# Algorithms for Scalable Content Distribution

Ernst W. Biersack (Ed.)[1], Anwar Al Hamra[1], Guillaume Urvoy-Keller[1], David Choi[1], Dimitrios N. Serpanos[2] and Apostolos Traganitis[3]

[1] Institut Eurecom
Departement of Corporate Communications
B.P. 193, 06904 Sophia Antipolis, France
{erbi, alhamra, keller, choi}@eurecom.fr
[2] University of Patras
Department of Electrical and Computer Engineering
serpanos@ee.upatras.gr
[3] University of Crete
Department of Computer Science
tragani@csd.uoc.gr

**Abstract.** In this chapter, we address how to achieve scalable content distributions. We present two contributions, each of which uses a different approach to distribute the content.

In the first part of this chapter, we consider a terrestrial overlay network and build on top of it a VoD service for fixed clients. The goal is to minimize the operational cost of the service. Our contibutions are as follows. First, we introduce a new video distribution architecture that combines open-loop and closed-loop schemes. This combination makes the overall system highly scalable, very cost-effective, and ensures a zero start-up delay. Our second contribution is a mathematical model for the cost of delivering a video as a function of the popularity of that video. Our analytical model, along with some extensions, allows us to explore several scenarios: (i) long videos of 90 min (movies), (ii) short videos of a few min (clips), (iii) the dimensioning of a video on demand service from scratch, and (iv) the case of the optimization of an already installed video on demand service (i.e. the limited resources scenario).

In the second part of this chapter, we consider a satellite distribution of contents to mobile users, or in general to users thar are occasionally connected. We consider a push-based model, where the server periodically downloads objects. We assume that clients register to the service off-line. Our goal is to minimize the mean aggregate reception delay over all objects where each object is weighted by its popularity. Our contibutions in this part are as follows. First, we provide a simple proof for the need of periodicity (equal distance in transmission) of popular objects in a cycle. Second, in contrast to existing results, we consider the scheduling problem for caching clients. To increase the performance of the system,

[1] Institut Eurécom's research is partially supported by its industrial members: Bouygues Télécom, Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom , Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales

we also evaluate a pre-emptive scheduling algorithm that allows interruption (pre-emption) of an object's transmission in order to transmit on schedule another more popular one.

## 1 Introduction

The Internet has evolved, in the last decade, from a research oriented network to a large scale commercial network. Still, most of the trading concerns non-real time documents or services and only two parties are involved in general (dealer/client). The deployment of content distribution networks faces two main obstacles:

- There currently exists no scalable and widely deployed means to serve simultaneously a great number of clients;
- The wide heterogeneity of clients, not only in terms of access bandwidth, but also in terms of reachability since clients might be fixed or mobile.

To achieve a scalable distribution (whose cost grows less than linearly with the number of clients), two approaches have been commonly used: (i) a dedicated overlay network to provide a multicast service at the application layer or (ii) a satellite distribution whose cost is essentially independent from the number of simultaneous clients. In this chapter, we present two contributions that use either one of these scalable distribution techniques with different hypotheses concerning the clients:

In the first part of this chapter, we use a terrestrial overlay network and build on top of it a video on demand service for fixed clients. Our objective is to minimize the operational cost of the service. To minimize the distribution (bandwidth) cost, we use an open-loop central server since its cost is roughly independent of the number of simultaneously connected clients. The only limitation of open-loop algorithms is that they induce a non-zero start-up delay, which accounts for the duration between the moment where the client starts receiving the video stream and the moment it is able to visualize the video on the screen. To achieve a zero start-up delay, we thus introduce servers that stream the beginning of the video (which is referred to as prefix) immediately upon connection of the client to the video on demand service. To build a cost-optimal video on demand service, we thus end up solving an optimization problem whose variables are: the popularity of the requested video, the number of video servers that we use to service the beginning of the video and the fraction of the video that the "prefix" represents.

Our contribution is the design of a mathematical model that allows to find the architecture (i.e. the optimal values of these variables) that minimizes the total operational cost of our video on demand service. In contrast to previous studies, our model considers realistic underlying physical network infrastructures

to accurately model all bandwidth costs and also includes both the I/O and storage requirements. Our analytical model, along with some extensions, allows us to explore several scenarios: (i) long videos of 90 min (movies), (ii) short videos of a few min (clips), (iii) the dimensioning of a video on demand service from scratch, and (iv) the case of the optimization of an already installed video on demand service (i.e. the limited resources scenario).

In the second part of this chapter, we consider a completely different setting since we assume a satellite distribution of content to mobile users that are occasionally connected. We consider a push model where a client has registered off-line to a service that can be modeled as the periodic download of objects representing, for instance, pages, pictures, or short videos. The objective is to minimize the mean aggregate reception delay of all objects where an object is weighted by its popularity. This metric is important since it ensures minimization of the energy consumption at the client side. Existing studies on broadcast systems generally assume that a client is not able to temporarily cache an object. In contrast to these studies we consider the case where clients are equipped with a cache, which is a realistic assumption with respect to the recent evolution of technology.

Our contributions in the second part of this chapter are the following. We first derive a simpler proof than the existing one justifying the periodic transmission of objects during a cycle is optimal for the case of cache-less clients. We then extend this proof to the case of caching clients. We also devise the corresponding optimal algorithm that computes the optimal schedule of each object. In practice, however, achieving a perfect periodicity of the broadcasting of objects within a cycle is an NP-hard problem because the optimal time to transmit an object may lie before the end of transmission of the previous one.

To improve the performance of the system, we investigate several empirical pre-emption strategies, some of them resulting in a real performance improvement while others, like the ones based on interleaving transmissions of multiple objects are proved to be inefficient.

# 2 Cost-optimal Dimensioning of a Large Scale Video on Demand System

## 2.1 Introduction

**Background** Video streams such as MPEG-2 encoded video require several Mbps and providing a VoD service to a large number of clients poses high resource demands to the server and the network. The bandwidth-intensive nature of video requires efficient distribution techniques that typically serve multiple clients who request the same video at approximately the same time via a single video stream that is multicast. VoD systems can be classified in *open–loop systems* [6, 43, 2, 25, 32, 33, 11] and *closed–loop systems* [3, 39, 26, 19, 37, 17].

- Open-loop VoD systems partition each video into smaller pieces called *segments* and transmit each segment at its assigned transmission rate. The first segment is transmitted more frequently than later segments because it is needed first in the playback. All segments are transmitted periodically and indefinitely. In open–loop systems there is no feedback from the client to the server and transmission is completely one–way. Open-loop systems are suitable for popular videos where multicast is efficient. However, open-loop systems introduce a start-up delay[4] and waste bandwidth in the case of non-popular videos.
- Closed-loop systems, on the other hand, require the client to contact the server. Closed–loop systems generally open a new unicast/multicast stream each time a client or a group of clients issue a request for a video. Whenever a new client arrives, the client joins an ongoing multicast stream that has been initiated for earlier clients, if there is any, and retrieves the missed part due to its later arrival via unicast, which ensures an immediate playout of the video. More often, Closed-loop schemes are suitable for videos with moderate popularity where the load on the server is reasonably low.

In the first part of this chapter, we present a new video distribution architecture that combines both, open-loop and closed-loop systems. This combination makes our video distribution architecture suitable for both, popular and non-popular videos. Moreover, having used a closed-loop system to deliver the first part of the video, our video distribution architecture ensures a zero start-up delay.

**Contributions and Related Work** Providing an efficient and scalable video distribution to a large client population has been investigated extensively over the years. The basic idea to achieve scalability is to serve multiple clients via a single stream, using multicast. Open-loop schemes take a full advantage of multicast. *Staggered broadcasting* [6] is the straightforward open-loop scheme where the server allocates for each video $C$ channels each of bandwidth equal to the play back rate $b$ of the video. On each channel, the whole video is broadcast

[4] We ignore here the *transmission* delay due to sending a request to a server or joining a multicast group.

at rate $b$. The starting points of the transmission on the different channels are shifted to guarantee a start-up delay of no more than $L/C$, where $L$ is the length of the video. Clients listen to only one channel at the same time and no storage capacity is needed at the client side. The start-up delay can be improved only by increasing the number of allocated channels $C$.

More efficient and complicated schemes have been proposed later on.
*Pyramid Broadcasting* (PB) [43] divides the video into $C$ segments of increasing size, where $C$ is also the total number of logical channels. The size of each segment is made $\alpha$ times larger than the size of the previous segment ($L_i = \alpha L_{i-1},\ i > 1$). The value of $\alpha$ is set to $\frac{B}{K \cdot C}$, where $K$ is the total number of videos in the system and $B$ is the total bandwidth over all the $C$ channels. Note that B is divided equally amongst the $C$ channels. Segments $i$ for all videos are multiplexed into the same channel $i$ and broadcast consecutively and periodically on that channel $i$ at rate $B/C$. At any moment, the client downloads from at most two channels. This scheme reduces the bandwidth utilization and the start-up delay as compared to the *staggered broadcasting* [6] while guaranteeing a continuous playback of the video. However, its main disadvantage is that segments are transmitted on the different channels at a high rate (i.e. $B/C$) which requires a high I/O capacity at the client side.

*Permutation-based Pyramid Broadcasting* (PPB) [2] addresses the problem of high I/O requirement at the expense of a larger start-up delay and a more complex synchronization. PPB uses the same geometric series proposed by PB to define the length of each segment ($L_i = \alpha L_{i-1},\ i > 1$). Unlike PB, PPB proposes to divide each channel into $K \cdot p$ sub-channels, where $p$ sub-channels are dedicated for the same video segment. Then, each segment is broadcast on $p$ sub-channels in such a way that the access time of segment $i$ is $L_i/p$. Thus, segments are transmitted at rate $B/(C \cdot K \cdot p)$ instead $B/C$ in the case of the PB scheme. On the other hand, at any moment, clients download from at most two separate sub-channels.

Another efficient scheme is the *skyscraper Broadcasting* (SB) [25]. SB uses a recursive function instead of a geometric series to generate the segment lengths. Each segment is then broadcast on a separate channel at the playback rate $b$. Clients listen to at most two channels at the same time. This scheme accounts for buffer space limitations of the clients by constraining the size of the last segment.

*Fast Broadcasting* (FB) [32] divides each video into segments according to a geometric series ($L_i = 2^i$). Each segment is broadcast on a separate channel at the playback rate $b$. Clients listen to all channels at the same time. FB provides the lowest bandwidth requirement at the server as compared to the above schemes.

*Harmonic Broadcasting* (HB) [33] presents another variant from the same general idea. In this scheme, the video is divided into $N$ segments of equal size. Segment $i$ is transmitted at the rate $1/i$ and clients download from all channels at the same time.

All the schemes cited above are constrained to highly regular designs which limits their flexibility. The *Tailored Transmission Scheme* [11] is a more flexible organization that can be adapted to meet different constraints in the system such as limited I/O capacities of the server/clients, limited storage capacity at the client side. This scheme will be detailed more in subsubsection 2.2. For more details on open-loop schemes, see [24].

While open-loop schemes broadcast the video regardless the request pattern of the clients, closed-loop schemes serve the video in response to client requests. Closed-loop schemes can be classified into batching, patching, and hierarchical merging schemes. The basic idea of batching [3, 39] is that the server groups clients that arrive within a given interval of time, in order to serve them via a single multicast stream. However batching introduces a start-up delay.

Patching techniques [26], on the other hand, ensure a zero start-up delay. The first client that requests a video receives a complete stream for the whole video from the server. When a new client arrives after the first one, we distinguish two cases:

- The complete stream that has been initiated for the first client is still active. In this case, the client needs to connect to that stream which changes from unicast to multicast. In addition, the client receives immediately from the server a unicast patch for the part it missed in the complete stream due to its late arrival.
- There is no active complete stream. In this case, the server initiates a new one and the process repeats for clients that arrive later.

Note that clients that are listening to the same complete stream form a session. One efficient extension of patching has been proposed in [19] with the introduction of a threshold policy to reduce the cost of the unicast patches. Whenever a new client arrives and a complete stream is active, the threshold value serves to decide whether to initiate a new complete stream for that client, or whether that client must join the last ongoing complete stream. This scheme will be explained more in subsubsection 2.2.

White et al. [45] propose OBP, a hybrid scheme that combines patching and batching techniques. As for the classical patching [26], in OBP, the server initiates a new complete stream for the first client. A later client either receives a new complete stream in case there is no active complete stream, or the client joins an already existing one. In this later case, in contrast to patching, the client does not receive immediately the missed part in the complete stream; instead, the server batches many clients that arrive within a given interval of time and serve them via multicast. Thereby, as compared to patching, OBP reduces the delivery cost of the missed parts at the expense of a larger commencement viewing delay of the video. Similar to the controlled multicast, the authors introduce a threshold to decide when a new session should start. They also extend the OBP to deal with the case of heterogeneous clients where each client chooses the start-up latency according to the price it is willing to pay for the service.

Hierarchical merging [17] is a more efficient closed-loop scheme. As its name indicates, this scheme merges clients in a hierarchical manner. When a new client

arrives, the server initiates a unicast stream to that client. At the same time, the client listens to the closest stream (target) that is still active. When the client receives via unicast all what it missed in the target stream, the unicast stream is terminated and the client merges into the target stream, and the process repeats. It has been shown that the server bandwidth for the hierarchical merging increases logarithmically with the number of clients.

In this work, we propose a scalable and efficient video distribution architecture that combines open-loop and closed-loop mechanisms to assure a zero start-up delay. Each video is partitioned into a prefix and a suffix. The suffix is stored at a central server, while the prefix is stored at one or more prefix servers. A client who wants to view a video joins an already on-going open-loop multicast distribution of the suffix while immediately requesting the prefix of the video as a patch [26] that is sent either via unicast or multicast [19]. We develop an analytical model for that video distribution architecture that allows to compute for a video with a given popularity the cost-optimal partitioning into prefix and suffix and the placement of the prefix servers in the distribution tree.

In contrast to previous studies (see for example [13, 21, 44]), we

- Model the network as a tree with outdegree $m$ and $l$ levels. In comparison, Guo et al. [21] consider only a two-level distribution architecture.
- Account in the model of the network transmission cost for the number of clients that are simultaneously served by the multicast distribution (either from the prefix servers or the suffix server).
- Allow for the prefix servers to be placed at any level in the distribution tree and not only at the last hop between client and network [44].
- Include in our cost model not only the network transmission cost but also the *server* cost, which depends on both, the storage occupied and the number of input/output streams needed. While the network transmission cost is a major cost factor, the server cost must be included in the overall cost model, especially when we try to design a cost-optimal video distribution architecture. Otherwise, independent of the popularity of a video, the obvious/trivial architecture will be the one where a large number of prefix servers are placed near the clients. While previous papers [21] have treated in their model the storage space of the prefix servers as a scarce resource, we feel that the cost model can be made more realistic by explicitly modeling the cost of the prefix servers.

A related cost model has been presented previously in [35]. The authors model the distribution network as a tree with $l$ levels. In this model, *caches* can be placed at any level in the network. However, these caches can only store the entire video. Our model is more flexible in the sense that any portion of the video can be stored at the prefix servers. The system cost is simply the sum over the storage cost, the I/O bandwidth cost of server/caches, and the network bandwidth cost. Moreover, the cost formulas developed are for simple scenarios with only unicast server/clients connections.

A recent paper by Zhao et al. [48] looks at different network topologies, such as fan-out $K$, daisy-chain, balanced tree and network topologies generated with

topology generators. The analysis focuses on schemes that provide an instantaneous delivery of the video. They derive a tight bound on the minimum network bandwidth requirement for each topology. They show that, at best, the minimum network bandwidth requirement scales as $O(lnN)$, where $N$ is the average number of clients during an interval of time equal to the duration of the video. They also show that it is possible to achieve simultaneously close to the minimum network and server bandwidth usage with practical distribution protocols.

The rest of this sub-chapter is organized as follows: in Subsection 2.2, we present the broadcasting algorithm and content distribution overlay that we use to build the video distribution service. In Subsection 2.3, we present the basic PS-model that allows to obtain the cost formulas for the single video case when there is no a priori constraint on the content distribution network. In Subsection 2.4, we apply the PS-model to the case of long videos, and investigate different scenarios such as heterogeneous or homogeneous bandwidth costs, zero servers costs, etc.. In Subsection 2.5, we prove that the PS-model is still advantageous in the case of short videos, e.g. news clips. In Subsection 2.6, we first study the dimensioning of the system with $n$ videos and no resource constraints. We next devise the algorithm that optimally assigns video to prefix servers in the case of limited resources (I/O, storage) and fixed placement of these servers. We conclude this sub-chapter in Subsection 2.7.

## 2.2 The system environment

**Prefix caching assisted periodic broadcast** Prefix caching assisted periodic broadcast[5] [21] assumes that clients are serviced by a main central suffix server and also by local prefix servers, which can be located throughout the network. A video is partitioned into two parts, the prefix and the suffix, which can be of arbitrary proportion. We denote by $L$ (min) the length of the video and $D$ (min) the length of the prefix. Hence, the suffix will have the length $L - D$. The entirety of the prefix is always *viewed before* the suffix. The main idea of the broadcast scheme is that prefix and suffix transmissions should be decoupled in order to transmit each most effectively. The reason why the prefix and suffix are transmitted differently is that the client must receive the prefix *immediately upon request* while the suffix needs not to be received until the prefix has been completely viewed.

Because the prefix must be immediately received, there is less flexibility in the choice of a transmission scheme for the prefix. In addition, transmitting the prefix from the central server to each client may be costly. In order to reduce transmission costs, the prefix can be stored locally at multiple prefix servers, which can more cheaply transmit the prefix to their local audiences. For the suffix, on the other hand, there is more leeway in the method of broadcast since it needs not to be received immediately. The allowable delay $D$ in transmitting

[5] The term broadcast is commonly used in the literature. In our model, broadcast really refers to multicast as the data that are sent only over links that reach clients who are interested in receiving the video.

the suffix permits to deliver it with an open-loop scheme. Therefore, the suffix is retained at the central server, benefiting from sharing amongst a relatively larger number of clients as well as avoiding the server costs incurred to replicate data across multiple servers.

