Towards Self-Organizing Multimedia Delivery

Anita Sobe

anita.sobe@itec.uni-klu.ac.at

Laszlo Böszörmenyi

laszlo@itec.uni-klu.ac.at

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

Institute of Information Technology
Klagenfurt University
Technical Report No TR/ITEC/12/2.08
September 2008

Abstract

A video calculus is introduced, describing videos as a composition of elementary, self-describing data units. With the help of the calculus we can describe videos as direct access media, rather than a sequential stream of frames. Direct access is meant in several dimensions: in time, space and quality. This flexible view is the prerequisite for a self-organizing video delivery mechanism. In a first evaluation we show that under the assumption that only a small fraction of available video scenes is getting popular, the flexible delivery mechanism is also substantially more efficient than traditional mechanisms.

1 Introduction

1.1 Self-organization in video delivery

We regard self-organization - to tell it short - as a kind of resource management, aiming at enhancing of efficiency and scalability, with the help of reducing complexity and raising the grade of distribution. In video delivery, the principle of self-organization is currently hardly applied. In recent years, a number of fairly well-established video delivery methods has been crystallized, as different compromises between the two extremes of (1) pure download-and play (zero sharing, long startup-delay and stable - network-independent - quality); and (2) client/server based streaming (maximal sharing, short startup-delay, varying quality). The improved methods are variations of proxy caching, content delivery network (CDN) and peer-to-peer (P2P) techniques (for more details see the chapter on related work). However, all these techniques suffer from the same drawback: They either disregard the timely character of videos, or they view at videos as one, long, continuous stream of data in time. In the first case, we can apply self-organizing methods, however, without any guarantee for quality. In the second case, the timely constraints make the usage of highly flexible delivery techniques practically impossible. We will show that in many newly emerging usage scenarios, these extreme views can and should be replaced by more flexible ones, enabling also correspondingly flexible resource management, based on self-organizing principles.

1.2 New usage scenarios

General adoption of digital multimedia [1] [1]- especially that of video - suffers considerably from a traditional view of usage; assuming users watching some previously selected videos for hours; in analogy to users going to a video shop and lending movies. Even though this "classical" usage pattern is of high relevance, the dictate of viewing videos as "long spaghetti in time" can constrain further innovative development. In the near future we may expect that radically new usage patterns will arise, which can be characterized by the following main features:

- Digital multimedia is produced at many sources and is injected at most different places into a fully distributed and multimodal environment.
- A huge "web" of multimedia data is produced and consumed in most different quality, with most different aims and requirements.
- Beside entertainment, professional use of digital multimedia will grow considerably.
- Production and consumption of multimedia data must be integrated much better in the computing environment than this is the case today.

In such a world, production, search, access, delivery, processing and presentation of multimedia data must become much more flexible; or even "spontaneous". While spontaneity is a valuable attitude in everyday life, it is extremely hard to apply this concept in technology. Research is confronted with new challenges. Let us consider an example scenario of this new multimedia world.
In case of a live event (e.g. the yearly "iron man"), where tens of thousands of interested people, including some professional camera teams and a great number of private people equipped with most different camera facilities follow the event on a large but limited geographical area for several hours. A huge number and diversity of multimedia data are generated and people with most different interests would like to consume them. Many of the users are watching just for fun; some others in order to follow a certain participant; further people, like the organizers, in order to get a global view of the whole event, etc. How can these users find easily and exactly what they need, without being bothered by long sequences of scenes they are not interested in, how can they get the required content without substantial delay and in exactly that

[1] The term multimedia is used for a mixture of modalities, at least one of them being continuous. In the center of interest are "natural" videos, taken by cameras.

quality they require (neither worse nor better)? Currently, there is no system that is able to cope with such a complex and spontaneous world, not even approximately.

