An Announcement-based Caching Approach for Video-on-Demand Streaming

Maxim Claeys*, Niels Bouten*, Danny De Vleeschauwer†,
Werner Van Leekwijck†, Steven Latré‡ and Filip De Turck*
*Department of Information Technology, Ghent University - iMinds, Belgium
Email: maxim.claeys@intec.ugent.be
†Alcatel-Lucent Bell Labs, Antwerp, Belgium
‡Department of Mathematics and Computer Science, University of Antwerp - iMinds, Belgium

***Abstract*—The growing popularity of over the top (OTT) video streaming services has led to a strong increase in bandwidth capacity requirements in the network. By deploying intermediary caches, closer to the end-users, popular content can be served faster and without increasing backbone traffic. Designing an appropriate replacement strategy for such caching networks is of utmost importance to achieve high caching efficiency and reduce the network load. Typically, a video stream is temporally segmented into smaller chunks that can be accessed and decoded independently. This temporal segmentation leads to a strong relationship between consecutive segments of the same video. Therefore, caching strategies have been developed, taking into account the temporal structure of the video. In this paper, we propose a novel caching strategy that takes advantage of clients announcing which videos will be watched in the near future, e.g., based on predicted requests for subsequent episodes of the same TV show. Based on a Video-on-Demand (VoD) production request trace, the presented algorithm is evaluated for a wide range of user behavior and request announcement models. In a realistic scenario, a performance increase of 11% can be achieved in terms of hit ratio, compared to the state-of-the-art.**

I. Introduction

Traditionally, channel-based networks (e.g., satellite, cable networks) were used to distribute linear TV, benefiting from the broadcast nature of these networks. Using multicast capabilities, packet-based networks also support a scalable way of distributing live video. Moreover, packet-based networks are able to support personalized on-demand video services where a video library is offered by means of a catalog and separate sessions are set up for each individual user. However, while in linear TV the amount of video traffic is proportional to the number of channels, in an on-demand system the traffic is proportional to the number of users. This leads to a major increase in capacity requirements for on-demand video delivery.

By deploying intermediary caches, closer to the end-users, popular content can be served faster and without increasing backbone traffic. Since video interests are shared between users and their viewing interests overlap in time, deploying these caches can substantially reduce the required capacity by taking advantage of these overlapping sessions. Caching networks (e.g., Content Delivery Networks (CDNs)) were traditionally used to deliver web content in a scalable way and reduce latency by temporally storing part of the content in a network of caches close to the end-users. However, the use of caching in a streaming environment differs from the traditional web-based caching, since the objects are typically much larger and are to be consumed directly after they have been requested. Therefore, video delivered over caching networks is typically temporally segmented into smaller chunks that can be accessed and decoded independently.

Designing an appropriate replacement strategy for such caching networks is of utmost importance to achieve high caching efficiency and reduce the network load. Most caching algorithms are based on observations and take into account the recency or frequency of requests to calculate the rank of each of the cached objects. When caching temporally segmented video, as is the case in HTTP-based streaming systems, the caching algorithm can also take into account the temporal structure of the video. That is, when the streaming application requests the n-th segment at a specific point in time, the caching strategy can take advantage of the knowledge that the (n+1)-th and subsequent segments will be consumed shortly after this segment. Therefore, when multiple streaming sessions request the same segmented content, the caching strategy can take advantage of this knowledge to keep the segments in cache that are to be consumed in the near future.

Caching strategies can be further improved by taking into account additional information, such as content popularity and regional differences. The optimal caching algorithm (known as Belady's MIN [1]) takes advantage of the knowledge on future requests to discard the objects for which the next request occurs the furthest in the future. In a real system, this information on future requests is not directly available. However, studies[1] on user behavior for over the top (OTT) streaming services report that respectively 88% and 70% of the Netflix and Hulu Plus users stream three or more consecutive episodes of the same TV show (referred to as *binging*). Furthermore, binging is reported to become the moderate behavior with users streaming on average 2.3 episodes per sitting[2], resulting in about 57% of the streaming sessions that could be announced in advance. The same trend can be expected for YouTube videos, where the autoplay feature recently became auto-enabled[3]. Taking into account these announcements allows

