2009 First International Conference on Advances in Multimedia

Non-sequential Multimedia Caching

Anita Sobe
Institute of Information Technology (ITEC)
Klagenfurt University
Klagenfurt
Austria
Email: anita.sobe@itec.uni-klu.ac.at

Laszlo Böszörmenyi
Institute of Information Technology (ITEC)
Klagenfurt University
Klagenfurt
Austria
Email: laszlo@itec.uni-klu.ac.at

***Abstract*—Non-sequential media is seen as a number of semantically meaningful units derived from several videos. A video notation is introduced to express and compare different transmission types including caching. Furthermore, the idea of grouping units based on user intentions is described, which is the basis for two cache admission policies (simple and rank-based). These policies are compared regarding efficiency and hit rate. The evaluation is based on simulation and shows that a replacement policy that considers the popularity of the units is needed.**

***Index Terms*—multimedia caching, non-sequential multimedia, video notation**

I. Introduction

"We assume that in the future virtually everybody may broadcast any kind of message at any time and can of course also receive any such message at any time, at any place, equipped with any kind of device ... We assume further that videos will be rarely watched sequentially. In many usage scenarios, especially in professional situations, people want to quickly find certain scenes and avoid watching long sequences they are not interested in ...." [1]
Therefore, we need a radically higher level of *flexibility* in the management and usage of videos than usual. We derive two basic new requirements from the above assumptions: (1) Videos should be regarded as non-linear, direct access data, and (2) The relation between data and metadata should become dynamic and bidirectional.
(1) Non-linear video. We regard videos as direct access media - instead of the traditional long spaghetti in time. A continuous video presentation should be dynamically composed of units. A unit could be a single frame, or a clip, which is short enough to be downloaded faster than to be streamed. From such units, spread over the Internet, arbitrary presentations can be composed on demand. The composition may fulfill quality-of-service constraints, if necessary. A classical video stream could be thus described as a sequential composition of frames (as units), e.g. under the constraint of 25 fps and a jitter of $\pm$ 10 ms. A much more interesting scenario could be e.g. a sport reporter, downloading and playing the goal-scenes of some soccer games in parallel.
(2) Relation between data and metadata revisited. Behind the notion of metadata lies the - usually subconscious - assumption that we know in advance what the data are and what the metadata (additional information to the data) are. In the highly dynamic scenarios of the future, when not only new data but also new kinds of data are continuously entering the scene we need radically new ways to inform the consumers, what is there, what could be watched. Thus, data may turn into metadata and the way around - thus forming a general hypermedia space. This fits very well to the idea of non-linear video: data units can not only be composed, but can also refer to each other.
Breaking with the conventional view of video representation and presentation brings a new dimension of flexibility. The question is of course: for what price? Using a flexible system promises good performance only, if we can indeed take advantage of this flexibility both (1) in the application patterns and (2) in the resource management. If we assume that in accordance with the laws of Zipf and Pareto, at most 20% of all videos, available somewhere over the Internet, will be downloaded or streamed more than once, and again only at most 20% portion of these will be really watched, we can assume that at most 4% of available video material is watched at all [1]. That means that if we could exactly identify this 4%, than we could concentrate in resource management on this part of the data (e.g. replicating them heavily to popular client locations). Of course we cannot predict the desired 4% exactly - however, a potential of two orders of magnitude is there. That means that it is worthwhile to do research based on the above assumptions. It promises a new class of video services, which is not only much more flexible, but also more efficient than the traditional approaches.

In this paper we define first a simple notation enabling sequential, parallel and QoS-constrained unit composition. Then we make first feasibility studies on the suggested model. The simplest possible realization of it is a single cache that performs caching based on video units and their popularity. We present first simulation results of this investigation.