1.3 Popularity revisited

We have known for many years that popularity of videos follows essentially the laws of Zipf and Pareto. This means - a bit simplified - that roughly 20 % of all videos, stored somewhere accessible over the internet, will be downloaded or streamed more than once. The remaining 80 % will be practically never used. It has been much less noticed that the same laws hold for scenes inside of videos. I.e. only 20 % portion of all downloaded videos will be watched. Putting these two observations together, we come to the result that ca. 4 % of all video material is watched by a second person (beyond the author), the rest is just there. However, in resource management, this issue is hardly considered - if, then at most on the level of the popularity of entire videos, but hardly on the level of individual scenes. We make a lot of effort to provide good resource management for the entire 100 % - although what we need is rather a good management for the relevant 4 %. Even if the resource management takes popularity into consideration on the level of entire videos, and even if techniques, such as partial caching, do consider popularity on the level of scenes to a certain extent, here is still a huge potential of savings, up to two orders of magnitude. The difficulty is of course obvious: we usually do not know which is that 4 % we should support the best. Therefore, we need a new model of video delivery. Instead of viewing videos as sequential streams of data (resembling the video tape paradigm), they should rather be regarded as direct-access media. Direct-access in a broad sense: users should get exactly (1) what they need, (2) when they need and (3) how (in what quality) they need. This implies a transition to a flexible management in heterogeneous environments.

1.4 Basic elements of a flexible video delivery model

We look for a new model of video delivery, with the following basic assumptions:

1. We assume that videos are mostly not watched sequentially. In many, especially professional, usage scenarios people want to quickly find certain scenes and avoid watching long sequences they are not interested in.

2. Data are decomposed into units of "meaningful" size (how large meaningful is, depends on the actual context); and can be composed under Quality-of-Service (QoS) constraints to continuous "movies". The units are self-describing. This enables composition at any place. For performance reasons the decomposition may be performed in a lazy way, in order to avoid decomposing data, which are never used or are used only for "traditional" streaming in full size.
   A traditional, long, continuous video is defined as a "special case"; as a sequential composition of data units, under certain QoS constraints (e.g. 25 frame/sec, jitter $<$ 10 msec). The interesting point is that we may compose any data units in any order under arbitrary QoS constraints. The user becomes thus from a passive consumer to an active "composer". This does not mean that the interactive human user always has to take the burden of the composition: pre-defined profiles of user-classes may serve with composition patterns. Essential is that the model basically supports this free and flexible composition. In real usage full freedom must be always reasonably restricted. Decomposition and composition are obviously not for free. Using them only to support traditional usage patterns is hardly a good idea. However, if we assume that videos are usually not watched sequentially from the beginning to the end, rather certain important or "popular" parts are watched often then we are confronted with new optimization possibilities. Popular parts may be replicated much more intensively than others. Moreover, the same data might be replicated in different qualities, following different replication strategies. Optimal data management - based on the principle of self-organization - is a most challenging research question.

The topics of this paper are: (1) Description of a video calculus that allows for a quantitative comparison of our suggested approach and traditional delivery techniques and (2) Presentation of first results on the feasibility of the flexible video delivery model, based on the calculus and in comparison with well-known delivery techniques.

2 Video Calculus

The first question is whether a "visionary" video delivery system as described in section 1 is feasible. Therefore, in the first step, we define a simple, formal "video calculus" enabling quantitative reasoning on different delivery techniques. As a first "test" we apply the calculus on well-known video delivery techniques. This is valuable per se, as currently a solid, quantitative comparison of theses techniques is hardly possible. In the second step we apply the calculus on the suggested, more flexible delivery model, outlined above.

2.1 Basic Elements

A video v is split up into a set of units u:

$$v = \bigcup u = u_1 \cup u_2 \cup ... \cup u_n \; where \; \forall u_i, u_j \in v \wedge i \neq j : u_i \cap u_j = \emptyset$$