[1]Nielsen - http://www.nielsen.com/us/en/insights/news/2013/binging-is-the-new-viewing-for-over-the-top-streamers.html

[2]Cinemablend - http://www.cinemablend.com/television/Unsurprising-Netflix-Survey-Indicates-People-Like-Binge-Watch-TV-61045.html

[3]MarketingLand - http://marketingland.com/autoplay-is-now-the-default-for-youtube-videos-122555

978-3-901882-77-7

us to predict future segment requests and thus approximating the theoretical optimal caching strategy. Furthermore, Service Providers (SPs) could give incentives (e.g., discounts, higher resolution video (4K) when requested upfront) to users when they announce their streaming sessions in advance.

In this paper, a novel caching strategy is proposed that takes advantage of the inferred knowledge of future segment requests when streaming segmented video and additionally exploits the added knowledge when streaming clients announce the videos that will be streamed in the near future. Using this additional information, we are able to estimate the future reuse times of the segmented content. These reuse times are then used to apply the Belady's MIN eviction strategy. This allows us to increase the caching efficiency compared to using only reuse time predictions based on the structure of the segmented video.

The main contributions of this paper are threefold. First, we propose a novel caching strategy that outperforms the current state-of-the-art strategies by including knowledge on segmented video streaming and announced future video requests. Second, we use user behavior models to emulate video interruptions and evaluate their impact on the proposed and state-of-the-art caching strategies. Finally, the proposed strategy is thoroughly evaluated in a distributed caching scenario.

II. Related Work

Web caching was introduced as a way to save traffic and minimize the perceived delay on the Internet by storing data closer to the end-user. Many caching algorithms have been proposed in the past and stem from replacement strategies for both web proxy caching and CPU caching. Least Recently Used (LRU) is based on the recency of the cached objects and discards the least recently used items first. This type of caching algorithms is know to attach too much importance to unpopular objects, awarding them the highest rank upon a single request. Least Frequently Used (LFU) determines the number of requests for an object over a period of time and discards the least frequently used objects first. This type of caching algorithms is known to attach as much importance to ancient as to recent history. To account for this, dynamic aging factors are introduced to improve the performance in LFU Dynamic Aging (LFU-DA). Adaptive Replacement Cache (ARC) balances between LRU and LFU by using more complex algorithms to determine the rank [2].

The aforementioned replacement strategies are widely used in web caches. Caching for video streaming differs from caching web content in a number of ways. First, video objects are typically much larger than traditional web objects such as text and images. Second, there is a difference in popularity evolution for streaming videos compared to web content. Furthermore, in contrast to web objects, having only the beginning of a video allows a video application to already start playout while the download continues [3]. This has led to the adoption of segment-based caching of multimedia streams, where data blocks of the video objects are grouped into segments. These segments can have variable size, e.g., smaller segments at the start of the video to decrease start-up delays. Cache admission and replacement policies have been proposed for such segmented video that take into account the distance from the start of the video and where segment sizes are exponentially increasing with their distance to the start of the video [4]. *Wu et al.* define three segmentation approaches for proxy caching: fixed length segments, pyramid with exponentially increasing segment sizes and skyscraper where sizes increase slower compared to the pyramid scheme by repeating each layer n times [5]. The authors conclude that segmentation approaches are particularly effective when the cache size is limited, media popularity changes over time, requests are spread over a large number of media objects and when the media file size and library is large.

Chen et al. propose a streaming proxy system called *Hyper Proxy*, which uses active prefetching for in-time delivery of uncached segments to address the proxy jitter problem [6]. This work was extended by adding segmentation strategies based on object popularity and by using predictions of future segment requests to preload uncached segments [7]. In previous work, a proactive cache management system for multi-tenant CDNs was proposed that optimizes content placement and server selection based on content popularity and geographical distribution of requests [8]. In this paper we focus on reactive caching approaches leveraging video structure and request announcements rather than proactively placing content into network caches. Other approaches use a two-tier cache that distinguishes between to-be-played and possibly played segments, partitioning the cache in two layers that are dynamically scaled [9]. The approach of dynamic cache partitioning has also been applied by *Wauters et al.*, where caching decisions in a time-shifted television service are based on content popularity metrics [10]. Popularity-based caching approaches have also been proposed by *De Vleeschauwer et al.* [11].

