An Overlay Network for Replica Placement within a P2P VoD Network

Kan Hung Wan, Chris Loeser

Abstract— **Generic file sharing P2P applications have gained high popularity in the past few years. In particular, P2P streaming architectures have also attracted attention. Many of them consider many-to-one streaming or high-level multicast techniques. In contrast to these models, we propose techniques and algorithms for point-to-point streaming in autonomous systems as it might occur in large companies, a campus or even in large hotels. Our major aim is to create a replica situation that inter-subnetwork RSVP streams are reduced to a minimum. Therefore, we introduce the architecture of an overlay network for interconnecting subnetworks. Each subnetwork contains a so-called local active rendezvous server (LARS) which does not just act as directory server but also controls availability of movie content in its subnetwork. Due to this, we consider data placement strategies depending on restrictions of network bandwidth, peer capabilities, as well as the movies's access frequency.**

Index Terms— **Peer-to-Peer, Streaming, Video on Demand, Content replication, Overlay network, Data placement**

I. Introduction

IN the last few years peer-to-peer (P2P) techniques became extremely popular for file sharing applications like Gnutella, eDonkey / Overnet, Kazaa, etc. When considering P2P applications, users typically talk about the sharing of resources like computation power, instant messaging, video-conferences, and (mostly) MP3 / video files. Each peer is client and server at the same time: it offers resources and is also able to occupy other peers' resources simultaneously. However, the idea of peer-to-peer communication is not new, application layer architectures in Intra- or Internet mainly base on client server techniques: e.g. Mail-, HTTP-, FTP-, LDAP-Server.

Within the P2P area we distinguish between pure and hybrid P2P architectures. Pure P2P systems do not have any central instance. Hybrid architectures, in contrast, use at least one directory or rendezvous server, which administrates references of all connected peers and their content. There are existing distributed P2P indexing and lookup services making use of distributed hash tables (DHTs) like Chord [1], CAN [2] or Pastry [3] which do not need any rendezvous server at all.

Though there has been much research activity in general P2P applications (e.g. content distribution, lookup services, or groupware support) there has recently been several work dealing with P2P streaming networks. The main difference between conventional P2P and P2P streaming applications is the instant time when content is used. Considering the first item, content is completely transferred before files are opened. In contrast to P2P streaming where content is decoded, personated immediately and afterwards (probably) is discarded.

Regarding P2P streaming architectures there are currently two major approaches. The first one proposes a high-level multicast technique mostly creating multicast trees with a single source streaming (mostly live) media over the network. The second approach introduces many-to-one streaming. Here streaming content is segmented and a single target is served by several sources (proxy peers).

In our P2P streaming scenario we consider large autonomous systems with many layer-2 subnetworks. Each peer maintains a set of movie content. This kind of scenario may be found in large companies, large hotels with set-top-boxes on each room, or a campus with e-learning movie content. We show how to eliminate the need of one (or a few) central movie streaming server to reduce hardware costs by creating a distributed streaming network.

The following architecture bases on the following simple assumption: *costs for local streaming* $<$ *costs for streaming/copy within layer-2 subnetwork (Gigabit ethernet)* $\ll$ *costs for RSVP based streaming over the backbone network.*

Thereby we presume the streaming over the backbone network to be very expensive. Thus we perform an sophisticated data placement strategy which creates replicas due to their individual access frequency. The set of movies may have a high variance of size and demand frequency. Regarding the peers, we presume different storage capacities, gigabit ethernet connections within a subnetwork, and different capabilities of simultaneous streams. Each peer offers so-called *service slots* (also named streaming slots). These may be used for content replication or movie streaming.

Each peer can become the (local) active rendezvous server for its LAN. This, so called LARS, does not only act as directory server, additionally it is aware of all other LAR server distributed in the whole network. It also controls availability of movie content and initiates content replication.

In this proposal we analyzed the demands of a P2P VoD network and adapted known solutions and algorithms from theoretical computer science. Some of these algorithms might have a high WC execution time. However, we have a limited amount of peers and movies. Thereby we can optimize the supply and demand scenario. Moreover, we do not consider content segmentation: When using RSVP/MPLS within the autonomous system, changing source peers too often may cause unpredictable problems for the media stream.

Manuscript received May 15, 2004; revised September 28, 2004.

Kan Hung Wan is with the Bertelsmann Inc., Germany. Email: kan_hung_wan@t-online.de

Chris Loeser is with the Cooperative Computing and Communication Laboratory (C-LAB), Germany. Email: loeser@c-lab.de

The outline of this paper is as follows: In the next section we describe related work, followed by the basic architecture of the overlay network. In the next sections, we describe the individual algorithms (cache algorithm, replication algorithm and peer selection algorithm). The following section deals with simulation results and within the last chapter, we conclude and give an outlook on future work.

II. Related Work

Within this chapter we give an overview of existing architectures and techniques. There has been a lot of work in general P2P networks, streaming network architectures, Video on Demand, multicast/broadcast algorithms, and quality of service techniques. However, these papers differ in several points from our architecture or they concentrate on just one to a few of these topics.

When considering some architectures of video networks there are quite a lot covering hierarchal structures in WAN networks: In [4] authors propose a central office (CO) that delivers movie content to the end-user. [5] suggests an architecture with several backup-servers, which deliver their content to local-cache-servers that are located in the middle of this hierarchy. They offer the videos with QoS support to the end-user. Also [6] proposes a central instance, a so-called video warehouse, which provides video content to intermediate storages which are located closer to the end-user. Here are scheduling strategies for video-on-reservation applications considered. Also [7] uses a hierarchical structure and uses besides several video-root- and proxy-server also one central directory server. The directory server administrates just references on the video content and is hence not an active part of the content replication and distribution.

A lot of work has been done in the past few years in generic P2P area: On the one hand there are P2P storage projects like e.g. CFS [8], Oceanstore [9] or PAST, on the other hand there are projects for indexing distributed content (some with efficient high level routing) like CAN [2], Pastry citerowstron01scalable [10], Chord [1], and Tapestry [11]. The aim of these projects is to eliminate the need of any central instance. These kinds of indexing problems can also be found outside the P2P world.

Lately there are also several proposals referring to P2P streaming: [12] introduces SpreadIt to build a multicast tree and to ensure good Quality of Service (not on IP-level). Also [13] proposes algorithms for the creation of a multicast tree (Zigzag). Using these multicast trees peers receive a media stream (from a unique root source or from other peers) and relay it to other peers. A very interesting kind of overlay architecture is proposed in [27] which consists of a hierarchy of layered deBruijn exchange (DXN) networks with additional edges. Nodes with a high bandwidth connection are located in higher levels of the hierarchy. This overlay is also suitable for high level multicast streaming.

[14] proposes an architecture of heterogenous peers with ”many to one” streaming. Moreover the authors say that peers do not act as servers. Also [15] uses multiple senders with just one receiver. However, this architecture is more a client-server system. In [22] authors recommend to use peers as proxy instances. Content that has been received lately is offered to all other peers. Additionally the content is segmented.

The resource-mapping problem within a distributed, replica containing/creating storage environment is considered in [16]. Here the current workload of different nodes is taken into account when replicas shall be created. However, the authors just describe the problem and do not give a real solution.

[17] proposes a replication algorithm for generic content distribution networks (CDN). When creating replicas the algorithm builds up a dissemination tree. They distinguish between the original source, a replica, and a cache instance.

In [18], [19] authors present a P2P QoS-aware replication service of distributed movie content for teaching purposes in a network. The communication bases on multicast. This has direct impact on the scalability of the architecture. To avoid excessive use of multicast-addresses authors propose to organize IP-addresses to smaller sub-groups.

III. Overlay Architecture

The overlay network is used as a virtual network layer which is usually implemented inside the application layer. Thus it is independent of the underlying network layers. The node connections in the overlay are composed of point-to-point connections (see Fig. 1). The service to establish and close a point-to-point connection is provided by the underlying routing- or network-layer.

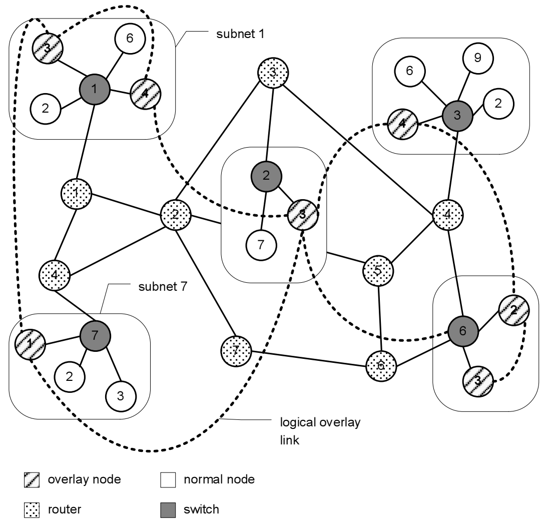


Fig. 1. Overlay network connecting several LAR-server.