Units can be of different size. Additionally, the units may be semantically meaningful, e.g. can represent scenes, shots, group of pictures or frames of the video.
We introduce three operators $\vdash$, $\leftarrow$ and $\|$. $u_1 \vdash u_2$ describes a sequential operation where u_1 is followed by u_2 *without* QoS constraints. A unit u_1 followed by u_2 *with* QoS constraints is denoted by $u_1 \leftarrow u_2$. Optionally, a time constraint Q can be added to specify type and value of QoS. As an example $u_1 \leftarrow_Q u_2$ can be specified (see section 2.2).
Parallel processing (e.g. delivery) of unit u_1 and u_2 is expressed as $u_1 \parallel u_2$. Sequential operations are ordered by definition; parallel operations are unordered by definition.
Grouping brackets () and [] are introduced for better readability of complex combinations and nestings of sequential and parallel delivery. Units that are connected sequentially are grouped by using (). A group n following sequentially (e.g. in a delivery process) another group $n-1$ (using $\vdash$ *or* $\leftarrow$) results in starting the delivery of group n after group $n-1$ is delivered completely. To specify QoS constraints for all units of a group a constraint Q can be added behind the closing bracket (see section 2.2). Parallel delivery from different sources can be expressed by using the $\|$ operator. Parallel operations can be enclosed by []. For sake of simplicity grouping brackets can be omitted in obvious cases.
Additionally, units can build atomic sets, which are denoted by {}. An atomic set is a single unit or a group of units connected by any operator. All operations within the atomic set are executed as a whole or not at all. The atomic set can be used e.g. for describing the play out of units.

2.2 Simple Examples

The notation described above is sufficient for expressing a range of existing delivery technologies. In the following client/server, content delivery networks (CDN) and peer-to-peer (P2P) are described. Furthermore, download-and-play and streaming for each of the delivery technologies are expressed.
In the client/server model it is assumed that a client downloads a full video from a server without QoS constraints. Using the notation introduced before, client/server download is expressed by:

$$download = u_1 \vdash u_2 \vdash ... \vdash u_n$$

The playback of a video starts after it is fully available, but introduces order and timeliness properties and is described by:

$$playback = \{u_1 \leftarrow u_2 \leftarrow ... \leftarrow u_n\} = \{v\}$$

In client/server streaming the playback starts at the moment a unit is available. The units must follow each other corresponding to some QoS constraints.

$$streaming = download \ \& \ playback = \{u_1\} \leftarrow \{u_2\} \leftarrow ... \leftarrow \{u_n\}$$

The & sign on the left side of an equation is just a shorthand for some combination of two operations - the exact meaning is defined by the right side. Play out buffer sizes are modeled by enclosing more than one unit with curly brackets, e.g. $\{u_1 \leftarrow u_2\} \leftarrow ... \leftarrow \{u_{n-1} \leftarrow u_n\}$ for a buffer size of two units.
A CDN is similar to client/server. Nevertheless, on a cache miss a second download from the origin server or another surrogate server is needed.

$$delivery\ on\ cache\ miss = replication \ \& \ download = (u_1 \vdash u_2 \vdash ... \vdash u_n) \vdash (u_1 \vdash u_2 \vdash ... \vdash u_n)$$

The first download denotes the replication from an origin server to a surrogate server; the second one denotes the download from a surrogate server to the client. The content may also be streamed from the surrogate server to the client.
Another example is peer-to-peer delivery. BitTorrent is especially of our interest, since it divides a video into segments by default and supports parallel delivery. Segments from the same peer are delivered sequentially; from different peers in parallel. BitTorrent can be described according to a simple example. Let us assume that a video v consists of five units that are delivered in groups from three different peers:

$$download = (u_1 \vdash u_4) \parallel (u_2 \vdash u_5) \parallel (u_3)$$

The playback with BitTorrent is the same as the playback of client/server download-and-play: $\{v\}$.
A video streamed over BitTorrent is delivered in the same way as in BitTorrent download-and-play. Nevertheless, the play out is restricted regarding the order of the units. A unit is only played out if the preceding unit is available (without considering unit loss), which is expressed by:

$$playback = \{\{... \{\{u_1\} \leftarrow u_2\} \leftarrow ...\} \leftarrow u_n\}$$

