Prefetching Schemes and Performance Analysis for TV on Demand Services

Du, Manxing; Kihl, Maria; Arvidsson, Åke; Zhang, Huimin; Lagerstedt, Christina; Gavler, Anders

Published in:
International Journal on Advances in Telecommunications

2015

Prefetching Schemes and Performance Analysis for TV on Demand Services

Manxing Du*†, Maria Kihl†, Åke Arvidsson‡§, Huimin Zhang¶, 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
¶Uppsala University, Sweden, Email: firstname.lastname.4997@student.uu.se

***Abstract*—TV-on-Demand services have become one of the most popular Internet applications that continuously attracts high user interest. With rapidly increasing user demands, the existing network conditions may not be able to ensure a 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, two datasets from different IPTV providers are used to analyse the TV program request patterns. According to the results, we propose a prefetching scheme at the user end to preload videos before user requests. For both datasets, our prefetching scheme significantly improves the cache hit ratio compared to passive 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. We also discuss if there is enough time for prefetching. Finally, more factors, which may have an impact on optimizing prefetching performance, are further discussed, such as the jump patterns over different time in a day and the the distribution of each video's viewing length.**

Keywords–TV-on-Demand services; user perceived latency; prefetching; jump patterns over time; viewing fractions;

I. INTRODUCTION

Internet has become a popular medium for distributing multimedia content like TV shows, movies, and user generated videos besides the tradional distribution channels such as air broadcasting, cable networks and physical media like Video Home System (VHS) or Digital Versatile Disc (DVD). The massive amount of multimedia traffic has imposed a significant burden on the Internet. Consequently, users sometimes have to endure long access delays for filling up the playout buffer before the content is displayed. Although web caching is widely used as a solution to lessen the web traffic congestion and improve network performance, the benefit of caches is limited. To further reduce user perceived latency, prefetching has become a popular technique. The objective of the prefetching system is to proactively preload certain content to the cache even before the user requests it. We have explored the subject previously in [1] and this paper extends the analysis and discussion presented previously.

Understanding the usage of IPTV services is very important when investigating prefetching schemes. In [2], the usage of a Peer-to-Peer IPTV service which includes both live and VoD content is presented. The daily VoD content receives more requests than live content. Videos belonging to different genres have different temporal distributions of popularity. In [3] [4] [5], the user behaviour for a VoD system and an IPTV service is investigated thoroughly in many aspects, such as user access rate, channel switching patterns, video popularity and so on. In [3], the impact of recommendations on the user viewing behaviour by recommending two sets of movies to the users is discussed. The result shows that recommending daily popular videos has a much more significant impact on the users choices than recommending popular videos over a longer period (15 days), which suggests that VoD content has short life time. We also know that the user's tolerance for downloading web pages is short. The results in [6] show that it is approximately 2 seconds and in [7], the result suggests that the more familiar the users are with a web site, the more sensitive they are to delays.

Thorough summaries of web caching and prefetching approaches and performance measures can be found in [8] [9] [10]. Domènech *et al.* in [11] compare different prefetching architectures and find that a 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 by more than 97.2%. However, the results are obtained based on the ideal scenario that the prediction is always correct. Thus, the results can be seen as the upper limit of latency reductions.

Most of the existing prefetching approaches are access-history based which predict the future user requests depending on the observed content access patterns. Márquez *et al.* applies a double dependency graph prediction algorithm to a mobile web and observes that the performance of prefetching approaches rely on the underlying networking technologies [12]. 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 [13]–[16]. The prefetching schemes proposed in [17]–[20] use data mining techniques to discover the users' access patterns.

Popularity-based prefetching approches are also widely used, especially for prefetching multimedia content. In [21], 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, a hit ratio of 80.88% can be obtained and while requiring only a 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.* [22] apply a prefetching top-100 video scheme on the Hulu traffic trace, one of the most popular streaming media provider in America, from a campus network and compare the performance of prefetching with conventional proxy caching. The results prove that prefetching

is very effective for online TV services.

From these papers, we note that the research focus is mainly on proxy prefetching whereas the performance of terminal prefetching is less well known. Generally, the closer the content is to the users, the shorter the delay. To the best of our knowledge, our analysis is the first that focuses on using terminal storage to do prefetching for a TV-on-demand service.

In this study, we have used two datasets from different TV-on-demand services. Each TV-on-Demand program consists of a series of episodes which have a high consistency regarding content, thus, the index of each episode is a good indicator of the user's future access. We propose to use the intrinsic structural information of the episodes belonging to a specific 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 datasets, followed by an investigation into the optimal choice of prefetching. We show that a high terminal gain can be obtained by prefetching two adjacent episodes in a series for each viewed episode with minimum cost. This study also shows the potential of implementing terminal prefetching for TV-on-Demand service both effectively and economically. In addition, we investigate whether there is enough time to perform prefetching and detail possible improvements for making prefetching decisions.

The remainder of the paper is organized as follows. Section II describes the infrastructure of a prefetching system and the evaluation metrics used. Our datasets 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 in Section VI, factors which may have impact on improving the prefetching system are considered. At the end, we conclude in Section VII.

II. VIDEO PREFETCHING SYSTEM

By implementing a prefetching system, multimedia streams can be retrieved from a 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: the prediction engine and the prefetching engine. The prediction engine is responsible for forseeing what the users will watch next before they request the content. The prefetching engine proactively stores the video prefix to the local cache. In this case, both 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: at the terminals [21], at the proxies [18] [22], at the content servers [23] [24] or between these elements [11] [12].