Marinca et al. use an analysis of the clients' request over passed time intervals to predict the content that will be requested in the near future [12]. By using a history window of the past 24 hours allows reducing the traffic with about 30%. Others evaluate the impact of dynamic sizing between prefix and suffix caching for segmented video [13]. *Hong et al.* propose a ranking scheme that follows the dynamicity of the video library [14]. The rank is based on the linearity of the video that if a chunk is requested, the next chunk will be requested with high probability in the near future. To this end each segment is assigned a value based on the number of viewers who are watching the video and will probably request the segment in the future. This value reflects the number of hits that this particular segment will receive in the future. Based on this value a rank is calculated that evicts the segments with the least chance of being reused in the future. In this paper, we additionally take into account the timing information of future requests by considering video request announcements to further increase the cache performance.

The MIN cache replacement algorithm proposed by *Belady et al.* is an offline eviction strategy that has been proven to be optimal [1]. It requires advanced knowledge of the requested content, and based on this information, chooses to evict the item in the cache whose next request occurs furthest in the future. *Van Roy et al.* propose a proof of optimality for the MIN cache replacement algorithm based on a dynamic programming argument [15]. *Wu et al.* make use of the natural linear time structure of segmented video to propose a caching algorithm based on the exact reuse time [16]. This reuse time indicates

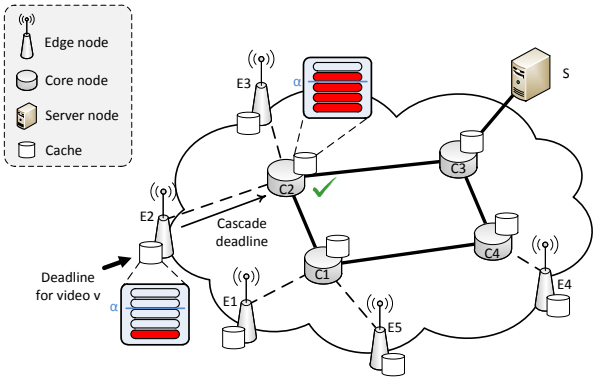

Figure 1. Illustration of the deadline announcement process.

when a cached segment will be reused and allows the eviction strategy to determine the segment request that will occur the furthest in the future. The authors do not take into account the delivery times between the different nodes in the network, only a single cache is assumed and the sessions are never interrupted. Our proposed approach not only uses the temporal structure of segmented video but also includes information of announced sessions. Furthermore, we also use interruption models to simulate user churn and use a hierarchy of caches where the delivery times between the different nodes are taken into account.

Recently, protocols have been proposed that support the announcement of deadlines, applied in this work. Shared Content Addressing Protocol (SCAP) is a transport protocol aimed at optimizing the delivery process that allows in-network intermediary elements to participate in delivering the content [17]. Furthermore it offers support for intermediary caches, multicast-like delivery of shared content requests and announcing deadlines for the delivery of the content. Similar features are also provided by Information-Centric Networking (ICN) architectures [18]. These architectures allow content to be duplicated everywhere in the network, enabling in-network caching at a fine granularity [19].

III. Deadline aware cache replacement

In the proposed approach, the caching nodes keep track of the start times of video streaming sessions. In the remainder of this paper, the start times of a video playout will be called *deadlines*. Based on these deadlines, a node can decide which segments to keep in its cache and which to remove. In Section III-A, the management of the deadlines is described. Section III-B describes the proposed novel cache replacement strategy.

A. Deadline management

Deadlines can either be announced by the client in advance of the streaming session, or be determined by the caching node at the arrival of a segment request. As both types of deadlines will play a different role in the replacement strategy, each caching node keeps track of both a set of *announced* deadlines D_A and *perceived* deadlines D_P.