2.3 Expressing QoS constraints

The notation needs to be extended by enabling explicit QoS constraints, in order to quantitatively compare delivery technologies. The QoS annotation language (QL) specified by Blair and Stefani in [2] is a simple events-based notation for describing QoS constraints. The occurrence of an event in time is described by the τ function, i.e. $\tau(e)$. We integrate a simplified version of the QL in our video calculus for expressing timeliness properties. For example, we use the QL for describing the time constraint Q for sequential delivery. A video that is streamed from a server to a client is expressed by $\{v\} = \{u_1\} \leftarrow_Q \{u_2\} \leftarrow_Q ... \leftarrow_Q \{u_n\}$. To express a jitter constraint Q is set to $\forall n,\ Q = \delta_1 \leq |\tau(u_n) - \tau(u_{n-1})| \leq \delta_2$ where δ denotes a duration in time. The time between the delivery of two units must not be smaller than δ_1 and must not exceed δ_2. QL can be used for describing further QoS constraints such as delay and throughput. For example, the duration of the delivery of a full video can be described by specifying the constraint $Q = \delta \leq |\tau(u_n) - \tau(u_1)|$; in other words the delivery of the whole video $v = (u_1 \leftarrow ... \leftarrow u_n)_Q$ must not exceed δ.

2.4 Applying the Video Calculus for self-organizing Video Delivery

Two basic operations are applied to videos in our "world". Firstly, each video is decomposed in semantically meaningful units. Secondly, units will be recomposed according to the user's demands. It is possible to recompose units from different sources and with different content.
In the following we apply the video calculus to some flexible delivery scenarios. To distinguish units from different videos an index for the related video is added. I.e. unit x of video y is denoted by $u_{y,x}$.

1. Assume a user knows that she wants to watch scene 3,4 and 5 of video x. A unit represents a scene. All units are located on different entities. The delivery can be parallel and is then described as follows: $u_{x,3} \parallel u_{x,4} \parallel u_{x,5}$. The playout includes the constraint that unit u_5 can only be played out if unit u_4 is available and unit u_4 can only be played out if unit u_3 is available:

$$playback = \{\{\{u_{x,3}\} \leftarrow u_{x,4}\} \leftarrow u_{x,5}\}$$

2. Assume a user wants to watch scene 4 and 5 of video 1, scene 1 of video 2 and scene 3 of video 3. A unit represents a scene. Let all units of video 1 and video 3 be located on the same host and all units be streamed to the user. The delivery including playout can be expressed by:

$$download\ \&\ playback = (\{u_{1,4}\} \leftarrow \{u_{1,5}\} \leftarrow \{u_{3,3}\})_Q \leftarrow \{u_{2,1}\}, \quad where\ Q = \delta \leq |\tau(u_{3,3}) - \tau(u_{1,5})|$$

   The delivery of the units of video 1 and 3 must not exceed δ. The delivery of unit $u_{2,1}$ starts after the units of video 1 and 3 are fully delivered.

3. Units that are popular are assumed to be often replicated. Assume a user wants to watch amongst other popular units $(u_{x,3}, u_{y,4})$ a unit $(u_{i,200})$ that has not been popular until now. The delivery of the popular units can start immediately. In the meantime the non-popular unit has to be replicated. The replication and delivery of unit $u_{i,200}$ must not exceed the delivery of the popular units. The delivery of the popular units depends on their location. Let the popular units be located on the same entity and be streamed: $\{u_{x,3}\} \leftarrow \{u_{y,4}\}$. Unit $u_{i,200}$ will be copied to a replication server and afterwards streamed to the client: $u_{i,200} \leftarrow \{u_{i,200}\}$. Both deliveries can be combined to:

$$download\ \&\ playback = (\{u_{x,3}\} \leftarrow \{u_{y,4}\}) \parallel (u_{i,200} \leftarrow \{u_{i,200}\})_Q \quad where\ Q = \delta \leq |\tau(u_{y,4}) - \tau(u_{x,3})|$$

2.5 Evaluation

It is obvious that the flexible unit model can be very efficient under some certain conditions; and can be wasteful under some other conditions. In the following we are looking for pre-conditions, under which such a video-delivery is profitable.