II. Related Work

A first step in the direction of non-sequential media was investigated by Zhao et al. in [2]. The authors are defining "non-linear" media as a video that consists of several parallel branches. The user can decide at certain points interactively which branch to follow, e.g., "alternative ending movies". The streaming system holds a channel per branch; the contents are transmitted using multicast. The authors observed as major problem that there is no possibility to explore bandwidth reduction by sharing connections, because it is not known if the client will choose the branch just transmitted in advance. Nevertheless, the authors showed that some hints regarding the client branch selection lead to remarkable server bandwidth and client data overhead reduction. In this paper it is shown, that clients form groups regarding the path they follow. For sure, some paths are more popular than others. Nevertheless, in this case the possible paths are predefined and limited in comparison to the possibilities regarding our unit model.

Videos are considered too large for being cached as a whole. A lot of research has been done on partial caching. In general the idea of caching only parts of a video fits our non-sequential media idea.

In [3], a detailed overview of different caching strategies is given. Prefix caching and segment-based caching are the most similar to our work. A prefix may be fixed or dynamic (investigations in this direction can be found in [4]). Segment-based caching starts with a minimal part of a video and based on popularity measurements the number of segments of this video increases in the cache. Segments may be uniformly sized or grow exponentially as shown in [5].

In [6] the authors measure the popularity between segments, which leads to an internal popularity distribution. Based on that the authors introduce a caching algorithm for streaming media. Considering fixed sized segments of one second, they observed that the popularity within a video follows a k-transformed Zipf-like distribution (for $k_x = 10$ and $k_y > 200$). Since the internal popularity shows, that the beginning of the video is most popular, the caching policy prioritizes prefix caching. However, we cannot expect that non-sequential media follows the same internal popularity pattern. A user may not be aware of which unit is the "beginning" of a video.

Another segment-based caching mechanism is described in [7]. The authors observed that no caching technology supports interactivity like "jumps" in a video stream. In this paper the authors introduce firstly a basic interleaved segment caching (BISC) policy. Every second segment is prefetched. This guarantees a higher hit rate if interactive jumps are common. On a miss, the cache delivers the closest cached segment. Since it is more likely that after a jump segments are accessed sequentially, BISC was extended to a dynamic interleaved segment caching (DISC) policy. DISC initially caches a whole object for observation. Later, it will be decided if segments are replaced or several segments are stored sequentially. The authors expect, although taking jumps into account, that a video will be watched sequentially, i.e. only forward jumps are possible. In our unit model approach we refrain from this restriction.

III. Expressing Transmission

We define the abstract notion of "transmission" of units, which may describe any kind of transportation, such as transmission over the network, delivery between two layers of a protocol stack, or rendering on a presentation device. Each transmission, sequential or non-sequential, can be described by a "video notation".

A video v comprises a number of units u that are indexed by $1 - N$ and have different byte lengths. When transmitting this video sequentially as in currently available systems, one can describe this as: $u_1 \vdash u_2 \vdash ... \vdash u_n$. Unit u_{i+1} follows after full transmission of u_i.

Sequential transmission can be restricted by QoS parameters, like delay and jitter - expressed as: $u_1 \leftarrow_Q u_2 \leftarrow_Q ... \leftarrow_Q u_n$. For the description of Q the QL (QoS annotation language) specified by Blair and Stefani in [8] is used[1]. E.g.

$$\forall i \; Q: \; \delta_1 \leq |\tau(u_i) - \tau(u_{i-1})| \leq \delta_2, \; 1 < i <= N$$

specifies jitter by restricting the delay between unit u_i and unit u_{i-1} to be between the lower limit δ_1 and the upper limit δ_2 (δ describes a duration in time).

A middleware that implements the functionality of the video notation has to meet all specified QoS requirements or has to ensure that a corresponding exception is thrown otherwise.

Besides sequential transmission, the notion of parallel transmission is also introduced. $u_1||u_2||...||u_n$ means that each unit is transmitted in parallel (usually) from different places and that there is no restriction regarding the order of the units. Sequential and parallel transmission can be combined arbitrarily - corresponding to the usual case in real systems, where some places hold some of the units. Units from different places are usually transmitted in parallel and units at the same location can be transmitted sequentially. E.g. the following expression describes the transmission of a video, divided into two halves, consisting of i respectively $n - i$ units:

$$(u_1 \vdash u_2 \vdash ... \vdash u_i)||(u_{i+1} \vdash u_{i+2} \vdash ... \vdash u_n)$$

The video notation is the basis for the calculation of transmission costs, since different transmission techniques can be described and therefore also be compared. The video notation will be extended to a video calculus (for further information see [9]).

