Analysis of Prefetching Schemes for TV-on-Demand Service

Manxing Du*†, Maria Kihl†, Åke Arvidsson‡§, Christina Lagerstedt* and Anders Gavler*
*Acreo Swedish ICT, Sweden, Email: firstname.lastname@acreo.se
†Dept. of Electr. and Inform. Technology, Lund University, Sweden, Email: firstname.lastname@eit.lth.se
‡Business Unit Support Solutions, Ericsson, Sweden, Email: firstname.lastname@ericsson.com
§Department of Computer Science, Kristianstad University, Sweden, Email: firstname.lastname@hkr.se

***Abstract*—TV-on-Demand service has become one of the most popular Internet applications that continuously attracts higher user interests. With rapidly increasing user demand, the existing network conditions may not be able to ensure low start-up delay of video playback. Prefetching has been broadly investigated to cope with the start-up latency problem which is also known as user perceived latency. In this paper, we analyse request patterns for TV programs from a popular Swedish TV service provider over 11 weeks. According to the analysis, we propose a prefetching scheme at the user end to preload videos before user requests. Our prefetching scheme significantly improves the cache hit ratio compared to terminal caching and we note that there is a potential to further improve prefetching performance by customizing prefetching schemes for different video categories. We further present a cost model to determine the optimal number of videos to prefetch. Finally, we discuss available time for prefetching and suggest that when to make prefetching decisions depends on the user demand patterns of different video categories.**

Keywords–TV-on-Demand services; user perceived latency; prefetching;

I. INTRODUCTION

Nowadays, other than air broadcasting, cable networks and physical media like Video Home System (VHS) or Digital Video Disc (DVD), Internet has become a popular medium for distributing multimedia content like TV shows, movies, and user generated videos. The massive amount of multimedia traffic has imposed a significant burden on the Internet. Consequently, users sometimes have to endure long access delay for filling up the playout buffer before the content is displayed. In [1], the result shows that user's tolerance of waiting time for downloading web pages is about 2 seconds. Results in [2] suggest that the more familiar users are with a web site, the more sensitive they are to delays. Although web caching is widely used as a solution to lessen the web traffic congestion and improve the network performance, the benefit of caches is limited. To further reduce the user perceived latency (UPL), prefetching has become a popular technique. The objective of the prefetching system is to proactively preload certain content to the cache even before users request it.

Thorough summaries of web caching and prefetching approaches and performance measures can be found in [3] [4] [5]. Domènech *et al.* in [6] compared different prefetching architectures and found that the maximum latency reduction of 67.7% can be obtained if the predictor is placed at a proxy while the collaborative prediction between proxy and server can reduce the latency more than 97.2%. However, the results are obtained based on the most ideal scenario that the prediction can be 100% correct, thus the results can be seen as the upper limits of latency reductions.

Most of the existing prefetching approaches are access-history based which predict user future requests depending on the observed content access patterns. Márquez *et al.* applied a Double Dependency Graph (DDG) prediction algorithm to a mobile web and observed that the performance of prefetching approaches rely on the underlying networking technologies [7]. Another history based model is the Markov model, which is an effective scheme to predict what users intend to request based on the sequence of the historical access [8]–[11]. The prefetching schemes proposed in [12]–[15] use the data mining technique to discover users' access patterns.

Popularity-based prefetching approches are also widely used, especially for prefetching multimedia content. In [16], a trace driven simulation was performed to investigate prefetching schemes for YouTube videos. Their prefetching scheme is to prefetch the top 25 videos from each video's related video list to a proxy server. Combining both prefetching and conventional proxy caching, 80.88% hit ratio can be obtained and it only introduces 2% increase in the network load.

However, this recommendation-based prefetching scheme requires an effective recommendation system which can be a big challenge. Krishnappa *et al.* in [17] applied a prefetching top-100 videos scheme on a trace of Hulu traffic in a campus network and compared the performances of prefetching and conventional proxy caching which proves prefetching is very effective for online TV service.

From these papers, we note that the research focus is mainly on proxy prefetching whereas the performance of terminal prefetching is less well known. To the best of our knowledge, our analysis is the first that focuses on using terminal storage to do prefetching for the TV-on-demand service. To prefetch content to the terminal can eliminate the delay between proxy and user and further reduce the playback start-up latency preceived by users.