2.5.1 Parallel and Sequential Delivery.

Consider the following example: Assume unlimited download bandwidth and limited upload bandwidth of 1000 kbps. Units 1-3 are delivered in parallel: $u_1 \parallel u_2 \parallel u_3$. The size of u_1 is 50 MB, the size of u_2 is 70 MB and the size of u_3 is 100 MB. The upload bandwidths of all sending sources can be summated and the delivery starts synchronously. Thus, the time needed to process all units equals the delivery time of the unit that takes longest. This can be expressed precisely with the video calculus: $\forall n, \delta = \max(\delta_{u_n})$ where $\forall n, \delta u_n = |\tau(u_n.finished) - \tau(u_n.start)|$. $u_n.start$ and $u_n.finished$ stay for the time of starting and finishing the delivery of unit n, respectively. Applying this to our example the delivery time equals to $\delta u_3 = 100\ MB * 8/1\ Mbit \approx 13\ minutes$.
If the same units are delivered sequentially the upload is constrained by 1000 kbps. A unit starts to be delivered if the preceding unit is already downloaded. Thus, the delivery time equals the sum of all unit delivery times: $\forall n, \delta = \sum \delta_{u_n} = (210\ MB * 8/1\ Mbit = 30\ minutes$.
The suggested delivery model is flexible enough to combine parallel and sequential delivery. Using the video calculus it is possible to clearly express and calculate the costs of delivery.

2.5.2 Startup Delay.

Referencing the popularity discussion in section 1 users are interested in approximately 20 % of a video. Assuming a video's size of 700 MB the users are interested in 140 MB. This is often the first part of the video, but not at all necessarily. In our model any 20 % of a video should be accessible easily (in integral number of units).
Using download-and-play the video is always fully downloaded. Additionally, the playback time to the point of the desired unit u_d has to be considered. Thus, the startup delay can be expressed by: $\delta = |\tau(u_n) - \tau(u_1) + \tau(u_d) - \tau(u_1)|$.
For playback and streaming the worst case scenario is represented by the desired unit located at the end of the video. In this case all units are delivered and played back, respectively. The startup delay of streaming is denoted by: $\delta = |\tau(u_d) - \tau(u_1)|$.
The best case scenario for our flexible model is represented by being able to predict exactly *what* unit is desired and *where* the unit is located within the video. Then, only the required unit has to be delivered. The startup delay is denoted as: $\delta = |\tau(u_d.finished) - \tau(u_d.start)|$. The worst case scenario is represented by the necessity of delivering a full video because of a bad prediction. The flexible delivery model will always perform better in comparison to download-and-play as long as the prediction does not cause a delivery of a full video. Regarding streaming the flexible approach performs better if the following holds: $u_x \leq u_d \leq u_y : \delta_{flex} = |\tau(u_x) - \tau(u_y)|$ is lower than $\delta_{streaming} = |\tau(u_d) - \tau(u_1)|$.

2.5.3 Replication Effort.

In order to provide good video quality, units have to be replicated near to the user. We measure the replication effort as the time needed to copy the requested units to a replication server. The overall costs are the sum of the replication costs and the delivery costs from the replication server to the client ($\delta = |\tau(u_n) - \tau(u_1)|$). Interested clients share the replication costs ($c = \frac{\delta}{clients}$). The higher the number of interested clients the lower are the costs that have to be paid for replication - the higher is the hit rate.
In the flexible delivery model we take into consideration that some units are more popular than others. Assume a video of 10 units. Let 500 users be interested in unit u_1 and 1000 users be interested in unit u_2. The rest of the video is less popular. Some of the 500 users that are interested in unit u_1 are also interested in unit u_2 and vice versa. However, there are also users that are interested in the whole video. For them the effort to replicate and search the whole video with our flexible model will be higher, than requesting the video as a single file. The users that are only requesting some of the popular units can take advantage of the high client interest rate. The costs for a user that is interested in unit u_1 and u_2 are $\frac{c_{u_1}}{500} + \frac{c_{u_2}}{1000}$.
Additionally, limited cache sizes have to be considered. A traditional delivery model that performs full caching can only store 1 video in a cache of 700 MB. The flexible delivery model can cache 5 videos. Even though, partial caching of videos is not new at all, it is usually restricted to the first part of a video (in time or quality) - see also the chapter 3 on related work. We release this restriction to the first few minutes of a video or lower layers of the quality domain. The strength of our approach is that we can - depending on the precision of the popularity prediction - cache more consequently.