The requirements for an overlay network differ depending on the field of application. For our P2P streaming network following requirements are relevant:

- **Self organization:** The network uses distributed algorithms to build up the overlay decentralized without the help of central network administration.
- **Locality awareness:** Locality awareness can significant improve the high level routing and information exchange in the application layer.
- **Fault tolerance & robustness:** If one or more nodes in the overlay network fail, the overlay has still to function

accurate. Failures have to be rapidly recognized and corrected. If nodes fail, logical connections which are incident to the nodes also fail. Thus the overlay has to seek for alternative connections.

- **Scalability & adaption:** The overlay network should endue of good scalability related to the number of nodes in the overlay. Nodes can join and leave anytime. Thus it has to be able to adapt to changes rapidly.

For better scalability of the overlay network a hierarchical approach is used. The tree just consist of two levels. The first level groups all nodes (peers) within a subnetwork to a cluster. Upon the peers one is chosen to be the *local active rendezvous server* (LARS).

All nodes in a cluster are sending their demand requests to the LARS in their subnetwork. Fig. 2 illustrates the logical hierarchical overlay connections. The darker nodes (LARS) and edges represent the higher level of the hierarchy structure, while the white nodes (peers) are building the first hierarchical level in the overlay.

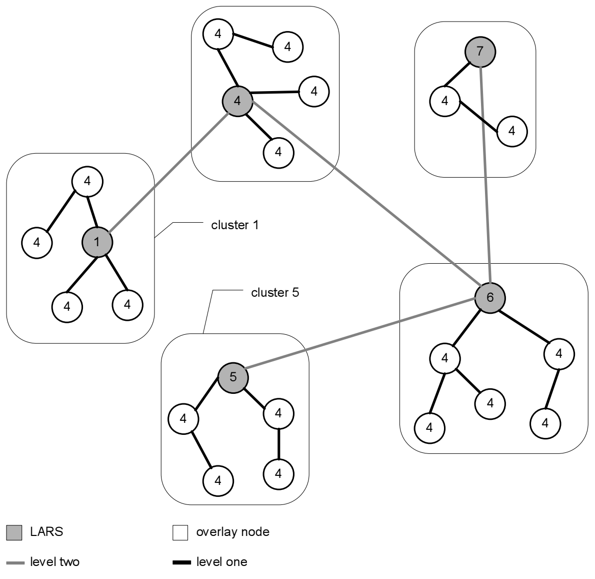


Fig. 2. Hierarchical overlay architecture.

Due to the introduction of several centralized instances the overlay network is more robust regarding node failures. Each peer can become the LARS. For that we use a modified Bully Algorithm.

Each LARS is responsible to have enough replicas of a movie in its own subnetwork. Due to the architecture the LARS is not aware of network load / demands of other subnetworks.

IV. Cache Algorithm

Fig. 3 illustrates the interconnection of the algorithms which are introduced in the following chapters. As mentioned before each subnetwork contains a dedicated peer which acts as the local active rendezvous server (LARS). It is managing all important information of the stored content of subnetwork s_i, e.g. where which content is stored, access frequency or peers, and (remaining) hard disk capacities. The cache algorithm is

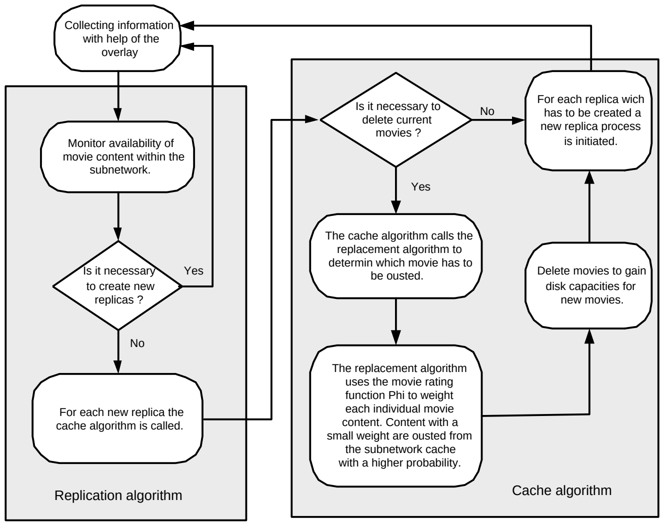

Collecting information with help of the overlay
Monitor availability of movie content within the subnetwork.
Is it necessary to create new replicas ?
Yes
No
For each new replica the cache algorithm is called.
Replication algorithm
Is it necessary to delete current movies ?
No
Yes
For each replica wich has to be created a new replica process is initiated.
The cache algorithm calls the replacement algorithm to determin which movie has to be ousted.
Delete movies to gain disk capacities for new movies.
The replacement algorithm uses the movie rating function Phi to weight each individual movie content. Content with a small weight are ousted from the subnetwork cache with a higher probability.
Cache algorithm

Fig. 3. Algorithm overview.

started each time when a new movie has to be stored in the network.

The cache of a subnetwork s_i is defined as the sum of all peers capacity $storage_{avail}(s_i) := \sum_{p\in P_{s_i,t}} storage(p)$, where $P_{s_i,t}$ is the set of peers in s_i at time t. When a new film has to be stored into s_i the cache algorithm determines whether and where it is possible to store film f_k with $size(f_k)$. Therefore it is necessary to find a peer which can provide both enough storage capacity and sufficient capacity for replication. If no peer is able to provide enough capacity the cache algorithm has to preempt one or more movie content by starting the cache-replacement algorithm.

Cache algorithm (subnetwork s_i, time t, film f_k)

1. Determine the size $size(f_k)$ of the new movie content f_k.
2. Let $L_{s_i,t}$ be the set of possible candidates (peers) within subnet s_i which are able to store the film f_k at point of time t . $P_{s_i,t}$ is the set of all peers within subnet s_i and $F_{p,t}$ is the set of all stored movies in peer p at time t. Thus $L_{s_i,t} := \{p|p \in P_{s_i,t} : f_k \notin F_{p,t}\}$.
3. $l_{max} := \max_{p\in L_{s_i,t}}(\Phi(p, f_k))$ is the maximal weight of all peers of the set $L_{s_i,t}$ The function Φ is defined as follows:

$$\begin{aligned}\Phi(p,f_k) \quad := \quad & w_3\cdot\min\left(\left\lfloor\frac{storage_{free}(p)}{size(f_k)}\right\rfloor,1\right)+\\ & w_2\cdot\min\left(\left\lfloor\frac{bw_{free}(p)}{bw_{min}^{repl}}\right\rfloor,1\right)+\\ & w_1\cdot avail(p)+\\ & w_0\cdot\frac{storage_{free}(p)}{size(f_k)}\end{aligned}$$

   w_0, w_1, w_2 und w_3 are constants with $w_3 \gg w_2 \gg w_1 \gg w_0$. The function $storage_{free}(p)$ determines the set of free disk space in peer p. $avail(p)$ delivers the availability of a peer.
4. **if** $(l_{max} > w_3 + w_2)$ **then begin**
5.   Store the movie f_k on the local hard disk of peer p_{max} for which is: $l_{max} = \Phi(p_{max})$.
6. **else if** $(l_{max} > w_3)$ **then begin**
7.   Place movie f_k in peers p_{max} queue.

```
 8. else
 9.     Start Cache-Replacement algorithm with
        parameters($s_i, t, f_k, \varepsilon$).
10. end
```

In step #2 the algorithm just considers peers which haven't this movie yet. Afterwards all other peers are rated with the $\Phi(p, f_k)$ function. This is done by weighting the following set of rules: The first addend checks whether the peer has enough storage capacity. The second addend verifies that the peer possesses a minimum of bandwidth capacity available to start a replication process. The last two addends just weight the peer due to its availability (e.g. 99.99 % uptime) and the storage quotient. Thus peers with a large amount of space are preferred.

