The Use of Stream Merging Mechanisms in a Hierarchical CDN

Carsten Griwodz
griff@ifi.uio.no
University of Oslo – Department of Informatics
Gaustadallen 23 – 0371 Oslo, Norway

***Abstract*— Content distribution networks (CDNs) are a popular service for the dissemination of multimedia content over wide areas. The existance of a centralized administrative structure makes them attractive for the commercial distribution of high quality content. By sharing resources, service providers can implement their services more efficiently than a single content provider could establish a distribution structure itself. An efficient operation requires cost estimations that allow service providers to determine the dimensioning of their infrastructure and the placement of content in the system. In case of video streaming, distribution mechanisms that exploit multicast, segmented and out-of-order delivery can be applied to merge streams and reduce resource consumption. Several applicable stream merging mechanisms exist in the literature and can be used.**

We examine three such mechanisms, namely patching, gleaning and prefix caching in a hierarchically organized CDN, and show that a co-optimization of movie placement and stream merging mechanism has an undesirable effect on quality by delivering highly popular movies over longer distances than less popular ones. We explore and compare two approaches for overcoming this problem by qualifying the placement optimization with additional conditions, and find that in this case, straight-forward sorting is a good solution.

I. Introduction

The controversy over peer-to-peer systems has shown that a demand for world-wide, borderless access to audiovisual content exists. It has shown as well that a rudimentary distribution infrastructure can be established over the current Internet. The problem is, of course, that this distribution is frequently infringing the copyrights of media companies. Since central control in peer-to-peer systems is somewhere between difficult to establish and undesired, it is unlikely that media companies will follow this approach to satisfy users' demands.

But the popularity of peer-to-peer applications can not be ignored as a demonstration of user demands. They provide world-wide access to an indiscriminate choice of content. Previous video-on-demand systems have not provided this but were restricted to a small subset of the existing titles. VoD trials were also proof that the replication of all accessible content to each regional content provider is not feasible, and an appropriate distribution system is required. Through content distribution networks (CDNs), a similar accessibility may be achieved. They allow a bigger amount of control than peer-to-peer networks because digital rights management issues are more easily addressable in CDNs. The issue of cost-effective deployment of such large-scale CDNs remains. To reduce the cost of deploying and operating a system that holds copies of the content and delivers it to a large number of users, the resource utilization in such a system must be high and predictable.

For fully replicated movies, we have investigated the cost-optimal placement of movies on servers in a distribution system analytically, the so-called *root servers of the movies*. As expected, the placement combined with a video-on-demand (VoD) approach that is implemented by unicast delivery from the root server chosen for each movie becomes optimal when popular movies have root servers closer to the clients, and less popular titles are assigned servers as root servers that are located further away from the clients. This is also the case when very simple streaming merging schemes, such as batching, are combined with optimized placement. Similarly, patching without restart of a multicast stream before completion of the previous one are examples of such simple mechanisms. If, however, the use of multicast and out-of-order transmission is optimized together with the placement, it is not necessarily the case that more popular movies are stored closer to the client than less popular ones.

As a result, more popular titles may be delivered to the clients with higher startup latency, jitter and loss rate than less popular ones if no means for guaranteeing QoS are applied. This reduces the average satisfaction of users with the services and should be prevented.

Besides optimal patching, we show that the problem appears also in gleaning (patching with caching) and optimized prefix caching. We then examine two means of enforcing that more popular titles are located closer to the user. One approach introduces an artificial cost penalty for QoS reduction into the cost computation. The other simply enforces that distances from the clients are sorted by popularity. In section II, we present stream merging mechanisms from the literature that can be applied in addition to the replication of content. This is followed by an introduction to the analytical model that we use for cost estimation of combined placement and stream merging in section III. Our optimization approach is explained as well. We present the cost computation for three stream merging mechanisms in the appendix V. For each of the mechanisms, the result of combined optimization of stream merging and placement for an overall minimal cost if shown in an example. In section IV, we show the placement results and the changes in overall cost when QoS is taken into account. Section V concludes the paper.

II. Related Work

Existing research can be applied to reduce the consumption of resources such as the number of concurrent streams that must be supported by a server and the bandwidth that is required for streaming to end users. This existing work can be divided into single server approaches and approaches that apply server hierarchies in a distribution system.

Research that considers single server systems has frequently focussed on the efficient use of streaming capacity. The near VoD (NVoD) approaches apply scheduling of movie transmissions without user interaction based on statistics about the popularity of movies. These approaches are collectively called periodic broadcasting schemes and many are derived from pyramid broadcasting [1]. True VoD (TVoD) approaches take user interaction into account in their resource allocation. They reduce the load of servers by answering to several user requests at once using multicast. Examples are batching [2], dynamic batching [3], piggybacking [4], content insertion [5,6], chaining [7] and patching [8]. Bradshaw et al. [9] refer to several original periodic broadcasting works and present an implementation as well as measurement results for these techniques, including patching.

Research that considers hierarchical systems did not attract much attention until recently, in spite of some early papers, especially by Nussbaumer et al. [10]. Stand-alone video-on-demand development had little influence [11] but integration of streaming into web services has attracted attention [12]. To increase the efficiency of the distribution system and to overcome the varying network quality of the Internet, partial caching approaches have been developed. There are two general approaches: caching a *part of the* entire movie's *length*, or caching only a *part of the quality*.

A look at a distribution system with several layers of caches is taken by Chan and Tobagi, who investigate several related approaches for the optimized delivery of movies in a hierarchical distribution system, where each server may hold part of the length of a movie [13].

The partial caching approach called proxy prefix caching [14] stores the first part of the movie in a proxy called a prefix cache and delivers the rest of the movie from a root server for that movie. Zhang et al. introduce video staging [15], a partial caching approach aimed at smoothing and reduction of resource requirements. Both approaches are vulnerable to changes in network quality for the uncached part of the movie. Layered caching [16,17] tries to overcome this problem by caching the entire length of the movie but reducing its quality if the movies popularity does not warrant its complete storage. This assumes the availability of layer encoded video to work. Since such codecs are not available at this time, we have investigated the combination of caching and stream merging mechanisms only for full movie caching in gleaning [18]. Eager et al. provide an optimization of a distribution system with one intermediate level of proxy caches [19]. A variety of the mentioned optimization options can be applied for the given hierarchical distribution system on the basis of existing research results or common practice.

In this paper, we take a closer look at optimized patching, gleaning and prefix caching in conjunction with the assignment of movies to servers. In the following, these approaches are described in more detail. [13] is not considered because its free use of servers makes a direct application unlikely for stability reasons. The smoothing achieved by video staging could be combined with any of the approaches. Layering is not desirable for a commercial system if quality changes are large. For small changes, the approach could be combined with any of the presented approaches as well.

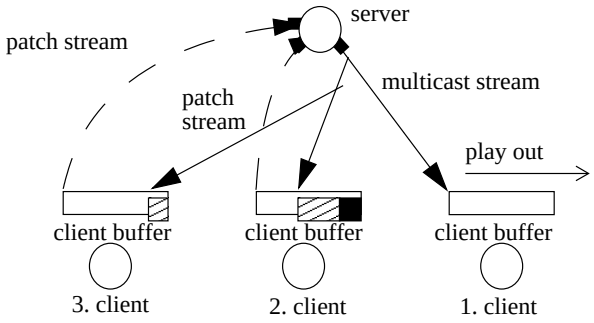

Fig. 1. Patching

A. λ-patching

Patching works by creating a multicast group for the delivery of a video stream to a requesting client. If another client requests the same movie shortly after the start of this transmission, this client starts storing the multicast transmission in a local cache immediately. The server sends a unicast stream to this client containing the missing initial portion of the movie, until the cached portion is reached. Then, the client uses its cache as a cyclic buffer. Figure 1 illustrates patching. The period in which the use of unicast patch streams is more resource efficient than the transmission of a new multicast is limited, and can be optimized based on the request interarrival time. The computation for this optimum has been found for λ-patching [20], controlled multicast [21] and optimal patching [22].