1) Deadline announces: When a client announces a deadline, he sends the video ID and the future start time of the playout to the first caching node on the path to the server hosting the video. Each node then decides whether to accept the deadline locally or to cascade it to the next hop on the path. A node will only decide to accept the *announced* deadline, and add it to D_A, if it estimates to be able to serve at least a relative part α of the video from its local cache. This estimation is based on the number of segments of the considered video that is already present in the cache, combined with the segments that are estimated to arrive before the new deadline, based on known earlier deadlines. The value of α serves as a deadline acceptance threshold and provides a basic form of coordination between the caches. If the node is not able to meet the threshold α, it does not accept the deadline and cascades it to the next hop on the path to the server. An illustration of this process is given in Fig. 1. A client sends a deadline announcement for video v to the edge node $E2$. As the number of cached segments for video v is below the deadline acceptance threshold α, $E2$ cascades the deadline to the next hop $C2$ where it can be accepted and the cascading process is terminated.

2) Deadline determination: Upon arrival of a segment request for a specific video v, a caching node can easily determine the deadline d_v of the video playout as defined in equation (1), where x, t and l_v respectively denote the requested segment number, the current time and the segment length of video v. When the node does not have an *announced* deadline for the specified video at time d_v, either because the client did not announce the deadline or because the node could not accept it, it stores d_v as a *perceived* deadline in D_P.

$$d_v = t - x \times l_v \tag{1}$$

3) Deadline expiration: When a streaming session ends, the corresponding deadline can be removed from the caching nodes. When a streaming session ends before the end of the video, the client will inform the first node on the path to the server about the finished session only if the streaming session was announced in advance. In this case, the notification is cascaded until it reaches the node that accepted the deadline, where the deadline is removed from the set of *announced* deadlines D_A. The nodes are not informed about finished streaming session that were not announced or that were not accepted. For this reason, *perceived* deadlines $d_v \in D_P$ are only removed when the session is known to have ended, even if it was interrupted prematurely. For a deadline d_v, this is the case when $t > d_v + L_v$, with t and L_v denoting the current time and the total length of video v respectively.

B. Replacement strategy

Using the information contained in the *announced* and *perceived* deadlines, each node can decide which segments to keep in its cache, and which can be removed. For this purpose, the concept of earliest reuse times is applied. For each deadline d_v for a specific video v, the reuse time $r^{d_v}_{v_x}$ of each segment v_x of that video can easily be calculated using equation (2), where x denotes the sequence number of segment v_x. If $r^{d_v}_{v_x} < t$, with t denoting the current time, the reuse time is set to infinity. The earliest *announced* reuse time $e^A_{v_x}$ is then obtained as the

lowest reuse time $r_{v_x}^{d_v}$ of all *announced* deadlines $d_v \in D_A$ for video v, a expressed in equation (3). Similarly, the earliest *perceived* reuse time is obtained using equation (4). When there are no *announced* or *perceived* deadlines for video v, respectively $e_{v_x}^A$ or $e_{v_x}^P$ are said to be equal to infinity for every segment v_x of that video.

$$r_{v_x}^{d_v} = d_v + x \times l_v \tag{2}$$

$$e_{v_x}^A = \min_{d_v \in D_A} r_{v_x}^{d_v} \tag{3}$$

$$e_{v_x}^P = \min_{d_v \in D_P} r_{v_x}^{d_v} \tag{4}$$

When a segment v_x of video v arrives at a node, the node has to decide whether or not to add it to its local cache C, if it is not already cached. If the remaining available caching capacity is insufficient to store the new segment v_x, another segment can be selected for removal. This procedure is outlined in Algorithm 1. First, the earliest *announced* and *perceived* reuse times $e_{v_x}^A$ and $e_{v_x}^P$ of the newly arrived segment are calculated as described above (line 1). Next, a segment $s' \in C \cup \{v_x\}$ is selected as a candidate for eviction. This segment s' is selected as the segment with the maximal earliest *announced* reuse time: $s' = \arg\max_{s \in C \cup \{v_x\}} e_s^A$ (lines 4-6). When multiple such segments exist, the segment with the maximal earliest *perceived* reuse time is selected: $s' = \arg\max_{s \in C \cup \{v_x\}} e_s^P$ (lines 7-9). The LRU order is used as a final tiebreaker (lines 10-13). When the evicted segment s' is a cached segment ($s' \in C$), it is removed from the cache and replaced by the new segment v_x (lines 18-20).