The constants w_i determine the priority of the individual addends/criteria. Thereby it is important that $w_3 \gg w_2 \gg w_1 \gg w_0$, e.g. $w_3 = 10^8, w_2 = 10^4, w_1 = 10^2$ and $w_0 = 10$

A. Replacement Algorithm

The replacement algorithm is started when no storage capacity is left in the whole subnetwork. Therefore all peers of the specific subnet are iterated sequently because other movie content has to be ousted. Let $A_i \subseteq F_{p_i,t}$ be the set of movie content which has to be deleted in peer p_i. Then the following inequation has to be fulfilled: $\sum_{f \in A_i} size(f) + storage_{free}(p_i) \geq size(f_k)$. The set of movies which have to be deleted is determined by the movie rating function ϕ. The higher a movie is rated the higher is the probability that the movie will not be deleted. Moreover the equation $\phi(A_i) := \sum_{f \in A_i} \phi(f)$ has to be minimized simultaneously. We call this the replacement problem (RPP) which is similar to the knapsack problem. It is easy to show that $RPP \in NPO$:

- with: $RPP < knapsack$
- and: $knapsack > RPP$

The knapsack problem can be solved with dynamic programming. DYN-knapsack uses the optimizing equation by Bellman: $F_j(\alpha) = \min\{F_{j-1}(\alpha - p) + vol(j), F_{j-1}(\alpha)\}$, where $F_j(\alpha)$ is the smallest needed knapsack volume to integrate a value of α. The DYN-knapsack takes $O(n \cdot OPT(I)) = O(n^2 \cdot P_{max})$ where P_{max} is the maximal price of the commodities. In the worst case it might be possible that DYN-knapsack takes exponential WC execution time.

To avoid this it is possible approximate the problem with a fully polynomial approximation scheme (FPAS). The result of this algorithm differs from the optimal result by a factor ϵ. The base idea of this technique reduce the prices of all commodities by the factor $1/k$. ϵ is then determined by the rounding error of the prices. This algorithm is called FPAS-knapsack [20], [21]. FPAS-knapsack can easily be adapted to the RPP: which we call FPAS-replacement. Due to space limitations this proof is left away.

For the cache-replacement algorithm each peer in the subnet has to be considered. However the algorithm just takes peers into account which currently don't store the movie. Therefore recall that $L_{s_i,t} := \{p | p \in P_{s_i,t} : f_k \notin F_{p,t}\}$ is the set of potential candidates where the movie may be placed. The algorithm calculates for each peer p_e the set σ_{p_e} of movies which could be deleted. Due to the fact that each peer has just limited service slots the costs $\phi(\sigma_{p_e})$ are weighted by the number of free service slots $\#slots^{serv}_{free}(p_e)$. Thereby peers with a larger number of free service slots are advantaged. The cache replacement algorithm is defined as follows:

```
Cache Replacement algorithm
(subnetwork $s_i$, time $t$, film $f_k$, $\varepsilon$)
1. Define $L_{s_i,t} := \{p | p \in P_{s_i,t} : f_k \notin F_{p,t}\}$
   and $g(p_e) := \max(\#slots^{serv}_{free}(p_e), \frac{1}{2})$
2. forall $p_e \in L_{s_i,t}$ do
3.     $\sigma_{p_e}$ := FPAS-replacement($p_e, t, f_k, \varepsilon$);
4. end
5. Let $\sigma_{p_{min}}$ be the solution with
   $\frac{\phi(\sigma_{p_{min}})}{g(p_{min})} = \min\left\{\frac{\phi(\sigma_{p_e})}{g(p_e)} | p_e \in L_{s_i,t}\right\}$
6. Delete all films $f_d \in \sigma_{p_{min}}$ on peer $p_{min}$
7. Place the new film $f_k$ into peer $p_{min}$
The WC execution time is $O(n^3 \cdot \frac{1}{\varepsilon} \cdot |P_{s_i,t}|)$
```

B. Movie Rating Function ϕ

To determine which movie f_k has to be deleted from the subnetwork s_i by the cache algorithm we use the movie rating function $\phi(f_k, s_i)$. Each movie $f_k \in F_{s_i,t}$ is assigned a weight $\phi(f_k, s_i)$. The higher $\phi(f_k, s_i)$ the higher is the probability for the movie to remain in the subnet. The main task for the LARS is to reduce the network load. Network load is created by RTP packets for video streaming $n_{video}(f_k, s_i)$ and replica creation streams $n_{repl}(f_k, s_i)$.

1) Streaming costs $n_{video}(f_k, s_i)$: Let $v = (p_{send}, p_{rec}, f_k, t_e) \in VS_t$ be a video stream from p_{send} to p_{rec} of film f_k which started at point of time t_e. Video streams with $p_{send}, p_{rec} \in P_{s_i,t}$ create no backbone network cost. Streams between two subnetworks $p_{send} \in P_{s_i,t}, p_{rev} \in P_{s_j,t}$ and $s_i \neq s_j$ have to be routed over the backbone network and are creating higher network costs depending on the (avg.) distance.

Moreover the access frequency has to be taken into account: Here we distinguish between internal and external accesses (see Fig. 4). Let $V_{int} := \{v | v = (p_s, p_r, f_k, t_j) \in V^k_{t-d,t} : p_r \in P_{s_i,t}\}$ be the set streams which have been demanded between time steps $(t - d)$ and t and the stream receiver peer is located in subnetwork s_i. Similar let $V_{ext} := \{v | v = (p_s, p_r, f_k, t_j) \in V^k_{t-d,t} : p_r \notin P_{s_i,t}\}$ be the set of streams where the receiver of the stream was outside subnetwork s_i. The access frequencies are then defined as $h^{int}_{k,i} := |V_{int}|$ and $h^{ext}_{k,i} := |V_{ext}|$.
The distance $dist(v)$ for a $v \in VS$ is due to the hop-count metric:

$$dist(v) := \begin{cases} d(p_{send}, p_{rec}) & p_{rec} \in P_{s_i,t}, p_{send} \notin P_{s_i,t} \\ 1 & p_{rec}, p_{send} \in P_{s_i,t} \\ 1 & p_{rec} \notin P_{s_i,t}, p_{send} \in P_{s_i,t} \end{cases}$$

Costs for streams within a subnetwork are set to 1. Also costs for streams from subnetwork s_i to s_j are defined as 1. This is done due to the fact that the same costs considered from the opposite subnetwork must not be taken into account twice.

Define $V_{out} := \{v | v = (p_s, p_r, f_k, t_j) \in V^k_{t-d,t} : p_r \in P_{s_i,t}, p_s \notin P_{s_i,t}\} \subseteq V_{int}$ and $h^{out}_{k,i} := |V_{out}|$. Then $h^{out}_{k,i}$

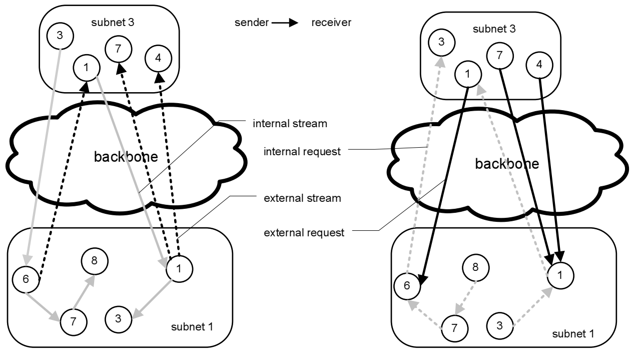


Fig. 4. View from subnet 1: Video streams (left) / Internal and external demands (right)

describes the number of demands in subnetwork s_i which could not be served through peers from the own subnetwork. This occurs when the demands resp. access frequencies are greater than the available service slots in s_i for film f_k. Let $\#slots^{serv}_{free}(f_k, s_i) := \sum_{p \in P_{s_i,t(f_k)}} \#slots^{serv}_{free}(p)$ be the number of free service slots in s_i for film f_k. The function $\#slots^{serv}_{free}(f_k, s_i)$ claims the supply for film f_k in s_i, while $h^{int}_{k,i}$ and $h^{ext}_{k,i}$ are representing the demands. If the supply is greater than the demands all requests can be served by peers of the own subnetwork. Otherwise $h^{int}_{k,i} + h^{ext}_{k,i} - \#slots^{serv}_{free}(f_k, s_i)$ many demands have to be served from other subnetworks, creating higher network costs.