In this study, we have used data from one of the most popular Swedish TV channels which provides all their broadcasting content online for subscribed users. Each TV-on-Demand program consists of a series of episodes which have high consistency regarding content, thus the index of each episode is a good indicator of user's future access. We propose to use the intrinsic structural information of episodes belonging to a TV series to make prefetching decisions. The criterion for prefetching is based on the index of episodes within each TV program. First, we analyse the potential of prefetching in our dataset, followed by investigating the optimal choice of prefetching. We show that high terminal gain can be obtained by prefetching two adjacent episodes in a series for each viewed episode with minimum cost. This study shows the potential of implementing terminal prefetching for TV-on-

 ISBN: 978-1-61208-396-4

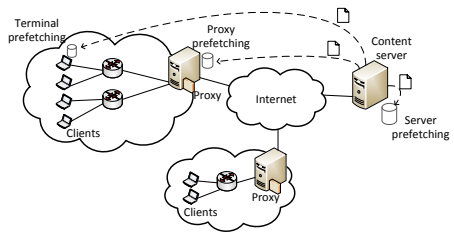


Figure 1. Network architecture with prefetching system

Demand service both effectively and economically. In addition, we investigate whether there is enough time to perform prefetching.

The remainder of the paper is organized as follows. Section II describes the infrastructure of a prefetching system and the evaluation metrics. Our dataset and prefetching methods are presented in Section III. Section IV shows the results and evaluations of our prefetching scheme. In Section V, we further discuss the available time for prefetching and we conclude in Section VI.

II. VIDEO PREFETCHING SYSTEM

By implementing a prefetching system, the multimedia streams can be retrieved from the content provider and saved in a cache. In this way, users can quickly get access to their preferred content. In this section, we first introduce the components of a prefetching system, and then we present the evaluation metrics used for measuring the performance of prefetching schemes.

A. The infrastructure of a prefetching system

The prefetching system consists of two elements: prediction engine and prefetching engine. The prediction engine is responsible for forseeing what users would watch next before they request the content. The prefetching engine proactively stores the video prefix to the local cache. In this case, both the first-time views and repeats of the prefetched videos can be served from the local cache with short start-up delay. These two engines can be placed at any part of the web architecture which is shown in Figure 1: terminals [16], proxies [13] [17], servers [18] [19] or between these elements [6] [7].

In order to bring content closer to end users, so as to the most extent reduce user's perceived latency, we assume the prefetching engine is implemented at the user end which is called terminal prefetching. This means that the prefetched content will be stored in a terminal cache at the user end. Anything from a short part of the video to the entire video can be prefetched. Typically, to reduce the start-up latency, there is no need to prefetch the entire video, which requires more network resources, thus it may introduce longer delay. A solution is to prefetch only parts of video streams. When a user requests a video, it can be played directly from the cache which gives more time for setting up the transport and filling up the playout buffer for the remaining parts of the video. Considering that users may not all favour the beginning of each video, in [20] [21], the performance of caching different parts of the videos are discussed. In this paper, we ignore the portion of prefetched video and treat each video as a video unit, thus the cache size is not considered. In this scenario, every video request from a user is first sent to a prefetching engine which checks whether the video has already been retrieved earlier or not. If so, the content will be served from the local cache instead of the remote server.

B. Evaluation metrics

In order to evaluate the performance of a prefetching scheme, three metrics are used in this study. The first metric is precision (P) which is defined as the number of videos that are prefetched and requested by users (h_v) over the total number of prefetched videos (p_v).

$$P = \frac{h_v}{p_v} \tag{1}$$

A high precision value suggests that more prefetched content is requested by end users, thus the prediction engine works efficiently.

The second metric is recall (R) which is the share of prefetched and requested videos (h_v) over total number of requested videos (r_v).

$$R = \frac{h_v}{r_v} \tag{2}$$

Higher recall indicates that a larger share of the content requested by users can be correctly predicted by the prediction engine.

Precision and recall are constrained by each other. In principle, higher recall could be obtained by prefetching as many videos as possible in order to cover more content. Thus it would be more likely that the prefetched content contains the videos which will be requested by users. However, in this case, p_v will increase. Consequently, the prediction engine's precision will decrease. To prefetch a great amount of data which will not be used by users will also deteriorate the network congestion. These two metrics need to be balanced when designing a prefetching system. F_1 score (balanced F-score) [22] is a weighted average of the precision and recall which is used in this study to show the effectiveness of a prediction. The closer F_1 score is to 1, the more effective the prediction is.

$$F_1 = 2\frac{PR}{P+R} \tag{3}$$

To evaluate whether the prefetched content are highly demanded by users, we introduce the last metric which is the cache hit ratio (H). It is defined as the number of requests to videos (h_r) which are retrieved from the prefetching cache over the total number of requests (t_r).

$$H = \frac{h_r}{t_r} \tag{4}$$

Cache hit ratio shows the share of repeated requests for videos which can be served directly from the cache. A high hit ratio suggests that more requests can be served with reduced start-up delay and a high utilization of the prefetched content.