B. Gleaning

Patching requires buffers at clients and the bandwidth to receive two streams in parallel at playout rate. Gleaning is the idea of exploiting patching efficiency with standard clients as shown in figure 2. At the cost of proxy installations close to clients, it enables the use of patching techniques without client support. The proxy close to the client buffer and re-order the multicast and patch streams of patching and deliver them to clients at playout speed and in playout order. The gleaning proxies are always present in this approach. After delivery to the client, the data is usually discarded from a proxy's buffer. To simplify the investigation, we assume that buffers are kept until the entire stream has been delivered. This implies that additional clients receive data from the buffer. Secondly, the caching algorithm of the system can determine caching of a movie at level 1. The buffer is never discarded and an entire movie's length of disk space is allocated.

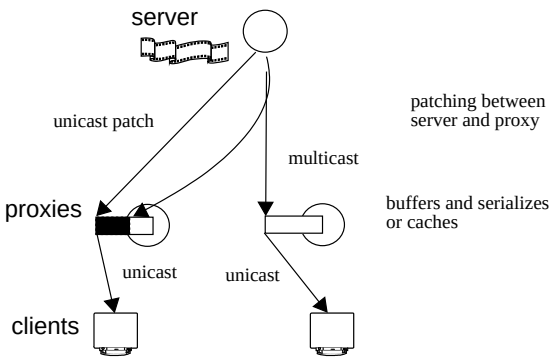

Fig. 2. Gleaning

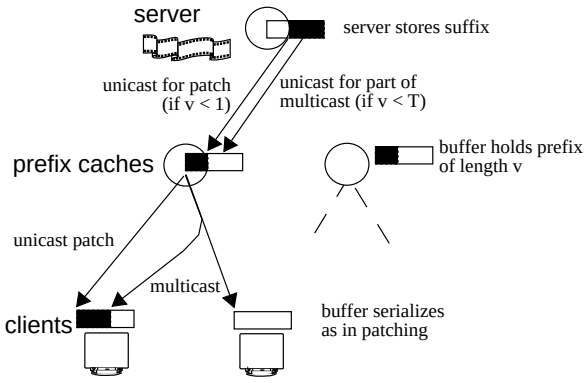

Fig. 3. MPatch

C. Prefix Caching

Prefix caching is a practically oriented scheme that caches the beginning of movies (the prefix) on proxies (called prefix caches) that are located close to the clients [14] and reduces problems of congestion and jitter in long-distance delivery. When the prefix cache receives a request for a movie, it starts the transmission of the prefix immediately and requests the rest of the movie (the suffix) from the root server for forwarding. The original does neither use multicast nor optimize the selection of the prefix length. Several variations have been introduced to reduce costs predictably. UCast, SBatch, UPatch and MPatch are presented in [23]. Very similar approaches have been introduced independently as mcache [24].

In this paper, we consider MPatch, which uses multicast between the prefix cache and the clients but not between root server and prefix caches. When a first client requests a movie, the prefix of length v is sent immediately by multicast, and the suffix is scheduled for transmission from the root server to arrive exactly at the end of the prefix. This suffix is sent by unicast from the root server to the proxy, and redistributed by multicast from the proxy to the clients. When another request for the movie arrives after an independently defined threshold time T, it is handled in the same manner. If it arrives before T, it joins the original multicast, and receives the missing start by unicast. MPatch allows values of T that are smaller or larger than v. If they are larger, part of the suffix that must be sent by unicast from the root server as well. [23] does not restrict the choice of v or T. In figure 3, we can see the situation when $T > v$. A section of length $T - v$ must be sent from the root server to the proxy by unicast, and is re-distributed by multicast from the proxy.

III. Modeling Deployment and Operation Cost

In this section, we will first present the topology model that we use in our cost estimation. This is followed by an introduction of the costs that we consider and a description of our optimization approach.

A. Topology model

In our approach to model the cost of deployment and operation of a distribution system we take into account that the network is already in place, topologies can not be changed easily, but servers can be deployed and dimensioned within the existing distribution system. Considering that a wide-area distribution system will involve networks of several network providers, an overlay network of servers is a likely choice. In such an overlay network, our model of a distribution system that is organized as a homogeneous tree is an appropriate model for planning, although it does not reflect a physical topology. Deployment of the service on an overlay network is consistent with the assumption of linear cost development and planning in terms of average costs. Our model aims at the computation of the cost for deployment and operation. Such a model must combine one-time installation costs with maintenance costs and resource consumption. Because of the different time-scales of these models, from a depreciation period for installations to the bandwidth needed for the transfer of a video frame, our entire model is timeless. This implies that we assume a constant average number of active clients, a constant average number of streams that are supported by a server, and a one-time cost for the installation of a server or network link. We assume that an average number of users accesses the system and do not model idle users at all. This number is constant but not necessarily a natural number. We use the Zipf-distribution to model popularity of movies which is usually proposed in the literature [13,25]. Its problems of temporal development [26,27] can be ignored here because we consider only average relevance. Without time in our model, the number of concurrent retrievals of a movie within a fraction of its length is determined directly by its popularity and the number of users, i.e., hit probability takes the place of interarrival times in the computations.

Since we investigate large systems with central control, we assume that a good prediction of hit probabilities is realistic. This allows to move movies to optimal positions within the system. Since we do not take temporal development into account in our cost estimation, we can ignore also the changes in the relative popularity of movies over time. The cost of movement due to popularity changes is currently ignored in our model. While this may be a considerable amount of traffic if system dimensions are faulty, it will be negligible in the optimized case. Even for approaches using autonomous caching, this traffic was shown to be negligible [28], and centralized systems can make better decisions.

Figure 4 shows our model of a distribution system annotated with costs. To discuss the model, we borrow from the