In order to bring content closer to the end users, so as to the largest extent reduce their perceived latency, we assume that 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

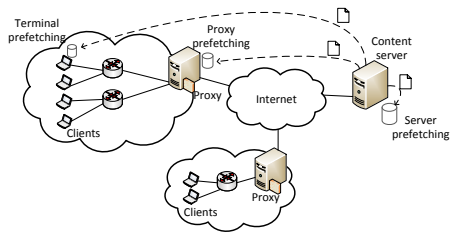


Figure 1. Network architecture with prefetching system

can be prefetched. Typically, to reduce the start-up latency, prefetching only part of the video stream is sufficient. 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 the users may not always favour the beginning of each video [25] [26], the performance of caching different parts of the videos is discussed. Since the total length of each video is unknown and the starting and ending points of each request are also not available, in this paper, we do not address which portion of each video should be prefetched. Instead, each video is treated 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 it has, the content will be served from the local cache instead of from the remote server. In Section VI, we made an assumption for video length and discussed the possible impact that viewing length proportion may have on improving the prefetching performance.

B. Evaluation metrics

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

$$P = \frac{v}{p} \tag{1}$$

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 (v) over the total number of requested videos (r).

$$R = \frac{v}{r} \tag{2}$$

A higher recall value 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, a higher recall value can 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 requested by the users. However, in this case, p will also increase. Consequently, the prediction engine's precision will decrease. To prefetch a great amount of data, which will not be requested by the users will also deteriorate

the network congestion. These two metrics need to be balanced when designing a prefetching system. The F_1 score (balanced F-score) [27] is a weighted average of the precision and recall, which is used in this study to show the effectiveness of a prediction. The closer the 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 is highly demanded by the users, we introduce the last metric, which is the cache hit ratio (H). It is defined as the number of requests for videos (h), which are retrieved from the prefetching cache over the total number of requests (t).

$$H = \frac{h}{t} \tag{4}$$

The 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 prefetched content.

III. Experiments

In this section, we will describe the experiments conducted using the prefetching method in this paper, which is to prefetch N adjacent episodes for each viewed episode. 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. Furthermore, we applied the prefetching scheme on another dataset to compare and validate the results.

A. Dataset

Our study is firstly conducted using the video requests from one of the most popular Swedish TV providers (*dataset 1*). The data was collected by Conviva who 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 have access to all the online TV content provided by the TV channel by paying a monthly fee. Thus, the users who do not have subscriptions are not included in this study. 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* categories for statistical analysis 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, we include TV-on-demand content categorized as *Children, Documentary, TV series, Home and leisure time* and *Entertainment* 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 for programs with only one season available.

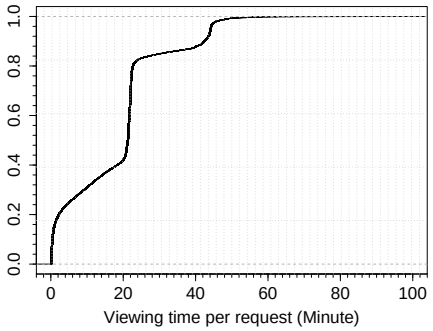


Figure 2. CDF of viewing time per request

TABLE I. Example of user request from Swedish TV provider

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

Table I shows an example of the available information contained in each data entry. There are 7933 subscribed users who generated 104845 requests for 2427 videos belonging to 253 series over a period of 11 weeks from December 31, 2012 to March 17, 2013. Figure 2 depicts the Cumulative Distribution Function (CDF) of the viewing time per request. Around 20% of the total requests result in viewing times shorter than 2 minutes. We filter out these short viewing sessions to eliminate the impact of random clicks by the users, which are not suitable to serve as predictors. We also note that the 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 of the dataset.

To test the proposed prefetching scheme on one more dataset and compare the results, a second dataset from a Portuguese catch-up TV service (*dataset 2*) is used. This dataset contains one month of request logs from its subscribers. The data was collected from June 1 to June 30 in 2014. Unlike traditional TV-on-demand services, this catch-up TV service has a 7 days access window for the content. The dataset contains over 17 million requests from about 570,000 users and more than 80 TV channels. Table II shows an example of a request log. Each video and each TV channel is identified by a unique ID, namely EpgPID and StationID. In contrast to *dateset 1*, the viewing time of each video is unknown. Since the category of TV programs provided by different channels varies, one of the most popular channels, which mainly contains TV series was chosen in the following study in order to be comparable with *dataset 1*. This channel has more than 3 million requests from about 275,000 users during the recording period.