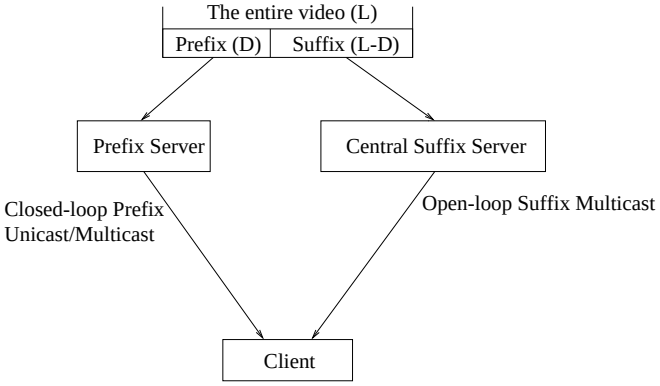



**Fig. 1.** The VoD distribution architecture

Once specific transmission schemes for prefix and suffix have been chosen, the remaining design parameters are the length of the prefix (and suffix) and the height of placement of the prefix servers in the network. The prefix length and the location of the prefix servers should be chosen so as to efficiently divide the workload between central suffix server and prefix servers.

In our prefix caching assisted periodic broadcast model, we choose to transmit the prefix via *controlled multicast*, while the suffix is delivered through *tailored periodic broadcast*.

**The distribution network** We assume that the distribution network is organized as an *overlay* network. An overlay network consists of a collection of nodes placed at strategic locations in existing network, i.e. the Internet. Overlay networks provide the necessary flexibility to realize enhanced services such as multicast [31] or content distribution [1] and are typically organized in a hierarchical manner.

In our model, we assume that the topology of our distribution network is a $m$-ary tree with $l$ levels (see figure 2). The suffix server is assumed to be at the root. The prefix servers may be placed at any level of the distribution network other than the highest level (i.e. leaves). If the prefix servers are placed at level $j$, there will be one at each node of that level, which makes a total of $m^j$ prefix servers. The clients are lumped together at the $m^l$ leaf nodes. The number of clients watching simultaneously a video is not limited to $m^l$ since a leaf node does not represent a single client but multiple clients that are in the same building.

We digress briefly to consider the practical aspects of a network tree model. A tree model captures the hierarchical structure of a large-scale network, where large backbone routers service many smaller service providers which in turn service the end-user clients. For example, a tree might include multiple levels, dividing the network into national, regional and local sub-networks.

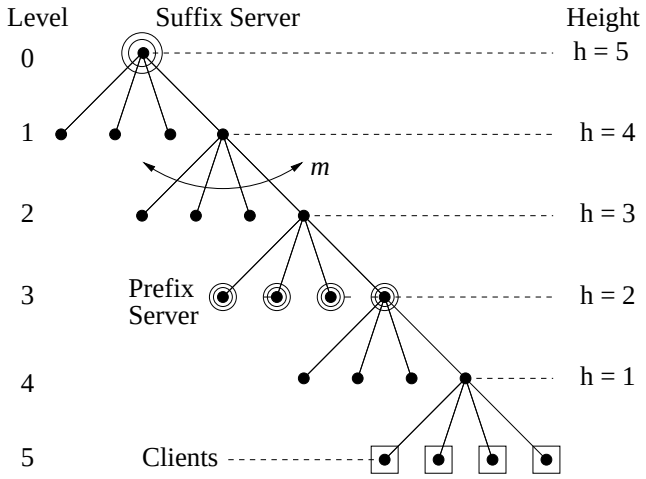



**Fig. 2.** Video distribution network

The distribution network is assumed to support both unicast and multicast transmissions. Unicast transmission occurs between a server and a single client, whereas multicast transmission occurs when multiple clients (possibly from different leaf nodes) all simultaneously receive the same single transmission from a server. We assume that for the duration of a transmission, a cost must be paid only for every link spanned between the server and its active client(s). The per-link cost may differ depending upon the specific links that are utilized.

For a multicast transmission, the cost may change over the duration of the transmission as users join and leave. Note also that if multiple clients reside at a single leaf node then the cost of multicast transmission is effectively the same as if there were only a single client at that node. For clients at different nodes, multicast still offers savings due to links shared at the lower levels by different nodes.

**Prefix transmission via controlled multicast** Patching was first proposed in [26] and then extended with the inclusion of a thresholding policy to produce **Controlled Multicast** [19]. The key idea of patching is to allow clients to share segments of a video stream when they arrive at different times. As the number of clients increases from one to several, the transmission stream is changed from a unicast stream to a multicast one so that late arrivals can still share in the

remainder of the stream. In addition, a separate unicast stream must also be transmitted to each client after the first one in order to deliver the data missed due to its later arrival.

For extremely late arrivals, the cost of the additional unicast transmission may outweigh the benefits of sharing in the remaining transmission. Controlled multicast modifies patching to allow for this scenario. Whenever a new transmission is started at time $t$, arriving clients are patched onto the stream until time $t+T$, where $T$ is a **thresholding** parameter. The first client to arrive after time $t+T$ is given a brand new transmission, and all future arrivals are patched onto the new transmission instead of the old one, until the threshold time passes again and the process is repeated. Figure 3 illustrates the operation of controlled multicast. At time $t_1$ the first client arrives: The prefix server starts transmitting the prefix via unicast. When the second client joins at time $t_2$, the remaining part of the prefix is multicast to clients 1 and 2. Client 2 additionally receives the initial part of the prefix that had been transmitted between $t_1$ and $t_2$ via a separate unicast transmission. Since client 3 arrives at time $t_3$, with $t_3 - t_1 > T$, the prefix server starts a new unicast transmission of the entire prefix.

The costs of controlled multicast have been shown to increase sub-linearly with the arrival rate of requests and the length of the prefix [37]; however, the analysis assumes a network of a single link between the server and all its clients. This is the case when the prefix servers are located one level above the clients. We refer to that case as **leaf prefix server placement**. Placing the prefix servers higher up in the distribution network increases the cost of transmission; however, it consolidates the arrivals to a smaller number of prefix servers and thereby allows for more sharing to occur amongst clients. Furthermore, placing the prefix servers higher up in the network reduces server costs since there are fewer copies of the prefix. One contribution of this paper is an analysis of the tradeoffs in prefix server placement for controlled multicast.

**Tailored periodic broadcast of the suffix** We use tailored transmission [11] to transmit the suffix. Thereby, we divide the suffix into segments of fixed lengths. If there are no clients then the suffix server does not transmit. As long as there is at least one client, each segment is periodically multicast at its own transmission rate. Arriving clients receive the multicast of each segment simultaneously. Clients are not expected to arrive at the starting point of each segment; instead, they begin recording at whatever point they arrive, store the data and reconstruct each segment as they receive the data.

The length and rate of each segment is chosen so as to minimize the bandwidth subject to the constraint that each segment must be completely received before its time of playback, and also subject to the storage and reception capabilities of the clients. Figure 4 illustrates the tailored transmission scheme for the case of minimal transmission rates. The client starts receiving all segments at time $t_0$. The shaded areas for each segment contain exactly the content of that segment as received by the client who started recording at time $t_0$.

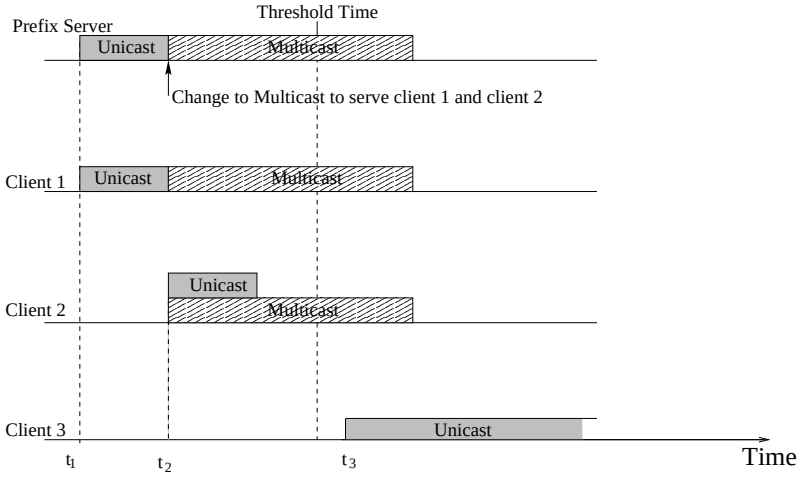



**Fig. 3.** Prefix transmission with multicast and patching

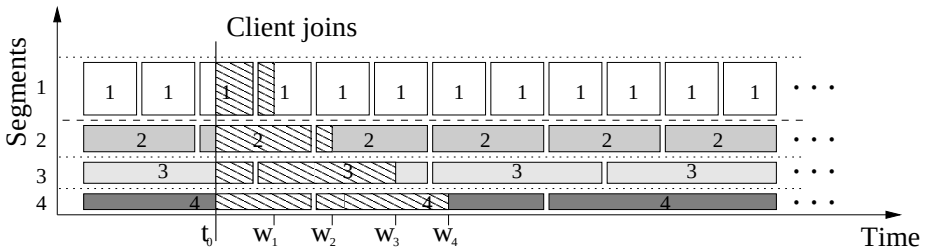



**Fig. 4.** Tailored Broadcast transmission

The total server transmission bandwidth is:

$$R_t^{min} = \sum_{i=1}^{N_s} r_i^{min}$$

where $N_s$ , $\lceil \frac{L-D}{D} \rceil$ is the number of segments the suffix consists of and $r_i^{min}$ is the minimal transmission rate of segment $i$. When we break the suffix of length $L - D$ into segments, each of the segments 1 to $N_s - 1$ has length $D$ and the last segment $N_s$ has a length of $(L - D) - (N_s - 1)D = L - N_sD$. Segment $i$ with length $D$ needs to be entirely received no later than $iD \cdot 60$ seconds after the start of the video consumption. The factor 60 in $iD \cdot 60$ is due to the fact that the lengths of the video, the prefix, and the suffix are expressed in *minutes*. Therefore, the minimal transmission rate for segment $i$ is $r_i^{min} = b \, \frac{D \cdot 60}{iD \cdot 60} = \frac{b}{i}$, where $b$ [Mbit/sec] is the consumption rate of the video and $bD \cdot 60$ is the amount of data [Mbit] in a segment of length $D$.

In the case where the length of the last segment $N_s$ is less than $D$ (i.e. $\frac{L-D}{D}$ is not an integer), the transmission rate $r_{N_s}^{min}$ is computed as follows depending on whether $N_s$ is larger than 1 or not.

- If $N_s > 1$, the segment $N_s$ should be entirely received $N_s D \cdot 60$ seconds after the start of the video consumption. As we will see later, we assume that all the segments are multiplexed onto a single multicast channel and the receiver stays tuned into that multicast channel until it has received the last segment. Therefore, clients will receive the first segments multiple times. Thus, in this case where the length of segment $N_s$ is less than $D$, it might be efficient to reduce the tuning time of clients. This can be done by reducing the transmission time of segment $N_s$ to $(N_s - 1)D \cdot 60$ instead $N_s D \cdot 60$ seconds. For a video with a low demand, reducing the service time of the client to $(N_s - 1)D \cdot 60$ increases the transmission rate $r_{N_s}$ of the last segment; however, the increase is offset by avoiding useless transmissions of some of the first segments. For a popular or very popular video, the prefix length is quite short which means that $N_s \sim (N_s - 1)$, and so the increase in $r_{N_s}$ has a negligible impact on the overall server transmission bandwidth $R_t^{min}$. So, using the fact that $N_s = \lceil \frac{L-D}{D} \rceil = \lfloor \frac{L}{D} \rfloor$, the transmission rate of segment $N_s$ then becomes $r_{N_s}^{min} = b \, \frac{(L-N_s D)\cdot 60}{(N_s-1)D\cdot 60} = b \, \frac{(\frac{L}{D} - \lfloor \frac{L}{D} \rfloor)D}{(N_s-1)D} = b \, \frac{\frac{L}{D} - \lfloor \frac{L}{D} \rfloor}{N_s - 1}$.
- If $N_s = 1$, the suffix consists of one segment of length $(L - D) < D$ that must be entirely received $D \cdot 60$ seconds after the start of the video consumption. The transmission rate of segment $N_s$ then becomes $r_{N_s}^{min} = r_1^{min} = b \, \frac{(L-D)\cdot 60}{D\cdot 60} = b \, \frac{L-D}{D}$.

The total server transmission bandwidth is therefore:

$$R_t^{min} = \begin{cases} b \, ( \sum_{i=1}^{\lfloor \frac{L-D}{D} \rfloor} \frac{1}{i} + \frac{\frac{L}{D} - \lfloor \frac{L}{D} \rfloor}{N_s - 1} ) & \text{if } N_s > 1 \\ b \, \frac{L-D}{D} & \text{if } N_s = 1 \end{cases}$$

Transmitting all the segments at their minimal transmission rate requires that the client starts receiving all segments simultaneously. As a consequence, parts of the video will be received some time before they are consumed and must be stored by the client in the meantime. Figure 5 plots the evolution with time of the amount of data stored for a video of length $L = 90$ min and a prefix length of $D = 2$ min. We see that at the beginning, more and more data will be received ahead of time so that the amount of data keeps increasing until it reaches a peak corresponding to nearly 40 percent of the video. From there on, data will be consumed at a rate higher than the aggregate rate at which new data are received and the storage curve decreases. Storing up to 40 percent of the video data does not pose any problem with todays equipment. In fact, there already exist products such as the digital video recorder by TiVo [40] that can store up to 60 hours of MPEG II encoded video.

The tailored periodic broadcast scheme also allows to support user interactions [10] such as fast forward if the transmission rate of each segment is increased by a small factor (less than twice the minimal transmission rate), provided that the client has enough buffer to store large parts of the video.

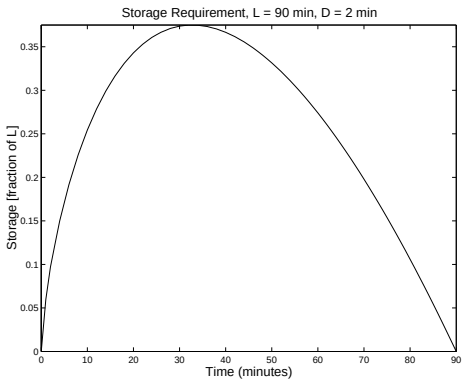



**Fig. 5.** Storage required at client as function of time, $L = 90, D = 2$

**Interaction between clients and servers** When a new client wants to receive a video, it will contact its closest prefix server in the distribution tree. The prefix server will either start a new transmission cycle of the prefix or extend the ongoing prefix multicast transmission to the new client and transmit a unicast patch to the client. However, the client does not need to contact the suffix server. The suffix server simply needs to know whether there is currently at least one client who needs to receive the suffix, which the suffix server can learn by communicating with the prefix servers. Other large scale video distribution schemes such as hierarchical stream merging [18] require that all client requests be handled by the central server, which makes the central server a potential bottleneck. Our video distribution scheme is not only *scalable* in terms of the network and server resources required to stream a video to a large number of clients but also in terms of processing incoming client requests.

### 2.3 PS-model, a cost model for the video transmission

**Introduction** We divide the costs of a VOD network into network and server costs. The network costs are proportional to the amount of network bandwidth which is transmitted over each link between a server and its clients. The server cost is dependent upon the necessary storage and upon the total number of input/output streams which the server(s) must simultaneously support over the network.

It is interesting to note that the storage and input/output stream capacity of a server cannot be purchased in arbitrary quantities. Instead, they can usually only be purchased in discrete increments. As a result, it is unlikely that a server can be purchased to exactly match both storage and streaming requirements; typically one constraint will be slack as the server will either have extra storage or extra streaming capability.

We will examine the expected delivery cost for a video of length $L$ as a function of the average request rate per *minute* $\lambda$ (1/min), the prefix length (and allowable suffix delay) $D$ and the topology of the network. The maximum output capacities should be fixed for each server; however, to facilitate the analysis we will assume that any number of streams can be allocated with the costs paid on a per-stream basis.

We divide the multicast tree into levels $1, \ldots, l$, where level 1 consists of the $m$ links from the root and level $l$ consists of the $m^l$ links connected to the leaf nodes. Arrivals to a link at level $j$ are modeled as a Poisson process with parameter $\lambda/m^j$. As a consequence, arrivals at the root will form a Poisson process with parameter $\lambda$.

We first derive the cost of prefix transmission with a single prefix server at the root. We next generalize the root case to the general case where the prefix servers are placed at some level between the root and the clients. The costs fall into three categories: network, storage capacity and I/O capacity.

We neglect the effects of network latency, even though they can be important from an operational point of view. One might treat latency effects by constraining the maximum number of hops allowed between prefix servers and their clients.

We will derive the cost for the prefix and for the suffix transmission separately and then combine both to obtain the overall system cost. We will refer to this cost model also as the **PS-model** to distinguish it from other cost models.

### Costs for prefix transmission

*Prefix server at the root* We first consider a single prefix server at the root of the multicast tree. We will afterwards generalize our results to the case where the prefix server is at an arbitrary height in the tree.

**Network bandwidth costs:**

A single server at the root combines many small request arrival streams (to the leaves) into a single large arrival stream (to the root); this should lower costs since the cost of prefix transmission via controlled multicast is sub-linear in the arrival rate because it promotes efficiency through shared streams; by combining all the requests arriving at the root, we increase the possibility for sharing. However, this is counterbalanced by the fact that sharing is no longer free; if two clients share the same I/O stream but reside on different leaves, a separate network cost must be paid for each of them. Of course, if clients are already active at every leaf node, then no new network costs must be paid for any future arrivals. However, this scenario is unlikely even for high arrival rates because high arrival rates produce short threshold times in order to reduce the length of the unicast streams.

Let $t_i$ be the time of the $i$th complete multicast transmission of the prefix without any patching. Arrivals between times $t_i$ and $t_{i+1}$ will share from the multicast transmission at time $t_i$ and will each receive a separate unicast transmission for the data which were missed. We can divide the patching process up into separate renewal cycles $(t_1 t_2], (t_2 t_3], \ldots$ which are independents and identically distributed in their usage of bandwidth. We analyze the bandwidth usage over a single renewal cycle.

Given the threshold time $T$, on average there will be $T\lambda$ arrivals which will each need partial transmission of the prefix in unicast. The average length of the unicast transfer will be $T/2$ since the arrivals are uniformly distributed over time. Finally a bandwidth cost must be paid for every link on the path between client (at the leaf) and the root server. As a result, the total amount of data transmitted for the unicast streams over one renewal cycle is

$$C_{netw}^{unicast} = b \cdot 60 \, \frac{lT^2\lambda}{2}.$$

Each arrival will also share from a single multicast network stream. A price must be paid for every link in use. Given a link at level $j$, let $\tau_j$ be the duration of time in which the link is active. For the multicast stream, a link is active from the time of the first arrival (before time $T$) to that link to the end of the prefix at time $D$. Arrivals to a link at level $j$ form a Poisson process with parameter $\lambda/m^j$.

