation, University of Copenhagen, 1995.

[8] S. Bradley and T. M. A. Hax, *Applied Mathematical Programming*. Addison-Wesley, 1977.

[9] P. Barth. (2002) OPBDP - a Davis-Putnam based enumeration algorithm for linear pseudo-boolean optimization. [Online]. Available: http://www.mpi-sb.mpg.de/units/ag2/software/opbdp/

[10] S. Acharya, B. Smith, and P. Parnes, "Characterizing user access to videos on the world wide web," in *Multimedia Conferencing And Networking*, 2000.

[11] E. Amir, S. McCanne, and H. Zhang, "An application level video gateway," *Proceedings of ACM Multimedia*, 1995.

[12] J. McManus and K. Ross, "Video-on-demand over ATM: Constant-rate transmission and transport," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1087–1098, 1996.

[13] R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet," in *Proceedings of INFOCOM*, 2000, pp. 980–989.

[14] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. F. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," in *Measurement and Modeling of Computer Systems*, 1996, pp. 222–231.

[15] Z.-L. Zhang, Y. Wang, D. H. C. Du, and D. Shu, "Video staging: a proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 429–442, 2000.

[16] W. Ma and D. Du, "Design a progressive video caching policy for video proxy servers," in *IEEE Trans. Multimedia*, To appear.

[17] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems*, vol. 4(5), pp. 197–208, 1996.

[18] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proceedings of ACM SIGCOMM*, 1997, pp. 89–100.

[19] L. Juhn and L. Tseng, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Transactions on Broadcasting*, vol. 44, no. 1, Mar 1998.

[20] ——, "Harmonic broadcasting for video-on-demand service," *IEEE Transactions on Broadcasting*, vol. 44, no. 3, Sep 1997.

[21] D. Eager, M. Ferris, and M.Vernon, "Optimized caching in systems with heterogeneous client populations," *Performance Evaluation Special Issue on Internet Performance Modeling*, September 2000.

[22] Y. Guo, S. Sen, and D. Towsley, "Prefix caching assisted periodic broadcast: Framework and techinques to support streaming for popular videos," in *Proceedings of IEEE International Conference on Communications*, vol. 4, 2002, pp. 2607 –2612.

[23] S. Ramesh, I. Rhee, and K. Guo, "Multicast with cache (mcache): An adaptive zero delay video-on-demand service," in *Proceedings of INFOCOM*, vol. 1, 2001, pp. 85–94.

[24] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," in *Proceedings of INFOCOM*, vol. 3, 2002, pp. 1726 –1735.