B. User access patterns

In this section, traffic patterns for both datasets on different time scales are presented. To make it comparable with *dataset*

TABLE II. EXAMPLE OF USER REQUEST FROM CATCH-UP TV PROVIDER

Title	Belmonte - Ep. 192
User Id	A0573D6D4F9D7BC5018B3D17DC6DCB3
Start time	2014-06-07 09:31:09
Program release time	2014-06-03 21:51:00
Program end time	2014-06-03 22:47:00
EpgSeriesID	24952
EpgPID	6373423
StationID	327398

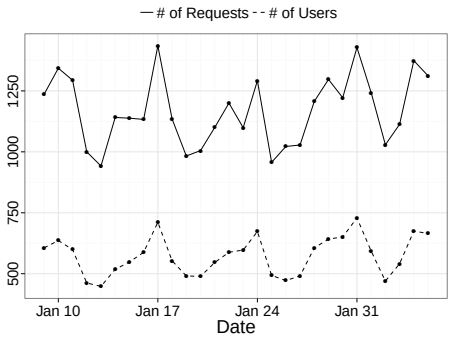


Figure 3. Daily pattern of dataset 1

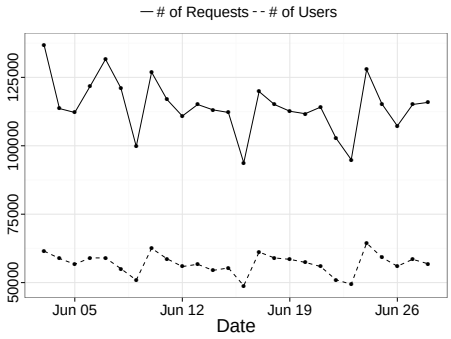


Figure 4. Daily pattern of dataset 2

2, one month of data from *dataset 1* was selected. As is shown in Figure 3 and Figure 4, on average, there are over 500 users online per day in *dataset 1* and over 50,000 users use the catch-up TV service in *dataset2* per day. Unlike the request pattern of VoD services such as YouTube in [28] in which the requests distributed evenly on weekdays, an interesting phenomenon here is that the number of users and the number of requests peaks on Thursdays in *dataset 1* and on Tuesdays in *dataset 2* and then drop rapidly. In *dataset 1*, each weekend has the least number of users and requests while in *dataset 2* users, the lowest usage is seen on Mondays.

Figure 5 and Figure 6 show the total number of requests and the total number of users for each hour of the day averaged over one month for both datasets. In *dataset 1*, peak hours are from 20:00 to 23:00 regardless of whether it is a weekday or a weekend. However, in *dataset 2*, multiple request peaks can be observed during the day. The peaks arrive earlier on weekends than on weekdays.

C. Video prefetching selection methods

In order to implement prefetching, the prediction engine needs to determine what content should be preloaded based on

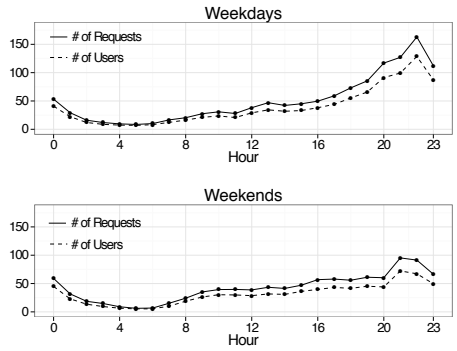


Figure 5. Average diurnal traffic patters in dataset 1

the user's viewing history of the same series. In the following, we propose and evaluate a scheme for how the prediction engine should make prefetching decisions.

In this study, we consider two extreme cases as baselines for evaluating the performance of the proposed prefetching scheme. One case is prefetching all of the available episodes in a series to the end user when the user has watched one 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 the users' viewing histories and collect unique video sets for each series. We assume that the videos in each set are all the available videos for 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 using conventional terminal caching only, which means passively caching all of the user demand and preloading nothing prior to requests from the user. To distinguish this from prefetching, passive caching is used in the following. When passive caching is enabled, repeated requests for videos ,which have not been prefetched previously, can be served directly from the local cache. Passive caching can help to reduce initial delay only if the cached video is requested more than once.

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

D. Cost model

Any prefetching scheme will make incorrect predictions and inevitably download more content than a system without prefetching. Consequently, the traffic overhead caused by prefetching may impose a burden on a bandwidth sensitive network. A 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 demands and potentially reduces the user perceived latency. Sometimes spare network capacity is available during off-peak hours, e.g, during nights. It can be

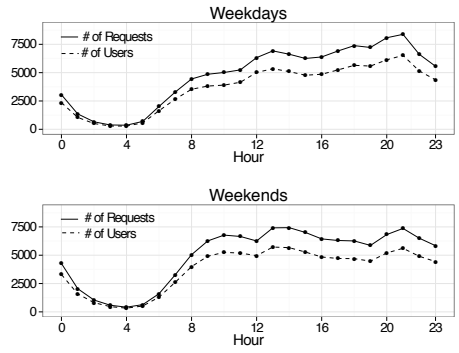


Figure 6. Average diurnal traffic patters in dataset 2

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 an optimal number of videos to prefetch.

We assume that the cost of real time video delivery equals 1 monetary unit per video. The cost of off-peak prefetching is x monetary units per video where x is smaller than 1 since prefetching can be done during off-peak hours costing less than real time downloading. In addition, 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 which is not prefetched as a video miss (M). The number of videos, which are prefetched by each user, is P. The subscript i indicates each user. The cost of prefetching for *k* users will then be:

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

IV. Results

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

We first use *dataset 1* to calculate the potential of the prefetching scheme. Then we compare the prefetching performances when different values of N are selected. The cost metric can be seen as a metric for measuring the performance of each prefetching scheme regarding transport cost, QoE cost and so forth. This aids in making optimal prefetching decisions when network condition change. Finally, we apply the same prefetching analysis to *dataset 2* for comparison.

A. The potential and benchmarks of prefetching

Before we go further into prefetching schemes, it is essential to evaluate the potential of prefetching.