3 Related Work

The video calculus was "tested" according to traditional delivery techniques like Content Delivery Networks (CDN) as described by Vakali and Pallis [3], [4]. An origin server is supported by strategically placed surrogate servers to which the content is replicated. The type of replication in a CDN varies from passively pulling the content to actively pushing it. The latter replication technique is rarely found in commercial CDNs [5]. CDNs are widely used for supporting web servers. A special video related CDN is described by Cahill et al. [6], where the content is automatically replicated near to the users. Akamai [7] also offers a special service for video streaming, but only for flash videos. In a CDN the client does not

contribute to the delivery, whereas in a Peer-to-Peer network (P2P) peer capacity is exploited. A survey on P2P delivery systems is given in [8], [9]. Streaming technologies in P2P networks were investigated in [10], [11], [12]. The received quality is still a matter of luck, since P2P systems traditionally suffer from peer churn. In the BitTorrent protocol [13] a tracker manages the available content and the connected peers. Efficiency is improved by splitting the content into segments and thus allowing simultaneous upload and download. Several authors investigated the BitTorrent protocol for streaming [14], [15]. The traditional download policy has been adapted from prioritizing the rarest segments to prioritizing segments near to the playback time. Qiu and Srikant [16] showed that P2P streaming is mostly used in order to support streaming servers since the peer upload capacity is not sufficient for videos of good quality. In contrary to the BitTorrent protocol our delivery model introduces semantically meaningful units, which allows for a higher flexibility for non-sequential media.
In our model network places are mentioned where content is replicated to. It does not matter if the entities are Proxy Caches, nodes of a CDN or a peer in a P2P network. In [17], [18] a Proxy to Proxy (X2X) network with some self-organizing capabilities has been investigated. The term affinity has been introduced to control the three available component classes (proxies, videos and clients). Proxies that handle similar requests and/or are located near to each others are grouped together by having high affinity to each other. Similarly, videos are replicated to proxy groups, to which they have a high affinity. An integration of the idea of affinity in a more general form into the unit delivery model will be further investigated.
In [19] partial caching and replication according to quality reduction has been investigated. With the flexible delivery model we are able to cache specific parts of a video according to popularity. In contrary to traditional models where only the beginning of the video is cached, or quality levels as described in [19]. In [2] the Quality annotation language (QL) is described in detail. Based on this we integrate the QL into our video calculus. With the pure video calculus all delivery techniques described before including our flexible delivery model can be expressed; the integration of QL allows for an analytical comparison of the technologies. Dynamic usage patterns and the requirement to increase scalability exclude a centralized management of replications. At a certain point self-organization is needed. Herrmann [20] describes self-organizing service replica placement. Furthermore, he compares them to self-organizing systems from biology. Each service executes the replication algorithm regularly and locates itself automatically. The authors in [21] investigate centralized and decentralized replication algorithms for self-organizing CDNs.

4 Conclusion and Future Work

We have introduced a novel video calculus enabling quantitative comparison of different video delivery techniques. The calculus also enables us to describe a new, flexible delivery mechanism. This flexibility is a key and the first step towards self-organizing video delivery.
We suggest taking advantage of the observation that only ca. 20% of all available videos, and only ca. 20 % of the scenes of the same are popular. As a result, only 4 % of all available video scenes can be regarded as popular - and should be handled preferential. Thus, principally, we could enhance the efficiency of delivery by two orders of magnitudes. The realistically reachable gain depends on the precision of forecasting the actually interesting 4 %. This again depends on the actual user behaviour, which will be investigated in the future.
The first step for future research is to extend the video calculus, to deal with effects such as data loss and churn in P2P environments. A further step is to implement the flexible video delivery mechanism, and gain knowledge about user behaviour. The traditional separation between video search and delivery should be overcome. Instead of assuming that the user knows by some previous actions (by mystery e.g.) exactly, which video to watch, we introduce a two-phase mechanism, consisting of video offering and video delivery. Offering should be fast, highly interactive and should provide orientation about available videos in a certain context (e.g. a live event). During offering the underlying resource management system can make preparations for efficient delivery. In a similar way, as in a good restaurant, where guests get first a bill of fare and some appetizer very quickly, enabling them making their choices comfortable - and the

kitchen to prepare the main dish.