As each renewal cycle begins with the activation of a stream between the root and a single client, we know that one link at each level will be active at time zero. Therefore $\tau_j = D$ with probability $1/m^j$. We will now write out an expression for $\mathrm{E}[\tau_j]$:

$$\begin{aligned} \mathrm{E}[\tau_j] &= D\mathrm{P}\{\tau_j = D\} + \mathrm{E}[\tau_j | \tau_j \neq D]\mathrm{P}\{\tau_j \neq D\} \\ &= D\frac{1}{m^j} + \mathrm{E}[\tau_j | \tau_j \neq D]\frac{m^j - 1}{m^j}. \end{aligned}$$

Given a Poisson process with parameter $\lambda/m^j$, the time of first arrival will have an exponential distribution with parameter $\lambda/m^j$ and a cumulative distribution $F(t) = 1 - e^{-\frac{\lambda}{m^j}t}$. We evaluate $\mathrm{E}[\tau_j | \tau_j \neq D]$ making use of the fact that

$\int_0^T t\frac{\lambda}{m^j}e^{-\frac{\lambda}{m^j}t}dt = \int_0^T (e^{-\frac{\lambda}{m^j}t} - e^{-\frac{\lambda}{m^j}T})dt.$

$$\begin{aligned}
\mathrm{E}[\tau_j|\tau_j \neq D] &= \int_0^T (D-t)\frac{\lambda}{m^j}e^{-\frac{\lambda}{m^j}t}dt \\
&= \int_0^T D\frac{\lambda}{m^j}e^{-\frac{\lambda}{m^j}t}dt - \int_0^T t\frac{\lambda}{m^j}e^{-\frac{\lambda}{m^j}t}dt \\
&= D(1-e^{-\frac{\lambda}{m^j}T}) - \int_0^T (e^{-\frac{\lambda}{m^j}t} - e^{-\frac{\lambda}{m^j}T})dt \\
&= D(1-e^{-\frac{\lambda}{m^j}T}) - (-\frac{m^j}{\lambda}e^{-\frac{\lambda}{m^j}t} - e^{-\frac{\lambda}{m^j}T}t)\Big|_{t=0}^{T} \\
&= D(1-e^{-\frac{\lambda}{m^j}T}) - \frac{m^j}{\lambda} + \frac{m^j}{\lambda}e^{-\frac{\lambda}{m^j}T} + Te^{-\frac{\lambda}{m^j}T}.
\end{aligned}$$

Substituting $\mathrm{E}[\tau_j|\tau_j \neq D]$ into $\mathrm{E}[\tau_j]$ produces

$$\begin{aligned}
\mathrm{E}[\tau_j] &= D\frac{1}{m^j} + \mathrm{E}[\tau_j|\tau_j \neq D]\frac{m^j-1}{m^j} \\
&= D\frac{1}{m^j} - \left(\frac{m^j}{\lambda} - \frac{m^j}{\lambda}e^{-\frac{\lambda}{m^j}T} - Te^{-\frac{\lambda}{m^j}T} - D(1-e^{-\frac{\lambda}{m^j}T})\right)\frac{m^j-1}{m^j} \\
&= D\left(\frac{1}{m^j} + (1-e^{-\frac{\lambda}{m^j}T})\frac{m^j-1}{m^j}\right) - (1-e^{-\frac{\lambda}{m^j}T})\frac{m^j-1}{\lambda} + (\frac{m^j-1}{m^j})Te^{-\frac{\lambda}{m^j}T} \\
&= D(1-e^{-\frac{\lambda}{m^j}T}(1-\frac{1}{m^j})) - (1-e^{-\frac{\lambda}{m^j}T})\frac{m^j-1}{\lambda} + (\frac{m^j-1}{m^j})Te^{-\frac{\lambda}{m^j}T}.
\end{aligned}$$

By summing over all the links in the tree we find the total multicast cost $C_{netw}^{multicast}$:

$$C_{netw}^{multicast} = b \cdot 60 \sum_{j=1}^{l} m^j \mathrm{E}[\tau_j]$$

and the average network bandwidth cost $C_{netw}^{root}$ can be found by dividing by the average duration of each renewal cycle $(T + 1/\lambda) \cdot 60$.

$$\begin{aligned}
C_{netw}^{root} &= \frac{C_{netw}^{multicast} + C_{netw}^{unicast}}{(T+1/\lambda)\cdot 60} \\
&= b\ \frac{\sum_{j=1}^{l} m^j \mathrm{E}[\tau_j] + lT^2\lambda/2}{T+1/\lambda}.
\end{aligned}$$

**Server costs for prefix server placed at the root:**

It is easy to see that the storage cost $C_{sto}^{root}$ of a single root server will be $b{\cdot}60D$. The I/O stream cost must be paid for the output capabilities of each server, i.e. the number of input/output streams which a server can simultaneously maintain. From the expression of $C_{netw}^{root}$ above, one can conclude that the average number of streams for a root server is equivalent to

$$C_{I/O}^{root} = b\ \frac{D + T^2\lambda/2}{T+1/\lambda}.$$

If $T$ is chosen to minimize the number of I/O streams then

$$C_{I/O}^{root} = b(\sqrt{2D\lambda + 1} - 1).$$

However, choosing $T$ to minimize the number of concurrent I/O streams will unnecessarily increase the network bandwidth. If $T$ is chosen to minimize the network bandwidth, then $C_{I/O}^{root} = b(\sqrt{2CD\lambda/l + 1} - 1 - \frac{D\lambda(C/l-1)}{\sqrt{2CD\lambda/l+1}})$, where $C$ is a scalar constant.

In case of a non-popular video that has on average, at most, one arrival in an interval of time $D$ ($\lambda < 1/D$), each request activates a new prefix transmission and then a new cycle. Hence, the threshold time $T$ becomes *zero* and the expressions for $C_{netw}^{root}$ and $C_{I/O}^{root}$ simplify to

$$C_{netw}^{root} = bl\lambda D$$
$$C_{I/O}^{root} = b\lambda D \ .$$

*Varying the placement of the prefix server* We now generalize the model to the case where the prefix servers can be placed at any height in the network tree. By placing the prefix servers at some height $h$ in the tree where $1 < h < l$, we divide the arrival process between $m^{l-h}$ servers, each of which can be considered as the root of a network tree with height $h$. We need only therefore to consider the root server case for a tree of the proper height $h$ with arrival rate $\lambda/m^{l-h}$, and then multiply the costs by the number of servers $m^{l-h}$. The resulting formulas are listed in table 1. The height $h = 1$ refers to the case of a leaf server, i.e. one level before the clients.

| **Cost terms** | |
|---|---|
| $C_{netw}^{prefix}$ | $b \cdot m^{l-h} \frac{\frac{hT^2\lambda}{2m^{l-h}} + \sum_{j=1}^{h} m^j \,\mathrm{E}[\tau_j]}{T + m^{l-h}/\lambda}$ <br> $(\mathrm{E}[\tau_j] = D(1 - e^{-\frac{\lambda}{m^j m^{l-h}} T}(1 - \frac{1}{m^j})) - (1 - e^{-\frac{\lambda}{m^j m^{l-h}} T}) \frac{m^{l-h}(m^j-1)}{\lambda} + (\frac{m^j-1}{m^j}) T e^{-\frac{\lambda}{m^j m^{l-h}} T})$ |
| $C_{sto}^{prefix}$ | $b \cdot 60 \cdot m^{l-h} \cdot D$ |
| $C_{I/O}^{prefix}$ | $b \cdot \frac{D + T^2\lambda/2}{T + 1/\lambda}$ |

**Table 1.** Summary of the prefix cost terms for the PS-model ($l$ levels, prefix servers at height $h$)

**Costs for suffix transmission with a multicast tree** The costs once again fall into three categories: bandwidth, storage capacity and streaming capacity.

The bandwidth cost is equal to the transmission rate $R_t^{min}$ multiplied by the average number of active links, which we will now calculate. For the periodic broadcast, each arrival is serviced for an amount of time $\mathrm{E}[T_s]$ (min), which equals the transmission time of the last segment:

$$\mathrm{E}[T_s] = \begin{cases} \lfloor \dfrac{L-D}{D} \rfloor \cdot D & \text{if } N_s > 1 \\ D & \text{if } N_s = 1 \end{cases}$$

We assume that all segments are multiplexed to a single multicast channel. As a consequence, each client will consume a bandwidth of $R_t^{min}$ during all the transmission of the suffix. If one multicast channel were dedicated to each segment, the bandwidth consumption could be reduced; the client would be connected only to channels corresponding to segments not yet received. However, this reduction in bandwidth cost comes at the expense of a more complex multicast transmission and a complex synchronization between channels. This study is left for future work.

From queuing theory, it can be shown that given an expected service time $\mathrm{E}[T_s]$ and memoryless arrivals with parameter $\frac{\lambda}{m^j}$, the probability of $n$ jobs simultaneously in progress is given by

$$\mathrm{P}\{n \text{ jobs}\} = \frac{e^{-\frac{\lambda}{m^j}\mathrm{E}[T_s]}(\frac{\lambda}{m^j}\mathrm{E}[T_s])^n}{n!},$$

which is a Poisson distribution. This result can be found through the derivation of the Erlang call-blocking formula commonly used in telecommunications. Arrivals to a link at level $j$ are memoryless with parameter $\lambda/m^j$. Define $P(j)$ as the probability that a link at level $j$ has any requests:

$$P(j) \;, 1 - \mathrm{P}\{0 \text{ jobs}\} = 1 - e^{-\frac{\lambda}{m^j}\mathrm{E}[T_s]}$$

Then

$$P(j) = \begin{cases} 1 - e^{-\frac{\lambda \lfloor \frac{L-D}{D} \rfloor D}{m^j}} & \text{if } N_s > 1 \\ 1 - e^{-\frac{\lambda D}{m^j}} & \text{if } N_s = 1 \end{cases}$$

The expected number of active links at any given time is therefore

$$\mathrm{E}[\text{active links}] = \sum_{j=1}^{l} m^j P(j),$$

and the bandwidth is

$$C_{netw}^{suffix} = R_t^{min} \sum_{j=1}^{l} m^j P(j). \tag{1}$$

On the other hand, the suffix is continuously and periodically multicast independent of the arrival rate as long as there is at least one user (if there are no users, the suffix server will not send the suffix). In contrast, the number of I/O channels does vary with $L$ and $D$. The I/O stream cost is equal to the rate $R_t^{min} \times P(0)$, where $P(0)$ is the probability that there is at least one user active at the root:

$$C_{I/O}^{suffix} = \begin{cases} b \left( \sum_{i=1}^{\lfloor \frac{L-D}{D} \rfloor} \frac{1}{i} + \frac{\frac{L}{D} - \lfloor \frac{L}{D} \rfloor}{N_s - 1} \right) \left(1 - e^{-\lambda \lfloor \frac{L-D}{D} \rfloor D)}\right) & \text{if } N_s > 1 \\ b \, \frac{L-D}{D} (1 - e^{-\lambda D}) & \text{if } N_s = 1 \end{cases}$$

The storage cost is proportional to the length of the suffix, which is $C_{sto}^{suffix} = b \cdot 60(L - D)$.

The terms of the suffix costs are given in table 2, while table 3 shows all the important mathematical terms used.

| **Cost terms** | | |
|---|---|---|
| $C_{netw}^{suffix}$ | $b \left( \sum_{i=1}^{\lfloor \frac{L-D}{D} \rfloor} \frac{1}{i} + \frac{\frac{L}{D} - \lfloor \frac{L}{D} \rfloor}{N_s - 1} \right) \sum_{j=1}^{l} m^j \left(1 - e^{-\frac{\lambda \lfloor \frac{L-D}{D} \rfloor D}{m^j}}\right)$ | if $N_s > 1$ |
| | $b \, \frac{L-D}{D} \sum_{j=1}^{l} m^j \left(1 - e^{-\frac{\lambda D}{m^j}}\right)$ | if $N_s = 1$ |
| $C_{sto}^{suffix}$ | $b \cdot 60 \ (L - D)$ | |
| $C_{I/O}^{suffix}$ | $b \left( \sum_{i=1}^{\lfloor \frac{L-D}{D} \rfloor} \frac{1}{i} + \frac{\frac{L}{D} - \lfloor \frac{L}{D} \rfloor}{N_s - 1} \right) \left(1 - e^{-\lambda \lfloor \frac{L-D}{D} \rfloor D}\right)$ | if $N_s > 1$ |
| | $b \, \frac{L-D}{D} (1 - e^{-\lambda D})$ | if $N_s = 1$ |

**Table 2.** Summary of the suffix cost terms for the PS-model.

**Overall system cost for the case of a single video** We divide the costs of a VoD service into network and server costs. The network costs are proportional

| Term | Definition |
|---|---|
| $m$ | Tree breadth |
| $l$ | Tree depth |
| $h$ | Prefix server height |
| $L$ | Video length (min) |
| $D$ | Prefix length (min) |
| $L-D$ | Suffix length (min) |
| $N_s$ | Number of segments of the suffix ($N_s$ , $\lceil \frac{L-D}{D} \rceil$) |
| $\lambda$ | Average client request arrival rate (1/min) |
| $N$ | Video popularity ($N$ , $\lambda L$) |
| $\tau_j$ | Active occupation duration for link at depth $j$ (prefix transmission) |
| $P(j)$ | Prob. link at depth $j$ is active (suffix transmission) |
| $T$ | Threshold time |
| $b$ | Consumption rate of the video [Mbit/sec] |
| $C$ | Scalar constant |

**Table 3.** Important Mathematical Terms

to the amount of network bandwidth that is expended for the transmission of the prefix and the suffix. The server costs depend upon the necessary storage and upon the total number of input/output streams needed for the suffix server and the prefix server(s).

The total cost of the system can be computed as the sum of the total network and total server costs:

$$C_{PS}^{system} = C_{netw}^{system} + \gamma C_{server}^{system} \tag{2}$$

To relate the network and the server costs, a normalization factor $\gamma$ is introduced that allows us to explore various scenarios for the cost of the servers as compared to the cost for the transmission bandwidth. We consider here the values of $\gamma = \{0, 0.1, 1\}$. The case of $\gamma = 0$ corresponds to the case that only the cost for network transmission is taken into account and the cost for the servers is not considered at all (considered to be zero). $\gamma = 0.1$ provides high network cost relative to the server cost, while $\gamma = 1$ represents the case where the network cost is relatively low as compared to the server cost.

The terms for the network and server costs are given by:

$$C_{netw}^{system} = C_{netw}^{prefix} + C_{netw}^{suffix}$$
$$C_{server}^{system} = C_{server}^{prefix} + C_{server}^{suffix}$$

The server cost depends on both, the required amount of storage $C_{sto}$ (in Megabit) and the amount of disk I/O bandwidth $C_{I/O}$ (in Megabit/sec).

$$C_{server}^{prefix} = \max(C_{I/O}^{prefix}, \beta C_{sto}^{prefix})$$
$$C_{server}^{suffix} = \max(C_{I/O}^{suffix}, \beta C_{sto}^{suffix})$$

To be able to relate the cost for storage and for I/O, we introduce the normalization factor $\beta$ that is determined as follows: If our server has a storage capacity of $d_{sto}$ [Megabit] and an I/O bandwidth of $d_{I/O}$ [Megabit/sec], then $\beta$ , $\frac{d_{I/O}}{d_{sto}}$. Since the server will be either I/O limited (I/O is the bottleneck and no more requests can be served) or storage limited (storage volume is the bottleneck and no more data can be stored), the server cost is given as the *maximum* of $C_{I/O}$ and $\beta C_{sto}$.

To model a case where the cost for the "last-hop link" towards the clients is not the same as the cost for the other links, we can set the cost for the last link to the clients (*lhc*) to a value different from the cost for the other links.

### 2.4 Results for long videos

**Introduction** We consider a distribution network with an out-degree $m = 4$ and a number of levels $l = 5$. We expect that such a topology is representative for a wide distribution system that covers a large geographical areas of the size of a country such as France or the UK. If one wants to model a densely populated metropolitan area such as NewYork, one would choose $l < 5$ (e.g. $l = 2, 3$) and $m > 4$ (e.g. $m = 10$). Our model has quite a few parameters and we present results only for a limited subset of parameter values that can provide new insights. For the rest of the paper, we will vary only the parameters $\gamma$, last-hop cost *lhc*, and video length $L$. The other parameters are chosen as follows: For the disk I/O cost to disk storage cost ratio $\beta$, we choose $\beta = 0.001$, which is a realistic value for the current disk technology such as the IBM Ultrastar 72ZX disk. The video length will be $L = 90$ min for most of the time, except when we consider very short video clips of lengths $L = 2$ and 7 min.

**Homogeneous link costs** For the first results presented, the server cost is weighted by $\gamma = 1$ and the network per-link costs are uniform at all levels of the network ($lhc = 1$). We first plot in figure (see figure 6) the optimal system cost as a function of the video popularity $N$. The video popularity $N$ represents the number of clients requesting a same video in an interval of time of duration $L$ ($N = \lambda L$).

In figure 6 we see that the total cost efficiency improves with increasing video popularity $N$: For a 10-fold increase in video popularity $N$ from 9,000 to 90,000 clients, the cost only doubles.

The optimal values for the prefix length and prefix server placement in the hierarchy as a function of the video popularity $N$ are given in figures 7(a) and 7(b). We see that both, the prefix server height and the prefix length decrease monotonically with $N$.