In *dataset 1*, we assume a clairvoyant scheme. This means that once an episode in each series has been watched by a user, the following episodes in that series, which the user watches at a later time, 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%. The optimal recall in this scenario equals 73% calculated by equation (2), which means that in principle, 73% of the clicks to new videos are predictable. The result suggests that if all the episodes that 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 that can be seen as the upper limit of the prediction effectiveness that our study can obtain based on *dataset 1*.

In order to evaluate the performance of a prefetching scheme, the two extreme cases described in Section III-C serve as benchmarks for comparison. First, we present the non-prefetching system, which only has passive caching enabled. The passive cache yields a hit ratio of 13.77% that means even with an infinite local cache, only 13.77% requests can be served locally. The result shows a great potential of prefetching since passive 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. In this case, the cache hit ratio can reach up to 77%, which 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 of 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 which content should be prefetched and stored.

B. Prefetch N episodes

In order to avoid prefetching too much data causing 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 7 shows the probability that a request for episode n will be followed by a request for episode $n+k$ as a function of k. We find that for a user who has watched episode n of a series, the probablity of that user watching episode $n+1$ next is over 50%. The number 0 on the x axis denotes that during the measurement period, there are approximately 26% of the users who will not watch any episode after watching episode n. According to Figure 7, prefetching videos with index of: $n+1$, $n+2$, $n+3$, $n-1$, $n+4$, $n+5$, and $n-2$ will account for 95% of the requests for a next video. Now, we need to decide the value of N and which videos to prefetch. In Figure 8, $N = 0$ represents passive caching when no videos are prefetched. We notice that when episode $n+1$ is prefetched, the hit ratio can reach 55%, which is a big improvement compared to the 13.7% hit ratio, which is reached using passive caching. To prefetch further videos after $n+1$ and $n+2$ gives very little increase in the hit ratio.

We also repeated the above analysis for different video categories. The request pattern for episodes in each video category were examined separately and we found that for the categories TV series and entertainment, the programs exhibit predictible consecutive request patterns while for the categories children's programs, documentaries and home and leisure time programs, a more random request pattern is exhibited. When we prefetch videos according to the request pattern for each video category, the hit ratio increase is about 1 percentage point compared to the results in Figure 8. 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

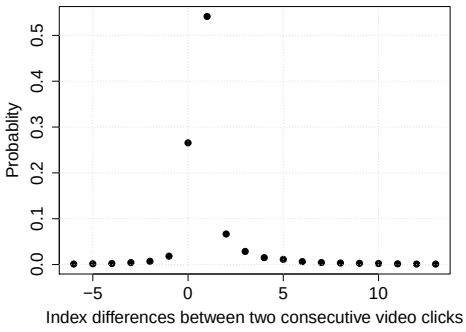


Figure 7. Transition probability of user requests within the same series in dataset 1

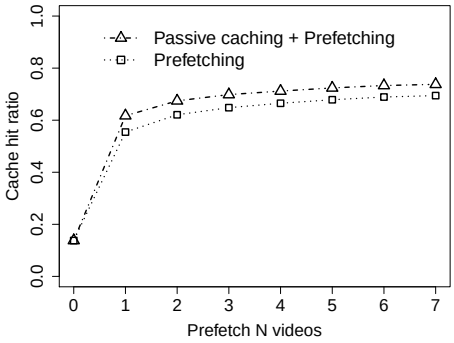


Figure 8. Hit ratios of terminal cache and prefetch in dataset 1

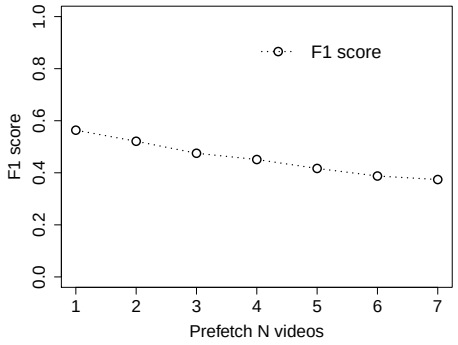


Figure 9. F1 score of prefetching system

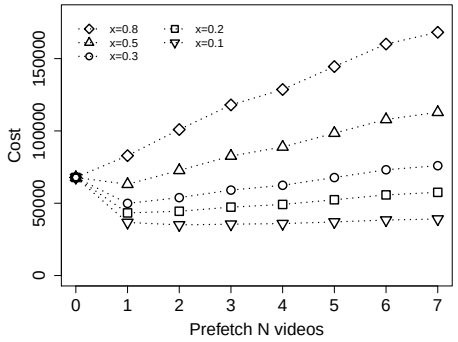


Figure 10. Total cost vs. Number of prefetched videos

according to video category should be further investigated by using larger datasets. In *dataset 2*, each video is tagged with multiple categories, thus, the categorization is too ambiguous to categorize videos for conducting a similar study.

In the lower curve in Figure 8, the videos, which are not prefetched, are not passively cached neither. They will be downloaded from the server during the playback and not stored locally. As is shown in the upper curve in Figure 8, the hit ratio increases about 6 percentage points which represents the gain for passively caching these videos. In our study, we treat all requests for the same video as cache hits. However, the users may watch part of a video and after some time continue to watch the rest. 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 caching will be more limited.

Now, we present the effectivness of prefetching schemes using varying values of N. Figure 9 shows the F_1 scores of the prefetching system when different N values are applied. In general, the more videos are prefetched, the less accurate and less effective the prefetching system will be. However, the hit ratio increases when more videos are prefetched, as shown in Figure 8. The decrease in effectiveness is caused by the amount of prefetched videos, which are not requested by users. If the network conditions are suitable to cope with this extra traffic, then a small hit ratio improvement with relatively large decrease of prefetching effectiveness is still profitable.