 ISBN: 978-1-61208-396-4

TABLE I. EXAMPLE OF USER REQUEST

Title	The bridge episode 13
User Id	1044197
Start time	2012-12-31 09:00:08
End time	2012-12-31 09:21:48
Viewing minute	22
avgBitrateMbps	2.599
Program	The bridge
Category	TV series

III. EXPERIMENT

In this section, we will describe the experiments conducted using the prefetching method in this paper. The objective of our analysis has been to investigate which episodes to prefetch for each viewed video to obtain the best performance of the prefetching system. The analysis is carried out by measuring the performance of prefetching and terminal caching in terms of effectiveness and hit ratio. To optimize the number of videos to prefetch, a cost model is proposed to find an appropriate trade-off between the cost of prefetching and the potential of response time improvement.

A. Dataset

Our study is conducted using the video requests from one of the most popular Swedish TV providers. The data was collected by Conviva which provides online video analytic solutions to media content providers around the world. The data is based on recorded TV requests for a subset of users in a city in Sweden. The users are so called subcribed users who can get access to all the TV content online provided by that TV channel by paying a monthly fee. Thus the users who do not have subscriptions are not included in this study either. All the users are anonymized and no data can be traced back to any specific user.

There are nine video categories defined by the service provider as follows: *Children, Documentary, TV series, Home and leisure time, Entertainment, Default, Mixed, Sports and News.* There is too little data in the *Default* and *Mixed* category and usually the *Sports* and *News* content are distributed by live streaming which cannot be prefetched like TV-on-demand content. Therefore, in the following sections, only the TV-on-demand content which is categorized as *Children, Documentary, TV series, Home and leisure time* and *Entertainment* are included and all the TV programs in each category consist of a series of episodes. To ensure unique identifiers for each episode, our dataset only includes requests of programs with only one season available.

Table I shows an example of the available information which is contained in each data entry. There are 7933 subscribed users who generated 104845 requests to 2427 videos which belong to 253 series over 11 weeks from December 31, 2012 to March 17, 2013. Figure 2 depicts the Cumulative Distribution Function (CDF) of viewing time per request. Around 20% of the total requests result in viewing times smaller than 2 minutes. We filtered out these short viewing sessions to eliminate the impact of user's random clicks which are not suitable to serve as predictors. We should also note that users may request the prefetched videos after the end of time period in our dataset. Thus, the result of hit ratio is underestimated due to the finite time period.

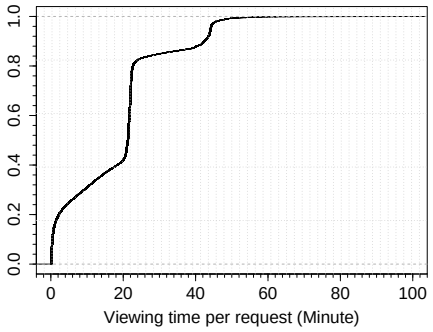


Figure 2. CDF of viewing time per request

B. Video prefetching selection methods

In order to implement prefetching, the prediction engine needs to determine what content should be preloaded based on user's viewing histroy of the same series. In the following, we propose and evaluate a scheme for the prediction engine to make prefetching decisions.

In this study, we consider two extreme cases as baselines for evaluating the performance of the proposed prefetching scheme. One is to prefetch all the available episodes in a series to the end user as long as a user watched an episode in that series. In reality, the content provider has the information of the number of episodes in each TV series, both released and unreleased. In our dataset, the total number of episodes in each series is unknown, so we scan users' viewing histories and collect unique video sets of each series. We assume the videos in each set are all the available videos in each series. This coarse scenario may waste significant amounts of bandwidth since some of the content would never be accessed by users.

The other extreme scenario is conventional terminal caching which passively caches all of user demand and nothing will be preloaded prior to user requests. Comparing with prefetching, terminal caching is also called passive caching. When terminal caching is enabled, the repeated requests for each video can be served directly from the local cache. Terminal caching can help to reduce the initial delay only if the cached video is requested again.

Our approach is based on the intrinsic structure of TV content. Since each TV program contains a series of episodes which have high consistency and similarity in content, we propose to prefetch N adjacent episodes for each viewed episode. Different from videos on the traditional VoD websites like YouTube, a TV series consists of a series of episodes which will be released regularly. User's request patterns of episodes in the series will be examined so that the prediction engine could decide which episodes to prefetch.