For videos that are rarely demanded ($N < 20$), the prefix server is placed at the *root* and the optimal prefix comprises the whole video of 90 min. Indeed, for $N < 20$ the storage cost due to a replication of the prefix in multiple prefix servers is not justified and the optimal architecture is a *centralized* one. On the other hand, for videos that are popular or very popular, the optimal architecture

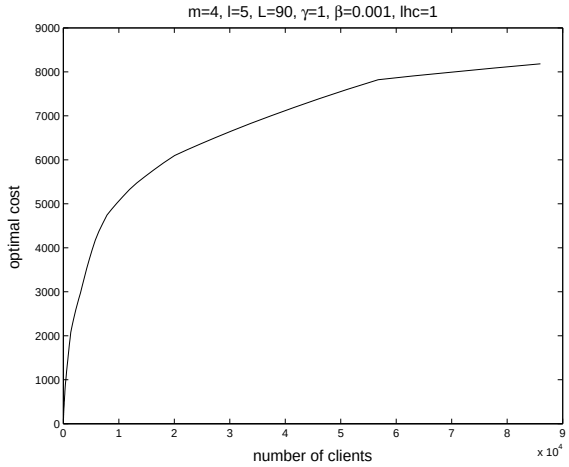
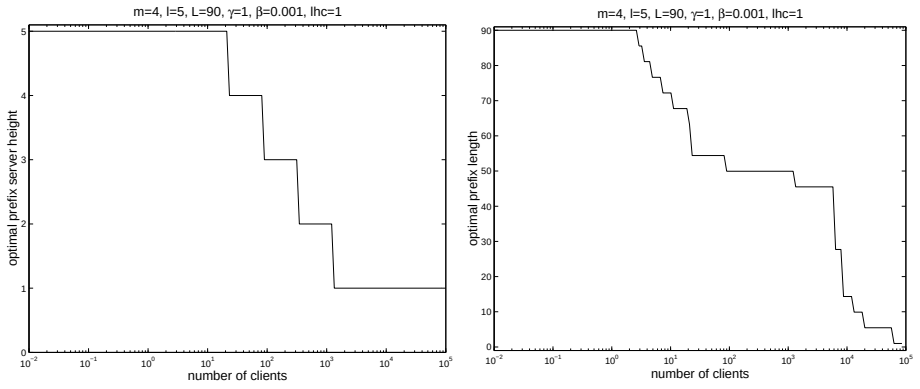



**Fig. 6.** Total system cost $C^{system}$ with optimal prefix server height and optimal prefix length, for $\gamma = 1$ and $lhc = 1$

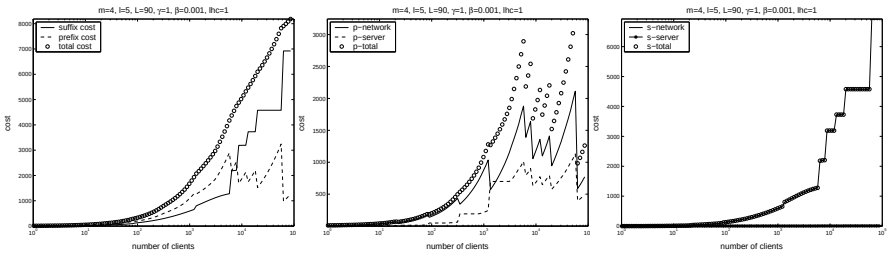
is a *distributed* one with the server for the suffix at the root and the prefix servers closer to the clients. As the popularity $N$ increases, the optimal prefix length decreases since the transmission bandwidth required by the prefix server increases with the square root of the number of clients served, while for the suffix server, the transmission bandwidth required depends for very high values of $N$ only on the length of the suffix and not the number of clients served.




(a) Optimal prefix server height (b) Optimal prefix length (min)

**Fig. 7.** Optimal prefix server height and optimal prefix length for $\gamma = 1, lhc = 1$.

We plot the prefix-suffix cost breakdown in figure 8. We see that the suffix cost $C^{suffix}$ is initially lower than the prefix cost $C^{prefix}$ since the prefix length is quite long. Eventually, for an increasing video popularity $N$, the suffix system becomes more cost efficient, and so the length of the prefix is significantly reduced and the suffix cost becomes greater than the prefix cost. From figure 8(c) we see that the suffix cost $C^{suffix}$ for higher values of $N$ is entirely determined by the suffix *network* cost. In the following, we will not present the suffix cost breakdown anymore since it does not provide any additional insight. For a given suffix length, the fact that $C^{suffix}$ does not change with $N$ indicates that *all* the links of the video distribution network are active, i.e. the multicast transmission is in fact a broadcast to all leaf nodes.




(a) total system cost

(b) prefix cost breakdown

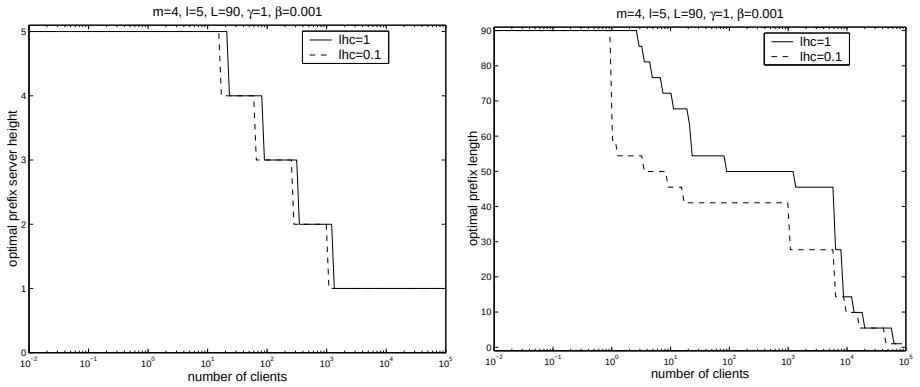
(c) suffix cost breakdown

**Fig. 8.** Breakdown of costs for $\gamma = 1$ and $lhc = 1$

If we take a closer look at the evolution of the prefix *server* cost for very popular videos with $N > 10^3$, we see (figure 8(b)) that the prefix server cost increases linearly with increasing $N$. In this case, to achieve an optimal system cost $C^{system}$, the prefix length is frequently shortened.

**Heterogeneous link costs** We now set the cost of transmission between levels 4 and 5 (the "last-hop" from the root to the clients at the leaves) to be one-tenth the normal network cost. A reduced last-hop link cost would apply to situations where the local service providers are much cheaper than their regional and national counterparts since they support less traffic. A recent study [8] contains a cost comparison for transmission links of different bandwidth that indicates that the cost in Mbit/hour for local access via ADSL or Cable is at least one order of magnitude lower than the cost for a high speed OC-3 or OC-48 link. We can model this case by using a reduced last-hop link cost, which is $\frac{1}{10}$ of the cost for the other links. We refer to this case as $lhc = 0.1$.

In figure 9, we plot the optimal prefix lengths and prefix server heights under this scenario, with $m = 4, l = 5, \gamma = 1$ and $lhc = \{1, 0.1\}$.




(a) optimal prefix server height (b) optimal prefix length

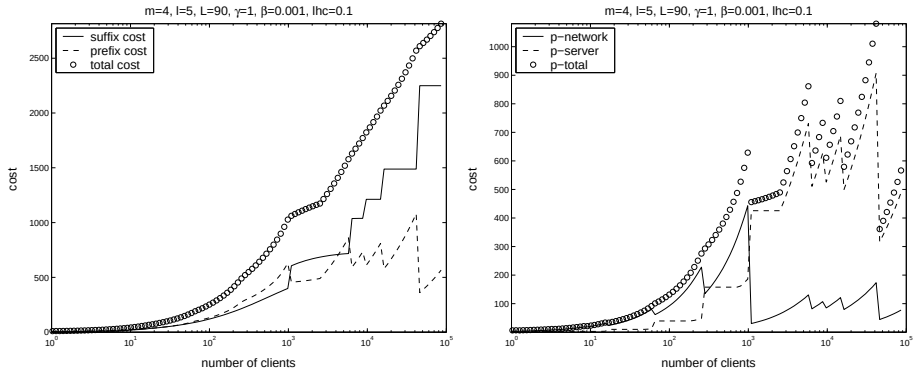
**Fig. 9.** Optimal prefix server height and optimal prefix length for $\gamma = 1$ and $lhc = \{1, 0.1\}$

Reducing the last-hop cost makes network transmission cheaper compared to server cost. We see that the prefix server heights are roughly the same while the prefix lengths decay faster than in the original setup ($lhc = 1$) to compensate for the relative increase in prefix server costs. This may seem counter intuitive since reducing the last-hop cost should make the prefix cost cheaper, especially if the prefix servers are at the leaves while the suffix server is at the root. However, we see in figure 10(b) that the *server* cost for the prefix $C_{server}^{prefix}$ now dominates the total prefix cost $C^{prefix}$, (in particular when the prefix servers are placed at the leaves, i.e. for $N > 10^3$) which was not the case for $lhc = 1$ (see figure 8(b)).

When we compare the suffix costs in figures 8 and 10, we note that reducing the last-hop cost reduces the suffix network cost by almost a factor of 4, which assures that the suffix system remains cost effective. This cost reduction is due to the fact that with $m = 4, l = 5$, there are 1024 links at the leaf level (last-hop) and only 340 links in the rest of the network. The network cost for the suffix system and the server cost for the prefix system are roughly in the same neighborhood. As a result, the server cost (i.e. the prefix server cost) is magnified in importance and the optimal policy is to compensate by shortening the prefix.

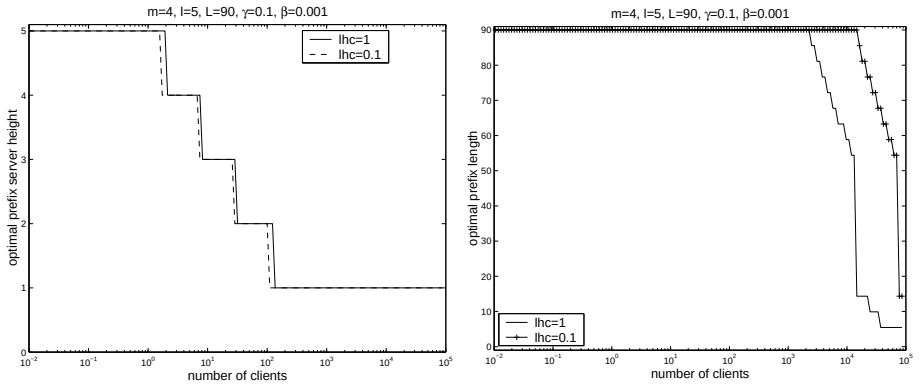
**Reducing the server costs relative to the network transmission cost** While the cost for servers is usually comparable in Europe and the US, the cost of network transmission is much *higher* in Europe than in the US. To account for the case where the network transmission cost relative to the server cost is higher, we set $\gamma$ to 0.1.

The results for $\gamma = 0.1$ (figure 11) indicate that reduced relative server cost allows to deploy more servers in order to reduce the impact of the expensive




(a) total system cost (b) prefix cost breakdown

**Fig. 10.** Breakdown of costs for $\gamma = 1$ and $lhc = 0.1$



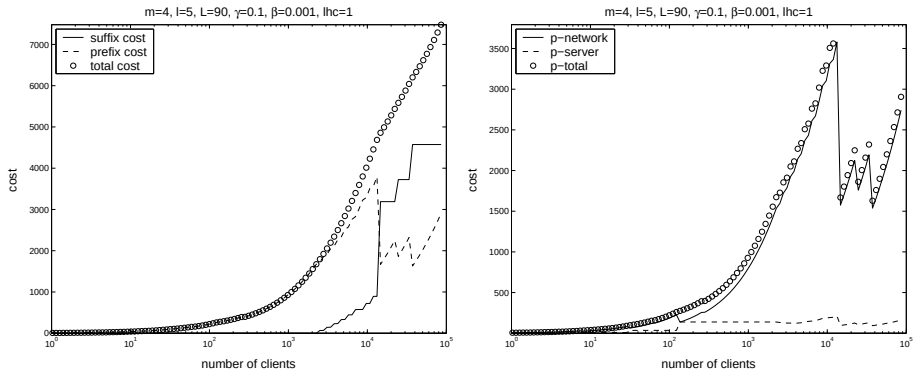

(a) optimal prefix server height (b) optimal prefix length

**Fig. 11.** Optimal prefix server height and optimal prefix length for $\gamma = 0.1$

network transmission cost. If we compare with $\gamma = 1$ (figure 9), we see that for $\gamma = 0.1$

- The optimal prefix (figure 11(b)) comprises the entire video for a much wider range of clients $N$. Only for a very high video popularity $N > 10^4$, the optimal prefix length decreases significantly.
- The prefix server height (figure 11(a)) drops faster to $h = 1$ since this helps to reduce the network costs and since the server costs, though they increase (the number of prefix server increases), have been discounted.

We also examine the case where $\gamma = 0.1$ and in addition the network cost for the last-hop is reduced to $lhc = 0.1$. We plot the optimal prefix lengths and prefix server positions in figure 11. Previously, we saw for $\gamma = 1, lhc = 0.1$ that although the reduced last-hop cost significantly reduced the cost of prefix service (and of suffix service), the high server costs associated with leaf servers limited the use of the prefix. Now the server cost has been discounted, we see that reducing the last-hop network cost will allow the prefix servers to deliver the entire video over a wider range of video popularities as compared to the $lhc = 1$ case (figure 11). This is interesting, as reducing the last-hop network cost *decreased* the prefix length in the $\gamma = 1$ case and is an example of the interplay between network and server costs. For $\gamma = 1, lhc = 0.1$ the prefix server cost dominates the overall prefix cost (see figure 10(b), while for $\gamma = 0.1, lhc = 0.1$ it is the prefix network cost that dominates the overall prefix cost (see figure 13(b)).

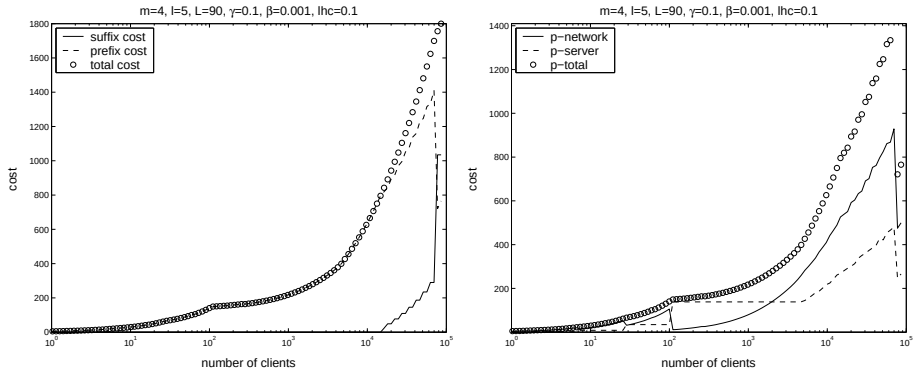



(a) total system cost (b) prefix cost breakdown

**Fig. 12.** Breakdown of costs for $\gamma = 0.1$ and $lhc = 1$

**Ignoring server costs** Previous studies of video distribution schemes [13] have often ignored the server cost and only considered the network cost. We can model this case if we choose $\gamma = 0$. When we ignore the server cost, placing the prefix servers at the leaves is always optimal since the prefix *network* cost is minimized in this case.

We plot the optimal prefix length for $\gamma = 0$ in figure 14(b). For both values of $lhc = \{1, 0.1\}$, the optimal prefix comprises the entire video for a large interval of the values of $N$. For large $N > 10^4$, the optimal prefix length becomes shorter as the centralized suffix system becomes more bandwidth efficient (despite the fact that the video must traverse 5 hops for the suffix as compared to 1 hop for the prefix) than the prefix transmission via controlled multicast.




(a) total system cost (b) prefix cost breakdown

**Fig. 13.** Breakdown of costs for $\gamma = 0.1$ and $lhc = 0.1$

If we compare the optimal values for the prefix length with the case of uniform link costs (i.e. $lhc = 1$) and large $N$ ($N > 10^4$) , we see that for $\gamma = 0$ the optimal values are only unnoticeably larger than for the case of $\gamma = 0.1$.

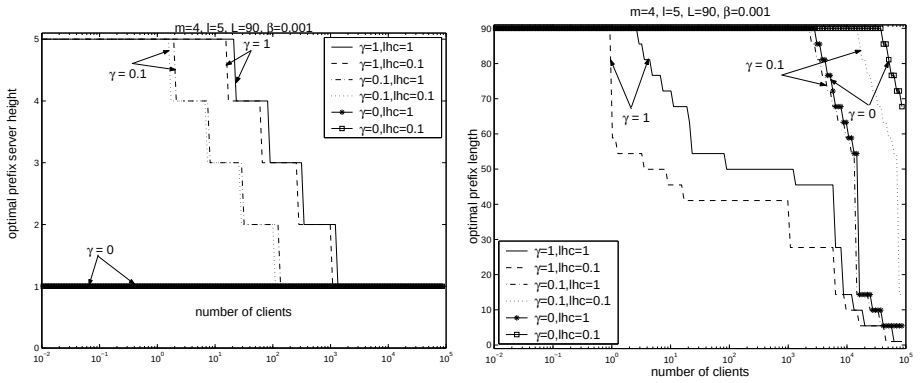
We plot the optimal prefix server height and optimal prefix lengths for all configurations covered so far in figure 14. We see how the system adapts the partitioning into prefix and suffix and the placement of the prefix servers as a function of the cost for the network and server resources. When the cost for the servers relative to the cost for network transmission is reduced ($\gamma < 1$) more prefix servers are deployed. For a given video popularity $N$, this happens in two ways

- The prefix servers are placed closer to the clients (see figure 14(a))
- The prefix is made longer (see figure 14(b))

In the case of $\gamma = 0$, the entire video is, over a wide range of the video popularity $N$, delivered by the prefix servers.

We conclude this subsection by showing in figure 15 the optimal total cost values under each scenario discussed. We see that

- for the topology $m = 4, l = 5$ chosen, the majority of the links are at the last-hop to the leaves. Therefore, for non-uniform link costs ($lhc = 0.1$) the cost of the overall system is considerably much smaller than for $lhc = 1$.
- There is surprisingly little difference in the cost of the overall system for $\gamma = 0.1$ and $\gamma = 0$ since in both cases it is the cost for the network transmission that dominates the total cost. For $\gamma = 0.1$, the impact of the prefix *server* cost is attenuated (for $1 < N < 100$) by using fewer prefix servers (see figure 14(a))




(a) optimal prefix server height

(b) optimal prefix length

**Fig. 14.** Optimal prefix server height and optimal prefix length for $\gamma = \{1, 0.1, 0\}$ and $lhc = \{1, 0.1\}$

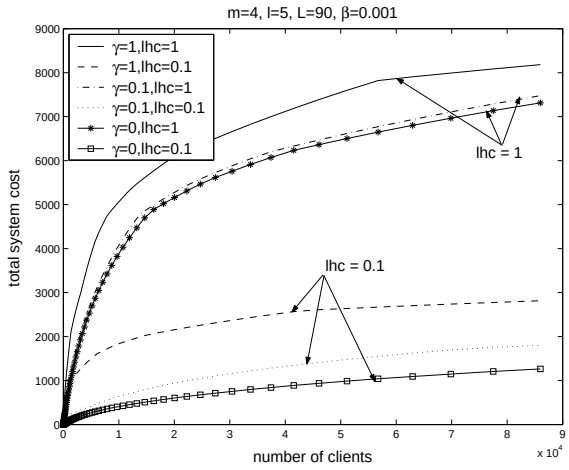



**Fig. 15.** Comparison of total system costs for $\gamma = \{1, 0.1, 0\}$ and $lhc = \{1, 0.1\}$

**Conclusions so far** We have seen how our distribution architecture gives us the cost-optimal configuration as a function of the video popularity $N$. For a non-popular video, the prefix server is placed at the root[6] and the length of the prefix comprises the whole duration of the video. As the video popularity $N$

[6] With the exception of $\gamma = 0$, where the prefix servers are always placed at height $h = 1$.

increases, the optimal prefix length becomes shorter and the optimal height for prefix servers becomes closer to the clients.