The quotient of supply and demands in s_i defines the relative availability of film f_k in s_i: $v_{rel,t}(f_k, s_i) := \frac{\#slots^{serv}_{free}(f_k,s_i)}{\#slots^{serv}_{req}(f_k,s_i)}$ with $\#slots^{serv}_{req}(f_k, s_i) := h^{int}_{k,i} + \varepsilon_1 \cdot h^{ext}_{k,i}$. With help of the constant $\varepsilon_1 \in [0, 1]$ it is possible to put a stronger weight on internal accesses in contrast to external accesses (e.g. $\varepsilon_1 < 1$). If the number of free service slots is larger or equal to the demanded set of service slots the availability is thus $v_{rel,t}(f_k, s_i) \geq 1$ and $h^{out}_{k,i} = 0$. For $v_{rel,t}(f_k, s_i) \geq 1$ the streaming costs are reduced to a minimum. $n_{video}(f_k, s_i)$ with $f_k \in F_{p2p,t}$ is defined as follows:

$$\begin{aligned} n_{video}(f_k, s_i) \quad := \quad & \min(v_{rel,t}(f_k, s_i), v^*_{rel,t}(f_k, s_i)) \cdot \\ & size(f_k) \cdot h_{k,\varepsilon_1} + \\ & \max(1 - v_{rel,t}(f_k, s_i), 0) \cdot size(f_k) \cdot \\ & dist^{av}_{s_i}(f_k) \cdot h_{k,\varepsilon_1} \end{aligned}$$

with $h_{k,\varepsilon_1} := h^{int}_{k,i} + \varepsilon_1 \cdot h^{ext}_{k,i}$ and

$$v^*_{rel,t}(f_k, s_i) := \begin{cases} \frac{1}{v_{rel,t}(f_k,s_i)} & v_{rel,t}(f_k, s_i) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

The streaming costs $n_{video}(f_k, s_i)$ for film f_k consists of two parts. The first addend describes the costs by extern and intern demands which can be served from the own subnetwork s_i. The second addend determines the costs created by intern demands which could not be served from the own supply in subnetwork s_i. Whether streaming costs are created by the first or second addend depends on the relative availability $v_{rel,t}(f_k, s_i)$ of the movie in the own subnetwork. Generally it may be said that higher relative availability $v_{rel,t}(f_k, s_i)$ results in smaller streaming costs $n_{video}(f_k, s_i)$. If $v_{rel,t}(f_k, s_i) = 1$ all demands can be served by the own subnetwork. Thus there are no streaming cost created by the second addend. Visa versa there are no costs created by the first addend if $v_{rel,t}(f_k, s_i) = 0$.

If $v_{rel,t}(f_k, s_i) \in (0, 1)$ the streaming costs consists of both parts. Outlier: If $v_{rel,t}(f_k, s_i) > 1$. Then s_i has more service slots available than needed. Because of the max-function of the second addend there are no costs created here. The min-function of the first addend creates lower weighted streaming cost. $dist^{av}_{s_i}(f_k)$ describes the average distance of a video stream which has to be served from another subnetwork.

2) Replication costs $n_{repl}(f_k, s_i)$: If the number of demands is higher than the number of offered content the generation of replicas is creating further network costs. The creation of replicas is initiated by the replication algorithm as soon as the availability sinks beneath a predefined threshold. $\#repl_{s_i,t}(f_k)$ describes the number of replicas which have to be created. This function is defined in the next section. $dist^{av}_{repl}(f_k)$ names the average distance each replication stream has to pass by. The replication costs are defined as: $n_{repl}(f_k, s_i) := \#repl_{s_i,t}(f_k) \cdot [size(f_k) \cdot dist^{av}_{repl}(f_k) \cdot time(f_k)]$. Here $time(f_k)$ describes the time to create a replication.

3) Network costs $n_{net}(f_k, s_i)$: The whole network costs may not just be summarized. Moreover it is important to divide the sum by the size of the movie. Then we gain information about the costs per MByte which has to be transferred.

$$n_{net}(f_k, s_i) := \frac{n_{video}(f_k, s_i) \ + \ n_{repl}(f_k, s_i)}{size(f_k)}$$

4) Movie rating function $\phi(f_k, s_i)$: The movie f_k is rated by the network cost function if the access frequency is greater than 0. Otherwise, the access frequency is equal 0, ϕ weights f_k due to the number of replicas in the whole subnetwork.

$$\phi(f_k, s_i) := \begin{cases} n_{net}(f_k, s_i) & h^{int}_{k,i} + h^{ext}_{k,i} > 0 \\ r_{max} - |P_{s_i,t}(f_k)| + 1 & \text{otherwise} \end{cases}$$

$r_{max} := \max_{f_k \in F_{s_i,t}}(|P_{s_i,t}(f_k)|)$ determines the max number of replicas over all movies within s_i. $P_{s_i,t}(f_k)$ is the set of peers in s_i which have a replica of film f_k. Thereby movies with a large amount of replicas in the whole s_i are preferred to be deleted.

V. Replication Algorithm

The replication algorithm considers supply and demands of movie content. It becomes active if a shortage of arbitrary movie content due to high demands is recognized. The replication algorithm inside the LARS is exclusively responsible for its own subnetwork. The major task is to ensure the availability of enough replicas that all demands within the subnetwork can be served. In best case there are no demands which have to be served from another subnet. This algorithm just determines the film f_k and the point of time when a replica has to be created. The decision where the new replica has to be placed is still made by the cache algorithm.

The absolute availability of film f_k in subnet s_i is calculated by dividing the number of free service slots by the number of all available service slots:

$$v_{s_i}^{abs}(f_k) := \frac{\#slots_{free}^{serv}(f_k, s_i)}{\#slots_{avail}^{serv}(f_k, s_i)}$$

If all service slots $\#slots_{avail}^{serv}(f_k, s_i)$ are engaged the availability is set to 0. It is possible to define a threshold $l_k \in [0..1]$ for each f_k. The number of necessary replicas which have to be created is appointed by the following function:

$$\#repl_{s_i,t}(f_k) :=$$

$$\left\lceil \frac{(\#slots_{avail}^{serv}(f_k,s_i) \cdot l_k) - \#slots_{free}^{serv}(f_k,s_i)}{sl_{av} \cdot (1-l_k)} \right\rceil$$

This equation is gained from the following inequation. sl_{av} describes the average number of available service slots of a peer. The algorithm presumes that in the best case a new replica creates sl_{av} further new service slots. r is the number of required replicas we are searching for (i.e. $\#repl_{s_i,t}(f_k)$).

$$\frac{\#slots_{free}^{serv}(f_k, s_i) + sl_{av} \cdot r}{\#slots_{avail}^{serv}(f_k, s_i) + sl_{av} \cdot r} \geq l_k$$

$F_{s_i,t} := \{F_{p,t} | p \in P_{s_i,t}\}$ is the set of all available films within subnet s_i. Moreover the availability $v_{s_i}^{abs}(f)$ is determined for each $f \in F_{s_i,t}$. However there is a problem regarding the number of films and the service slots considering an individual peer: Due to the fact that each peer maintains usually more than one file the situation results in a competition of the films for the service slots. With respect to the definition of available service slots the change of availability of one movie may change the availability of other films though neither supply nor demands have changed.

To reduce this competition situation for the service slots we just consider films which have recently been accessed. A film $f_k \in F_{s_i,t}$ is called *active* if the access frequency $h_{s_i,t}^{d}(f_k)$ is greater than 0 (during the past d time steps, i.e. between $(t-d)$ and t) . $A_{s_i} := \{f | f \in F_{s_i,t}$: $h_{k,i}^{int} > 0\}$ is the set of all active films in s_i. Fig. 5 illustrates the competition situation: The three figures above illustrate the bad replication situation (for movie 2). Below just active movies are considered. The replication algorithm is described as follows:

LARS replication algorithm (subnetwork s_i, time t)

```
1. Let $A_{s_i} := \{f | f \in F_{s_i,t}$: $h_{k,i}^{int} > 0\}$ be the set of active
   films within subnet $s_i$.
2. For each film $f_k \in A_{s_i}$ the absolute availability
   $v_k := v_{s_i}^{abs}(f_k)$ is determined.
3. Let $R := \{f_k | f_k \in A_{s_i} : v_k < l_k\}$ be the set of films for
   which new replicas have to be created.
4. forall $f_k \in R$ do
5.    Let $r_k := \#repl_{s_i,t}(f_k)$ be the number of replicas
      which have to be created for $f_k$.
6.    for $i := 1$ to $r_k$ do
7.       start Cache algorithm with ($s_i$, $t$, $f_k$)
         as parameters
8.    end
9. end
```

VI. Peer Selection Algorithm

In this section we describe how a peer is selected as source peer. This is done in two parts. If a target peer demands a specified movie first of all the source subnet is determined. Afterwards the source peer within this subnetwork is selected. In the following a target peer p_i in subnetwork s_i is seeking for a source peer p_j in subnetwork s_j.