IV. A Flexible Caching Model

As described in section II, traditional partial caching exploits the fact that a video is considered sequential media. Once a video is started to be streamed, either the cache can proactively pull or the video server can push the next parts of the video. For non-sequential media it is not known, what the next parts actually are. A user has the possibility to request whatever content of different videos he/she likes to watch. A proactive cache for non-sequential media needs information about the content and the users to be able to predict the next unit to be loaded.

We consider a simple architecture that consists of an origin server, a proxy cache and a number of users. The contents are streamed directly from the cache on a hit, which can be described as $(u_1 \leftarrow u_2 \leftarrow u_3 \leftarrow ... \leftarrow u_i)$ using the video

[1]For the sake of simplicity Q can be omitted if not required explicitly

notation. The cache acts as proxy that forwards the requests to the server on a miss, which corresponds to:

$$(...((u_{1,s} \vdash u_{1,c})||u_{2,s}) \leftarrow u_{2,c})||...||u_{i,s}) \leftarrow u_{i,c}$$

After a full unit is copied to the cache (c), the cache forwards this unit to the client. At the same time the next unit from the server (s) starts to be transmitted to the cache and if this one is available, it can be forwarded to the client, a.s.o.

Our proposed cache assumes units to be self-contained, equipped with metadata as further information about the content. User intentions are considered as information about semantic roles to which each user can be categorized to. A role of a user can be mapped to the interest in specific units. E.g. a police officer wants to see if car drivers violate the minimum distance to another car on a highway. This police officer is most likely not interested in parts of the video where the highway is empty. An engineer of the motorway company might be interested in the state of the highway and wants to see parts of the video where the highway is empty.

We define therefore: A semantic group G is a set of units out of all units U, where the content of each unit is of interest for a category of users ($G \subseteq U$) - e.g. $G_1 = \{u_1, u_7, u_8, u_{10}\}$. Groups are not disjoint; a unit u may be part of more than one group. As an example let there be a unit u where an athlete from Austria and another athlete from Italy is shown. The coach of the Austrian team and the family of the Italian athlete may be interested in the same unit although their role is considered to be different.

A user categorized to a specific role is expected to request units of a group that is mapped to that role. Units of different groups are competitive with the space available in the cache. The fact that some unit groups may be more popular than others can be exploited in the caching policy.

Furthermore, for better prediction an ordering within each group is needed. One solution is to order the units according to their popularity. The popularity is characterized by the frequency of requests for a unit and by external hints - e.g. by a smart application layer that has direct contact to the user. The units within the example group mentioned above can be expressed as $G_1 = u_7 \vdash u_{10} \vdash u_1 \vdash u_8$ in descending order of popularity by using the video notation.

Depending on the number of unit groups the cache prefetches the most popular units of each group or the most popular units of the most popular groups. If a request arrives, the next fitting unit of the matching group will be prefetched from the server (if necessary). The next fitting unit is the unit following the current unit regarding popularity within the current group. Furthermore, it is assumed that the request contains information about the group the unit belongs to.

We propose a simple caching admission policy:

$$\text{prefetch=} \begin{cases} u_{next} & \text{if hit } u_{current} \\ 0 & \text{if hit } u_{current} \text{ \&\& hit } u_{next} \\ u_{current} \vdash u_{next} & \text{else} \end{cases}$$

However, the rate of prefetching efficiency is low when following this simple admission policy, since units even with low popularity might be prefetched. Thus, the admission policy has to include a threshold that must not exceed a specific popularity. The fact that the same unit represents different popularities in different groups can be exploited by formulating a general popularity metric. Assuming the popularity rank r of the unit within all groups is known, we map this to a global rank r_{all}. The impact of low popularity is minimized using the logarithm of the group rank.

$$r_{all} = \frac{1}{n} \sum_{i=1}^{n} \ln r_i$$

Let a unit be in the top 5 of one group and less popular in another group. This unit is more likely to be cached than a unit, that has an average popularity within the same two groups.