Using the suffix server at the root of the distribution tree becomes economically very interesting for very popular videos, where the prefix length becomes much shorter than the whole duration of the video. However, the optimal suffix length used is quite sensitive to changes in the popularity (see figure 14(b) for $10^4 \leq N \leq 10^5$). Therefore, to operate the distribution system in a cost-optimal fashion, one needs to periodically estimate the current popularity of a video and adapt, if necessary, the prefix length.

**Costs for non-optimal prefix server placement** For a large video distribution system serving many movies, it will be feasible to place prefix servers at every level of the distribution tree; however, for smaller systems, the prefix server locations will probably be fixed as there may not be prefix servers at every level of the distribution network. The placement of the prefix servers at an arbitrary level may also be impossible for other reasons. For instance, the facilities necessary for installing a server may not be available or accessible. Thereby, it is worth evaluating the cost performance of a video distribution system where the prefix servers are placed non-optimally. We will examine how the system adjusts the prefix length to find the most cost-efficient solution for the case where

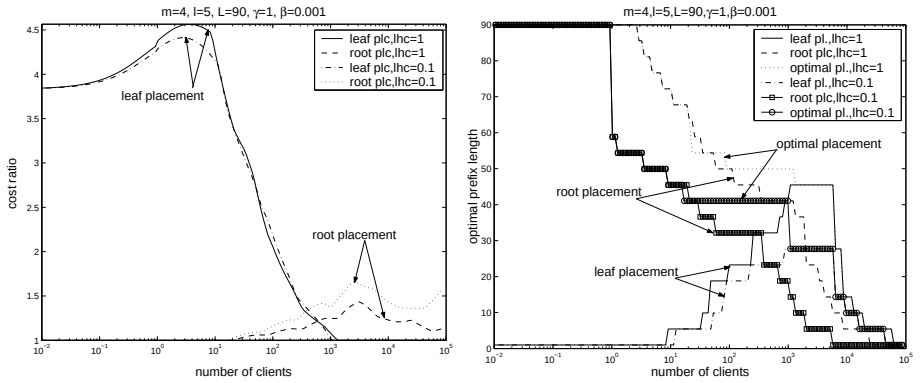
- The prefix server is placed at the root, which will be referred to as **root placement**
- The prefix servers are placed at the leaves (i.e. at height h = 1, one level above the clients), which will be referred to as **leaf placement**

We always assume that the cost for the server is taken into account (i.e. $\gamma \neq 0$) and consider the following scenarios

- Network is cheap, $\gamma = 1$ and $lhc = \{1, 0.1\}$.
- Network is expensive, $\gamma = 0.1$ and $lhc = \{1, 0.1\}$.

*Network is cheap* We first discuss the case with $lhc = 1$, where the per-link network costs are the same across the network. We know from figure 7(a) that for values of $N, N < 20$, the optimal placement of the prefix server is at the root. For increasing values of $N > 20$, the optimal placement finds the best trade-off between network and server cost by moving the prefix servers closer to clients and reducing the length of prefix. When we fix the placement of the prefix server at the root, the only degree of freedom left to minimize the cost is to adjust the prefix length. We see in figure 16(b) that the prefix length for the root placement case decreases more rapidly than when the prefix placement is chosen optimally.

In figure 16(a) we plot the ratio of the total system costs. For the case when the prefix server is always at the root as compared to when it is optimally placed, the total system cost increases by up to 60%: For very popular videos, placing the prefix server at the root results in a longer suffix in order to limit the high cost for the prefix distribution.




(a) Cost ratio for prefix server, $\gamma = 1$, $lhc = \{1, 0.1\}$

(b) Optimal prefix length, $\gamma = 1$, $lhc = \{1, 0.1\}$

**Fig. 16.** System cost ratio for non-optimal placement to optimal placement system cost and optimal prefix length for $\gamma = 1$ and $lhc = \{1, 0.1\}$.

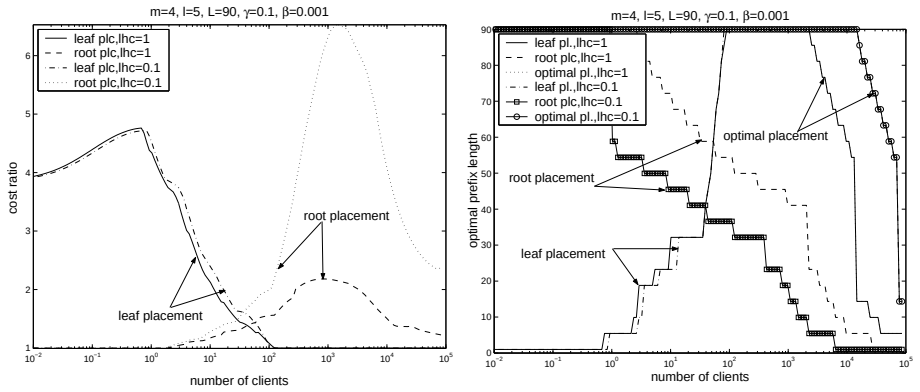
When the prefix servers are always at the leaves, the cost increase as compared to the optimal placement is much higher than for root placement and can be up to 450% since leaf placement is very expensive for non-popular videos ($N < 10^2$). This big difference in cost is due to the fact that keeping copies of a video in all the prefix servers placed at the leaves is very inefficient for videos that are not very popular, i.e. $N < 10^3$. When the prefix servers are placed at the leaves and the video popularity is low ($N < 10^1$), the system chooses the minimal possible prefix length of $D = 1$ to limit the overall cost of keeping the many copies of that prefix (see figure 16(b)). For larger values of $N$, the optimal prefix server placement is at the leaves, and so the performance of the non-optimal system is the same as the optimal one for $N > 10^3$.

When we compare the case where all links of the network have the same cost ($lhc = 1$) to the case where the last-hop links are less expensive for $lhc = 0.1$, we see little difference in the results. For $lhc = 0.1$, the worst case cost increase due to leaf placement is a little bit less than for $lhc = 1$. For the root placement, the fact that the last-hop links are less expensive ($lhc = 0.1$) increases the cost of root placement as compared to $lhc = 1$.

Overall, when the network is cheap and we must choose between root and leaf placement, root placement is preferable.

*Network is expensive* We examine now the opposite situation, where the network is expensive as compared to the cost for servers (see figure 17(a)).

When the network is expensive, the worst case cost performance of leaf placement deteriorates only slightly as compared to the case when the network is cheap, since the cost for the servers placed at the leaves dominates the total




(a) cost ratio for prefix server, $\gamma = 0.1$, $lhc = \{1, 0.1\}$

(b) Optimal prefix length, $\gamma = 0.1$, $lhc = \{1, 0.1\}$

**Fig. 17.** System cost ratio for non-optimal placement to optimal placement and optimal prefix length for $\gamma = 0.1$ and $lhc = \{1, 0.1\}$

system cost. For root placement, the worst case cost is significantly higher as in the case of $\gamma = 0.1$ , in particular for $lhc = 0.1$, since the network transmission has become more expensive, which is directly reflected in an increase of the total system cost. The optimal placement for $\gamma = 0.1$ moves the prefix servers for increasing $N$ rapidly towards the clients and chooses a prefix that comprises the entire video for all except the very popular ones $N > 10^4$ (see figure 14). Since the root placement can not move the prefix servers, it shortens the prefix length drastically with increasing $N$ to offset the cost-increase due to the prefix placement at the root (see figure 17(b)).

*Conclusion* The video delivery architecture has normally two degrees of freedom: the prefix length and the prefix server placement can be varied to determine the cost optimal solution. When we remove one degree of freedom and fix the placement of the prefix server, the total system cost can increase significantly, in particular for the case of leaf placement, where the server cost dominates as compared to the network cost.

### 2.5 Short videos

**Introduction** So far we have looked at videos of length $L = 90$ min, which corresponds to feature movies. Besides movies, there are news clips or clips for product promotion, which are much shorter in length, that can be distributed via a video distribution system. For these clips it is interesting to evaluate how efficiently the video distribution architecture supports the distribution of these

clips. We consider clips of $L = 7$ min[7]. As before, the network has an outdegree $m = 4$ and a number of levels $l = 5$. We vary the popularity of the video between $10^{-2} \leq N \leq 10^5$.

The optimal configuration is computed using the PS-model. The PS-model allows to determine which fraction of the video should be stored at the prefix servers and where to place the prefix servers to minimize the delivery cost.

In addition, we consider two more cases where we remove one degree of freedom:

- $D = L$, i.e. the prefix comprises the full video, which we refer to as **full video caching**. However, the height of the prefix servers is chosen such to minimize the overall system cost.
- The prefix server is fixed at the root, which we introduced in subsection 2.4 as **root placement**. However, the length of the prefix is chosen such to minimize the overall system cost.

**Results** Figure 18 shows the optimal prefix server placement and the optimal prefix length in the hierarchy as a function of the video popularity $N$. A comparison with the values obtained for long videos of $L = 90$ min (see figure 14) shows that the video distribution system behaves the same way, for both short and long videos:

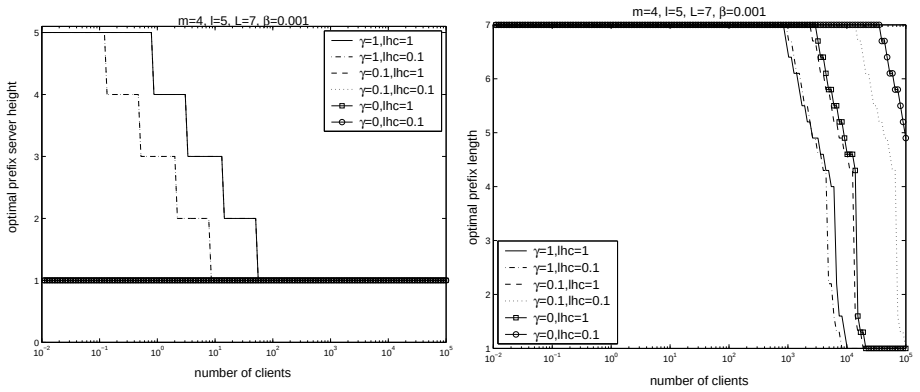
- With increasing video popularity $N$, the placement of the prefix servers moves closer to the clients
- For all but the most popular videos, the optimal prefix comprises the whole video.

As we can observe from figures 18(b) and 19(a), full video caching is the optimal solution, except for the most popular videos.

In Figure 19(b) we plot the ratio of the delivery cost of a video obtained in the case of root placement as compared to the delivery cost obtained when both, the prefix length and the prefix server placement are chosen optimally[8]. We see that there is an additional cost of fixing the prefix server at the root for a values of video popularity $N$ except very small ones, where placing the prefix server at the root is the optimal choice. The additional cost due to root placement is lowest when the network transmission is cheap ($\gamma = 1$) and highest when the relative cost for the prefix servers is low ($\gamma = 0.1$) **and** the transmission cost over the last hop is reduced ($lhc = 0.1$). When the network is expensive ($\gamma = 0.1$) the cost ratio is worst for the values of $N$ where the optimal prefix server placement puts the prefix servers at the leaves ($h = 1$) and chooses a prefix that comprises the entire video ($D = L$). The shape of the curves is similar to the ones observed for long videos (see figures 16(a) and 17(a)).

[7] We also looked at clips of 2 min length. The results we have obtained for $L = 2$ are very similar to the ones for $L = 7$ and are therefore not present here.

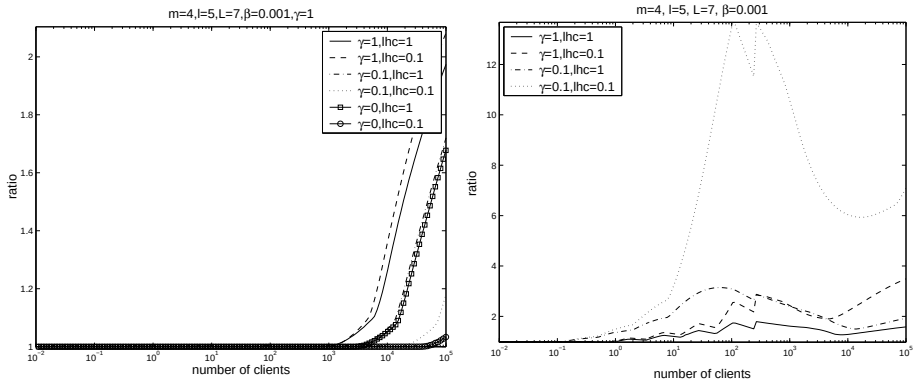
[8] We do not plot the cost ratio for $\gamma = 0$, which can reach a value up to 40 for small values of $N$.




(a) optimal prefix server height

(b) optimal prefix length

**Fig. 18.** Optimal prefix server height and prefix length for $\gamma = \{1, 0.1, 0\}$, $lhc = \{1, 0.1\}$, and $L = 7$ min.




(a) Cost ratio for full video caching

(b) Cost ratio for root placement

**Fig. 19.** Cost ratio for full video caching system and cost ratio of root placement for $\gamma = \{1, 0.1, 0\}$, $lhc = \{1, 0.1\}$, and $L = 7$ min.

## 2.6 Video distribution system for a set of videos

**Introduction** So far we have looked at a single video. We studied the case of how to determine the optimal prefix length and the optimal prefix placement as a function of the video popularity $N$. However, a complete video distribution system will offer to the clients a host of different videos $\mathcal{V} = \{1, \ldots, K\}$ to choose from.

We will now extend the PS-model to deal with the following scenarios:

- Provisioning.
  The PS-model is used to solve the provisioning problem for a given set of videos whose popularities are known: We just need to execute the model for each video separately to determine the optimal prefix length and the placement of the prefix servers.
- Video assignment problem for an existing configuration.
  Very often, the situation will be such that the video distribution system has been already deployed, i.e. the central suffix server and the prefix servers have been installed and changing the location (placement) or the capacities of prefix servers is not possible. Given that the placement and the capabilities of the prefix servers are *fixed*, we then want to determine the cost-optimal prefix length and prefix server placement for a set of videos and popularities.

For a set $\mathcal{V} = \{1, \ldots, K\}$ of $K$ videos, the PS-model computes separately the optimal system cost of each video $i \in \mathcal{V}$. The total system cost will be the sum of the system costs over all $K$ videos of the system. In the following, we will consider that the popularity $N_i = \frac{A}{i^\alpha}$ of video $i \in \mathcal{V}$ follows a Zipf distribution the popularity of the most popular video and $\alpha$ the slope of the popularity distribution; the bigger $\alpha$, steeper the slope, i.e. the more biased or skewed the popularity distribution. In figure 20 we plot the popularity for different values of $A$ and $\alpha$.

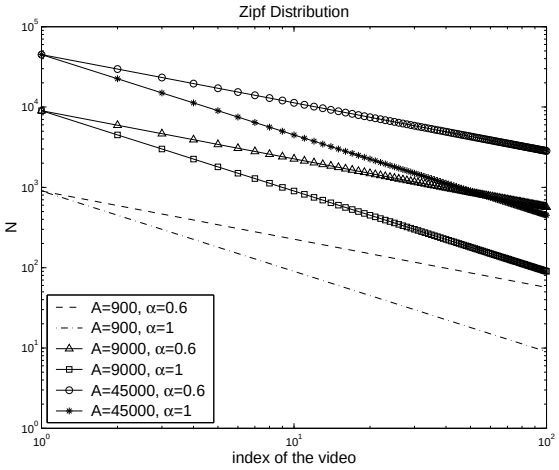



**Fig. 20.** The popularity $N_i$ of video $i$ as a function of the index $i$, according to a Zipf distribution, on a log-log scale.

**Provisioning of the prefix servers** The aim of provisioning is to compute the resources required in terms of video server storage and video server I/O

bandwidth for a given set of videos $\mathcal{V} = \{1, \ldots, K\}$ with popularities $N_i$, for $i \in \mathcal{V}$[9]. We will concentrate here on the provisioning of the prefix servers inside the network. However, the provisioning of the servers at the root can be done in a similar way.

We use equation 2 (subsection 2.3) to compute the optimal prefix length $D_i$, the optimal threshold $T_i$, and the optimal prefix server level $l_i$ for each video $i \in \mathcal{V}$. Let $\mathcal{L}(j)$ , $\{i \in \mathcal{V} \mid l_i = j\}$ denote the subset of videos whose prefix will be optimally served by the prefix servers of level $j$ and $I/O_i$ denote the amount of I/O bandwidth needed for each prefix server at level $j$ to serve the prefix of video $i \in \mathcal{V}$. The value of $I/O_i$ can be computed the same way as $C_{I/O}^{prefix}$ in subsection 2.3. With $\lambda_i = \frac{N_i}{L}$ and $i \in \mathcal{L}(j)$ we have

$$I/O_i = b\ \frac{\lambda_i}{m^j} \frac{2D_i + \lambda_i T_i^2/m^j}{2 + 2\lambda_i T_i/m^j}$$

At each level $j$, the total storage $PS_{st}(j)$ and I/O $PS_{I/O}(j)$ capacity of the prefix servers are computed as the sum of the prefix lengths and the amount of I/O bandwidth needed over all prefixes placed at that level. Hence, at level $j$, the resource requirements of the prefix servers are given by:

$$PS_{st}(j) = \sum_{i\ \in\ \mathcal{L}(j)} b \cdot 60\ D_i \qquad \forall j \in \{1, \ldots, l-1\}$$

$$PS_{I/O}(j) = \sum_{i\ \in\ \mathcal{L}(j)} I/O_i \quad \forall j \in \{1, \ldots, l-1\}$$

Since we assume a homogeneous client population, the load will be uniformly distributed over the prefix servers at a particular level. Therefore, all prefix servers at a particular level $j$ will have the same capabilities.