Algorithm 1 Outline of the proposed cache eviction strategy on arrival of a new segment v_x.

```
1: Calculate e^A_{v_x} and e^P_{v_x}
2: s' ← v_x
3: for s ∈ C do
4:     Calculate e^A_s and e^P_s
5:     if e^A_s > e^A_{s'} then
6:         s' ← s
7:     else if e^A_s = e^A_{s'} then
8:         if e^P_s > e^A_{s'} then
9:             s' ← s
10:        else if e^P_s = e^A_{s'} then
11:            Calculate LRU ranks LRU_s and LRU_{s'}
12:            if LRU_s > LRU_{s'} then
13:                s' ← s
14:            end if
15:        end if
16:    end if
17: end for
18: if s' ≠ v_x then
19:    Remove s' from C
20:    Add v_x to C
21: end if
```

The rationale behind this approach is to keep the segments in the cache that will be needed in the nearest future. However, accepted *announced* deadlines are always prioritized, even if another segment has an earlier *perceived* deadline. This is done to ensure that the cache will not violate the intent expressed when accepting the deadline. It is important to note that when no deadlines are announced, the algorithm behaves as a purely reuse time based replacement strategy without coordination between the caches, only taking into account the temporal structure of videos in the form of earliest *perceived* reuse times.

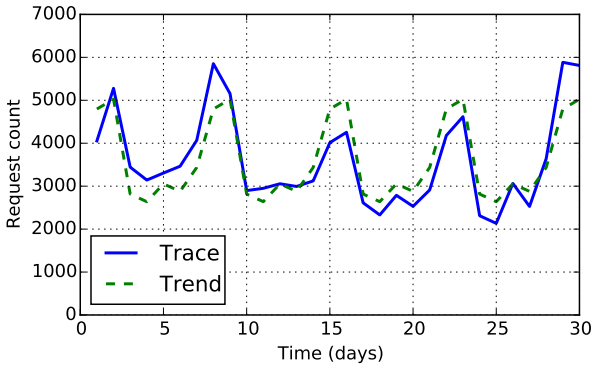

Figure 2. Number of daily requests in the considered VoD trace.

IV. Evaluation scenario description

To evaluate the proposed approach, a request trace of the Video-on-Demand (VoD) service of a leading European telecom operator has been used. Section IV-A describes the characteristics of this trace, while the applied session duration models, simulating the user behavior, are described in Section IV-B. Finally, the considered network topology is described in Section IV-C.

A. VoD trace characteristics

The trace was collected over a period of 30 days between Saturday February 6, 2010 and Sunday March 7, 2010. Due to a failure of the probing nodes, a couple of hours of monitoring data was missing for February 12, 2010 and February 19, 2010. These gaps have been filled by considering the request pattern of the same period in the previous week, mapped on the content popularity of the last day. The resulting trace contains monitoring information of 108,392 requests for 5644 unique videos, originating from 8825 unique users, spread across 12 cities. In this work, all videos are considered to have an equal length of 50 minutes and a bitrate of 1Mbit/s, resulting in a size of 375Mbyte for each video. The entire video catalog thus measures about 2Tbyte.

Fig. 2 depicts the evolution of the number of video requests per day over the considered time period. In this graph, a clear weekly pattern can be observed. The five peaks in the graph correspond to the five weekends, with increased activity on Friday, Saturday and Sunday. As Fig. 2 only shows per-day aggregated data for sake of visibility, the underlying diurnal trend cannot be seen. For Wednesdays and Sundays, the activity peak is situated between 4:30pm and 6:30pm, while for the other days of the week, the largest number of requests is reported between 8pm and 10pm.