References

[1] Böszörmenyi, L.: Public Adoption of Digital Multimedia - Why Is It Lagging behind Expectations? (Invited Talk). In: Proceedings of 2nd International United Information Systems Conference (UNISCON 2008) LNBIP Vol. 5, Springer Verlag,. (2008) 52–58

[2] Blair, Stefani: Open Distributed Processing and Multimedia. Addison-Wesley (1997)

[3] Vakali, A., Pallis, G.: Content Delivery Networks: Status and Trends. IEEE Internet Computing **vol. 7**(6) (2003) pp. 68–74

[4] Pallis, G., Vakali, A.: Insight and perspectives for Content Delivery Networks. Commun. ACM **vol. 49**(1) (2006) pp. 101–106

[5] Buyya, R., Pathan, A.M.K., Broberg, J., Tari, Z.: A Case for Peering of Content Delivery Networks. IEEE Distributed Systems Online **7**(10) (2006) 3

[6] Cahill, A.J., Sreenan, C.J.: VCDN: A Content Distribution Network for high Quality Video Distribution. In: Proc. Information Technology & Telecommunications, Letterkenny, Ireland. (2003)

[7] Akamai: Akamai. url: http://www.akamai.com

[8] Li, J.: On Peer-to-Peer content delivery. Peer-to-Peer Networking and Applications **vol. 1** (2008) pp. 45–63

[9] Androutsellis-Theotokis, S., Spinellis, D.: A Survey of Peer-to-Peer Content Distribution Technologies. ACM Computing Surveys **34**(4) (2004) 335–371

[10] Liu, Y., Guo, Y., Liang, C.: A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications **vol. 1** (March 2008) pp. 18–28

[11] Guo, Y., Suh, K., Kurose, J., Towsley, D.: A Peer-to-Peer on-demand Streaming Service and its Performance Evaluation. In: ICME 03: Proceedings of the 2003 International Conference on Multimedia and Expo. (2003)

[12] Jiang, X., Dong, Y., Xu, D., Bhargava., B.: Gnustream: A P2P Media Streaming System Prototype. In: Proceedings of the International Conference on Multimedia and Expo (ICME). (2003) 325–328

[13] specification, B.: URL: http://www.bittorrent.org/beps/bep_0000.html; accessed 06/08 (06 2008)

[14] Vlavianos, A., Iliofotou, M., Faloutsos, M.: BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In: 9th IEEE Global Internet Symposium 2006. (April 2006)

[15] Choe, Y.R., Schuff, D.L., Dyaberi, J.M., Pai, V.S.: Improving VoD server efficiency with BitTorrent. In: MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia, New York, NY, USA, ACM (2007) 117–126

[16] Qiu, D., Srikant, R.: Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In: SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA, ACM (2004) 367–378

[17] Spielvogel Ch., B.L.: Quality-of-Service based Video Replication. In: Mylonas P., Wallace M., Angelides M. (Hrsg.): Proceedings of Second International Workshop on Semantic Media Adaptation and Personalization (SMAP 2007) London, UK. (2007) 21–26

[18] Spielvogel, C., Böszörmenyi, L.: Active and passive replication of Multimedia Content in a Proxy-to-Proxy Network (X2X). In: Proceedings of Parallel and Distributed Computing and Networks (PDCN2007). (2007)

[19] Podlipnig, S., Böszörmenyi, L.: Replacement Strategies for Quality Based Video Caching. In: International Conference on Multimedia and Expo (ICME). Volume 2. (2002) 45–53

[20] Herrmann, K.: Self-Organizing Replica Placement - A Case Study on Emergence. In: SASO '07: Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems, Washington, DC, USA, IEEE Computer Society (2007) 13–22

[21] Coppens, J., Wauters, T., Turck, F.D., Dhoedt, B., Demeester, P.: Evaluation of Replica Placement and Retrieval Algorithms in self-organizing CDNs. In: Conference Proceedings of IFIP/IEEE International Workshop on Self-Managed Systems & Services SelfMan 2005. (2005)

Institute of Information Technology
Klagenfurt University
Universitaetsstr. 65-67
A-9020 Klagenfurt
Austria

http://www-itec.uni-klu.ac.at

Klagenfurt University