**Assignment of a videos to prefix servers with limited storage and I/O capabilities** We now consider the case that the prefix servers have been installed and that it is not possible to add new prefix servers or to modify their placement in the distribution hierarchy. The values of $PS_{st}(j)$ and $PS_{I/O}(j)$ are known $\forall j \in \{1, \ldots, l-1\}$. We will refer to them as **prefix server constraints**. However, we allow for modifications to the servers installed at the *root*. This means that there are no constraints on the resources of the central suffix server or the prefix server at the root ($l = 0$).

To solve the prefix assignment problem for a set of videos $\mathcal{V}$, we need to find the placement that satisfies the prefix server constraints of the system and minimizes the total system cost.

[9] For sake of simplicity we assume that all videos have the same length $L$.

We formulate the assignment problem for a set $\mathcal{V}$ of videos as follows:

$$\begin{cases} \min\limits_{\theta_{ij}} (\sum\limits_{j=0}^{l-1} \sum\limits_{i \in \mathcal{V}} C_{PS}^{system}(i,j) \times \theta_{ij}) & \\ s.t. \quad \sum\limits_{i \in \mathcal{V}} D_{ij} \times \theta_{ij} \leq PS_{st}(j) & 1 \leq j \leq l-1 \\ \quad \sum\limits_{i \in \mathcal{V}} I/O_{ij} \times \theta_{ij} \leq PS_{I/O}(j) & 1 \leq j \leq l-1 \\ \quad \sum\limits_{j=0}^{l-1} \theta_{ij} = 1, & \forall\ i \\ \quad \theta_{ij} \in \{0,1\}, & \forall\ i,j \end{cases}$$

where (i) $C_{PS}^{system}(i,j)$ is the lowest total system cost achievable when placing video $i$ at level $j$, (ii) $D_{ij}$ is the corresponding optimal prefix length when placing video $i$ at level $j$, (iii) $I/O_{ij}$ is the amount of I/O bandwidth needed when placing video $i$ at level $j$, and (iv) $\theta_{ij}$ is a binary variable. $\theta_{ij}$ is equal to 1 if the prefix of the video $i$ is placed at level $j$ and 0 otherwise. $\sum\limits_{j=0}^{l-1} \theta_{ij} = 1$ indicates that no video prefix can be stored at more than one level in the hierarchy.

Both, the objective function and the constraints are linear functions of the binary variables $\theta_{ij}$. This optimization problem can be resolved using dynamic programming. We use the *XPress-MP* package [15] to solve the assignment problem.

*Results* Moving a video prefix from an optimal level to a non-optimal one in order to satisfy the constraints increases the delivery cost of the video and consequently the overall system cost. It is interesting to evaluate how the prefix server constraints will impact the overall system cost of the video distribution system. To this purpose, we compute the overall system cost $C_{opt}^{vds}$ *without* any constraints and the overall system cost $C_{constr}^{vds}$ with prefix server constraints, which are defined as

$$C_{opt}^{vds} = \sum_{i \in \mathcal{V}} C_{PS}^{system}(i)$$

$$C_{constr}^{vds} = \sum_{j=0}^{l-1} \sum_{i \in \mathcal{V}} C_{PS}^{system}(i,j) \times \theta_{ij}$$

We use the cost ratio $C_{constr}^{vds}/C_{opt}^{vds}$ to evaluate how well the video distribution architecture can adapt to changes in the video popularity and the number of videos. We plot in figure 21 the cost ratio for $\alpha = \{0.2, 0.6, 1\}$ and different numbers of videos $K = \{100, 120, 150\}$. We set the normalization constant to $A = 9000$. The prefix server constraints were computed for the case of $K = 100$

videos with $\alpha = 0.6$ for the Zipf distribution. This initial distribution ($\alpha = 0.6$) is skewed or biased enough to fit well with small systems.

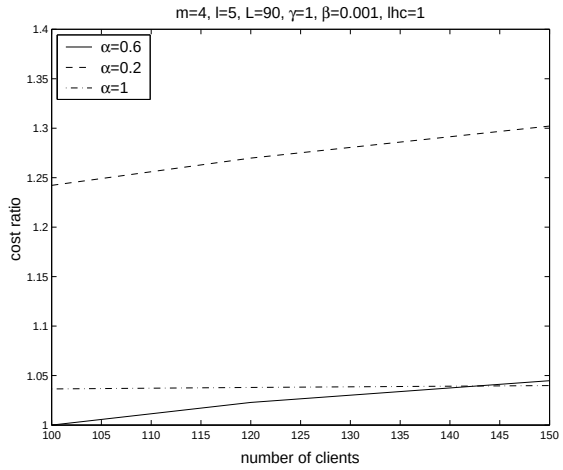



**Fig. 21.** Cost ratio $C^{vds}_{constr}/C^{vds}_{opt}$.

Figure 21 shows the following interesting features:

- The cost ratio increases sub-linearly with the number of videos.
- The increase of the system cost strongly depends on the parameter $\alpha$.

These results fit well with the intuition. The larger the popularity $N$ of a video, the closer is the placement of the prefix to the clients. If there is no more place for the popular videos at the higher levels of the prefix hierarchy, video must be moved closer to the root.

For a given Zipf distribution, increasing the number of videos adds videos at the *tail* of the distribution. Those additional videos are the least popular ones, and must be moved near the root to free resources for the more popular ones at the higher levels. The placement of the least popular videos close to the root will use up very few of the constraint resources, which explains the small change in the cost ratio with increasing number of videos. At some point, all the additional videos will have such a low popularity that their prefixes will all be placed at the root and the cost ratio will no longer change with increasing number of videos.

For $\alpha = 1$, the popularity distribution is very skewed, and increasing the number of videos $K$ in the system has no impact on the cost ratio since those additional videos are all optimally placed at the root. If we compare $\alpha = 0.2$ and $\alpha = 0.6$, we find that, as the number of videos $K$ increases, the cost ratio increases more rapidly for $\alpha = 0.2$ than for $\alpha = 0.6$. The reason is that, for $\alpha = 0.2$, the videos added to the system are relatively more popular than those for $\alpha = 0.6$, and as a consequence, moving them closer to (or placing them at) the root to satisfy the constraints comes out more costly.

We also evaluated other scenarios such that, $\gamma = 0.1$, or $\gamma = 0$ and for a reduced cost of the last hop link $lhc = 0.1$. The results obtained were similar and are therefore not presented here.

*Conclusion* The PS-model adapts itself very well in case of limited prefix server resources. Non-popular videos are moved close to the root to free a place for the more popular ones. For $\alpha = 0.6$ and $\alpha = 1$, the increase in the overall system cost for increasing number of videos is very low, less than 5%. Even when the popularity distribution differs a lot from the one used when determining the prefix server resources (as is the case for $\alpha = 0.2$) the extra cost incurred will be small, 25%– 30%.

## 2.7 Conclusion and Outlook

**Summary** We have presented a scalable video distribution architecture that combines open-loop and closed-loop schemes and assure a zero start-up delay. Under this architecture, each video is split into two parts, the prefix and the suffix. The prefix is transmitted via controlled multicast (closed-loop scheme) while the suffix is transmitted via tailored periodic broadcast (open-loop scheme). The architecture is very cost-effective since the cost for the prefix transmission increases only with the square root of the number of clients and the suffix distribution cost is, at high request rates, independent of the number of clients and simply a function of the number of segments the suffix is decomposed into.

Another advantage is that our architecture is highly scalable in terms of serving incoming client requests. A client who wants to receive a video contacts his closest prefix server. The central server only needs to know if there is at least one client connected, which the central server can learn by communicating with the prefix servers. This interaction between the clients and the servers avoids having a bottleneck due to handling all the requests by a single server.

We have developed an analytical cost model for that architecture that we called PS-model. In the cost model, we include not only the network bandwidth cost, but also the costs for the server I/O bandwidth and server storage. Using the PS-model we can determine the

- Bandwidth and streaming costs for prefix and suffix transmissions
- Optimal prefix length
- Optimal position of the prefix servers.

The PS-model makes the trade-off between the server and network bandwidth costs to determine the cost-optimal prefix length and the optimal prefix server placement.

As key results, we found that

- The cost efficiency increases with increasing video popularity $N$. For a 10-fold increase in $N$ from 9,000 to 90,000 clients the total system cost only doubles.

- A popular video is replicated at many prefix servers that are placed close to the clients.
- A non-popular video is placed at the root or very close to the root to reduce the server storage cost incurred for replicating the prefix across many prefix servers.
- The central suffix server is highly efficient to serve very popular videos for which the optimal the suffix comprises the major portion of the video.

The PS-model has two degrees of freedom: It can adjust the length of the prefix as well as the placement of the prefix server in the hierarchy so as to divide efficiently the workload between the suffix server and the prefix servers. Thus, if we remove one degree of freedom, for instance, we fix the placement of the prefix server at either the root or the leaves, the only degree of freedom left is to adjust the length of the prefix. It is worth to discuss the cost for a non-optimal placement of the prefix servers since for small systems, it might not possible to have prefix servers at any level in the network. By comparing between leaf placement and root placement for the prefix servers, we found that

- The system cost can increase significantly as compared to the optimal system cost. This increase is up to 450% for leaf placement.
- Root placement outperforms leaf placement in the case where the network cost is cheap as compared to the server cost. In this case, the cost increase is relatively small, up to 60%.
- Root placement becomes very costly when the network cost is high relative to the server cost, and the increase in the system cost can exceed 600% in case of a reduced last-hop cost.

We also evaluated the PS-model for the case of short videos such as video clips or clips for product promotions. In fact, even for short videos, it is always cost-optimal to divide the video into prefix and suffix in the case very popular clips.

Moreover, we extended the PS-model to looked at the following scenarios:

- Provisioning:
  We showed how the PS-model can be used to compute the cost-optimal for a system with a set $\mathcal{V}$ of videos. Given the popularity of each video, the PS-model determines the system resources required in terms of network bandwidth, server I/O bandwidth, and server storage to optimize the total system cost.
- Assignment of prefixes into prefix servers:
  We studied how the PS-model adapts the prefix length and the prefix placement for a set $\mathcal{V}$ of videos when the amount of resources for the prefix servers is given, which is for instance the case when a new set of videos must be optimally placed. The cost increase due to predefined capabilities of the prefix servers is very low over a wide region of our model, less than 5%. Even when the popularity distribution of the videos differs a lot from the initial distribution used to determine the prefix server resources, the increase in the system cost remains low, 25-30%.

The PS-model has allowed us to study different scenarios. However, several extensions to the PS-model are possible that allow to make the overall model more realistic. The current model assumes that client requests for a single video are homogeneously distributed among all clients. A possible extension would consider the case where a particular video is more popular with a sub-group of the clients which would lead to heterogeneous request patterns. The current video distribution network has a very regular structure with all clients being at the same distance from the root and the distribution tree being very regular, while a real distribution network most likely has not such a regular structure. We intend to extend our model in the future and evaluate the impact of these extensions. These extensions will clearly change the absolute cost values. However, we do not expect that they will change the broad conclusions that we could draw using the PS-model.

We would also like to evaluate different architectural choices. Today, digital VCRs with at least one hundred Gigabyte of local storage are commercially available [40]. Given local storage, one can proactively download the prefixes of the most popular videos directly into the VCR. We intend to extend the PS-model to evaluate the overall cost reduction due to the use of local storage in the VCR. Another very attractive architectural option is a satellite distribution of the suffix of the videos.

**Outlook** In the system model we have analyzed, we had assumed that both, the suffix server and the prefix servers, are *dedicated*. These assumptions hold true for a "commercial" CDN. In recent years, a new paradigm called peer-to-peer [34] emerged where a particular machine can assume both roles, i.e. be client and server at the same time. Popular peer-to-peer systems such as Gnutella, Napster, or KaZaa have been used by Millions of users to share digital content such as MP3 files. P2P architectures are very interesting for scalable content distribution. They offload all or at least the majority of the work for storage and distribution onto end-systems that are *not dedicated* to the purpose of content distribution and therefore, significantly reduce capital expenditures. P2P architectures are also inherently *self-scaling*: In case of a sudden increase in the number of requests, the number of peers that have received the content will also increase, which in turn will increase the total capacity of the P2P system to serve new clients. P2P systems are therefore much more suitable than centralized server-based systems to handle "flash crowds".

The research community has made numerous proposals on how to use the P2P paradigm to provide overlay multicast distribution trees [14, 7] and to perform scalable video distribution. The $P^2$Cast scheme [22] partitions the video into prefix and suffix as we do in our model and proposes to distribute the suffix by a central server via application layer multicast while the prefix is delivered via unicast by another client that has already received the prefix previously and is currently viewing the suffix.

Constructing application layer multicast trees for video distribution is very challenging as clients that are part of the tree may leave at any time, which may

disrupt the video reception of the clients that are downstream while the tree is re-built. Various proposals such as Coopnet [36] and SplitStream [12] propose to encode the video signal using multiple description encoding techniques [20] where the video signal is encoded as $N$ independent descriptions or sub-streams that are each transmitted on a separate multicast tree. In this case, a receiver will be able to play out the video, albeit with reduced resolution, if it receives only a subset of the descriptions. When using such an approach, it is important that the multicast trees for transmitting the different descriptions are constructed in such a way that the failure of a node will not affect different receivers for each of the descriptions. This is assumed in SplitStream by constructing the trees in such a way that an interior node in one tree is a leaf node in all the other trees.

A hybrid architecture that combines a CDN (with its servers installed at certain places) and peer-to-peer based streaming was proposed in [47]. Such an architecture allows to considerably reduce the amount of CDN resources required since peers that have received a video will in turn serve other clients, which reduces that load on the CDN server.

# 3 Scheduling Objects for Broadcast Systems

## 3.1 Introduction

Technological progress in high speed communications, embedded systems and consumer electronics has led to the availability of several *thin*, personalized user terminals capable to store, process, transmit and receive information, such as mobile phones, personal digital assistants (PDA), palmtops, tablet PCs, etc. Their powerful characteristics enable a wide range of services and applications, which deliver information to users efficiently.

Broadcast systems constitute a popular communication infrastructure to deliver services to thin, personalized *client* devices. Broadcast systems are capable to deliver multimedia services to a wide client population, because they are scalable in terms of user population and end-user bandwidth; furthermore, they accommodate heterogeneous user technologies, including mobile and wireless [4]. There exist several architectures for broadcast systems differing in the data delivery methods and mechanisms, such as the push or pull data transmission model, periodic or aperiodic transmission, etc. Each architecture provides advantages for specific environments and/or set of services.

In this work, we consider a popular and powerful broadcast system, suitable for satellite systems: an asymmetric, push-based system with receive-only i clients (i.e. without uplink bandwidth), who are mobile, or in general, connected occasionally. Typical broadcast systems in this category include the deployed systems Pointcast [28], Traffic Information Systems [38], Stock, Weather and News dissemination systems, DirecPc by Hughes [16] and Internet-over-Satellite (IoS) by Intracom [29]. In analogy to a web-type environment, we assume that application information is composed of logical objects (e.g., pages, pictures, text, segments, etc.), where each object is characterized by a different "popularity".

An important technical issue in such broadcast systems is scheduling, i.e. the order in which data objects are transmitted. The server transmits objects, so that their mean aggregate reception delay is minimized for all users. This criterion, the minimized mean aggregate reception delay, is important not only for performance but for energy consumption as well: minimized delay implies that client devices are switched on for a minimized time, and thus, energy consumption of users is minimized.

In push-based broadcast systems, servers broadcast objects in periodic cycles consisting of $T$ time units. Several algorithms have been used to construct the exact transmission schedule in a cycle, given as optimization criterion the minimization of the mean aggregate delay to start receiving an object, also called the access time (the results presented in the following, also apply to non-cyclic schedules, effectively when $T \to \infty$). The schedule of a cycle (period) includes multiple transmissions of each object, depending on its popularity [42]. It is known that, the optimal mean access time is achieved when all appearances of an object are equally distanced within the cycle [30].

Existing analyses of broadcast systems consider *memory-less* clients, i.e. clients which are not equipped with any storage. However, technological ad-

vances have led to the deployment of clients with memory today, and actually, memory sizes in the client devices are continuously increasing; in the following, we call this memory a *cache*, because it is not used for long-term storage. Existence of a cache at a client changes the model of a broadcast system, because a caching client is able to start collecting an object even if he/she turns on their device during the transmission of the desired object. This can be achieved provided that, each object transmission is done using packets that have headers with the appropriate packet information. This provision is the state of the art though, because often, transmission is done with fixed size cells, also called radio units, where each cell has an appropriate header, e.g. in GPRS [23], 802.11, etc.

In this chapter subsection, we present three main contributions. First, we provide a simple proof for the need of periodicity (equal distance in transmission) of popular objects in a cycle; the existent proof [30] uses arguments which are very complex. Second, in contrast to existing results, we consider the scheduling problem for caching clients. As we show, the existence of cache not only reduces the reception time of an object, but leads to an optimal broadcast schedule that is different from the optimal schedule for cache-less (traditional) clients. In our work, we calculate the reduced reception delay and we derive the i property that the optimal schedule for caching clients is based on, which is different from the one for cache-less clients. We also describe the scheduler that achieves the optimal schedule. For our work, we use as optimization parameter the mean aggregate reception delay, i.e. the sum of the access delay and the actual object reception time. In prior analyses [42],[46], where models similar to teletext were used, caching clients were not considered, because it was assumed that access time is significantly larger than actual object reception delay; however, in modern systems, this assumption does not hold because it is often necessary to transmit large objects with high popularity. Importantly, our optimization parameter reflects power consumption more realistically than existing models and calculations. This occurs because objects have variable sizes and thus, reception delay is not equal for all objects; however, prior work does not include this extra delay (and corresponding power consumption), because they consider environments with cache-less clients, where the only variable time is the access time.

Our third contribution refers to the analysis of pre-emptive scheduling, among other heuristics, for broadcast systems with or without caching clients. Since perfect periodicity of the broadcasting of all objects within a cycle is an NP-hard problem [9], we prove that the mean aggregate tuning delay of an object in a broadcast schedule may be further reduced, if we allow interruption (pre-emption) of an object's transmission in order to transmit on schedule another more popular one. This is contrary to the usual practice to transmit objects non-preemptively (without interruption). We deduce the conditions under which pre-emption is advantageous and we show that switching the transmission order of two consecutive objects is beneficial in some cases. Finally, we prove that interleaving transmission of two consecutive objects is not advantageous in any

case; such interleaving is tempting, considering object packetization and the popularity of interleaving in ATM networks.