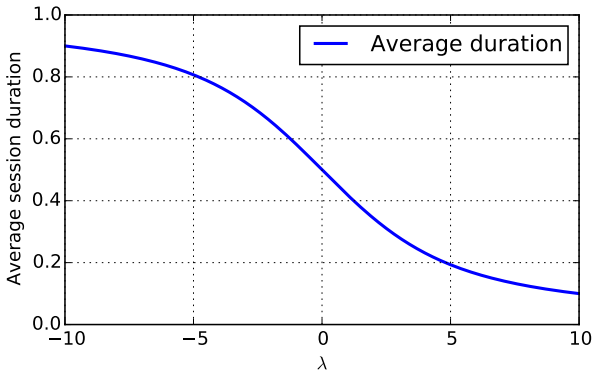

Figure 3. Average session duration for the normalized exponential model in terms of λ.

B. Session duration

In contrast to most other work, we take into account the fact that users do not necessarily stream a video entirely, but may interrupt the session mid-stream. Multiple models have been represented in literature to represent the session duration in video streaming services. *Ullah et al.* provide a survey of user behavior measurements in several types of video streaming services [20]. According to this survey, the session duration of most VoD services can be modelled using an exponential distribution. To deal with the fixed content length in this work, a normalized exponential distribution was used to model the session duration in our experiments. The cummulative probability $p(x)$ of a session to last at most a relative part $x \in [0;1]$ of the video is calculated using equation (5), where the value of λ depends on the video type. The average relative session duration in function of λ is presented in Fig. 3. To show the general applicability of our approach, evaluations have been performed using different values for λ, resulting in both shorter and longer average session durations. The scenario where all videos are streamed entirely is modelled as $\lambda = -\infty$.

$$p(x) = \frac{1 - e^{-\lambda x}}{1 - e^{-\lambda}} \tag{5}$$

C. Network topology

To evaluate the performance of the proposed algorithm in a distributed scenario, a GÉANT-based topology[4], consisting of 23 nodes, has been used. As described in Section IV-A, the employed VoD request trace contains 12 cities, which we map onto 12 edge nodes. One node is assigned as server node S, hosting all video content. The 10 remaining nodes are modelled as core nodes, for which the 10 most connected nodes were selected. The resulting topology is shown in Fig. 4.

Unless otherwise stated, the total caching capacity in the network is equal to 5% of the total catalog size (i.e. 105,750Mbyte). In each experiment, the total caching capacity was spread uniformely across the topology, with the edge nodes having half of the capacity of the core nodes (i.e. core node capacity of 6,610Mbyte, edge node capacity of 3,305Mbyte).

[4]GÉANT Project - http://www.geant.net

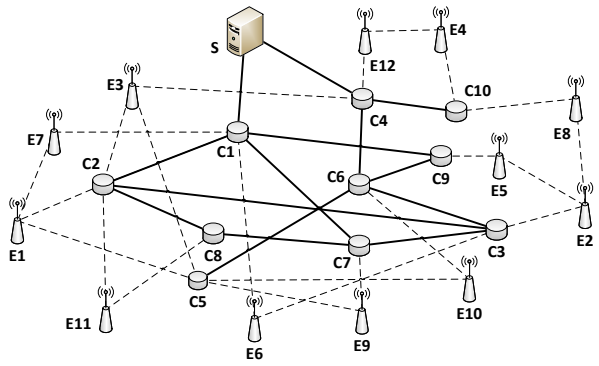

Figure 4. Evaluated network topology, based on the GÉANT topology.

Table I. EVALUATED PARAMETER CONFIGURATIONS.

Parameter	Values
Norm. exp. dur. model λ	-∞, -7, -3, 0.05, 3, 7
Announcement ratio	0%, 20%, 40%, 60%, 80%, 100%
Announcement timeframe	1-10min, 20-50min, 1-24h
Acceptance threshold α	0%, 25%, 50%, 75%

V. EVALUATION RESULTS

To thoroughly evaluate the performance of the proposed approach, experiments have been performed for a wide range of scenarios. Multiple values of λ have been used to simulate various session duration models, based on (5). Furthermore, different deadline announcement strategies have been investigated, ranging from short-term annoucements (1-10 minutes prior to playout) to long-term announcements (1-24 hours prior to playout). For each of the evaluated timeframes, the deadlines are announced at a specific time prior to video playout, uniformely distributed between the timeframe bounds. The ratio of streaming sessions for which a deadline is announced in advance is also varied throughout the experiments. Finally, the impact of the deadline acceptance threshold α was evaluated. A summary of the evaluated parameter configurations is presented in Table I.