C. Cost model definition

Any prefetching scheme makes inaccurate predictions which inevitably downloads more content than a system without prefetching, consequently, the traffic overhead caused by prefetching may impose more burden on a bandwidth sensitive network. The congested network may lead to packet loss, longer transmission delay and poor quality of experience (QoE). Nervertheless, prefetching more content increases the probablity of meeting user's demand and potentially reduces the user's perceived latency. Sometimes spare network capacity

 ISBN: 978-1-61208-396-4

is available during off-peak hours, e.g, during nights. It can be profitable to prefetch as much content as possible during that time to achieve high hit ratio. Hence, we propose a cost model to quantify the cost of prefetching in order to choose optimal number of videos to prefetch.

We assume the cost of real time video delivery equals 1 monetary unit per video and the cost of off-peak prefetching is x monetary unit per video which is smaller than 1, since prefetching can be done during the off-peak hours which costs less than the real time downloading. Besides, the possible cost for poorer QoE can also be seen as a reason that real-time downloading is more expensive than prefetching.

We define the number of videos which are requested by each user but not prefetched as video miss (M). The number of videos which are prefetched by each user is P. The cost of prefetching for *n* users is:

$$C = 1 \cdot \sum_{i=1}^{n} M_i + x \cdot \sum_{i=1}^{n} P_i, \quad 0 < x < 1 \tag{5}$$

IV. RESULTS

In this section, we present how to find the optimal number (N) of episodes to prefetch in order to achieve a high hit ratio which represents the potential of improving the response time.

We first show the potential of the prefetching scheme, followed by comparing the prefetching performances when different values of N are selected. The cost metric can be seen as a metric of measuring performance of each prefetching scheme regarding transport cost, QoE cost and so forth. It helps to make optimal prefetching decision when network condition changes.

A. The potential and benchmarks of prefetching

Before we get into depth of prefetching schemes, it is essential to evaluate the potential of prefetching. We assume a clairvoyant scheme that once an episode of each series has been watched by a user, the following episodes in that series which he will watch later can be 100% predicted and prefetched by the prefetching system. In this case, only the first requested video in each series cannot be predicted and preloaded. The precision of this prediction is 100%. In this case, the optimal recall equals 73% which means in principle, 73% of the clicks to new videos are predictable. This result suggests that, if all the episodes which are watched after the first one can be correctly predicted and stored in the local terminal cache, 73% of the requests to new videos will be served without delay. The corresponding F_1 score equals 0.84 which can be seen as the upper limit of the prediction effectiveness that our study can obtain based on this dataset.

In order to evaluate the performance of a prefetching scheme, the two extreme cases decribed in section III-B serve as benchmarks for comparison. First, we present the non-prefetching system which only has enabled terminal caching. The terminal cache yields a hit ratio of 13.77% which means even with an infinite terminal cache, only 13.77% requests can be served from the local cache. From this result, it shows a great potential of prefetching since terminal caching leaves over 85% of the requests unattended.

The other extreme case is to prefetch all the episodes in a series when an episode is watched, the cache hit ratio can reach up to 77%. This result is the upper limit of the hit ratio that the prefetching system can achieve in this study. When all the episodes are prefetched, the maximum recall value 73% is obtained. However, the precision is only 17% since the prefetching engine prefetched too much redundant data which users are not interested in. As a result, the effectiveness of this prefetching scheme is only 0.28. Clearly we need to intelligently select the content to prefetch and store.

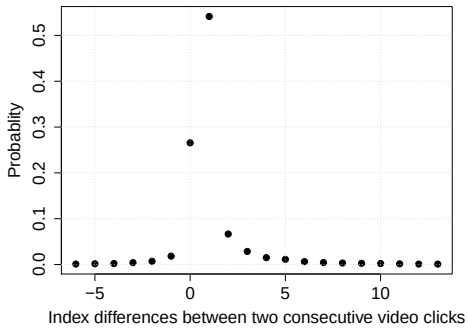


Figure 3. Transition probability of user requests within the same series

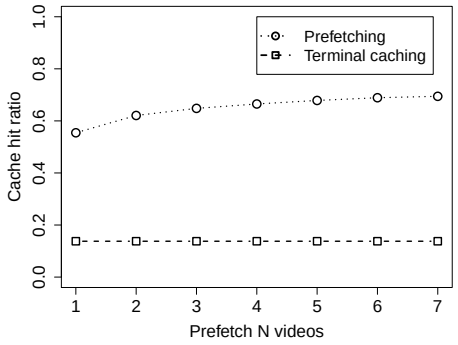


Figure 4. Hit ratios of terminal cache and prefetch

B. Prefetch N episodes

In order to avoid prefetching too much data and deteriorating network congestion, we propose to limit the number of videos to be prefetched by using a prefetching scheme, which only prefetches N videos in each series for each user.