C. Cost

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

Each point on the curves in Figure 10 shows the prefetching cost value (see Section III-D) versus the number of videos prefetched. The five curves represent the prefetching cost x in equation 5 equaling 0.8, 0.5, 0.3, 0.2, and 0.1, respectively. The total cost of passive caching when nothing is prefetched is shown when $N = 0$. Even though the top four curves show that the total cost of prefetching has a linear increase as more videos are prefetched, prefetching up to 7 videos is still cheaper than passive caching when off-peak downloading costs less than 30% of real time downloading. When off-peak downloading costs half of real time downloading, prefetching more than one video costs more than passive caching. However, in this case, prefetching only one video still outperforms passive caching. The cost for $x = 0.1$ is shown separately in Figure 11. An interesting phenomenon in this figure is that the total cost of prefetching declines when N increases from 1 to 2. This suggests that when prefetching two videos, the decrease in cost generated by a video miss (M) is larger than the cost increase generated by prefetching more videos. However, when N is larger than 2, the cost of prefetching starts to rise again. This 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, prefetching two videos yields the lowest cost.

D. Comparison

A similar study was conducted for *dataset 2*. First, we computed each user's requests and plotted the accumulated fraction of requests and the fraction of users. As can be seen in Figure 12, approximately 30% of the users generated 80% of requests. We define these 30% of the users in *dataset 2* as active users and in the following study, only the data from these active users is included.

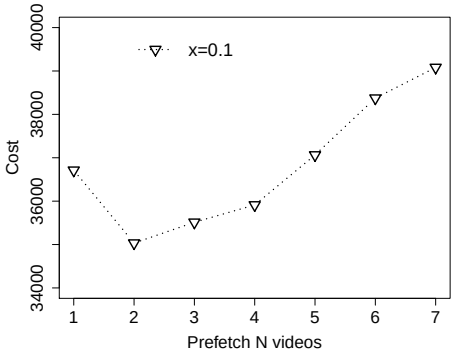


Figure 11. Total cost vs.Number of prefetched videos when cost factor = 0.1

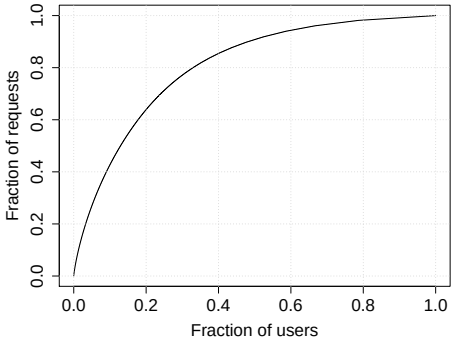


Figure 12. The distribution of requests between users

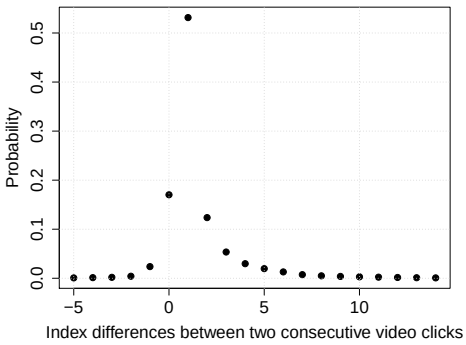


Figure 13. Transition probability of user requests within the same series in dataset 2

As was done for *dataset 1*, we examined the probability of what the user would watch next by using one month of data in *dataset 2*. As is shown in Figure 13, if episode n is watched by a user, then episodes with index of: $n+1$, $n+2$, $n+3$, $n-1$, $n+4$ have higher probabilities to be watched next. The probability that after watching episode n, no more episode from the same series will be requested by the user is nearly 20%.

Next, based on the request pattern, we applied our prefetching scheme on *dataset 2*. In Figure 14 is shown that if nothing is prefetched ($N = 0$), the hit ratio of passive caching is on average 27%. Prefetching the next episode, the hit ratio can reach up to 53%. With more than one episode prefetched, the hit ratio increase is more significant when two episodes with index $n+1$ and $n+2$ are prefetched. Prefetching 5 episodes for each request, a hit ratio of nearly 80% can be obtained, which is 3 times more than with passive caching. Even though the content and the group of users are different from those in

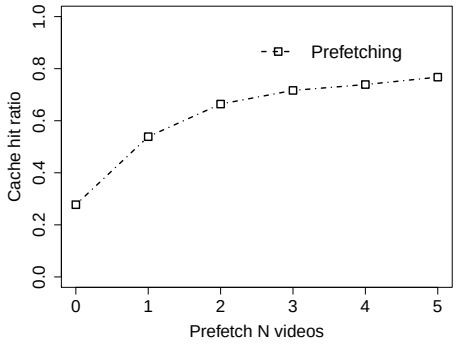


Figure 14. Hit ratios of terminal prefetching in *dataset 2*

dataset 1, our prefetching scheme can still be applied to *dataset 2*. From the results of both datasets, prefetching one episode can already greatly improve the cache hit ratio.