A. Source-Network Selection

When a target peer requests a movie it first of all checks whether it is storing the movie itself, or whether the content is available on its own subnetwork. If none is the case, the LARS sends a not binding request to all other LARS. They reply containing the following (also not binding) information:

- The abs. availability $v_{s_j}^{abs}(f_k)$ of the req. movie in s_j.
- The workload of the service capacities. It is destined by

$$a_{serv}(s_j) = 1 - \frac{\sum_{p \in P_{s_j}} \#slots_{free}^{serv}(p)}{\sum_{p \in P_{s_j}} \#slots_{avail}^{serv}(p)}$$

- The outer network load of s_j into the backbone network. It is described by the maximal available and the currently available bandwidth. It is determined by

$$n_{load}^{out}(f_k) := \frac{bw_{use}(s_j)}{bw_{s_j}^{avail}}$$

  The constant $bw_{s_j}^{avail}$ describes the maximal available bandwidth of s_j to the backbone network. The free bandwidth capacity is

$$bw_{use}(s_j) := \sum_{v \in V_{s_j,t}^{out}} bw(v)$$

  The function $bw(v)$ delivers the necessary bandwidth for stream v. The set $V_{s_j,t}^{out}$ combines all running streams in s_j which are consuming bandwidth into the backbone.

Basing on these information the source subnetwork $s_j \in R$ is determined. R is the set of all subnetworks containing the dedicated movie on at least one of their peers. This algorithm

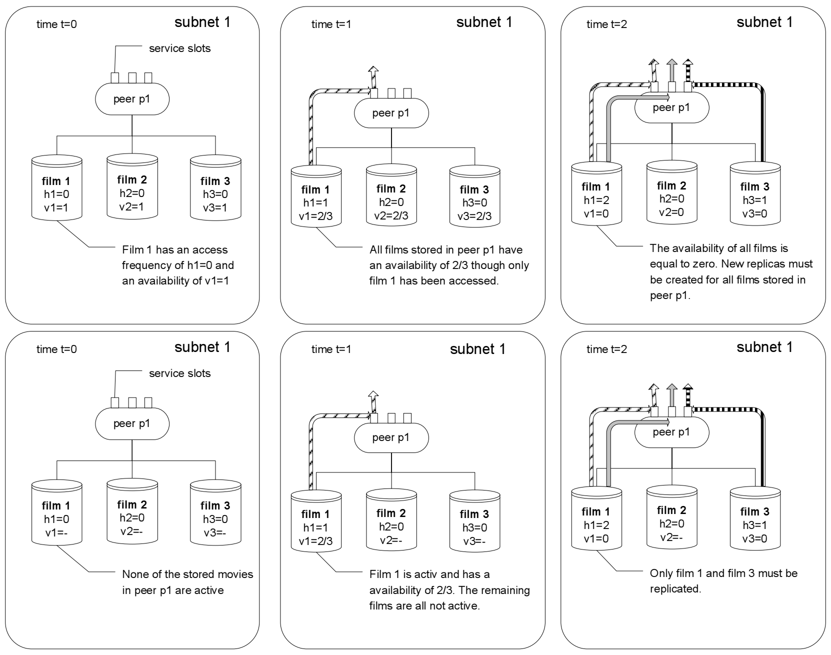

time t=0
subnet 1
service slots
peer p1
film 1 h1=0 v1=1
film 2 h2=0 v2=1
film 3 h3=0 v3=1
Film 1 has an access frequency of h1=0 and an availability of v1=1
time t=1
film 1 h1=1 v1=2/3
film 2 h2=0 v2=2/3
film 3 h3=0 v3=2/3
All films stored in peer p1 have an availability of 2/3 though only film 1 has been accessed.
time t=2
film 1 h1=2 v1=0
film 2 h2=0 v2=0
film 3 h3=1 v3=0
The availability of all films is equal to zero. New replicas must be created for all films stored in peer p1.
film 1 h1=0 v1=-
film 2 h2=0 v2=-
film 3 h3=0 v3=-
None of the stored movies in peer p1 are active
film 1 h1=1 v1=2/3
film 2 h2=0 v2=-
film 3 h3=0 v3=-
Film 1 is activ and has a availability of 2/3. The remaining films are all not active.
film 1 h1=2 v1=0
film 2 h2=0 v2=-
film 3 h3=1 v3=0
Only film 1 and film 3 must be replicated.

Fig. 5. Above: Considering all movies results in a bad availability. Below: Here just active movies are considered.

is quite similar to the cash algorithm. Thus we just describe the subnetwork rating function g:

$$\begin{aligned} g(s_j, f_k) \quad := \quad & w_4 \cdot \left\lceil v_{s_j}^{abs}(f_k) \right\rceil \cdot \\ & \min\left(\left\lfloor \frac{bw_{s_j}^{avail} - bw_{use}(s_j)}{bw(f_k)} \right\rfloor, 1 \right) + \\ & w_3 \cdot v_{s_j}^{abs}(f_k) + \\ & w_2 \cdot (1 - a_{serv}(s_j)) + \\ & w_1 \cdot (1 - n_{load}^{out}(f_k)) + \\ & w_0 \cdot (\max_{s \in R}(dist(s, s_i)) - dist(s_j, s_i)) \end{aligned}$$

The first addend ensures that just subnetworks with enough availability $v_{s_j}^{abs}(f_k)$ and outer bandwidth $n_{load}^{out}(f_k))$ are potential candidates. All other addends are weighing functions.

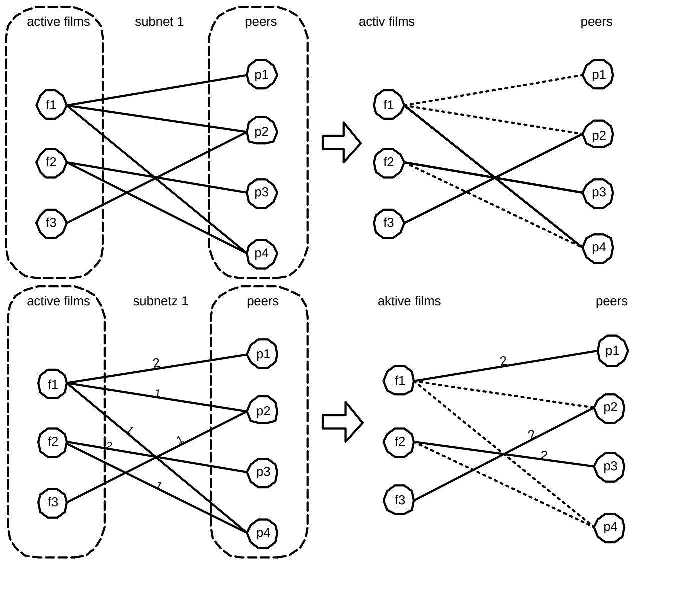


Fig. 6. Maximal (above) and maximal weighted (below) matching

B. Source-Peer Selection

Presuming that a source subnetwork has been selected it is afterwards important to select the ”right” source peer. If there are several peers containing the demanded movie an arbitrary peer containing a free service slot could be selected. However this may result in a bad availability concerning the remaining movies which are held by the selected peer. (after this selection all service slots might be in use.) There are two important criteria concerning the peer selection:

- Let $A_{s_i} := \{f | f \in F_{s_i,t}: h_{k,i}^{int} > 0\} \cup \{f_k\}$ be the set of all active films in s_j. The peer selection algorithm has to take care that after the selection all movies in A_{s_j} should be able to be served over other free service slots. Because of this criteria the competition between the movies due to the limited service slots is reduced.
- Let $P_{s_i,k} := \{p | p \in P_{s_i,t} : f_k \in F_{p,t} \wedge \#slots_{free}^{serv}(p) > 0\}$ be the set of all peers in s_j holding free service slots for film f_k. For the selection of a peer in $P_{s_j,k}$ the algorithm has to choose the source peer with the least active movies on its local hard disk.

In the following we describe an algorithm which takes both criteria into account. Therefore we transfer the problem in a bipartite graph for which we want to find a bipartite maximal matching. A matching in $G = (V, E)$ is a subgraph with maximal degree 1. A perfect matching is a matching where each node has a degree of 1. Thus we seek for a maximal weighted matching in a bipartite graph $G = (L \cup R, E)$.