Figure 3 shows the probability that a request for an episode n will be followed by a request for episode $n+k$ as a function of k. We find that for a user that has watched episode n of a series, the probablity of that user watches episode $n+1$ next is over 50%. Over our measurement period, there are about 26% of users that will not watch any episode after watching episode n. According to Figure 3, if we prefetch the videos with index of: $n+1$, $n+2$, $n+3$, $n-1$, $n+4$, $n+5$, and $n-2$, this will account for 95% of the requests for a next video.

Now we need to decide the value for N and which videos to prefetch. In Figure 4, we notice that when episode $n+1$ is prefetched, the hit ratio can reach up to 55% which is a big improvement comparing to the 13.7% hit ratio with terminal caching. To prefetch more videos besides $n+1$ and $n+2$ gives very little increase in hit ratio.

We also repeated the above analysis for different video categories. The request pattern of episodes in each video category

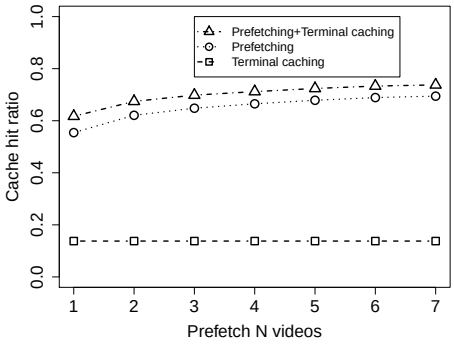


Figure 5. Hit ratio of combining prefetching and terminal caching

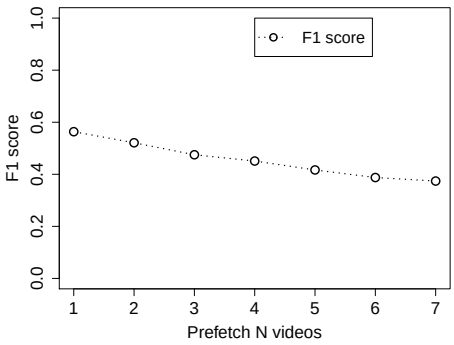


Figure 6. F1 score of prefetching system

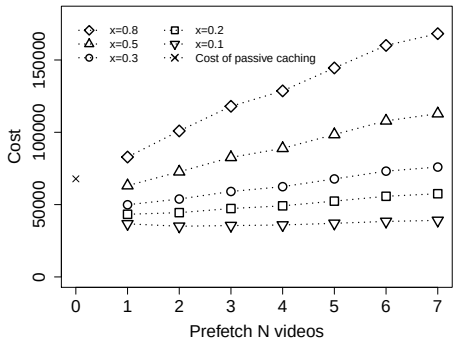


Figure 7. Total cost vs. Number of prefetched videos

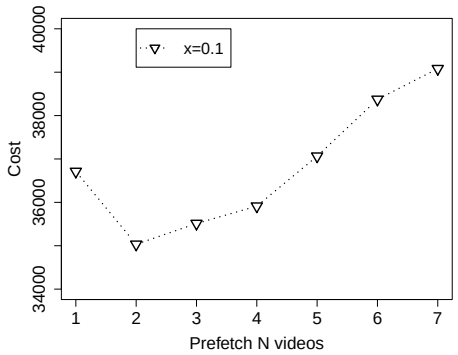


Figure 8. Total cost vs.Number of prefetched videos when cost factor = 0.1

were examined separately and we found that TV series and entertainment programs exhibit predictible consecutive request patterns while children's programs, documentaries and home and leisure time programs exhibit more random request pattern. When we prefetch videos according to the request pattern for each video category, the terminal hit ratio increase is about 1 percentage point comparing to the results in Figure 4. Even if the improvement is not significant, which may due to the limited number of videos in each category, the impact of customizing prefetched videos according to video category should be further investigated by using larger datasets.

Considering that prefetched videos and cached videos are two different sets, combining prefetching and terminal caching will increase the cache hit ratio, thus more content can be served from the cache with short start-up delay. Figure 5 shows the hit ratio improvement when prefetching and terminal caching are both adopted in a system. When a user requests a video which is not prefetched, the video will be downloaded from the server and at the same time cached at the user's device. By adding the passive terminal cache, hit ratio increases about 6 percentage points which represents the gain for not deleting the videos which are requested by users but not prefetched. In our study, we treat all requests of same video as cache hits. However, users may watch a part of a video and after some time continue to watch the rest of it. If we only treat the views after a user finishes watching the whole video as hits, the contribution from caching will be lower and the benefit brought by conventional passive terminal caching will be more limited.