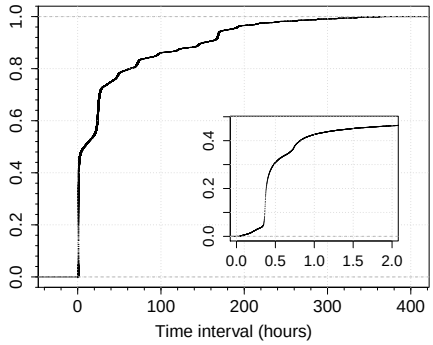
V. TIME TO PREFETCH

In this section, we estimate the available time for prefetching by measuring the time interval between two consecutive viewing sessions. Since in *dataset 2*, the length of each viewing session is unknown, we use only *dataset 1* in the following study.

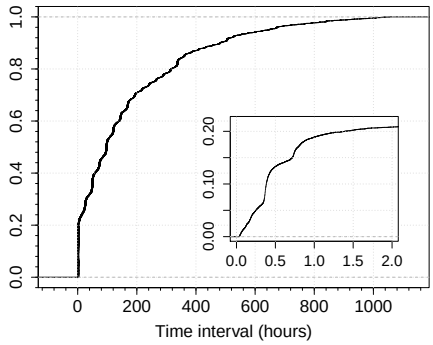
The time between the start of a viewing session and the start of the next one is defined as the upper bound for the time to make prefetching decisions. The time between the end of a viewing session and the start of the next session serves as the lower bound for the prefetching time. We differentiate between the behaviour of watching the $n+1$ video and watching any video not in sequential order and denote them as sequential views and non-sequential views correspondingly.

As is shown in Figure 15(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 in the first two hours. This shows that 30% of the sequential views arrive within 20 minutes. Compared with the results in Figure 15(b) for non-sequential views, only approximately 15% of the requests are generated within 20 minutes. Figure 16 shows the CDF of the lower bound time. In general, it shows a similar trend as the one in Figure 15. The difference is seen in the inner figures, which suggests that about 40% of sequential views are generated within 1 minute after the end of a viewing session and only 15% of the non-sequential views are generated within the same time period. This indicates that if we choose to prefetch videos at the end of the current session, we have a limited time frame. As is shown in Figure 7, the risk of prefetching for sequential views is lower. Thus, it is more reasonable to prefetch the next video in order during the current session.

Prefetching for sequential views means prefetching one episode only beforehand, whereas prefetching for non-sequential views means prefetching more episodes after the next one in order. In this case, for example, when we decide to prefetch 2 episodes and episode n has already been requested, then episode $n+1$ and $n+2$ will be prefetched accordingly. If the user watches episode $n+1$ next, this requires episodes $n+2$ and $n+3$ to be prefetched, since episode $n+2$ is already

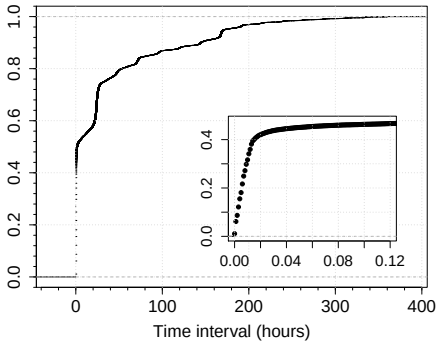


(a) Sequential views

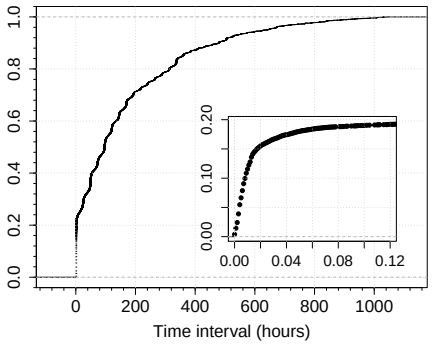


(b) Non-sequential views

Figure 15. CDF of time interval between two view events (Upper bound)



(a) Sequential views



(b) Non-sequential views

Figure 16. CDF of time interval between two view events (Lower bound)

stored in the cache. In this case there is a longer time period available to prefetch the second episode. For non-sequential views, the request pattern shows that people watch episodes from the same program on daily basis. This leaves us a longer time period for prefetching and we can delay until off-peak time.

Finally, we perform the same analysis for different video categories focusing particularly on the lower bound time. We observe that for TV series, 40% of the requests for the $n+1$ video come within 1 minute and 30% of the requests within 24 hours. This 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 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, prefetching during the video playback is more critical since the user has a high probablity to immediately request another video after watching the current one.

VI. Discussion

In this section, two factors that have potential to facilitate prefetching decisions are discussed. All the analysis in this section is based on *dataset 1*. We first investigate if the time of day can be used as a factor to adjust prefetching decisions. The other factor we discuss is the fraction of viewing length of a video. It implicitly reflects the users interests, which may be used as an indicator to predict the user's next move.