Fig. 6 illustrates a scenario where four peers (right site) contain three active movies. There is an edge between a peer and the movie if the peer is holding this movie. Peers which are not having at least one active movie are left away. If we just consider the unweighed case we get a max. matching, just the first criteria is considered here. Due to this we have to integrate a weight for the edges.

The function $w(e)$ with $e = (s, u) \in E$ defines a weight for the edges. Thereby $d_{max} := \max_{p \in P_{s_i,t}}(\#film_{active}(p))$ describes the maximal number of active movies stored on a peer.

$$w(e) := 2 \cdot (d_{max} - \#film_{active}(u) + 1) + i_k(e)$$

Moreover the indication function $i_k : E \to \{0, 1\}$ is defined as:

$$i_k((s,u)) := \begin{cases} 1 & \text{if } s = f_k \\ 0 & \text{otherwise} \end{cases}$$

The weight of the edge $e = (s, u) \in E$ is just determined by the nodes $u \in R$ on the right site. The less active movies a node maintains the larger is the weight for the incident edges. When s describes the demanded film f_k all edges which are incident to s are incremented by one.

Due to the indication function all nodes which are incident regarding node f_k gain a higher weight (by 1). Thus each edge which is incident to f_k has a weight of at least 2. If a matching contains two edges with weight 1, then the preferred edge has the same weight like the other two edges. This, however, violates the first criteria. Multiplying with factor 2 in the weighting function ensures that the first criteria is not neglected.

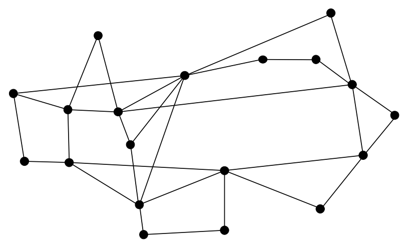
Each edge in the bipartite graph has at least a weight of 2. Two edges in the graph have a weight of at least 4, while the preferred edge has a weight of at least 3. Thus the first criteria is not violated.

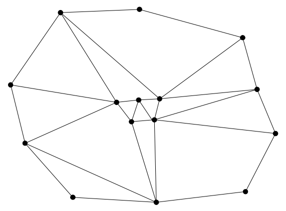
VII. Simulations

For performing simulations we have applied the P2P Network Simulation Environment from [23] (which bases on [24]). This application was designed to simulate the IP communication within autonomous systems. Similar to ns2 the Simulation Environment implements each individual layer: Besides the lower protocols it is possible to used different Interior Gateway Routing protocols (IGPs) like RIP, OSPF or DPS. Additionally it is RSVP enabled.

A. Network Topologies

Besides many other attributes the selected network topology is a major task for simulation results. We are making use of the following three practical router topologies: MCI-, Cluster- and Core-topology. The MCI topology bases on [25] and is illustrated in 7(a). The Cluster topology consists of four clusters interconnected with the other clusters by a high-bandwidth link 7(c). The Core topology consists of five core routes which are located in the middle of the network 7(b). The remaining routers are placed outside creating a ring. The


(a) MCI topology


(b) CORE topology

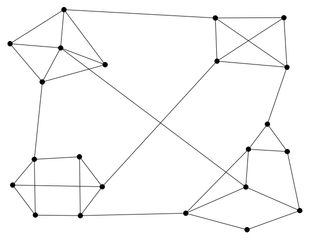

(c) CLUSTER topology

Fig. 7. Router topologies

number and size of L2 subnetworks which are interconnected to these routers are described hereafter.

Each of these topologies is used in three different variations, determined by the number and size of the subnetworks. We distinguish between *Small*, *Big* and *Mix*. The first version consists of many small subnetworks, whereas the big version consists of some larger subnetworks. The mix variance is a mixture of both smaller and larger subnetworks. A small subnetwork maintains 5-10 peers, whereas a large subnetwork has 30-60 peers. Overall there are nine different kinds of networks which are the base of the simulations. Tabular VII-A shows the attributes of the different topologies.

TABLE I
SIMULATION NETWORK TOPOLOGIES.

Network	Type	Router	Peers	Subnetworks
MCI	SMALL	19	100	14
	MIX	19	166	16
	BIG	19	296	10
CORE	SMALL	15	86	10
	MIX	15	135	7
	BIG	15	115	6
CLUSTER	SMALL	22	156	19
	MIX	22	176	11
	BIG	22	203	11

B. Access Modelling

The query model simulates the demands which are requested to the whole P2P network. It consists of a two-stage process: Within the first step the process determines *w*hether a movie is requested at time step t. This is done by a random variable. If the result of the first random experiment is positive the second step is executed. The probability for a positive result is p, for a negative result is $p-1$. The number of requests within the whole network is determined by the probability p; the number of experiments is n.

Due to the fact that the simulation period is predefined the overall number of requests just depends on the probability p. The requests are Poisson distributed. Consider the random variable $X \sim Po(\lambda)$. The constant expected value is thus $E(X) = V(X) = \lambda$. Regarding a simulation period of n time steps the expected number of overall requests is λ.

The second step generates the request at time step t. The peer requesting a movie has to be determined. This is also performed by a random experiment. In contrast to the former experiment all peers have the same probability to be ”elected”. All peers which are not playing / receiving a movie are possible candidates. If it should happen that all peers are engaged no request is performed.

To determine *w*hich movie content is requested we introduce a popularity function. Depending on the weight which is assigned to each movie the probability to be requested is calculated. The function pop models the typical lifecycle of a movie. It is consistent with the movie popularity function from [26] but has been modified slightly regarding the parameters of the curve.

$$pop(x) = \exp\left(a - \frac{b}{x} - \frac{x}{b}\right) + c$$

x describes the age of the movie and thus the popularity within the whole network. Fig. 8 illustrates a typical popularity process for different movies during the simulation: At the very beginning the popularity increases quite rapidly. With increasing time popularity reduces and converges against a constant value. The parameter a describes the gradient of the curve, whereas b determines the ”broadness” and thus the span of life of the movie. c is the mentioned remaining popularity when the movie is not active any more.

Do simulate the insertion of new movie content movies are divided into two parts: *passive* movies have a rest popularity of c. After time periods of s seconds a randomly selected movie is moved from the passive group to the *active* group and the $pop(x)$ values are determined. The parameters $a \in [0.5, 3]$ and $b \in [0.5, 3]$ are also randomly destined.

C. Results

For a Video on Demand System it is essential to ensure the ”on demand” character, i.e. we have to verify that almost all requests are answered with a minimal waiting time. In the best case there is no single request which has to be rejected. We test the replication algorithm for different network topologies and different request variations. Thereby we enable and disable the replication algorithm. The quality criteria is the number of positive answered requests. For the simulations we have created three simulation ”packets”:

1) In this simulation the requests for the movies are almost evenly distributed. This means that we have no e.g. ”blockbuster”. This scenario is not very realistic and thus is not observed here in detail.
2) Here we have just a few movies which are highly popular. Most of the movies are passive.
3) The request scenario inhere lies in-between packages one and two.

In the following we will just describe the results of packages 2 and 3 due to the fact that it is more likely to find them the real world.

Tabular VII-C illustrates the results of simulation packet 2 when the replication algorithm is disabled. The rest popularity c of each movie is set to a minimum value. Furthermore we insert one new movie within a 24 h period. The requests are distributed according to just a few movies. The time period between two requests is thus quite short and the few active movies are causing high stress. Without the replication algorithm just a few demands can be accomplished.

Fig. 9 shows the complete process of the simulation regarding one movie y within the Cluster network (replication algorithm disabled). The upper graph illustrates the requests for movie y during the simulation. The other graph describes the development of the free service slots of this film. The number of these slots collapses when the number of requests arises. Simultaneously the number of failed requests arises due to the lack of missing free service slots.

Simulation packet 3 is most likely to be found in the real world: The parameters are located in between the parameters of the first two simulation packets. Here we consider different remaining popularity values c for a broad set of passive movies. Roughly 1,5 new movies are inserted within a 24 h period (average value). The results of enabled and disable replication algorithms can be seen in tabulars VII-C and VII-C.