Now, we present the effectivness of prefetching schemes with different values of N. Figure 6 shows the F_1 scores of the prefetching system when different N values are applied. In general, the more videos prefetched, the less accurate and less effective the prefetching system can be. However, the hit ratio increases when more videos are prefetched, as shown in Figure 4 and Figure 5. The decrease of effectiveness is caused by the amount of prefetched videos which are not requested by users. If the network condition is suitable to cope with these extra traffic, then a small hit ratio improvement with relatively large decrease of prefetching effectiveness is still profitable. Next, we apply the proposed cost model in the next section in order to quantify the cost of prefetching and to find the optimal number of videos to prefetch.

C. Cost model

In this section, we present the measurement of cost of prefetching as a performance metric to find the optimal number of videos to prefetch.

Each point on the curves in Figure 7 shows the prefetching cost value (see section III-C) versus the number of videos prefetched. Three curves represent the prefetching cost x in equation 5 equals 0.8, 0.5, 0.3, 0.2, and 0.1, respectively. The cross mark to the left shows the total cost of passive terminal caching when nothing is prefetched. Even though the top four curves show that the total cost of prefetching has a linear increase as more videos are prefetched, when off-peak downloading costs less than 30% of real time downloading, to prefetch up to 7 videos is still cheaper than passive caching. When off-peak downloading costs half of the real time downloading, to prefetch more than one video costs more than passive caching. But in this case, to prefetch only one video still outperforms passive caching. The curve of $x = 0.1$ is shown separately in Figure 8. An interesting phenomenon in Figure 8 is that the total cost of prefetching declines when N increases from 1 to 2. It suggests that when prefetching two

videos, the decrease of cost generated by video miss (M) is faster than cost increase generated by prefetching more videos. However, when N is larger than 2, the cost of prefetching starts to raise again. It indicates that there are more additional prefetched videos which are not used by users. When the cost of off-peak downloading is 10% of the cost of real-time downloading, to prefetch two videos yields the least cost.

V. TIME TO PREFETCH

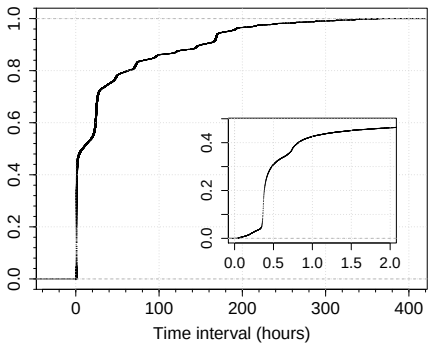
In this section, we estimate the available time for prefetching by measuring the time interval between two consecutive viewing sessions. The time between the start of a viewing session and the start of the next one is defined as the upper bound of the time to make prefetching decisions. The time between the end of a viewing session and the start of the next session serves as the lower bound of prefetching time. We differentiate the behaviour of watching the $n+1$ video and watching any video not in sequential order as regular view and irregular view correspondingly.

As is shown in Figure 9(a), two steep increases occur at 24h and 1 week respectively. Some programs are released on a daily or weekly basis and the increases suggest that a number of users follow these programs at the same pace. The inner graph shows the same CDF but zooms into two hours scale. It shows that 30% of regular views arrive within 20 minutes. Comparing with the results in Figure 9(b) for irregular views, within 20 minutes, only less than 15% of the requests are generated. Figure 10 shows the CDF of the lower bound time. In general, it shows similar trend as in Figure 9. The difference is shown in the inner figures, which suggests that about 40% of regular views are generated within 1 minute after the end of a viewing session and only 15% of the irregular views are generated within the same time period. It indicates that if we choose to prefetch videos at the end of the current session, we have rather limited time. As is shown in Figure 3, the risk of prefetching for the regular view is less. Thus, it is more reasonable to prefetch the next video in order during the current session. For irregular views, the request pattern shows that people watch episodes in the same program on daily basis, which leaves us longer time for prefetching and it can be delayed until off-peak time.

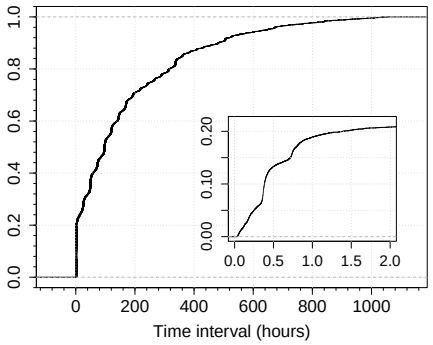
Finally, we perform the same analysis for different video categories and we focus particularly on the lower bound time in this part. We observe that for TV series, 40% of the requests of the $n+1$ video are within 1 minute and 30% of the requests within 24 hours. It suggests that the next episode can be either pre-downloaded during the current episode's playtime or be downloaded during off-peak time. The entertainment programs show a very similar pattern to the TV series. For children's programs, 60% to 70% of the requests are generated within 3 minutes no matter which episode is watched next. Similar patterns are observed for the videos of home and leisure time programs and documentaries. It suggests that in this case to prefetch during the video playback is more critical since user has a high probablity to immediately request another video after he watches the current one.