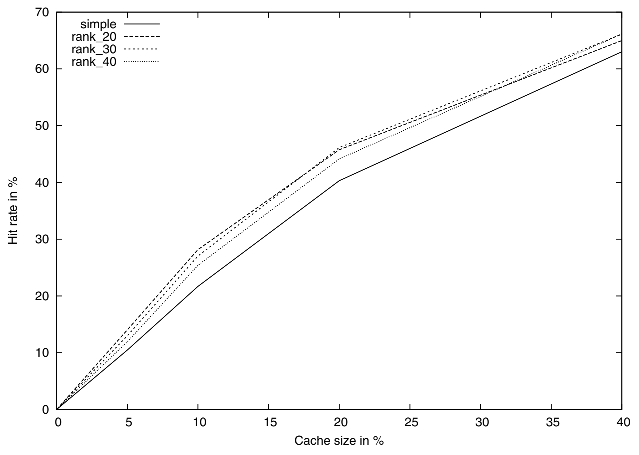
We evaluate the presented approach in a discrete event-based simulator. For the user request generation Medisyn [10] is used, a synthetic traffic generator. For 100 units with different popularity, we model 10000 requests in two competitive groups. As replacement algorithm LRU is used. The popularity threshold is mapped to the rank and was set to rank 20, 30 and 40.

The hit rate of the simple admission policy and the rank-based admission policy is compared in figure 1a. The rank-based algorithm shows an average hit rate increase of 5 % for all cache sizes. This is due to the lower request rate of the rank-based admission policy that results in a longer time a unit remains in the cache. It has to be noted that the difference regarding the hit rate is very small between the different thresholds of the rank-based policy.

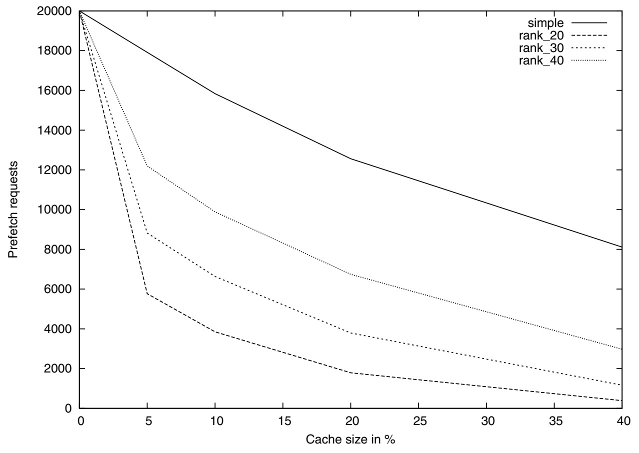
As shown in figure 1b the number of prefetches is remarkably reduced using the rank-based admission policy with all thresholds, which means an increase of prefetching efficiency. We observed a lower prefetching efficiency when the cache size is low, because a higher rate of replacement is reached. Although, LRU supports popular units to remain longer in the cache, for small cache sizes even popular units are often replaced in the case of prefetching. Figure 1c shows the proportion of requests sent to the server in comparison to the client requests. For small cache sizes it would be even better to serve all requests directly from the server. Since the number of total requests to the server exceeds the number of requests from the client. Using the simple admission policy, the load of the server is reduced having a cache size of around 30 %. Whereas the rank-based policy decreases the load already at a cache size of 10-15 %. Thus, for more efficiency, LRU has to be substituted by a replacement policy that considers the unit popularity in order to reduce the number of unnecessary replacements. Additionally, since the hit rate difference is small the rank threshold of 20 is sufficient.