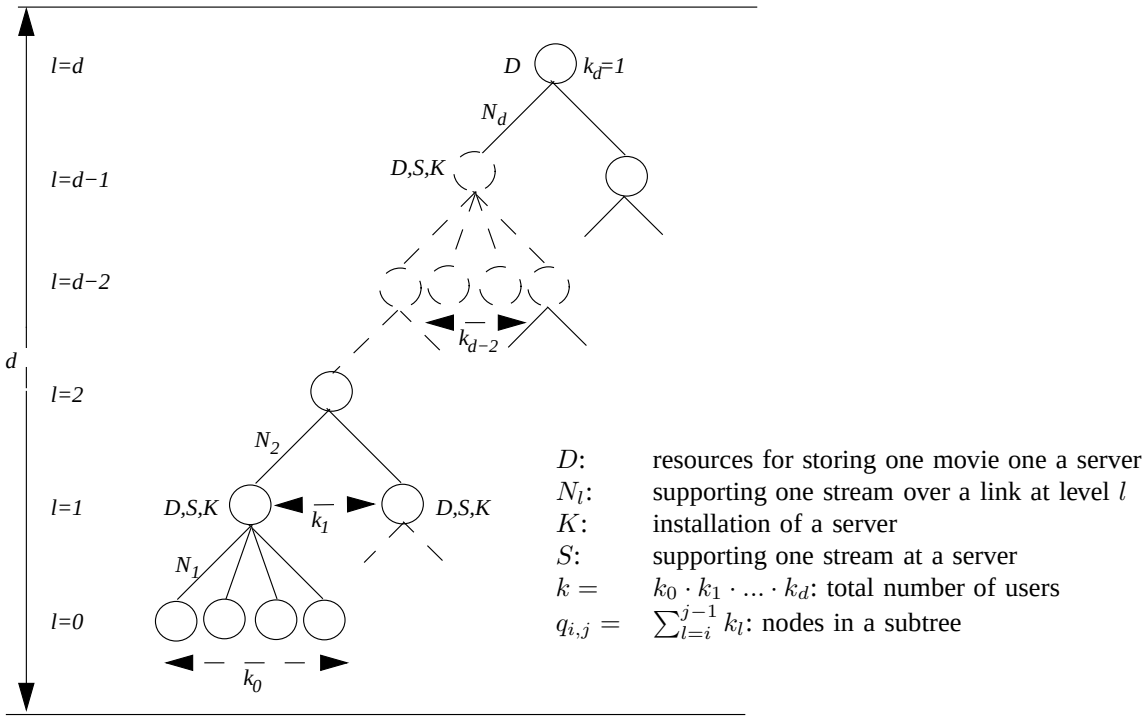

Fig. 4. Homogeneous distribution system

terminology of CPU caches, and call the servers closest to the end-users first-level servers, the next closer to the origin server a second-level server, and so on. Clients are located at level 0, the origin server at level d, i.e., including the levels of the clients and the origin server, the model has $d+1$ levels. The root servers for a movie may be placed anywhere between levels 1 and d. The tree is homogenous, meaning that the number of subtrees k_l of each node at a level $l+1$ is constant.

The cost for storing one movie on a server is denoted D, the cost for the support of one stream over a link that connects levels l and $l-1$ is denoted N_l, and each supported stream has a fixed cost S at the server from which it originates. The main reason for layer-dependent values N_l is that we want to be able to distinguish N_1, the consumer's access link and the consumer's storage from other levels. We add a fixed cost K for each installation of a server that is necessary, i.e., we assume that a server is installed at level l of the hierarchy if and only if movies are placed into the servers at level l. It may be unclear why the cost K is not integrated into D or S at first, but the cost of deploying and operating a set of servers at a whole level in a distribution system is not marginal, starting with the required floor space and operators. In the case of a multicast stream that traverses a node at level $l+1$ to two directly connected nodes at level l, the link costs between levels l and $l+1$ remains fixed in our model. This definition is consistent with overlay approaches to content distribution that use application-specific or application-level multicast [29].

B. Cost model and computation

We want to minimize the cost of deployment and operation, which implies that several timescales have to be optimized together. The deployment must be considered in a reasonable time for an amortization of investments that is measured in years. The length of a movie is measured in minutes, and stream segments may be measured in seconds. For the combination we consider only average cost, which is timeless.

$C_{l,D}$	storage costs at level l
C_D	total storage costs
$C^u_{l,N}$	unicast network costs at level l
$C^m_{l,N}$	multicast network costs at level l
$C_{l,N}$	network costs at level l
C_N	total network costs
$C^u_{l,S}$	unicast interface costs at level l
$C^m_{l,S}$	multicast interface costs at level l
$C_{l,S}$	interface costs at level l
C_S	total interface costs
$C_{l,K}$	server installation costs at level l
C_K	total server installation costs

TABLE I
OVERVIEW OF PARTIAL COSTS

The model presented in Section III-A provides us with the means to describe a homogeneous, hierarchical distribution system. To use it for the optimization of placement, we must use it to compute costs. We consider four cost factors: the storage space, the bandwidth that must be allocated to the system, the networking equipment necessary to support the bandwidth, and the cost of a server installation. We use to C denote the sum of the various partial costs the we compute. An overview of the partial costs is given in Table I. We model the popularity of movies according to the Zipf distribution.

To choose a resource assignment that achieves the minimal

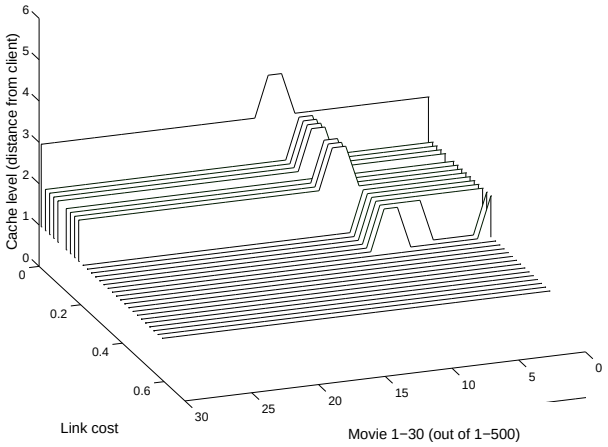

Fig. 5. Optimal placement for l-patching

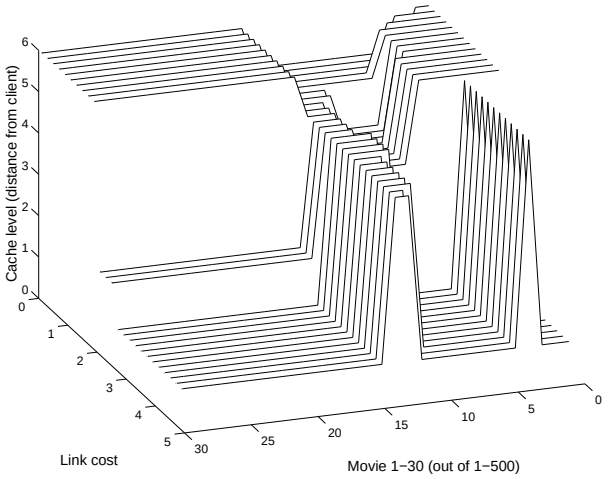

Fig. 6. Optimal placement for gleaning

deployment and operational costs for a given topology, we use an optimization program. The program takes as arguments one of the cost functions that are presented in the appendix V, the number of movies M, the number of levels l, the arity of the homogeneous tree at each level k_l, the storage and interface cost per movie, the bandwidth cost per movie at each level and the cost for the installation of a server. We assume that each leaf of the tree is an active user so the number of users is computed from the arity of the tree. We have chosen the parameter ζ of the Zipf distribution as $\zeta = 1$, which determines the popularity for each movie. For each set of parameters, the optimization program yields the assignment of movies to hierarchy levels that achieves the minimal cost, and the cost itself.

We use branch-and-bound for the optimization because the number of valid assignments is relatively small. The cost for server installations, $C_{l,K}$, is not introduced into the computation of cost for an individual assignment decision. Rather, we check all possible subsets of levels, add $C_{l,K}$ and compare those results. By checking the subsets separately we can find a global cost minimum through branch-and-bound. For the experiments that we use in this paper to demonstrate the effects of combined optimization in placement and stream merging mechanism, we assume fixed values: $M = 500$, $k_0 = 5000$ and $k_l = 10, \forall l > 0$. All movies are supposed to have identical length and throughput requirements. We optimize the placement for various link costs, and keep the storage cost fixed ($D_l = 1$).

C. Discussion

We have used the equations developed in the appendix V as the basis of our placement optimization algorithm. We have not used current or projected link and storage costs. Since these change frequently and CDN providers receive massive discounts anyway, we concentrate on qualitative observations instead.