VI. CONCLUSIONS

In this paper, in order to explore the potential of reducing the start-up latency of streaming media serices, we have proposed a prefetching scheme and performed an analysis to evaluate its performance based on data from a Swedish TV-on-demand service.



(a) Regular views



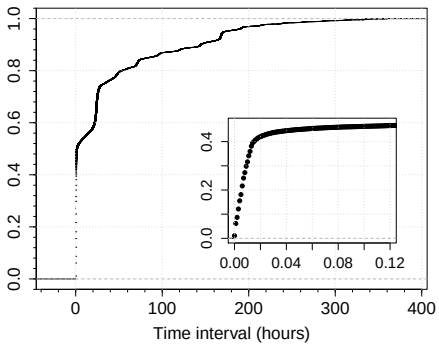
(b) Irregular views

Figure 9. CDF of time interval between two view events (Upper bound)

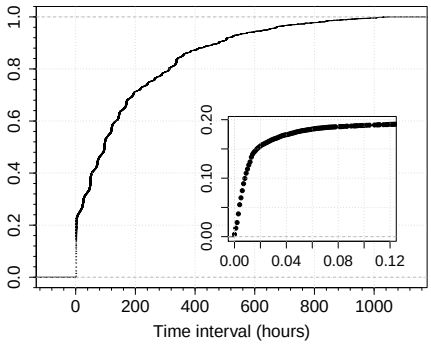
First, the paper has demostrated that in the ideal scenario with 100% prediction accuracy, 73% of the requests are predictable. It suggests a great potential for prefetching. We proposed to use the intrinsic structure of TV series in our data set and prefetch N adjacent videos to terminal devices. We found that the more videos are prefetched the higher hit ratio can be obtained which implicates that more requests can be served directly from the local cache with short delay. A cost model was proposed to quantify the cost of prefetching and to provide an optimal solution for prefetching. The result shows that prefetching two adjacent videos yields 62% hit ratio which is more than four times as terminal caching can obtain. We also demostrate that with a simple prefetching scheme as we proposed, 69% of all the requested videos can be correctly predicted which is very close to the ideal value 73% in this study. When prefetching costs 10% of the cost of real time downloading, to prefetch two videos costs the least which is the optimal choice of prefetching in this study. Moreover, prefetching combined with terminal caching can further improve the hit ratio. We also found that, the time for prefetching depends on user request patterns. For TV series, it is more reasonable to prefetch the next episode before the end of the current viewing session. For irregular requests, videos can be prefetched during off-peak hours. For programs which have a more random request pattern, like children's programs, it is better to make prefetching decisions during the current video playback time, even for irregular views.

In this work, only viewing sessions longer than 2 minutes can trigger prefetch. However, users may be less tolerant

 ISBN: 978-1-61208-396-4



(a) Regular views



(b) Irregular views

Figure 10. CDF of time interval between two view events (Lower bound)

to delays in short sessions than in long sessions. Thus, to prefetch for short session is worth to be further investigated. Another prediction limit of our study is the number of first seen episodes in each series. In future work, we plan to extend our research into cluster-based prefetching mechanisms to find user clusters and make prefetching decisions based on similar users' behaviour to even predict the first seen episode in each series.

ACKNOWLEDGMENT

This work has partly been financed by the Swedish Governmental Agency for Innovation Systems(VINNOVA) in the EFRAIM project and the NOTTS project. Maria Kihl is a member of the Lund Center for Control of Complex Engineering Systems(LCCC) and the Excellence Center Linkping - Lund in Information Technology(eLLIIT)

REFERENCES

[1] F. F.-H. Nah, "A study on tolerable waiting time: how long are web users willing to wait?" Behaviour & Information Technology, vol. 23, no. 3, 2004, pp. 153–163.

[2] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, "Web site delays: How tolerant are users?" Journal of the Association for Information Systems, vol. 5, no. 1, 2004, pp. 1–28.

[3] W. Ali, S. M. Shamsuddin, and A. S. Ismail, "A survey of web caching and prefetching," Int. J. Advance. Soft Comput. Appl, vol. 3, no. 1, 2011, pp. 18–44.