The paper is organized as follows. Subsection 3.2 presents an overview of data delivery architectures for broadcast systems and introduces the scheduling problem. Subsection 3.3 introduces the model of the broadcast system, which we analyze, and the notation used in the paper. Subsection 3.4 presents our simple proof of perfect periodicity. Subsection 3.5 presents our analysis for caching clients, including the calculation of aggregate reception delay and the scheduler for achieving optimal schedules in the new model. Finally, Subsection 3.6 describes the conditions under which pre-emptive scheduling provides improved results.

### 3.2 Data Delivery Architectures

Broadcast systems can be classified, in general, using 3 parameters:

- the data request model (push vs. pull);
- the timing properties of their scheduling scheme (periodic vs. aperiodic);
- the connection model between server(s) and client(s) (unicast vs. 1-to-N).

Every broadcast system is characterized by a specific choice for each of these parameters, leading to 8 possible system configurations. In the following, we elaborate on these three parameters.

A broadcast system is characterized by its data request model, where either the client requests (pulls) data from the server by posting a request, or the server sends (pushes) information to client(s) without explicit requests from the clients. In the pull model, the client posts an explicit request to the server, which, in turn, responds to the client with the requested data; i.e. the client initiates the data transfer. In contrast, in a push system, servers transmit data to clients without a prior request, using some schedule, i.e. the data transfer is initiated by the server itself. The push system is more appropriate for satellite broadcasting to mobile clients, because it eliminates the need for clients to transmit, which is power consuming, especially for the case of GEO satellites.

Data delivery can be periodic or aperiodic in a broadcast system. In periodic systems, data are transmitted according to a predefined schedule (off-line algorithm, similarly to the classic TDMA systems), while in aperiodic systems, data delivery is event-driven, i.e. a data transfer is performed when an event occurs (on-line algorithm, e.g., a request in a pull system or a server decision in a push system). Periodic systems with very long periods can be considered as aperiodic, although they are constructed using an off-line algorithm.

Furthermore, broadcast systems may use different connection models: data transfers may occur in a unicast fashion or in a multicast (broadcast) fashion. In unicast systems, data transfers between servers and clients occur over a one-to-one connection, where no other client can receive the transmitted data. In contrast, in 1-to-N systems, the delivered data are transferred to a set of N clients, which constitute a group. If N is the complete population, then the

system follows a broadcast delivery method; however, the broadcast method is typically used in environments where N is unknown.

Several known systems can be classified using the above scheme. Pull-based systems are used, in general, for on-demand services, e.g. Video-on-Demand (VoD), news-on-demand, etc., where clients make requests for a specific service. Actually, these services are typically pull-based and aperiodic, since requests typically arrive at random time instances. Depending on the specific network configuration, the system can be based either on unicast or multicast (1-to-N) connections (a multicast connection is considered as broadcast, when N is the total population). Push-based systems include the characteristic examples of news services and newsgroup services as well as some forms of VoD, where clients have subscribed to receive specific information. Such systems can be either periodic (sending specific information at designated time intervals) or aperiodic (sending update information only, or information at random intervals).

We focus on push-based, periodic, 1-to-N broadcast systems. These systems are popular in environments where N is unknown. We focus on periodic systems (off-line scheduling algorithms) but our results also apply to effectively non-periodic systems with very large periods. We focus on push-based, because we consider environments where data transmission from clients to servers is expensive and power hungry, e.g. satellites, and client-to-server communication occurs off-line, probably using a less expensive medium, such as a telephone connection. Scheduling in broadcast systems has received a lot of attention.

In pull-based systems, scheduling is necessary to specify the transmission sequence of objects (to choose the object to transmit next). Several scheduling algorithms have been developed for servers [5]: FCFS, Most Requested First (MRF), Most Requested First Lowest (MRFL), Longest Wait First (LWF) and RxW.

Several scheduling algorithms have been developed for push-based systems as well [4, 42, 41]. The criterion is to minimize average access time or tuning time.

One method to minimize access or tuning time, and thus power consumption, is to use indexing methods. In such environments, all objects are identified with a unique key and the system transmits indexing information along with the data; the index is a sequence of pairs of the form (key, location), where the key identifies an object and the location identifies the position of the object in a broadcast cycle. In this fashion, a client can tune in, find the appropriate index entry for the object required and then, it can tune in at the appropriate time, in order to receive the desired object. Several indexing methods have been developed, which differ according to the indexing method or the amount and transmission method of the indexing information. Methods such as (1,m)-Indexing, Distributed Indexing and Flexible Indexing [27] transmit index information in various ways (multiple transmissions of the complete index, distributed transmission of index portions, etc.), while hash-based techniques substitute the indexing information with hashing information that allows the client to calculate the appropriate position of an object in the broadcast cycle.

### 3.3 Model

We assume a **server** $S$ that broadcasts information **objects** to a set of clients (users), who have subscribed for reception of specific objects off-line. We consider a Web-like environment, where the server transmits objects from a set $O = \{O_1, O_2, \ldots, O_N\}$ and each object $O_i$ is characterized by a **popularity** $p_i$, which is a probability that quantifies the number of clients waiting to receive a specific object. Transmission is performed using fixed size cells, also called radio units. Objects have arbitrary length and $l_i$ denotes the length of $O_i$, measured in radio units.

Data transmission is performed in periodic cycles $T$. In every cycle $T$, all objects are transmitted and each object may appear more than once in $T$, depending on its popularity and its length. An appearance of an object in the broadcast cycle is denoted as an **instance** of the object. The **spacing** $s_{ij}$ between two consecutive instances of an object $O_i$ is the time between the beginning of the $j$-th instance and the beginning of the $(j+1)$-th instance.

Clients in the model are devices which are switched on arbitrarily in time. When a client is switched on, it remains on until it receives the desired object(s), and then it is turned off. The *switch-on* activity of a client can be modeled as posting a request for the object, which the client is waiting for; so, in the following, we will refer to the client switch-on as a *request*. We assume that, during a broadcast cycle $T$, client requests for an object $O_i$ appear uniformly distributed.

We consider two different types of clients in the system: *caching* clients (equipped with cache) and *cache-less* ones. Cache-less clients need to receive an object from beginning to end, in order to consume it, because they do not have any storage capability. In contrast, caching clients have storage capability and can store partial object information; thus, they are able to store the last part of an object first and then wait for the reception of its beginning later.

We define as **tuning time** for an object, the **access time** of the object (the time until the beginning of reception of the object) plus the actual **reception time** of the object. Thus, we define *waiting time* differently from others, who consider the waiting time equal to the access time. This definition does not affect the construction of the optimal broadcasting schedule for cache-less clients, which are considered in prior work. In our work, we use as optimization parameter the *mean aggregate tuning time*, which is defined as the average of the mean tuning times of all users, i.e. $\sum p_i D_i$, where $i = 1, \ldots, M$, $p_i$ is the *popularity* of object $O_i$ (i.e., the fraction of the user population waiting for $O_i$ at a given time) and $D_i$ is the mean tuning time.

### 3.4 Periodic Object Transmission in a Cycle

Using the system model with cache-less clients, it has been shown that, for optimal broadcast scheduling (i.e., for minimization of the mean aggregate access delay), all instances of an object should be equally spaced within a cycle $T$ [30].

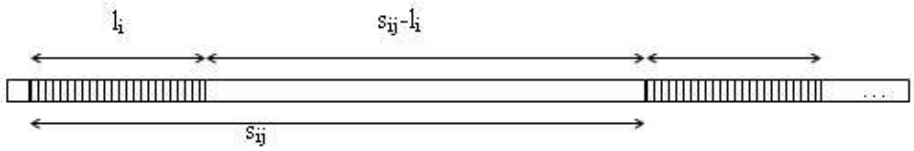



**Fig. 22.** Calculating the mean tuning delay of object $O_i$

In the following lemma, we provide a simple proof of this requirement, in contrast to [30], where the proof is quite obscure. Furthermore, we provide a proof that the same requirement exists for optimal scheduling with caching clients who can start collecting radio units of the desired item as soon as they appear (they start storing parts of the desired object $O_i$, even if they are switched on during transmission of $O_i$).

**Lemma 1.** *The spacing $s_i$ between any two consecutive instances of the same object $O_i$ should be equal in a transmission cycle $T$.*

*Proof.* We consider two different cases, depending on whether clients have caches or not.

*Cache-less clients*

Assume that object $O_i$ is been broadcasted $f_i$ times in a broadcast cycle $T$. The instances of $O_i$ are at spacings $s_{i1}, s_{i2}, \ldots, s_{if_i}$, where $\sum s_i = T$. If there is **one** request for $O_i$ during $T$ then there is a probability that it will appear during the spacing $s_{i1}$; this probability is $s_{i1}/T$, and the mean delay to receive all of $O_i$ is $[l_i + (s_{i1}/2)]$. The same holds true for every spacing $s_{ij}$. Therefore, the average delay $D_i$ to receive object $O_i$ during a broadcast cycle $T$ is:

$$\begin{aligned} D_i &= (1/T)\{s_{i1}[l_i + (s_{i1}/2)] + s_{i2}[l_i + (s_{i2}/2)] + \cdots + s_{if_i}[l_i + (s_{if_i}/2)]\} = \\ &= (1/T)\{s_{i1}^2/2 + s_{i2}^2/2 + \cdots + s_{if_i}^2/2 + l_i(s_{i1} + s_{i2} + \cdots + s_{if_i})\} = \\ &= (1/T)\{s_{i1}^2/2 + s_{i2}^2/2 + \cdots + s_{if_i}^2/2 + l_i T\} \end{aligned}$$

It is well known that the sum of squares of numbers with a constant sum is minimized when the numbers are equal. Therefore: $D_i = \min \text{for } s_{i1} = s_{i2} = \cdots = s_{if_i} = T/f_i$, and so:

$$D_{i\min} = (f_i/2T)(T/f_i)^2 + l_i = (s_i/2) + l_i$$

*Caching clients*

Under the same assumptions as above (for cache-less clients), the average delay

to receive object $O_i$ in a broadcast cycle is:

$$\begin{aligned}
D\text{cache}_i &= (1/T)\Big\{l_i s_{i1} + (s_{i1} - l_i)[(s_{i1} - l_i)/2 + l_i] \\
&\quad + l_i s_{i2} + (s_{i2} - l_i)[(s_{i2} - l_i)/2 + l_i] \\
&\quad \cdots \\
&\quad + l_i s_{if_i} + (s_{if_i} - l_i)[(s_{if_i} - l_i)/2 + l_i] \\
&= (1/T)\Big\{(l_i \sum s_{ij}) + (s_{i1}^2/2) + (s_{i2}^2/2) + \cdots + (s_{if_i}^2/2) \\
&\quad - l_i(\sum s_{ij}) + (l_i^2/2) f_i + l_i(\sum s_{ij}) - l_i^2 f_i\Big\} \\
&= (1/T)\Big\{s_{i1}^2/2 + s_{i2}^2/2 + \cdots + s_{if_i}^2/2 + l_i(\sum s_{ij}) - (l_i^2/2) f_i\Big\} \\
&= (1/T)\Big\{s_{i1}^2/2 + s_{i2}^2/2 + \cdots + s_{if_i}^2/2 + l_i T - (l_i^2/2) f_i\Big\}
\end{aligned}$$

This expression is minimized when $s_{i1} = s_{i2} = \cdots = s_{if_i} = s_i = T/f_i$, and so:

$$\begin{aligned}
D\text{cache}_{i\,\min} &= f_i (T/f_i)^2/2T + l_i - (l_i^2/2)(f_i/T) = (s_i/2) + l_i - (l_i^2 f_i/2T) \\
&= D_{i\,\min} - (l_i^2 f_i/2T) = D_{i\,\min} - (l_i^2/2s_i)
\end{aligned}$$

The last expression shows that, local caching clients receive the desired object $O_i$ with reduced (shorter) delay. The scheduling algorithm presented in [42], which requires that $s_i^2 p_i/l_i =$ constant, does not take into account local memory (cache). If one wants to take advantage of a client's cache, one must modify the condition, on which the scheduling algorithm is based, accordingly. As we prove in the following subsection, the optimal broadcast schedule for caching clients requires that $s_i^2 p_i/l_i + l_i p_i =$ constant. The difference between the two conditions becomes significant, if there exist lengthy objects with high popularities in the system.

### 3.5 Broadcast System with Caching Clients

**Theorem 1.** *The optimum broadcast schedule in a system with caching clients requires that*

$$s_i^2 p_i/l_i + l_i p_i = \text{constant}$$

*Proof.* Our proof follows the lines of the proof for cache-less clients [42] Assume that the following holds for all objects scheduled in a cycle $T$: the spacing $s_i$ between all pairs of consecutive transmissions of the same object $O_i$ within $T$ is equal (as we will see this is not always possible, but this is approximately true for large broadcast cycles). Then, the average delay to receive object $O_i$ is:

$$D\text{cache}_i = (s_i/2) + l_i - (l_i^2 f_i/2T)$$

So, the mean aggregate delay for all items is:

$$\begin{aligned} D\text{cache} &= \sum D\text{cache}_i p_i = \\ &= \sum (s_i p_i/2) + \sum l_i p_i - \sum (l_i^2 f_i p_i/2T) = \\ &= \sum l_i p_i + (1/2) \sum p_i [s_i - (l_i^2 f_i/T)] = \\ &= \sum l_i p_i + (1/2) \sum p_i l_i [(s_i/l_i) - (l_i f_i/t)] \end{aligned}$$

We denote as $q_i$ the quantity $q_i = l_i f_i/T$. Clearly: $\sum q_i = \sum (l_i f_i/T) = T^{-1} \sum l_i f_i = 1$. Then, $s_i/l_i = s_i f_i/l_i f_i = T/l_i f_i = q_i^{-1}$. This results to:

$$D\text{cache} = \sum p_i l_i + (1/2) \sum p_i l_i [q_i^{-1} - q_i]$$

In order to find the $q_i$ which minimize the mean aggregate delay $D$cache, we set

$$(\partial D\text{cache}/\partial q_i = 0, \text{for } i = 1, 2, \ldots, M$$

So, we find that the following relation must be true:

$$p_1 l_1 (1 + q_1^{-2}) = p_2 l_2 (1 + q_2^{-2}) = \cdots = p_M l_M (1 + q_M^{-2}) = \text{constant}$$

This leads to:

$$\begin{aligned} p_i l_i (1 + q_i^{-2}) = p_i l_i + p_i l_i (T^2/l_i^2 f_i^2) = p_i l_i + p_i l_i (s_i^2 f_i^2/l_i^2 f_i^2) = \\ = p_i l_i + p_i s_i^2/l_i = \text{constant} \end{aligned}$$

Considering the condition of the theorem, one can easily create an on-line scheduling algorithm that constructs optimal schedule for caching clients. We follow the method and the notation of [41]:
*let $Q$ denote the current time; the algorithm below decides which object to transmit at time $Q$. Let $R(j)$ denote the time at which an instance of object $O_j$ was most recently transmitted and $H(j)$ denote the quantity $H(j) = \{[Q - R(j)]^2 p_j/l_j\} + p_j l_j, j = 1, 2, \ldots, M$ ($R(j) = -1$ if $Q_j$ was never transmitted before).*
**On-line scheduling algorithm:**
Step 1: Calculate $H_{\max} = \max\{H(j)\}$, for all $j$
Step 2: Choose $O_i$ such that $H(i) = H_{\max}$ (if this equality holds true for more than one object, then choose one of them arbitrarily)
Step 3: Transmit object $O_i$ at time $Q$
Step 4: $R(i) = Q$, go to Step 1

Observe that $[Q - R(j)]$ is the spacing between the current time and the time at which $O_i$ was previously transmitted. The algorithm tries to keep constant the quantity

$$H(j) = \{[Q - R(j)]^2 p_j/l_j\} + p_j l_j$$

This quantity is similar to $p_i s_i^2/l_i + p_i l_i$, which must be constant according to the theorem.

### 3.6 Pre-emptive Object Scheduling

The optimum schedule requires that the consecutive instances of an object $O_i$ are equally spaced within a broadcast cycle $T$. This is a very desirable property, especially for energy-limited users, because it reduces *busy-waiting* (as opposed to *stand-by* or *doze-waiting*) for the desired object. However, the design of an optimum schedule is an NP-hard problem [9], leading to use of heuristics that do not achieve the "perfect" schedule. So, as illustrated in Figure 23, it is very probable that an instance of an object ($O_2$ with length $l_2$ in the figure) will be broadcasted after the expected time (for perfect periodicity) with high probability, because the transmission of another object ($O_1$ in the figure) is in progress and must be completed first.

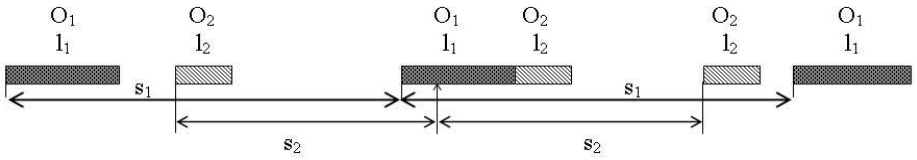



**Fig. 23.** A case where perfect scheduling is impossible

One can attempt to improve the schedule with the following heuristics:

1. Interrupt the transmission of object $O_1$, transmit part of $O_2$, complete the transmission of $O_1$ and then of $O_2$ (see Figure 24). As we prove below, this *interleaving* is not the right approach, because it always increases the total delay. It is easy to see that the same conclusion holds true if we attempt to do a finer interleaving.
2. Interrupt the transmission of $O_1$, transmit the complete $O_2$, and then resume and complete the transmission of $O_1$ (see Figure 25). This is an example of *pre-emptive transmission*. As we prove below, this approach may decrease the total delay under the right circumstances.
3. Transmit $O_2$ first (ahead of schedule) and then $O_1$ (behind schedule) (see Figure 3.6). Again, this ahead-of-schedule transmission decreases the total delay under certain conditions.

The results mentioned for each heuristic above hold for both client models, caching and cache-less. In the remaining subsection, we analyze all heuristics for the case of cache-less clients, because this analysis is much easier to present. We also present, for comparison, as case 4, the analysis of pre-emptive transmission for caching clients and derive the condition under which the total delay is reduced. In all cases analyzed below, only the delays of clients waiting for objects $O_1$ and $O_2$ are influenced. Remaining clients do not experience any difference in performance.