When the replication algorithm is switched off about 25% of all requests could be served. With enabled replication algorithm about 90% were operated positively. The replication

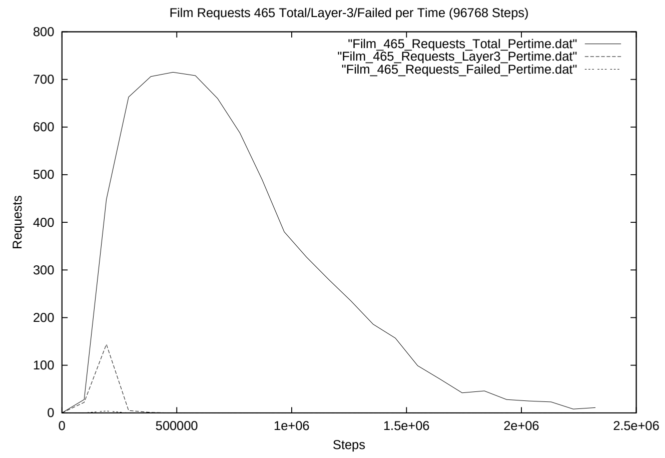


Fig. 8. Popularity values of a movie.

TABLE II
PACKET 2 ENABLED REPLICATION ALGORITHM

network	requests	positive requests	failed requests	L3 streams	replications
MCI-SMALL	35072	4283 (12,2%)	30789 (87,8%)	3204 (9,1%)	0
MCI-MIX	36332	3350 (9,2%)	32982 (90,8%)	2481 (6,8%)	0
MCI-BIG	35903	5025 (14%)	30878 (86%)	3698 (10,3%)	0
CORE-SMALL	34898	5050 (14,5%)	29848 (85,5%)	3528 (10,1%)	0
CORE-MIX	35984	3578 (9,9%)	32406 (90,1%)	2345 (6,6%)	0
CORE-BIG	36087	4410 (12,2%)	31677 (87,8%)	2553 (7,1%)	0
CLUSTER-SMALL	36213	3559 (9,8%)	32654 (92,2%)	2832 (7,8%)	0
CLUSTER-MIX	36056	4213 (11,7%)	31843 (88,3%)	3017 (8,4%)	0
CLUSTER-BIG	36064	3672 (10,2%)	32492 (89,8%)	2708 (7,5%)	0

TABLE III
PACKET 3 ENABLED REPLICATION ALGORITHM

network	requests	positive requests	failed requests	L3 streams	replications
MCI-SMALL	10023	9101 (90,8%)	922 (9,2%)	6192 (61,8%)	141
MCI-MIX	16568	14922 (90,06%)	1646 (9,94%)	8263 (49,9%)	320
MCI-BIG	29197	26020 (89,11%)	3177 (10,89%)	5731 (19,6%)	774
CORE-SMALL	8637	7859 (90,99%)	778 (9,01%)	4378 (50,7%)	148
CORE-MIX	13084	12063 (92,19%)	1021 (7,81%)	2678 (20,5%)	352
CORE-BIG	11427	10686 (93,51%)	741 (6,49%)	1867 (16,3%)	263
CLUSTER-SMALL	15553	14246 (91,59%)	1307 (8,41%)	8917 (57,3%)	215
CLUSTER-MIX	16864	15364 (91,10%)	1478 (8,9%)	4958 (21,4%)	293
CLUSTER-BIG	20493	18878 (92,11%)	1615 (7,89%)	5033 (24,6%)	368

TABLE IV
PACKET 3 DISABLED REPLICATION ALGORITHM

network	requests	positive requests	failed requests	L3 streams	replications
MCI-SMALL	32792	8708 (26,6%)	24084 (73.4%)	6614 (20,2%)	0
MCI-MIX	33959	11039 (32,5%)	22920 (67,5%)	8034 (23,7%)	0
MCI-BIG	35675	10239 (28,78%)	25436 (71,3%)	6912 (19,3%)	0
CORE-SMALL	32675	6256 (19,1%)	26419 (80,9%)	4320 (13,2%)	0
CORE-MIX	33363	8421 (25,2%)	24942 (74,8%)	4679 (14%)	0
CORE-BIG	33072	8787 (26,6%)	24285 (73,4%)	5509 (16,6%)	0
CLUSTER-SMALL	35277	7043 (20%)	28234 (80%)	5619 (15,9%)	0
CLUSTER-MIX	34469	9500 (27,6%)	24969 (72,4%)	6521 (18,9%)	0
CLUSTER-BIG	35436	9007 (25,4%)	26429 (74,6%)	6160 (17,3%)	0

algorithm was thus able to provide almost a sufficient number of replicas.

Fig. 10 illustrates the simulation development of one specific movie within the MCI network with enabled replication algorithm. The dashed line describes the number of overall requests for this movie. It was possible to serve most requests. The dashed line in graph a) illustrates the part of requests which had to be streamed over the layer 3 from one subnetwork to another. The lower dashed line exhibits the number of requests which could not be served. After a simulation period of four days (about in the middle of the graph) we have about 70 free services slots in the whole network available. Taking the number of subnetworks into account (10) this means that we have about 7 free service slots as reserve for further requests. Considering the further simulation this number arises due to a deceasing number of requests for this movie and a not changing number of replicas in the network. The number of replicas is not reduces just because the number of requests declines. Movie replicas stay on a peers disk until they are removed for another movie by the cache algorithm.

D. Comments on the Results

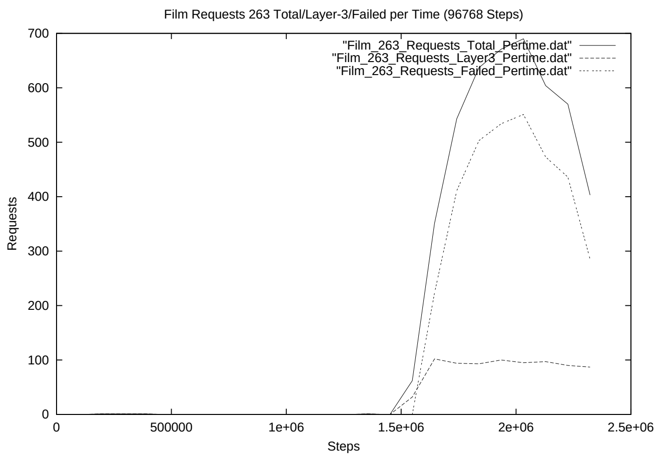
It turned out that package 1 performed too many replications which would not have been necessary. Due to the fact that demands have been distributed equally many created replicas are not accessed: The time period between two accesses regarding one movie is too large in this case.

In packages 2 and 3 more than 70% of the request could not be delivered when the replication algorithm is disabled. In contrast to package 1 there are just a few highly popular movies and many ”not so popular” content items. Because of this there was a high load for the ”main movies”. When the replication algorithm is switched off no replicas are created for package 2 and 3. Due to this many requests could not be answered.

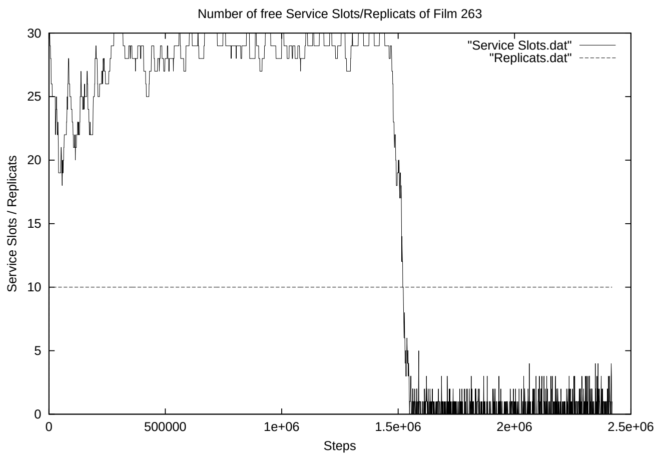
Overall the replication algorithm was able to answer ca. 90% of all requests. Nevertheless when content is not requested at a temporal locally awareness the algorithm creates too many algorithms.

VIII. CONCLUSION AND FUTURE WORK

In this paper we presented an approach for algorithms for P2P streaming networks in autonomous systems (AS). Thereby we considered data placement and source peer selection strategies which depend on the supply and demand situation. The major task is to reduce the number of RSVP based inter-subnetwork streams as they are considered to be ”expensive”. Though the proposed algorithms may have high worst case

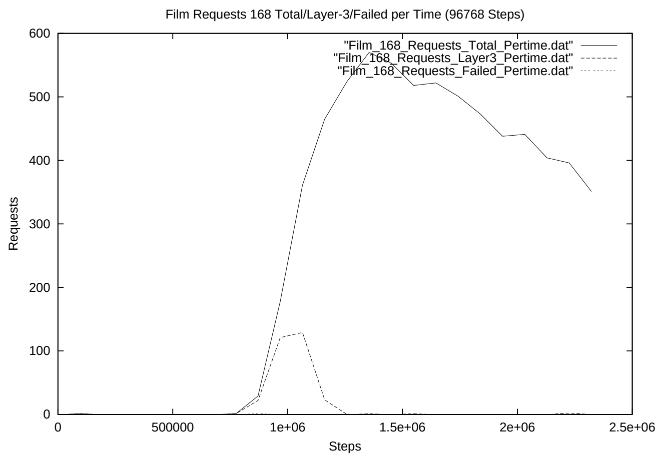


(a) Distribution of requests of an active movie.