The figures in this section show series of experiments in optimizing the levels at which movies in a distribution hierarchy should be kept. On the X-axis it shows movies

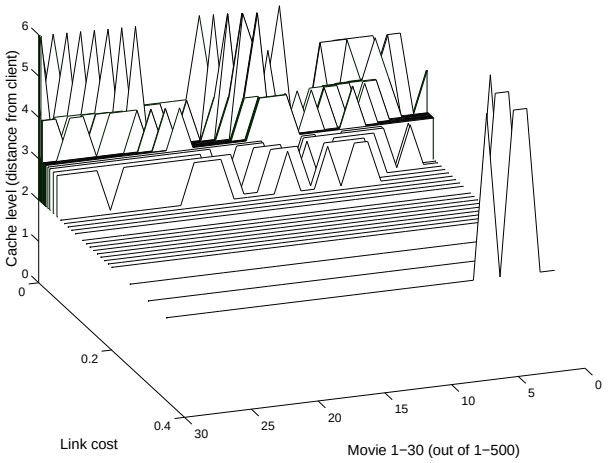

Fig. 7. Optimal placement for mpatch

sorted according to their popularity, where m_0 has the highest popularity and m_M the lowest. Each experiment that we perform results in a cost-optimal assignment of all movies to caching levels, i.e., one line in the X-Y plane shows the result of one experiment. The Figures 5, 6 and 7 show only the subset of highly popular movies that behave unexpectedly. Only a subset of movies is shown. The reason is that the placement for less popular titles holds no surprises. The experiments are performed with several link costs to observe the development depending on the relation of storage and link costs. We do not show the result for link cost 0, which places all content at the origin server.

1) λ-patching: For λ-patching, we observe optimal placements like in figure 5. More movies are stored closer to the client with an increasing link cost, but less relevant movies remain on higher levels of the distribution hierarchy. The reason is that the average network bandwidth that is consumed by accesses to these movies incurs less cost than storing more copies of these movies at the lower levels, i.e., closer to the client.

Figure 5 shows clearly that due to the cost reduction by efficiently using multicast, the optimal placement for λ-

patching is not always achieved by placing the most popular movies closer to the clients. Since the number of client requests is growing exponentially when the movie is stored at a higher level, but savings are achieved on the multicast links, the chosen patching windows is considerably smaller in comparison to the one chosen under server load optimization. The load generated by unicast streams, and thus the overall cost is smaller, and the incentive for storing the content close to the client is reduced. However, if a movie is extremely popular, the top of the distribution tree is reached and the exponential growth of requests that is associated with moving the movie towards the origin server stops, and the optimal position is found closer to the clients again.

2) Gleaning: The presence of the proxies at level 1 of the distribution hierarchy that uses gleaning (Figure 6) intensifies the effects that have we observe for λ-patching (Figure 5). The link cost in gleaning must be much higher to counteract the efficiency of multicast delivery, and find a cost-optimal placement of the movies at level 1. Since a large server installation cost is always present at level 1, fast changes in the placement decisions are more likely. In figure 6, the effect describes for l-patching is even repeated twice. Our model does not reflect that batching is the appropriate technique for very small inter-arrival times. This would result in a zero unicast cost and incentive to distribute the movie from a higher level of the distribution tree. The optimal placement would trade multicast link costs against storage costs.

3) MPatch: In mpatch (Figure 7), the possibility of storing movies partially at the prefix caches at level 1, which are always present, could be expected to have similar effects as gleaning. With this expectation, the low link costs that are required to force an optimal placement of complete movies at level 1 needs an explanation. The reason for this is that gleaning exploits multicast between the root server and the proxy at level 1, but mpatch uses multicast only between the prefix cache at level 1 and the clients, but not between root server and prefix proxies. Thus, the increasing link cost forces a placement closer to the client.

It is particularly noteworthy that the mpatch placement is more erratic that that of λ-patching and gleaning. The reasons is that in mpatch, not only multicast and placement can be adapted to reduce costs. Additionally, the length of a multicast section T_i and the length of the prefix at level 1, v_i, can be chosen per movie. With four competing optimizations and small margins, the placement changes frequently.

4) Interpretation: Storing highly popular movies further away from clients than less popular ones has consequences. Users will experience an increase in start-up latency, whether the system can rely on quality-of-service (QoS) guarantees in the network or not. Without QoS guarantees in the network, viewers will also experience more jitter and more loss than for a closer placement. In a commercial system, this may be considered acceptable if the movies that are moved away from the clients have only few viewers. We have shown in 3 examples that the co-optimization of placement and stream merging mechanism results in a situation where highly popular movies are placed further away from clients than less popular ones. As a result, the average user satisfaction with the service will be reduced.

While all of the presented approaches could address the problem by applying prefetching, retransmission or forward error correction approaches, this would increase the latency of the system in many cases. We consider an integration of user satisfaction into the placement mechanism a more general solution to the problem. In the following section, we examines this approach.

IV. QoS Maintenance

In this section, we consider two possibilities to increase the average user satisfaction in a system that optimizes placement and stream merging mechanism together. The parameter that determines user satisfaction in case of placement is the distance of a client from the root server of a requested movie. Because of the dominance of requests to popular movies, we consider it appropriate to identify this goal with the condition that popular movies should not be stored further away from the clients than less popular ones.

To achieve this, we compare two approaches. In the first, we introduce a penalty for distance into the cost formulas developed in the appendix V. In the second, we enforce a placement order in such a way that more popular movie are never placed further away from clients than less popular ones.

While the enforced order approach will always result in the expected order of placements, the penalty approach can not prevent non-intuitive ordering of movies with medium popularity. The penalty approach would be an appropriate way to go if such a penalty could actually be defined in terms of cost that customers are not willing to pay in case of quality degradation. In that case, this approach would be appropriate because revenue reductions can be considered in the cost function. In our case the penalty remains virtual. It is not included in the computation of the overall cost but only in the assignment of movies to layers.

We do not present graphs of the placement that results from the use of the placement modifiers. The placement results mostly in the expected strict order to placements, where more popular movies are closer to the clients than less popular ones. Spuriously, the penalty approach shows a minor deviations by one level. The effect of the placement modifiers on the overall cost is more important for planning, so we present the increase of total costs between an unmodified system and one that is modified with either of the two approaches.

To put the cost differences into perspective, we show the development of total cost with increasing link cost vs constant storage cost for the three unmodified techniques in Figure 8. When compared with the deviation of the cost modifiers, it becomes clear that it is small in all cases that are shown in figures 9, 10 and 11. A common point of all three stream merging mechanisms is that the penalty approach does not work very well for small link costs and becomes more efficient when the link cost grows. This is due to the way in which the penalty is introduced: as an additional, virtual cost for each link that a requested movie has to cross for delivery to the client. Although this distance must be covered only for part of the movie in prefix caching, we consider the complete viewing