V. Summary and Future Work

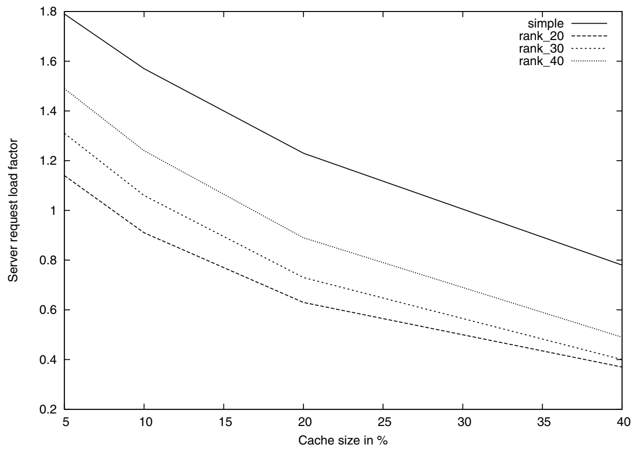
In this work we shortly introduced a video notation for expressing the transmission of non-sequential and sequential media including the possibility of setting preconditions regarding QoS. The video notation is based on the idea that each video is divided into units. We are able to express all traditional



(a) Hit rate comparison of simple and rank-based admission



(b) Number of prefetching requests



(c) Factor of server requests compared to user requests

transmission types like Client/Server streaming (sequential transmission) or download via BitTorrent (mixed transmission) [9]. In the future the video notation will be extended to express transmission costs for traditional transmission and for non-sequential transmission.

Furthermore, we introduced a proactive non-sequential cache that is based on the idea that user intentions can be semantically categorized. This information is used to predict user behavior by mapping the user categories to unit groups. The prediction is based on the popularity rank of each unit within all groups. The next fitting unit regards the next popular unit within a group. The number of requests is reduced by introducing a general rank over all groups. The decision of which unit to load next is based on the general rank of the current unit.

First evaluation scenarios showed, that LRU is not sufficient as a replacement strategy. A popularity-based replacement policy is under investigation that reduces on the one hand the load on the server and on the other hand the replacement frequency.

Another important point will be the relation between the number of unit groups and the size of each group. Since groups containing a high number of units may often replace units from smaller groups. Additionally, the number of users categorized to a role can differ and has impact on the number of units cached. These issues are under evaluation.

Furthermore, the evaluation will be extended to a comparison of dynamic segment-based caching (see section II) and our proposed approach. We expect a better hit rate of our approach since it is more flexible. Although, the complexity increases due to the prediction.

References

[1] H. Kosch, L. Böszörmenyi, G. Hölbling, D. Coquil, and J. Heuer, "Personalization of mobile multimedia broadcasting," *International Journal of Digital Multimedia Broadcasting*, vol. 2008, 2008.

[2] Y. Zhao, D. L. Eager, and M. K. Vernon, "Scalable on-demand streaming of nonlinear media," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1149–1162, 2007.

[3] S. Podlipnig, "Video-caching in verteilten multimedia-systemen," Ph.D. dissertation, Klagenfurt University, Austria, 2002.

[4] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 1999, pp. 1310–1319 vol.3.

[5] K. L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," in *WWW '01: Proceedings of the 10th international conference on World Wide Web*. New York, NY, USA: ACM, 2001, pp. 36–44.

[6] J. Yu, C. Chou, Z. Yang, X. Du, and T. Wang, "A dynamic caching algorithm based on internal popularity distribution of streaming media," *Multimedia Systems*, pp. 135–149, October 2006.

[7] L. Guo, S. Chen, Z. Xiao, and X. Zhang, "Disc: Dynamic interleaved segment caching for interactive streaming," *Distributed Computing Systems, International Conference on*, vol. 0, pp. 763–772, 2005.

[8] G. S. Blair and J.-B. Stefani, *Open Distributed Processing and Multimedia*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[9] A. Sobe and L. Böszörmenyi, "Towards self-organizing multimedia delivery," Reports of the Institute of Information Technology, Klagenfurt University, TR/ITEC/12/2.08, Tech. Rep., 2008.

[10] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, "Medisyn: a synthetic streaming media service workload generator," in *NOSSDAV '03*. New York, NY, USA: ACM, 2003, pp. 12–21.