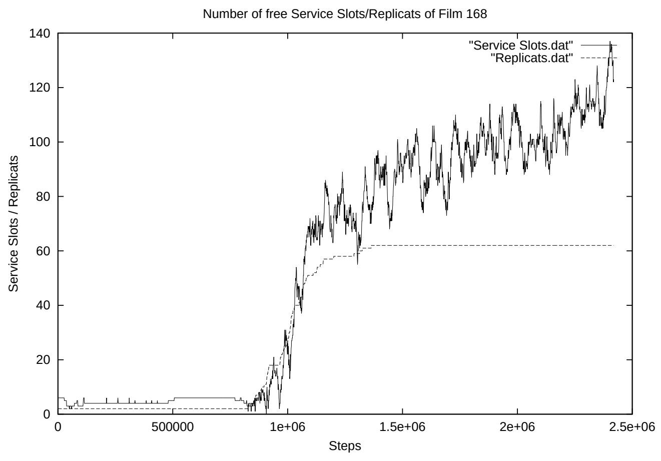


(b) Number of free service slots and available replicas of an active movie in the network.

Fig. 9. Simulation results of an active movie within the Cluster network (BIG) with disabled replication algorithm.



(a) Distribution of requests of an active movie.



(b) Number of free service slots and available replicas of an active movie in the network.

Fig. 10. Simulation results of an active movie within the MCI network (BIG) with enabled replication algorithm.

execution times in theory, they may nevertheless be used due to a relatively low number of subnetworks, peers (a few hundred all in all) and active movies.

We are currently including ARIMA timeseries analysis into the content distribution algorithms. With help of ARIMA it is possible to perform popularity forecasts for individual content and thus taking movie genre and time dependent access frequencies into account.

ACKNOWLEDGEMENT

Parts of the work described herein was funded by the German Federal Ministry of Education and Research (BMBF) within the ITEA-SIRENA project (01ISC09E). For further information visit: http://www.sirena-itea.org

REFERENCES

[1] I. Stoica and R. Morris and D. Karger and F. Kaashoek and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. of ACM SIGCOMM*, 2001
[2] S. Ratnasamy and P. Francis and M. Handley and R. Karp and S. Shenker, "A Scalable Content-Addressable Network," *Proc. of ACM SIGCOMM*, 2001
[3] A. Rowstron and P. Druschel, "Pastry: Scalable Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. of IFIP/ACM Middleware*, 2001
[4] A.D. Gelman and H. Kobrinski and L.S. Smoot and S.B. Weinstein, "A Store-and-Forward Architecture for Video-On-Demand Service," *Proc. of the International Communications Conference*, 1991, pages 27.3.1-27.3.5.
[5] Lars-Olaf Burchard and Reinhard Lüling, "An Architecture for a Scalable Video-on-Demand Server Network with Quality-of-Service Guarantees," *Proc. 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, pages 132–143, 2000, Springer
[6] Youjip Won and Jaideep Srivastava, "Strategic Replication of Video Files in a Distributed Environment," *Multimedia Tools and Applications Journal*, volume 8, number 2, pages 249-283, 1999
[7] Meng and Mostafa H. Ammar and Ellen W. Zegura, "Selecting among Replicate Batching Video-on-Demand Servers," *Proc. of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (NOSSDAV), published by ACM, May 2002
[8] F. Dabek and M. Kaashoek and D. Karger and R. Morris and I. Stoica, "Wide-area cooperative storage with CFS," *Proc. of ACM Symposium*

on Operating Systems Principles (SOSP), 2001

[9] J. Kubiatowicz and D. Bindel and Y. Chen and S. Czerwinski and P. Eaton and D. Geels and R. Gummadi and S. Rhea and H. Weatherspoon and W. Weimer and C. Wells and B. Zhao, ”Oceanstore: An Architecture for Global-State Persistent Store,” *Proc. of Support for Programming Languages and Operating Systems* (ASPLOS) 2000

[10] A. Rowstron and P. Druschel, ”Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility,” *Proc. of ACM Symposium on Operating Systems Principles* (SOSP), 2001

[11] B. Zhao and J. Kubiatowicz and A. Joseph, ”Tapestry: An infrastructure for fault-tolerant wide-area location and routing,” *Tech. Rep.* UCB/CSD-01-1141, UC Berkeley

[12] H. Deshpande and M. Bawa and H. Garcia-Molina, ”Streaming Live Media over a Peer-to-Peer Network,” *Technical Report* Stanford Database Group 2001-30

[13] Duc A. Tran and Kien Hua and Tai Do, ”Peer-to-Peer Streaming Using A Novel Hierarchical Clustering Approach,” *Proceedings of the tenth ACM international conference on Multimedia*, 2002

[14] D. Xu and M. Hefeeda and S. Hambrusch and B. Bhargava, ”On Peer-to-Peer Media Streaming,” *Technical Report*, Pursue Computer Science, Apr. 2002

[15] T. Nguyen and A. Zakhor, ”Distributed Video Streaming Over Internet,” *Proc. of SPIE/ACM MMCN*, 2002

[16] J. Chuang, ”Resource allocation for Stor-Serv: Network storage services with QoS guarantees”, *Proc. of NetStore Symposium*, 1999

[17] Y. Chen and R. Katz and J. Kubiatowicz, ”Dynamic Replica Placement for Scalable Content Delivery,” *Proc. of International Workshop on Peer-to-Peer Systems*, 2002

[18] Giwon On and Jens Schmitt and Ralf Steinmetz, ”Design and Implementation of a QoS-aware Replication Mechanism for a Distributed Multimedia System”, *Proc. of the Workshop on Interactive Distributed Multimedia Systems* 2001 (IDMS’01), Lancaster, UK, pages 38-49, ISBN 354042530

[19] Giwon On and Jens Schmitt and Michael Liepert and Ralf Steinmetz, ”Replication with QoS support for a Distributed Multimedia System,” *Proc. of the 27th EUROMICRO Conference* (Workshop on Multimedia and Telecommunications), 2001, Warszaw, Poland, ISBN 0769512364

[20] D. S. Hochbaum, ”Approximation Algorithms for NP-Hard Problems,” PWS, Boston, 1996

[21] O. H. Ibarra and C. E. Kim, ”Fast approximation for the knapsack and sum of subset problems,” *Journal of the ACM*, 22:463-468, 1975

[22] W. Jeon and K. Nahrstedt, ”Peer-to-peer Multimedia Streaming and Caching Service,” *Proc. of ICME 2002*, Lausanne, Switzerland, August 2002

[23] Chris Loeser, ”Peer-to-Peer Network Simulation Environment,” (German) *Technical Report* 2003. C-LAB, Paderborn University, Germany

[24] Klaus Volbert, ”Simulative Analysis of Communication Strategies in mobile ad hoc Networks”, *Diploma Thesis* 2001

[25] Qingming Ma, Peter Steenkiste, ”Quality-of-Service Routing for Traffic with Performance Guarantees,” Computer Science Department, Carnegie Mellon Univerity, Pittsburgh, PA 15213, USA, 1998.

[26] Carsten Griwodz, ”Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure,” *PhD Thesis* 2002 TU Darmstadt, Germany.

[27] Ankur Bhargava, Kishore Kothapalli, Chris Riley, Christian Scheideler, and Mark Thober, ”Pagoda: a dynamic overlay network for routing, data management, and multicasting,” *Proc. of the sixteenth annual ACM symposium on Parallelism in Algorithms and Architectures*, 2004, ISBN: 1-58113-840-7, pages 170–179, Barcelona, Spain.

Kan Hung Wan has received a BS degree in 2002 and MS degree in CS from Paderborn University, Germany in 2004. email: kan_hung_wan@t-online.de

Chris Loeser has received a Diploma in CS from Paderborn University, Germany in 2001 and is currently a PhD student at the Cooperative Computing and Communication Laboratory. The C-LAB is a joint venture between Paderborn University and Siemens Business Services. Personal webpage: http://www.c-lab.de/vis/loeser, email: loeser@c-lab.de