A. Temporal pattern of jumps

In this section, we examine whether a user's watching behaviour shows any preference depending on the time of day. Each day is divided into 5 time periods where period 1 is from 00:00 to 6:00, period 2 is from 6:00 to 12:00, period 3 is from 12:00 to 18:00, period 4 is from 18:00 to 21:00, and period 5 is from 21:00 to 00:00. The user's behaviour is categorized using *-1.0:* to denote watching the previous episode, *0:* to denote not watching any further episode within the same series, *1.0:* to denote watching the next episode and *IG:* to denote watching any other episodes within the same series. First, we plot the request distribution within each period in Figure 17. Not surprisingly, the most probable behaviour within each time period is to continue watching the next episode. The second highest possibility is not to request any further episode from the same series. Figure 18 shows the viewing behaviour distribution for the time periods as defined above. For instance, watching the previous episode is most likely to happen between 12:00 and 18:00 (time group 3) than during other time periods during the day. All the other behaviours are more preferable between 21:00 and 00:00 (time group 5), which is also the time of day when the users are most active. From Figure 17 and Figure 18, we can conclude that the jumps within each series do not show a clear temporal pattern in our dataset. Our observation confirms the results in [2], which suggest that if users watch an episode published in the late evening, they request further episodes from the same series in late afternoon of the following day. Thus, within time groups 3 to 5, the users may request episodes randomly from the same series or the next episode in order if it has been published.

Since watching next episode is the most probable be-

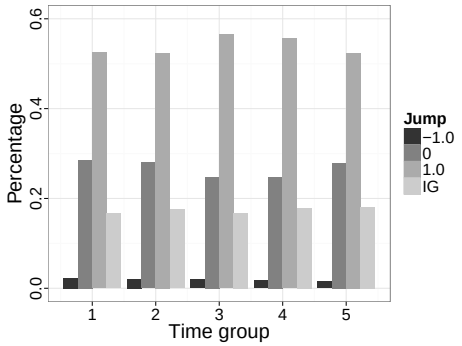


Figure 17. Jumps distribution within time periods

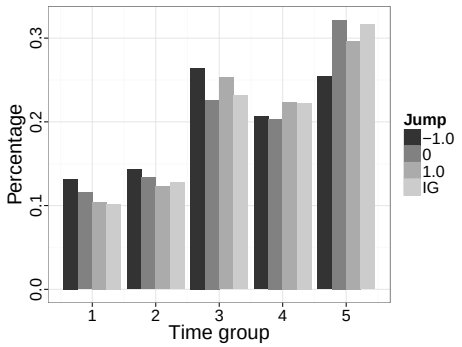


Figure 18. Jumps distribution over time periods

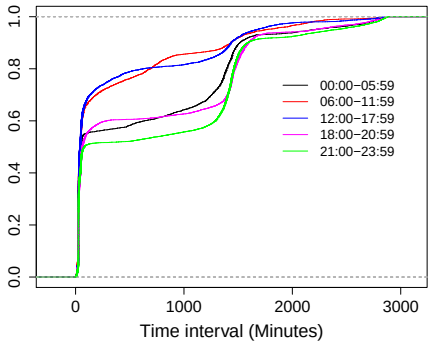


Figure 19. CDF of time interval watching n+1 video

haviour, we further investigate the time interval between the requests for episodes n and $n+1$. In Figure19, the longest time interval included in the graph was set to be two days and the CDF shows the cumulative distribution for the different time groups. It is clear that most requests for the $n+1$ episode are generated within the time period from 6:00 to 18:00 compared to the rest of the day. The three curves in the lower part of the figure show that nearly 40% of the requests for the next episode are generated within 24 hours. One explanation can be that people watch an episode regularly every day at a specific time and watch the next one the next day. The time of request also depends on the video availability. For instance, some series publish new episodes on a daily basis. Therefore, before new episodes are released, other older episodes from the same series have a larger chances of being requested. If the next episode has already been published, it accounts for 50% of the requests for the next episode being generated within a short time. However, in *dataset 1*, the video release time is unknown.

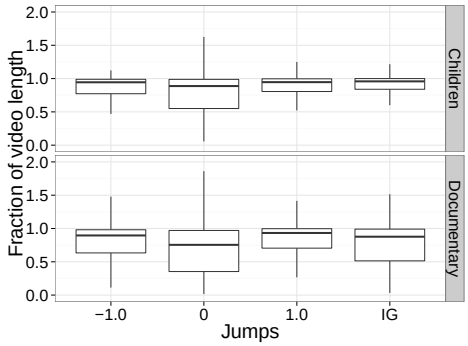


Figure 20. Boxplot of viewing length vs. jumps Part 1

B. *Viewing length distribution*

In the previous section, it was shown that the user viewing behaviour is almost evenly spread out across different time periods during the day. Having a model that shows whether the user behaviour is correlated with the viewing length of the current episode can help the prefetching system to make dynamic prefetching decisions as the video playback time accumulates. In this section, we investigate whether the viewing length of the current episode can be used as an indicator to predict what the user will watch next. Since the total video length is unknown in *dataset 1*, we assume that for each episode, at least one of the users has watched it until the end. Thus, the maximum viewing length of each episode over all the users who have requested it is estimated as the length of the episode.

We also investigate the viewing time fractions versus the watching behaviour for the different video categories. The viewing time of one episode is calculated by adding together the viewing length of consecutive requests for the same episode before the user requests the next episode. As can be seen in Figure 20, for childrens program, the users usually watch one episode until the end before requesting another one. However, if the user terminates the video before watching half of it, the user will not request any more episodes from the same series during the measurement period. Another interesting observation is, if a user watching the same episode more than once (the fraction of video length is larger than 1), then there is a higher probability that the user is only interested in this particular episode and will not watch anything more from the same series. Very similar behaviour can be observed for the documentary category. When users abort at the very beginning of the video playback, there is a higher probability that the user will not watch the next episode in order. Instead the user may browse other episodes from the same series. The short view time implies a low user interest, which is followed by a random viewing behaviour. The other three video categories have very similar patterns. The plot for TV series is shown in Figure 21 as an example. As the viewing time increases, it is hard to tell what the user's next move will be using only the viewing time as a factor.