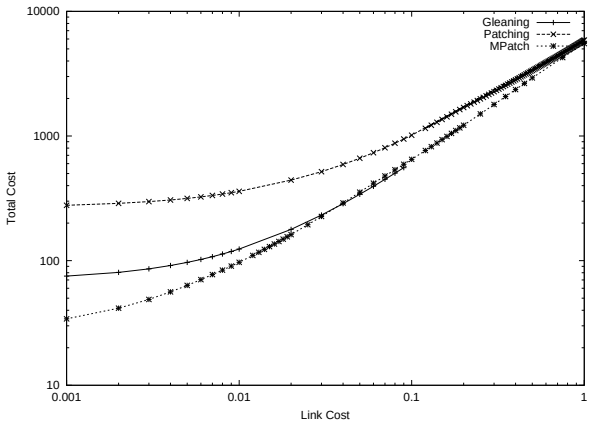

Fig. 8. Cost comparison

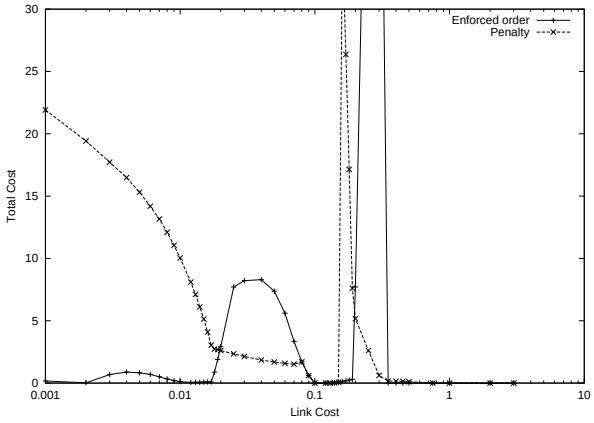

Fig. 9. Cost difference for l-patching

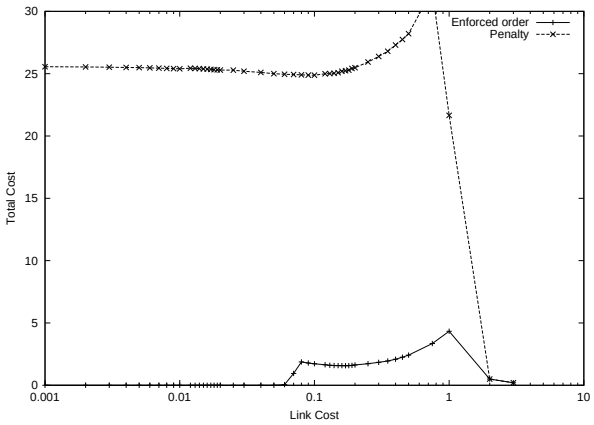

Fig. 10. Cost difference for gleaning

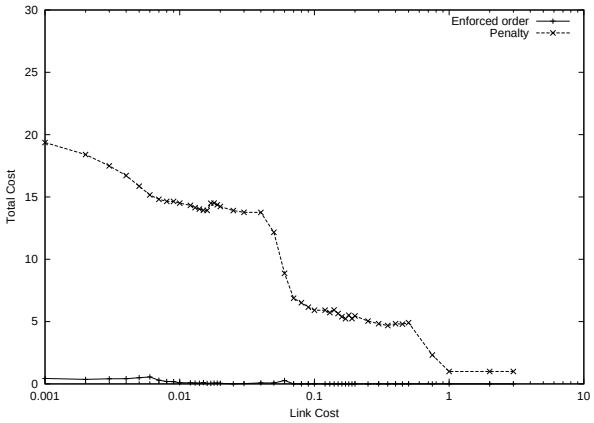

Fig. 11. Cost difference for MPatch

experience penalized by a quality reduction of the suffix. Since the penalty is fixed, it is obvious that it has a higher influence on the placement optimization when the link cost (and thus the total cost) of a placement is low. Enforced order, on the other hand, does not influence the cost directly but by preventing some placement evaluations entirely.

The inconsistent cost development in λ-patching in figure 9 coincides with the necessity to remove undesirable placements that we can see in figure 5. When the enforced order modifier exceeds the cost deviation of the penalty modifier, we have a situation where the penalty modifier does not achieve perfect order. The high double peak relates to the optimal placement that stores all but a few popular titles at level 1. The big deviations can be explained by the lack of flexibility of λ-patching: the only parameter that can be chosen is the placement itself.

Figure 10 shows the cost development in gleaning. The penalty modifier performs badly compared to the enforced order modifier. Enforced order can ignore optimal placements at higher levels because the installation cost of the proxies at level 1 is so high that movies can either be stored at level 1 or 6. The margin for placement at all other levels is so small that the cost deviation remains low. The penalty approach, on the other hand, acts like an additional unicast cost and cancels the savings of multicast delivery from level 6. Movies are placed inefficiently at level 1, resulting in the last deviation.

Mpatch is highly adaptive to all kinds of cost development, as shown in Figure 11. Since 3 parameters can be manipulated separately (placement, v_i and T_i), a restriction of the placement options penalizes the total cost only slightly. The penalty approach, on the other hand, leads the optimizer to choose these parameters in efficiently. The cost deviation for enforced order is smaller than for gleaning because mpatch does not use multicast between root server and prefix cache, so the penalty does not affect the optimizer as badly as in the gleaning case.

In general we can conclude that the simplest approach for providing the most cost-efficient placement while maintaining the highest average user satisfaction is an appropriate choice. A penalty should only be introduced into placement optimization if users actually pay for quality. But in both cases, the cost computation functions that we developed in the appendix V allow a combined optimization of placement and stream merging mechanism. We have furthermore seen that stream merging mechanisms with several free parameters such as mpatch are flexible in adapting to additional conditions. Although they could not be used for optimization in real-time, their use for capacity planning can keep cost low especially if further conditions must be met.

V. CONCLUSION

We have presented our approach for estimating the cost of deployment and operation of a CDN with several hierarchical levels. Our investigation was made under the assumption that overlay networks will increase in relevance, and that centrally controlled, hierarchical distribution systems will be a popular approach to their deployment. The approach determines the minimal cost and the placement of movies that achieves the minimal cost. Such an approach allows to determine whether an optimal placement is feasible in reality and whether the cost development is stable. We have applied this approach to three stream merging mechanisms that apply multicast, partial delivery, and out-or-order delivery of movie segments to reduce the resource use of the distribution system. Examining the results of optimizing these mechanisms together with the placement of movies on servers in a CDN, we found that the cost-optimal placement decision places popular movies further away from clients than less popular ones. This would result in a reduced average quality of service for the clients.

To modify this decision, we introduce placement modifiers into the cost computation. One modifier enforces the desired placement by preventing the evaluation of other orders. The other modifier adds a penalty for distance to the cost computation. We have found that the simple 'enforcement' approach provides the better results if quality if not actually paid for by the client. If quality influences the revenue that is generated with the CDN, the 'penalty' approach can be implemented intuitively and would then provide the best, although not necessarily an ordered placement. We have furthermore observed that stream merging mechanisms with several variables that can be chosen freely can adapt more easily to modified cost computation conditions. By reconfiguration they achieve costs that are similar to the optimal cost. On the other hand, the optimal placement becomes less predictable and less stable when more variables can be chosen freely. Near optimal solutions may be achieved by making major changes to the placement decision.

In our future work, we will extend the combination of caching and delivery to layered video approaches which we consider necessary to cope with heterogeneous end systems and unpredictable congestion problems in real-world implementations. Furthermore, we intend to integrate the optimization of internal server resources with stream stream merging mechanisms.

REFERENCES

[1] Charu C. Aggarwal, Joel L. Wolf, and Philip S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS)*, pages 118–126, Hiroshima, Japan, June 1996. SPIE.

[2] Asit Dan, Dinkar Sitaram, and Perwez Shahabuddin. Scheduling policies for an on-demand video server with batching. Technical Report RC 19381, IBM, 1993.

[3] Asit Dan, Perwez Shahabuddin, Dinkar Sitaram, and Don Towsley. Channel allocation under batchiing and vcr control in video-on-demand systems. Technical Report RC 19588, IBM, September 1994.

[4] Leana Golubchik, John C. S. Lui, and Richard R. Muntz. Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers. *ACM/Springer Multimedia Systems*, 4(3):140–155, June 1996.

[5] Dinesh Venkatesh and Thomas D. C. Little. Dynamic service aggregation for efficient use of ressources in interactive video delivery. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 113–116, Durham, New Hampshire, USA, November 1995.

[6] Rajesh Krishnan, Dinesh Venkatesh, and Thomas D. C. Little. A failure and overload tolerance mechanism for continuous media servers. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 131–142, Seattle, WA, USA, November 1997.