[4] J. Xu, J. Liu, B. Li, and X. Jia, "Caching and prefetching for web content distribution," Computing in Science Engineering, vol. 6, no. 4, July 2004, pp. 54–59.

[5] Y. Jiang, M.-Y. Wu, and W. Shu, "Web prefetching: Costs, benefits and performance," in Proceedings of the 7th international workshop on web content caching and distribution (WCW2002). Boulder, Colorado. Citeseer, 2002.

[6] J. Domenech, J. Sahuquillo, J. A. Gil, and A. Pont, "The impact of the web prefetching architecture on the limits of reducing user's perceived latency," in Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, ser. WI '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 740–744. [Online]. Available: http://dx.doi.org/10.1109/WI.2006.166

[7] J. Marquez, J. Domenech, J. Gil, and A. Pont, "Exploring the benefits of caching and prefetching in the mobile web," in Second IFIP Symposium on Wireless Communications and Information Technology for Developing Countries, 2008.

[8] M. Deshpande and G. Karypis, "Selective markov models for predicting web page accesses," ACM Trans. Internet Technol., vol. 4, no. 2, May 2004, pp. 163–184. [Online]. Available: http://doi.acm.org/10.1145/990301.990304

[9] X. Chen and X. Zhang, "Popularity-based ppm: an effective web prefetching technique for high accuracy and low storage," in Parallel Processing, 2002. Proceedings. International Conference on, 2002, pp. 296–304.

[10] D. Joseph and D. Grunwald, "Prefetching using markov predictors," SIGARCH Comput. Archit. News, vol. 25, no. 2, May 1997, pp. 252–263. [Online]. Available: http://doi.acm.org/10.1145/384286.264207

[11] Z. Ban, Z. Gu, and Y. Jin, "An online ppm prediction model for web prefetching," in Proceedings of the 9th Annual ACM International Workshop on Web Information and Data Management, ser. WIDM '07. New York, NY, USA: ACM, 2007, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/1316902.1316917

[12] G. Pallis, A. Vakali, and J. Pokorny, "A clustering-based prefetching scheme on a web cache environment," Computers & Electrical Engineering, vol. 34, no. 4, 2008, pp. 309–323.

[13] W.-G. Teng, C.-Y. Chang, and M.-S. Chen, "Integrating web caching and web prefetching in client-side proxies," Parallel and Distributed Systems, IEEE Transactions on, vol. 16, no. 5, May 2005, pp. 444–455.

[14] Z. Su, Q. Yang, and H.-J. Zhang, "A prediction system for multimedia pre-fetching in internet," in Proceedings of the Eighth ACM International Conference on Multimedia, ser. MULTIMEDIA '00. New York, NY, USA: ACM, 2000, pp. 3–11. [Online]. Available: http://doi.acm.org/10.1145/354384.354394

[15] C. Bouras, A. Konidaris, and D. Kostoulas, "Predictive prefetching on the web and its potential impact in the wide area," World Wide Web, vol. 7, no. 2, 2004, pp. 143–179.

[16] S. Khemmarat, R. Zhou, D. K. Krishnappa, L. Gao, and M. Zink, "Watching user generated videos with prefetching," Image Commun., vol. 27, no. 4, Apr. 2012, pp. 343–359. [Online]. Available: http://dx.doi.org/10.1016/j.image.2011.10.008

[17] D. K. Krishnappa, S. Khemmarat, L. Gao, and M. Zink, "On the feasibility of prefetching and caching for online tv services: A measurement study on hulu," in Proceedings of the 12th International Conference on Passive and Active Measurement, ser. PAM'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 72–80. [Online]. Available: http://dl.acm.org/citation.cfm?id=1987510.1987518

[18] Z. Zeng and B. Veeravalli, "Hk/t: A novel server-side web caching strategy for multimedia applications," in Communications, 2008. ICC '08. IEEE International Conference on, May 2008, pp. 1782–1786.

[19] Z. Zeng, B. Veeravalli, and K. Li, "A novel server-side proxy caching strategy for large-scale multimedia applications," J. Parallel Distrib. Comput., vol. 71, no. 4, Apr. 2011, pp. 525–536. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2010.06.008

[20] S. Seny, J. Rexfordz, 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, Mar 1999, pp. 1310–1319 vol.3.

[21] W. Liu, C. T. Chou, Z. Yang, and X. Du, "Popularity-wise proxy caching for interactive streaming media," in Local Computer Networks, 2004. 29th Annual IEEE International Conference on, Nov 2004, pp. 250–257.

[22] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in Proceedings of the 27th European Conference on Information Retrieval, 2005, pp. 345–359.

 ISBN: 978-1-61208-396-4