VII. Conclusions

In this paper, we have proposed a prefetching scheme and performed analysis to evaluate its performance based on a dataset from a Swedish TV-on-demand service (*dataset 1*) in order to explore the potential of reducing start-up latency of streaming media services. The same method is applied to a

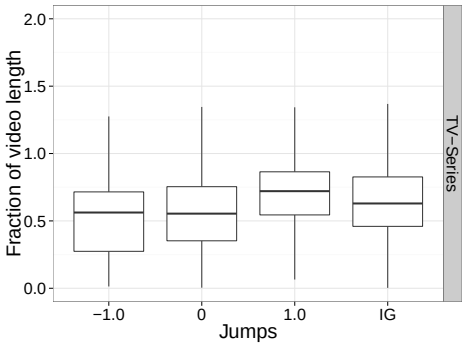


Figure 21. Boxplot of viewing length vs. jumps Part 2

second dataset from a catch-up TV service, (*dataset 2*), for comparison purposes and further analysis.

First, the paper has demonstrated that for *dataset 1,* 73% of the requests are predictable in an ideal scenario with 100% prediction accuracy. This suggests a great potential for prefetching. We propose to use the intrinsic structure of TV series in our dataset and prefetch N adjacent videos to terminal devices. A cost model is 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 much as terminal caching can obtain. We also demonstrate that with the simple prefetching scheme we propose, 69% of all the requested videos can be correctly predicted, a number which is very close to the ideal value of 73%. When prefetching costs 10% of the cost of real time downloading, prefetching two videos has the lowest cost and is the optimal choice for prefetching based on the dataset analyzed in this study. By comparing the prefetching results using two different datasets, we find that the more videos that are prefetched, the higher is the obtained hit ratio. This implicates that more requests can be served directly from the local cache with a short delay. Even though the two datasets we used are from different regions and different types of TV services, the prefetching scheme can easily be applied to both of them and the hit ratio improvement is significant.

We also found that the time of day suitable for prefetching depends on the user request patterns. For TV series, it is more reasonable to prefetch the next episode before the end of the current viewing session. For non-sequential requests, videos can be prefetched during off-peak hours. For programs, which have a more random request pattern, such as children's programs, it is better to make prefetching decisions during the current video playback time, even for non-sequential requests.

Two more factors are discussed as possible improvements for making prefetching decision. The first one is that user viewing behaviour does not show any strong preference regarding the time of a day. Approximately 10% more requests for the next episode is generated within a short time during the day from 6:00 to 18:00. From 18:00 to 0:00, the later during the day that the user watches the current episode, the higher probability that he will request the next episode in order within 24 hours. We also studied whether the viewing length of each video has any influence on what the user would watch next. For future work, more factors such as the number of episodes from the same series, which have already been watched by the user should also be considered to improve the prediction accuracy and hit ratio.

In this work, only viewing sessions longer than 2 minutes can trigger prefetch. However, users may be less tolerant to delays in short sessions than in long sessions. Thus, using prefetching to improve performance for these short sessions is worth being further investigated. Another prediction limit of our study is the number of first seen episodes in each series. In future work, we plan to use *dataset 2* to extend our research into cluster-based prefetching mechanisms to find user clusters and to make prefetching decisions based on similarity in user behaviour to predict even 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] M. Du, M. Kihl, Å. Arvidsson, C. Lagerstedt, and A. Gavler, "Analysis of prefetching schemes for tv-on-demand service," in Proceedings of the Tenth International Conference on Digital Telecommunications, 2015, pp. 12–18.

[2] Y. Elkhatib, M. Mu, and N. Race, "Dataset on usage of a live amp; vod p2p iptv service," in Proceedings of Peer-to-Peer Computing (P2P), 14th IEEE International Conference, 2014, pp. 1–5.

[3] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems. ACM, 2006, pp. 333–344. [Online]. Available: http://doi.acm.org/10.1145/1217935.1217968

[4] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an ip network," in Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement. ACM, 2008, pp. 71–84. [Online]. Available: http://doi.acm.org/10.1145/1452520.1452529

[5] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth, "Analysis and characterization of a video-on-demand service workload," in Proceedings of the 6th ACM Multimedia Systems Conference. ACM, 2015, pp. 189–200. [Online]. Available: http://doi.acm.org/10.1145/2713168.2713183

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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, ser. WCW, 2002.

[11] 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 IEEE/WIC/ACM International Conference on Web Intelligence, 2006, pp. 740–744. [Online]. Available: http://dx.doi.org/10.1109/WI.2006.166

[12] 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.

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

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

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

[16] 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. ACM, 2007, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/1316902.1316917

[17] 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.

[18] 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, 2005, pp. 444–455.

[19] 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. ACM, 2000, pp. 3–11. [Online]. Available: http://doi.acm.org/10.1145/354384.354394

[20] 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.

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

[22] 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, 2011, pp. 72–80. [Online]. Available: http://dl.acm.org/citation.cfm?id=1987510.1987518

[23] Z. Zeng and B. Veeravalli, "Hk/t: A novel server-side web caching strategy for multimedia applications," in IEEE International Conference on Communications, 2008, pp. 1782–1786.

[24] 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, 2011, pp. 525–536. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2010.06.008

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

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

[27] 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.

[28] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement. ACM, 2007, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298310