[7] Simon Sheu, Kien A. Hua, and Wallapak Tavanapong. A generalized batching technique for video on demand. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 110–117, Seattle, WA, USA, June 1997. ACM Press.

[8] Kien A. Hua, Yin Cai, and Simon Sheu. Patching: A Multicast Technique for True Video-on-Demand Services. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 191–200, Bristol, UK, September 1998. ACM Press.

[9] Michael Bradshaw, Bing Wang, Subhabrata Sen, Lixin Gao, Jim Kurose, Prashant J. Shenoy, and Don Towsley. Periodic broadcast and patching services - implementation, measurement, and analysis in an internet streaming video testbed. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 280–290, Ottawa, Canada, September 2001. ACM Press.

[10] Jean-Paul Nussbaumer, Baiju V. Patel, Frank Schaffa, and James P. G. Sterbenz. Networking requirements for interactive video on demand. *IEEE Journal of Selected Areas in Communications*, 13(5):779–787, 1995.

[11] DAVIC 1.4.1 Specification. Part 4: Delivery system architecture and interfaces, June 1999.

[12] Renu Tewari. *Architectures and Algorithms for Scalable Wide-area Information Systems*. PhD thesis, University of Texas, Austin, TX, USA, August 1998.

[13] S.-H. Gary Chan and Fourad A. Tobagi. Distributed Server Architectures for Networked Video Services. *IEEE/ACM Transactions on Networking*, 9(2):125–136, April 2001.

[14] Subhabrata Sen, Jennifer Rexford, and Don Towsley. Proxy Prefix Caxching for Multimedia Streams. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1310–1319, New York, NY, USA, March 1999. IEEE Press.

[15] Zhi-Li Zhang, Yuewei Wang, David H.C. Du, and Dongli Su. Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area networks. *IEEE/ACM Transactions on Networking*, 8(4):429–442, 2000.

[16] Shantanu Paknikar, Mohan Kankanhalli, K. R. Ramakrishnan, S. H. Srinivasan, and Lek Heng Ngoh. A caching and streaming framework for multimedia. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 13–20, Marina del Rey, CA, USA, October 2000.

[17] Reza Rejaie, Haobo Yu, Mark Handley, and Deborah Estrin. Multimedia proxy caching for quality adaptive streaming applications in the internet. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 980–989, Tel-Aviv, Israel, March 2000.

[18] Carsten Griwodz, Michael Zink, Michael Liepert, Giwon On, and Ralf Steinmetz. Multicast for savings in cache-based video distribution. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, San Jose, CA, USA, January 2000.

[19] Derek L. Eager, Michael C. Ferris, and Mary K. Vernon. Optimized regional caching for on-demand data delivery. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, pages 301–316, San Jose, CA, USA, January 1999.

[20] Carsten Griwodz, Michael Liepert, Michael Zink, and Ralf Steinmetz. Tune to lambda patching. *ACM Performance Evaluation Review*, 27(4):202–206, March 2000.

[21] Lixin Gao and Don Towsley. Supplying instantaneous video-on-demand services using controlled multicast. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS)*, volume 2, pages 117–121, Florence, Italy, March 1999.

[22] Derek Eager, Mary Vernon, and John Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. In *Proceedings of the International Workshop on Multimedia Information Systems (MIS)*, Indian Wells, CA, USA, October 1999.

[23] Bing Wang, Subhabrata Sen, Micah Adler, and Don Towsley. Proxy-based Distribution of Streaming Video over Unicast/Multicast Connections. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, New York, NY, USA, June 2002. IEEE Press.

[24] Sridhar Ramesh, Injong Rhee, and Katherine Guo. Multicast with cache (mcache): An adaptive zero-delay video-on-demand service. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Anchorage, Alaska, USA, April 2001.

[25] Ann Chervenak. *Tertiary Storage: An Evaluation of New Applications*. PhD thesis, University of California at Berkeley, CA, USA, December 1994.

[26] Scott A. Barnett, G. J. Anido, and H. Beadle. Caching policies in a distributed video on-demand system. In *Proceedings of the Australian Telecommunication Networks and Applications Conference*, Sydney, Australia, 1995.

[27] Carsten Griwodz, Michael Br, and Lars C. Wolf. Long-term movie popularity in video-on-demand systems. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 340–357, Seattle, WA, USA, November 1997.

[28] Carsten Griwodz. *Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure*. PhD thesis, Darmstadt University of Technology, Darmstadt, Germany, April 2000.

[29] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. *ACM Computer Communication Review*, 26(4):117–130, August 1996.

APPENDIX

A. Numbers of nodes

This number of nodes in the distribution hierarchy shown in figure 4 can differ from one level to another. For simpler handling in the cost computations, we introduce the total number of clients in the system k in equation 1 and, more generally, the number of nodes $q_{i,j}$ at level i that have a common parent node at level j in eq. 2, which in turn allows us to express k as in eq. 3.

$$k = k_0 * k_1 * ... * k_d \tag{1}$$

$$q_{i,j} = \prod_{l=i}^{j-1} k_l \tag{2}$$

$$k = q_{0,l} * q_{l,d} \tag{3}$$

B. The Zipf distribution

The Zipf distribution is shown in eq. 4, where η_i is the probability of choosing movie m_i from a set $m_1, m_2, ..., m_M$ of M movies that are sorted by their popularity (of course, this implies $\sum_{i=1}^{M} \eta_i = 1$). We call this probability also the popularity of m_i. The parameter ζ is a skew factor for the distribution.

$$\eta_i = \frac{C}{i^\zeta}\ ,\ C = 1/\left(\sum_{i=1}^{M} \frac{1}{i^\zeta}\right) \tag{4}$$

C. Specific cost equations

In the following sections we develop the cost computation equations for several distribution techniques. Each of the approaches applies a combination of unicast and multicast delivery of sections of movies to reduce the server and network resource consumption. This reduction can be exploited in dimensioning network links and servers in a distribution system, which is reflected in the cost computations.

D. λ-patching

λ-patching has been briefly introduced in Section II. Here we provide a cost computation for λ-patching in combination with placement of the movie on a root server that is specific to each movie m_i. To calculate the multicast link cost that is generated at each level for each movie, we assume a random distribution of the clients that share a stream of movie m_i in the overall set of clients. With this we determine the probability that a multicast stream uses a network link [1].

The expected number of links that receive m_i at level l is shown in eq. 5.

$$\begin{aligned} & E(\textit{number of links at level } l \textit{ that serve } m_i) \\ = \ & (1 - P(\textit{link at level 1 does not serve } m_i)) * q_{l-1,d} \\ = \ & (1 - (1 - \eta_i)^{q_{0,l-1}}) * q_{l-1,d} \end{aligned} \tag{5}$$

All clients in such a session except the first one are late and allocate the link only part of the time. The probability that the link is required is reduced linearly for each individual client, which is taken into account in eq. 6.

$$q_{l-1,d} * N_l * \int_0^1 1 - (1 - t * \eta_i)^{q_{0,l-1}} dt \tag{6}$$

As in [20], we assume for simplicity that the multicast transmission is cyclic, i.e., that it operates like an NVoD system rather than a true Patching system. We call the time interval after the start of one multicast stream and before the start of the next the *patching window*. Unicast streams that are started between these two multicast streams are said to be started in the window. In [20], we have specified the cost of patching for a movie of length L and with an interarrival time $1/\lambda$ in terms of server resources. It is shown in eq. 7 and consists of one multicast stream setup cost S_M per patching window ω, the unicast setup cost S_U for requests that arrive according to the frequency λ, the multicast stream cost Γ_M for the entire length of the movie, and the unicast stream cost Γ_U for the patch streams that are started per patching window.

$$\textit{Cost}_{\lambda\textit{-patching}} = \frac{S_M}{\omega} + S_U * \lambda + \Gamma_M * \frac{L}{\omega} + \Gamma_U * \frac{\omega * \lambda}{2} \tag{7}$$

In this paper, we can not assume that Γ_M and Γ_U are constant, since we examine the network costs, but we assume that $S_M = 0$, $S_U = 0$ and $L = 1$. To optimize ω, we have to solve eq. 8.

$$0 = \left(\frac{d}{d\omega}\Gamma_M\right) * \frac{1}{\omega} - \frac{\Gamma_M}{\omega^2} + \frac{\Gamma_U * \lambda}{2} \tag{8}$$

Γ_U is shown in eq. 9.

$$\Gamma_U = \eta_i * \sum j = 1 l N_j \tag{9}$$

It expresses the unshared cost of the link that connects a client with a server at level l, qualified with the request probability. The definition of Γ_M in eq. 10 is based on eq. 6.