**Case 1: Interleaving (Cache-less clients)**

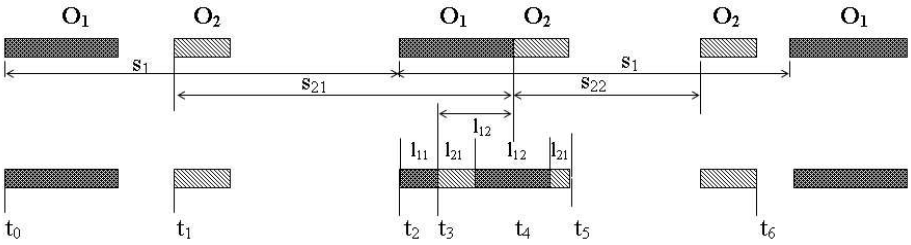



**Fig. 24.** Interleaving objects $O_1$ and $O_2$

*Calculation of the increase of the average delay for object $O_1$:*
Requests for object $O_1$ appearing in the interval $[t_0, t_2]$ will experience a delay **increase** to receive the object; this increase is equal to $l_{21}$. Remaining requests are unaffected. Therefore, the increase of the mean aggregate delay is equal to: $p_1 s_1 l_{21}/T$.

*Calculation of the increase of the average delay for object $O_2$:*
Requests for $O_2$ that appear in the interval $[t_3, t_4]$ will experience a delay **increase** to receive the object equal to: $t_6 - t_5 = s_{22}$ ($s_{22}$ is the spacing between instances #2 and #3). The remaining requests are unaffected. Given that the interval $[t_3, t_4]$ has length $l_{12}$, the increase of the mean aggregate delay is equal to: $p_2 s_{22} l_{12}/T$.

Combining the results of the two previous calculations, we see that it is not appropriate to interrupt the transmission of $O_1$ in order to transmit part of $O_2$, because this decision always increases the mean aggregate delay by the amount:

$$\text{Total delay increase} = [p_1 s_1 l_{21} + p_2 s_{22} l_{12}]/T > 0$$

**Case 2: Pre-emption (Cache-less clients)**
In this case, the transmission of $O_1$ is interrupted, $O_2$ is transmitted and then the transmission of $O_1$ is completed, as depicted in Figure 25.

*Calculation of the increase of the average delay for object $O_1$:*
Requests for $O_1$ appearing in the interval $[t_0, t_2]$ will experience a delay **increase** equal to $l_2$, in order to receive $O_1$. The remaining requests are unaffected. Therefore, the increase of the mean aggregate delay for $O_1$ is equal to: $p_1 s_1 l_2/T$.

*Calculation of the increase of the average delay for object $O_2$:*
Requests for $O_2$ appearing in the interval $[t_1, t_3]$ (with length $s_{21} - l_{12}$) will experience a delay **decrease** equal to $l_{12}$, in order to receive $O_2$. Requests made during $[t_3, t_4]$ will experience a delay **increase** equal to $s_{22}$ to receive $O_2$. The remaining requests are unaffected. Therefore, the **increase** of the mean aggregate delay is:

$$[-p_2(s_{21} - l_{12})l_{12} + p_2 l_{12} s_{22}]/T$$

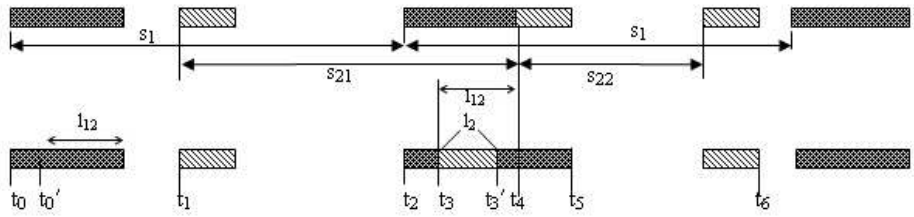



**Fig. 25.** Pre-emptive transmission of $O_2$

Given the above calculations, the total **increase** of the mean aggregate delay $\Delta D$ is:

$$\begin{aligned}\Delta D &= [p_1 s_1 l_2 - p_2(s_{21} - l_{12})l_{12} + p_2 l_{12} s_{22}]/T = \\ &= [p_1 s_1 l_2 + p_2 l_{12}^2 - p_2(s_{21} - s_{22})l_{12}]/T\end{aligned}$$

So, the delay increase depends on $l_{12}$. If we set $d\Delta D/dl_{12} = 0$, then we find that $\Delta D$ takes a minimum value, if $l_{12} = (s_{21} - s_{22})/2$. This minimum value is:

$$\Delta D_{\min} = [p_1 s_1 l_2 - p_2(s_{21} - s_{22})^2]/4T$$

and is negative (= **maximum decrease of delay**) if

$$(s_{21} - s_{22})^2 > 4p_1 s_1 l_2 / p_2$$

Thus, if this inequality holds, then we can reduce the mean aggregate delay, if we interrupt the transmission of $O_1$ after $l_1 - l_{12} = l_1 - (s_{21} - s_{22})/2$ radio units, in order to transmit $O_2$.

**Case 3: Ahead-of-Schedule Transmission (Cache-less clients)**
In this case, we transmit $O_2$ ahead of schedule and $O_1$ behind schedule.

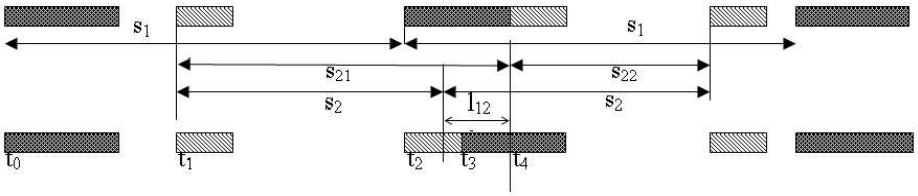



**Fig. 26.** Switching the transmission order of $O_1$ and $O_2$

*Calculation of the increase of the average delay for object $O_1$:*
Requests for $O_1$ appearing in the interval $[t_0, t_2]$ will experience a delay **increase**

equal to $l_2$, in order to receive $O_1$. Requests in the interval $[t_2, t_3]$ will experience a delay **decrease** equal to $(s_1 - l_2)$, while all remaining requests are unaffected. Therefore, the increase of the mean aggregate delay is equal to:

$$[p_1 s_1 l_2 - p_1 l_2 (s_1 - l_2)]/T$$

*Calculation of the increase of the average delay for object $O_2$:*
Requests for $O_2$ appearing in the interval $[t_1, t_2]$ will experience a delay **decrease** equal to $l_1$ to receive $O_2$. Requests in the interval $[t_2, t_4]$ will experience a delay **increase** equal to $s_{22}$. All remaining requests are unaffected. Therefore, the increase of the mean aggregate delay is equal to:

$$[-p_2 l_1 (s_{21} - l_1) + p_2 l_1 s_{22}]/T$$

The total delay increase is:

$$\begin{aligned} \Delta D &= [p_1 s_1 l_2 - p_1 l_2 (s_1 - l_2) - p_2 l_1 (s_{21} - l_1) + p_2 l_1 s_{22}]/T = \\ &= [p_1 l_2^2 + p_2 l_1^2 - p_2 l_1 (s_{21} - s_{22})]/T \end{aligned}$$

This expression becomes negative (**decrease of the delay**) if

$$(s_{21} - s_{22}) > l_1 + (p_1 l_2^2)/p_2 l_1$$

Since $(s_{21} - s_{22}) = 2l_{12}$ (see Figure 3.6), we obtain:

$$l_{12} > [l_1 + (p_1 l_2^2)/p_2 l_1]/2$$

If this inequality holds, the mean aggregate delay is **reduced** when the transmission order of $O_1$ and $O_2$ is switched.

The results of this analysis hold true for the case of caching clients as well; however, the analysis is more involved. We present the analysis of pre-emptive transmission with caching clients.

**Case 4: Pre-emption (Caching clients)**
Consider the pre-emptive transmission of object $O_2$, shown in Figure 25, for the case of caching clients. We derive the conditions under which this pre-emptive transmission reduces the total tuning time.

*Calculation of the increase of the average delay for object $O_1$:*
Clients requesting $O_1$ and appearing in the interval $[t_0', t_2]$ will experience an **increase** of the tuning time by $l_2$. Requests during the interval $[t_2, t_3]$ will not be affected. Requests during $[t_3, t_3']$ will have a mean **decrease** of their tuning time by $l_2/2$. Similarly, requests during $[t_3', t_4]$ will have a **decrease** of their tuning time by $l_2$, while requests during $[t_4, t_5]$ will have a mean decrease by $l_2/2$. Putting these together, we see that the mean increase of the tuning time of clients requesting $O_1$, due to the pre-emptive transmission of $O_2$ is:

$$\Delta D_1 = (p_1/T) l_2 [(s_1 - l_1 + l_{12}) - l_2/2 - (l_{12} - l_2) - l_2/2] = p_1 l_2 (s_1 - l_1)/T$$

*Calculation of the increase of the average delay for object $O_2$:*
Clients requesting $O_2$ and arriving in the interval $[t_1, t_3]$ will have a **decrease** of their tuning time by $l_{12}$. Requests during the interval $[t_3, t_3']$ will have a mean **increase** of their tuning time by $(s_{22} - l_2/2)$, requests during the interval $[t_3', t_4]$ an **increase** by $s_{22}$, and requests during the interval $[t_4, t_5]$ a mean **increase** by $l_2/2$. Thus, the mean **increase** of tuning time for object $O_2$ is:

$$\begin{aligned}\Delta D_2 &= (p_2/T)[-(s_{21} - l_{12})l_{12} + (s_{22} - l_2/2)l_2 + (l_{12} - l_2)s_{22} + l_{22}/2] = \\ &= p_2 l_{12}(s_{22} - s_{21} + l_{12})/T\end{aligned}$$

Working similarly to the case of cache-less clients, we find that the condition under which pre-emptive transmission **reduces** the total tuning time is:

$$(s_{21} - s_{22})^2 > 4p_1(s_1 - l_1)l_2/p_2$$

Thus, the maximum reduction of the tuning time is achieved when: $l_{12} = (s_{21} - s22)/2$.

### 3.7 Conclusions

We showed that, as expected, a local cache reduces the time required for the reception of an object. We also proved that the optimal broadcast schedule for these caching clients is different from the optimal schedule for cache-less clients (it must satisfy the condition $s_i^2 p_i/l_i + p_i l_i =$ constant, rather than the condition $s_i^2 p_i/l_i =$ constant). Considering a broadcast schedule constructed with heuristic algorithms that are based on these conditions, we proved that, for caching or cache-less clients, the mean aggregate delay for the complete reception of an object may be further reduced, if we allow the interruption of the transmission of an object in order to transmit on schedule another more popular one. We deduced the conditions under which this pre-emption reduces the mean aggregate delay. Furthermore, we showed that under some conditions the switching in the transmission order of two neighboring objects may also be advantageous. Finally, we proved that the interleaving of the transmissions of two neighboring objects is not beneficial in any case.

## 4 ACKNOWLEDGMENTS

The authors would like to thank professor Jon Crowcroft for his detailed comments.

## References

1. "FreeFlow: How it Works", Akamai, Cambridge, MA, USA, November 1999.
2. C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems", In *proceedings of ICMCS*, pp. 118–126, June 1996.
3. C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On Optimal Batching Policies for Video-on-Demand Storage Servers", In *Proceedings of ICMCS*, pp. 253–258, Hiroshima, Japan, June 1996.
4. D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang, and S. Zdonik, "Research in Data Broadcast and Dissemination", In *Proceedings of the First International Conference on Advanced Multimedia Content Processing (AMCP'98)*, Lecture Notes in Computer Science, pp. 194–207, Springer Verlag, 1999.
5. D. Aksoy and M. Franklin, "Scheduling for Large-Scale On-Demand Data Broadcasting", In *Proceedings of INFOCOM 1998*, San Francisco, CA, 1998.
6. K. C. Almeroth and M. H. Ammar, "On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service", *IEEE Journal on Selected Areas in Communications*, 14(6):1110–1122, April 1996.
7. S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast", In *Proceedings of ACM SIGCOMM*, August 2002.
8. S. Banerjee, J. Brassil, A. C. Dalal, S. J. Lee, E. Perry, P. Sharma, and A. Thomas, "Rich Media from the Masses", HPL-2002-63R1, HP Lab, May 2002.
9. A. Bar-Noy, B. Randeep, J. Naor, and B. Schieber, "Minimizing service and operation cost of periodic scheduling", In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 11–20, 1998.
10. E. Biersack, A. Jean-Marie, and P. Nain, "Open-loop Video Distribution with Support of VCR Functionnality", *Performance Evaluation*, 49(1-4):411–428, September 2002.
11. Y. Birk and R. Mondri, "Tailored Transmissions for efficient Near-Video-on-Demand Service", In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 226–231, june 1999.
12. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment", In *proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2003.
13. G. Chan and F. Tobagi, "Distributed Servers Architectures for Networks Video Services", *IEEE/ACM Transactions on Networking*, 9(2):125–136, April 2001.
14. Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture", In *proceedings of ACM SIGCOMM*, San Diago, CA, August 2001.
15. Dash Optimization, *Xpress-Mp Essentials*, 2001.
16. DirecPC, URL: http://www.direcpc.com/index.html.
17. D. Eager, M. Vernon, and J. Zahorjan, "Optimal and Efficient Merging Schedules for Video-on-Demand Servers", In *proceedings of the 7th ACM Multimedia Conference*, November 1999.

18. D. Eager, M. Vernon, and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand data delivery", *IEEE Transactions on Knowledge and Data Engineering*, 2001.
19. L. Gao and D. Towsley, "Threshold-Based Multicast for Continuous Media Delivery", *IEEE Transactions on Multimedia*, 3(4):405–414, December 2001.
20. V. Goyal, "Multiple description coding: compression meets the network", *IEEE Signal Processing Magazine*, 18(5):74 –93, September 2001.
21. Y. Guo, S. Sen, and D. Towsley, "Prefix Caching assisted Periodic Broadcast: Framework and Techniques for Streaming Popular Videos", In *proceedings of IEEE ICC*, April 2002.
22. Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: Peer-to-peer Patching Scheme for VoD Service", In *Proceedings of the 12th World Wide Web Conference (WWW)*, Budapest, Hungary, May 2003.
23. G. Heine, *GPRS from A to Z*, Artech House Inc., April 2000.
24. A. Hu, "Video-on-Demand broadcasting protocols: A comprehensive study", In *proceedings of Infocom*, volume 1, pp. 508–517, Anchorage, Alaska, USA, April 2001.
25. K. A. Hua and S. Sheu, "Skyscraper Broadcasting for Metropolitan VOD", In *proceedings of Sigcomm*, August 1997.
26. K. A. Hua, Y. Cai, and S. Sheu, "Patching : A Multicast Technique for True Video-on-Demand Services", In *proceedings of ACM Multimedia*, pp. 191–200, 1998.
27. T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Energy Efficient Indexing on Air", *ACM SIGMOD Record*, 23(2):25–36, June 1994.
28. Infogate, URL: http://www.pointcast.com/.
29. Intracom S.A., URL: http://www.intranet.gr/en/products/internet/ios.htm.
30. R. Jain and J. Werth, "Airdisks and airRAID: Modeling and scheduling periodic wireless data broadcast", , DIMACS Technical Report 95-11, May 1995.
31. J. Jannotti, D. K. Gifford, and K. L. Johnson, "Overcast: Reliable Multicasting with an Overlay Network", In *proceedings of the 4-th Symp. on Operating Systems Design and Implementation*, Usenix, Octobre 2000.
32. L. Juhn and L. Tseng, "Fast Data Broadcasting and Receiving Scheme for Popular Video Service", *IEEE Trans.on Broadcasting*, 44(1):100–105, March 1998.
33. L.-S. Juhn and L.-M. Tseng, "Harmonic Broadcasting for Video-on-Demand Service", *IEEE Transactions on Broadcasting*, 43(3), September 1997.
34. K. Kong and D. Ghosal, "Mitigating Server-Side Congestion in the Internet through Pseudoserving", *IEEE/ACM Transactions on Networking*, pp. 530–544, August 1999.
35. J. Nussbaumer, B. Patel, F. Schaffa, and J. Sterbenz, "Networking Requirements for Interactive Video on Demand", *IEEE Journal on Selected Areas in Communications*, 13(5):779–787, 1995.
36. V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking", In *proceedings of NOSSDAV*, May 2002.
37. S. Ramesh, I. Rhee, and K. Guo, "Multicast with Cache (MCache): An Adaptive Zero Delay Video-on-Demand Service", *IEEE Transactions of Circuit and System of Video Transmission*, 11(3), March 2001.
38. S. Shekhar and D. Liu, "Genesis and Advanced Traveler Information Systems ATIS: Killer Applications for Mobile Computing", MOBIDATA Workshop, 1994.
39. S. Sheu and K. A. Hua, "Virtual Batching: A New Scheduling Technique for Video-on-Demand Servers", In *Database Systems for Advanced Applications*, pp. 481–490, April 1997.

40. TiVo, "What is TiVo: Technical Aspects", 2003.
41. N. Vaidya and S. Hameed, "Data broadcast in asymmetric wireless environments", In *Proceedings of Workshop on Satellite-based Information Services (WOSBIS)*, New York, November 1996.
42. N. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments", *ACM/Baltzer Wireless Networks*, 5(3):171–182, 1999.
43. S. Viswanathan and T. Imielinski, "Pyramid Broadcasting for Video On Demand Service", In *Proceedings of Multimedia Conference*, San Jose, CA, February 1995.
44. B. Wang, S. Sen, M. Adler, and D. Towsley, "Proxy-based Distribution of Streaming Video over Unicast/Multicast Conncetions", In *proceedings of Infocom*, june 2002.
45. P. P. White and J. Crowcroft, "Optimized Batch Patching with Classes of Service", *ACM Communications Review*, October 2000.
46. J. W. Wong, "Broadcast delivery", *Proceedings of the IEEE*, 76:1566–1577, December 1999.
47. D. Xu, H.-K. Chai, C. Rosenberg, and S. Kulkarni, "Analysis of a Hybrid Architecture for Cost-Effective Streaming Media Distribution", In *proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, Santa Clara, CA, January 2003.
48. Y. Zhao, D. Eager, and M. Vernon, "Network Bandwidth Requirements for Scalable On-Demand Streaming", In *proceedings of Infocom*, june 2002.