[1] This is independent of λ. Whenever a client requests a movie, it will become part of a multicast. It depends on λ how long this multicast is.

$$\Gamma_M = \sum_{j=1}^{l} \left(q_{j-1,l} * N_j * \int_0^1 1-(1-t*\eta_i)^{q_{0,j-1}} dt \right) \quad (10)$$

Since the derivation of Γ_M in ω depends recursively on ω, we replace the definition with a slightly overestimated value $\widehat{\Gamma_M}$, which is given in eq. 11.

$$\widehat{\Gamma_M} = \sum_{j=1}^{l} (q_{j,l} * N_j * (1-(1-\eta_i)^{q_{0,j-1}})) \quad (11)$$

The estimation considers the number of links that are involved in the multicast at the end of the interval ω instead of the average during the interval. The definition $L = 1$ determines time, so we can define the request rate according to eq. 12.

$$\lambda = \eta_i * q_{0,l-1} \quad (12)$$

With the ω-independent definition of $\widehat{\Gamma_M}$, we can use eq. 8 to compute ω depending on a level l and a movie m_i. The result, which is limited by the movie length $L = 1$, is shown in eq. 13.

$$\omega = min\left(\sqrt{\frac{2 * \widehat{\Gamma_M}}{\lambda * \Gamma_U}}, 1 \right) \quad (13)$$

For further use, it is convenient to use a fraction $r_{i,l}$ of the movie length as defined in eq. 14 instead of the window size.

$$r_{i,l} = \frac{L}{\omega} = max\left(\sqrt{\frac{\lambda * \Gamma_U}{2 * \widehat{\Gamma_M}}}, 1 \right) \quad (14)$$

Note that $r_{i,l}$ can be interpreted as the number of patching windows that fit into one movie length. We define the probability $\mu_{i,l}$ of a request in a window in eq. 15.

$$\mu_{i,l} = \frac{1}{r_{i,l}} * \eta_i \quad (15)$$

These definitions allow cost computations. Unicast patches must be distributed to the clients. They require a direct transmission from the root server to the end-user, which is on average $1/2$ of the maximum patch length ω. For m_i, this results in unicast stream costs at level l and unicast interface costs at the root server as shown in eq. 16 and 17, respectively.

$$\frac{1}{2}(\mu_{i,l} k N_l) \quad (16)$$

$$\frac{1}{2}(\mu_{i,l} k S) \quad (17)$$

For all movies and levels, this adds up to eq. 18 and 19, respectively.

$$C_N^u = \sum_{i=1}^{M} \left(\frac{k}{2} * \mu_{i,l_i} * \sum_{j=1}^{l_i} N_j \right) \quad (18)$$

$$C_S^u = \frac{k}{2} * S * \sum_{i=1}^{M} \mu_{i,l_i} \quad (19)$$

The interface that supports the multicast sessions at the root server of m_i is needed for the entire duration of the multicast. We have to consider that the join probability is lower than η_i for $r_{i,l} < 1$ but more multicast streams are started. Thus, the usage probability of the interface depends only on the access probability of each m_i, as given in eq. 5, the number of restarts and the window size ω, which is a function of $r_{i,l}$. This results in the multicast interface cost given in eq. 20.

$$C_S^m = \sum_{i=1}^{M} [q_{l_i,d} * r_{i,l_i} * (1-(1-\mu_{i,l_i})^{q_{0,l_i}}) * S] \quad (20)$$

The computation of the multicast link cost is less straightforward because the number of links that are involved in a multicast transmission increases the first $1/r_{i,l}$ fraction of the movie, and remains stable for the remaining $1-1/r_{i,l}$. This is the case for all concurrently active multicast streams, yielding eq. 21 for each level j and movie m_i with a root server at level l_i.

$$\begin{aligned} & r_{i,l_i} \cdot q_{j-1,l_i} \cdot \left(\frac{N_j}{r_{i,l_i}} \cdot \int_0^1 1-(1-t\cdot\mu_{i,l_i})^{q_{0,j-1}} dt \right. \\ & \left. + \left(N_j - \frac{N_j}{r_{i,l_i}} \right) \cdot (1-(1-\mu_{i,l_i})^{q_{0,j-1}}) \right) \end{aligned} \quad (21)$$

Using this, the overall multicast link cost can be computed according to eq. 22.

$$\begin{aligned} C_N^m = \sum_{i=1}^{M} (\sum_{j=1}^{l_i} [& q_{j-1,d} \cdot N_j \cdot (\int_0^1 1-(1-t\cdot\mu_{i,l_i})^{q_{0,j-1}} dt \\ & + (r_{i,l_i}-1) \cdot (1-(1-\mu_{i,l_i})^{q_{0,j-1}}))]) \end{aligned} \quad (22)$$

For the storage cost, we have to take the storage cost on the root server of m_i as well as at the clients into account, yielding eq. 23.

$$C_D = \left(k + \sum_{i=1}^{M} q_{l_i,d} \right) * D \quad (23)$$

E. Gleaning

For gleaning with placement, the proxy installations at level 1 of the hierarchy are always present as explained in Section II. The root server that holds the full copy of a movie m_i is determined separately for each movie. All calculations must distinguish whether the root server of a movie is a first level server or a higher level server. If the movie is stored in the first level server, there is no additional cost for the serialization of the movie for the customer. Since unicast is used over the last link, it is delivered directly from the stored copy. To reduce the complexity, we discard the buffer holding the unicast patch stream for a movie only when a new multicast stream is started.

One portion of the unicast cost is always introduced at the level 1 proxy for delivering a unicast stream to each client. This is independent of whether the movie is stored at $l = 1$ or at a different level. For $l > 1$, another unicast interface cost must be taken into account for the patch streams. Since we are holding the buffer after the first additional request, the average patch stream length is only $1-(1-\eta_i)^{k_0} * \omega$ for each patching window. The multicast interface costs are identical to eq. 20, except that the cost for the last link is not calculated. See eq. 24 for the unicast interface costs and eq. 25 for the multicast costs.

$$C_S^u = k * S + \sum_{i=2}^{M} q_{1,d} * (1-(1-\eta_i)^{k_0}) * S \tag{24}$$

$$C_S^m = \sum_{i=2}^{M} q_{l_i,d} * r_{i,l_i} * (1-(1-\mu_{i,l_i})^{q_{0,l_i}}) * S \tag{25}$$

The multicast link cost for $l_i > 1$ is derived from eq. 21, but it is 0 for level 1. The result is shown in eq. 26.

$$\begin{array}{ll} C_N^m & = \sum_{i=1}^{M} C_{l_i,N} \textit{ where} \\ C_{l_i,N}^m & = \sum_{j=2}^{l_i} [q_{j-1,d} * N_j * (\int_0^1 1-(1-t*\mu_{i,l_i})^{q_{0,j-1}} dt \\ & +(r_{i,l_i}-1) * (1-(1-\mu_{i,l_i})^{q_{0,j-1}}))] \end{array} \tag{26}$$

The unicast link cost is similar to the unicast interface cost in eq. 24, except that the movie-dependent length of the path between root server and proxy must be considered rather than the root server's interface. The result is shown in eq. 27.

$$C_N^u = k*N_1 + \sum_{i=2}^{M} \left(q_{1,d} * (1-(1-\eta_i)^{k_0}) * \sum_{j=2}^{l_i} N_j \right) \tag{27}$$

In case $l_i > 1$, storage must be allocated both in the root server of m_i and in the proxy. In the root server of m_i, the entire movie must be stored, while the length of the longest requested patch stream must be allocated at the proxy if at least one client requests the movie. Rather than assuming the worst case situation that always the entire patch must be kept, only the patch length for the average latest patch request in an interval must be considered, expressed by eq. 28.

$$q_{l,d} \cdot D + \int_o^1 1-(1-t\cdot\mu_{i,l})^{q_{0,1}} dt \cdot \frac{D}{r_{i,l}} \cdot q_{1,d} \tag{28}$$

The storage cost is not affected by whether the full patch is transmitted from the root server to the proxy or only the missing patch; the decision to overwrite the previous patch is independent of it. With this approach, the storage cost is as in eq. 29.

$$\begin{array}{lll} C_D = \sum_{i=1}^{M} C_{l_i,D} \textit{ where} & & \\ \textit{for } l_i > 1 & C_{l_i,D} & = (q_{l_i,d} + \int_0^1 1-(1-t*\mu_{i,l_i})^{q_{0,1}} dt \\ & & *\frac{1}{r_{i,l_i}} * q_{1,d}) * D \\ & C_{1,D} & = k * D \end{array} \tag{29}$$

F. MPatch

As with gleaning, the presence of prefix caches at level 1 is mandatory for mpatch. The root server, which holds the suffix, is chosen independently for each movie m_i. For m_i, the portion that is stored at level 1, v_i, and the patching window size T_i, can be chosen as well. When a request arrives, it is served from the proxy using multicast. After time v_i, the root server sends a unicast stream to the prefix cache, which reflects it to its clients as multicast. Furthermore, a restart-threshold window of length T_i is defined. If a new client arrives less than T_i after the start of a transmission, it receives a unicast patch stream. If it arrives later, a new multicast is initiated. In case that the patch stream is used, and $T_i > v_i$, the portion of the patch that is not stored on the prefix cache is transmitted from the root server of m_i. The costs are computed in several steps.

If we assume $l_i = 1$, $v_i = 1$ is implied and the costs can be computed according to eq. 30, 31, 32, 33 and 34.

$$C_{1,N}^u(i) = k/2 \cdot T_i \cdot \eta_i \cdot N_1 \tag{30}$$

$$C_{1,S}^u(i) = k/2 \cdot T_i \cdot \eta_i \cdot S \tag{31}$$

$$C_{1,S}^m(i) = q_{1,d} \cdot T_i \cdot (1-(1-T_i\cdot\eta_i)^{q_{0,1}}) \cdot S \tag{32}$$

$$C_{1,N}^m(i) = k \cdot N_1 \cdot T_i \cdot T_i \cdot \eta_i \tag{33}$$

$$C_{1,D}(i) = q_{1,d} \cdot D \tag{34}$$

If we assume $l > 1$, we know $v_i < 1$. Since we know that the patching window size is T_i, we can use the patching equations. For this we define $r_{i,l}$ and $\mu_{i,l}$ as shown in eq. 35 and eq. 36, respectively.

$$r_{i,l} = L/T_i \tag{35}$$

$$\mu_{i,l} = T_i/L \cdot \eta_i = T_i \cdot \eta_i \tag{36}$$

Two probabilities are needed in several cost computations. The probability that a movie is requested from a prefix proxy at level 1 during a single window length T_i is computed in eq. 37.

$$H_i = 1-(1-T_i\cdot\eta_i)^{q_{0,1}} \tag{37}$$

The probability that part of a patch stream that is requested for the case $T_i > v_i$ requires part of its unicast from the root server of m_i which serves the tail is computed for one window length T_i in eq. 38.

$$G_i = 1-(1-T_i\cdot\eta_i + v_i\cdot\eta_i)^{q_{0,1}} \tag{38}$$

The storage space required depends only on v_i and is computed as in eq. 39.

$$C_{l,D}(i) = (q_{l_i,d}\cdot(1-v_i) + q_{1,d} * v_i) \cdot D \tag{39}$$

The length of the multicast streams depends only on T_i but interface costs for multicast occur only at the proxy since unicast is used for delivery from the root server of m_i. Thus, eq. 40 can be derived from eq. 20.

$$C^m_{1,S}(i) = q_{l_i,d} \cdot 1/T_i \cdot H_i \cdot S \tag{40}$$

Since we use multicast only between the proxy and the clients, the multicast cost can be computed as in eq. 21, with a window size T_i and simplified as shown in eq. 41.

$$C^m_{1,N}(i) = k \cdot \eta_i \cdot \left(\frac{T_i}{2} + (1 - T_i)\right) \cdot N_j \tag{41}$$

In all cases we know the storage cost. The movie is partially stored at level 1 and partially at level l. We know also that the multicast cost is identical to that in the $l = 1$ case, because multicast is only performed from the proxy to the clients.

With $T_i < v_i$, the unicast link cost consists of the cost for transmitting the suffix to the proxy that has to be repeated every T_i interval at every server at level l, and the patches that are delivered from the proxy and have an average length $T_i/2$. It is shown in eq. 42.

$$C^u_{l,N}(i) = q_{1,d} \cdot H_i \cdot \frac{1 - v_i}{T_i} \cdot \sum_{j=2}^{l} N_j + k \cdot \eta_i \cdot \frac{T_i}{2} \cdot N_1 \tag{42}$$

The interface cost is similar, see eq. 43. The multicast costs are identical to those in patching from level 1.

$$C^u_{l,S}(i) = q_{1,d} \cdot H_i \cdot \frac{1 - v_i}{T_i} \cdot S + k \cdot \eta_i \cdot \frac{T_i}{2} \cdot S \tag{43}$$

For $T_i > v_i$, we have to add unicast cost between the proxy and the server at level l, that occurs for each client. This results in the costs shown in eq. 44 and 45, respectively.

$$\begin{aligned} C^u_{l,N}(i) = \quad & q_{1,d} \cdot H_i \cdot \tfrac{1-v_i}{T_i} \cdot \textstyle\sum_{j=2}^{l} N_j \\ + \quad & q_{1,d} \cdot G_i \cdot \tfrac{T_i - v_i}{2} \cdot \textstyle\sum_{j=2}^{l} N_j + k \cdot \eta_i \cdot \tfrac{T_i}{2} \cdot N_1 \end{aligned} \tag{44}$$

$$\begin{aligned} C^u_{l,S}(i) = \quad & q_{1,d} \cdot H_i \cdot \tfrac{1-v_i}{T_i} \cdot S \\ + \quad & q_{1,d} \cdot G_i \cdot \tfrac{T_i - v_i}{2} \cdot S + k \cdot \eta_i \cdot \tfrac{T_i}{2} \cdot S \end{aligned} \tag{45}$$