In all of the evaluations, the performance of the proposed approach is compared to the LRU caching strategy. Unless otherwise stated, relative performance gain is always expressed with respect to the LRU approach. In the remainder of this paper, the hit ratio will be defined as the ratio between the number of requests that are served from a cache in the network and the total number of segment requests sent by the clients.

A. Impact of deadline acceptance threshold α

As described in Section III-A, the deadline acceptance threshold α is used to decide whether to accept an announced deadline or to cascade it to the next hop. To evaluate the pure impact of this parameter, without interference of the user behavior, the scenario where all videos are viewed in totality ($\lambda = -\infty$) is considered. Deadline announcement times are uniformely distributed between 20min and 50min prior to video playout.

Fig. 5 shows the hit ratio of the proposed approach, relative to the hit ratio of the LRU approach, for different values of α under varying request announcement ratios. It can be seen that the best results can be achieved with a deadline acceptance threshold α of 25%. Similar results are obtained when the

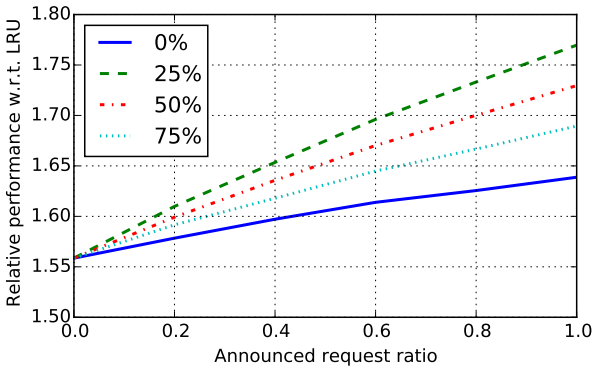

Figure 5. Impact of the deadline acceptance threshold α on the relative hit ratio compared to LRU, when deadlines are announced 20min to 50min prior to playout.

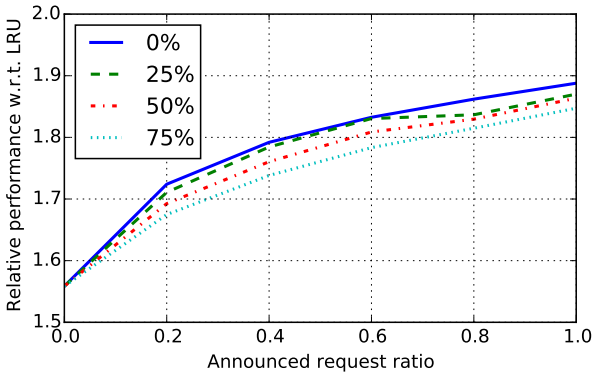

Figure 6. Impact of the deadline acceptance threshold α on the relative hit ratio compared to LRU, when deadlines are announced 1h to 24h prior to playout.

deadlines are announced closer to the session start (1min to 10min prior to playout).

However, as can be seen in Fig. 6, the approach with $\alpha = 25\%$ is slightly outperformed by the approach with $\alpha = 0\%$ when deadlines are announced more in advance (1h to 24h prior to playout). This behavior can be explained by the inter arrival times of requests in the considered VoD trace. The median time between two requests for the same content in the entire network amounts to 1h11min, while the median time between two requests for the same content from a single location amounts to 6h48min. When deadlines are announced 1h to 24h in advance, the average time between the announcement of a deadline and the video playout significantly exceeds the average inter arrival time on a single location. Therefore, a next deadline for a video will likely be announced prior to the end of an active session. In this way, only few deadlines will not be able to be reused, so all deadlines can be accepted directly. When announcements are made in a shorter timeframe, deadlines have higher reusage probability on the core nodes, as these perceive requests from multiple edge nodes. Therefore, higher values of α yield better performance. Given this bad performance using $\alpha = 0\%$ with shorter term announcements, the deadline acceptance threshold $\alpha = 25\%$ can generally be selected as the best configuration.

The impact of the threshold α on the deadline acceptance

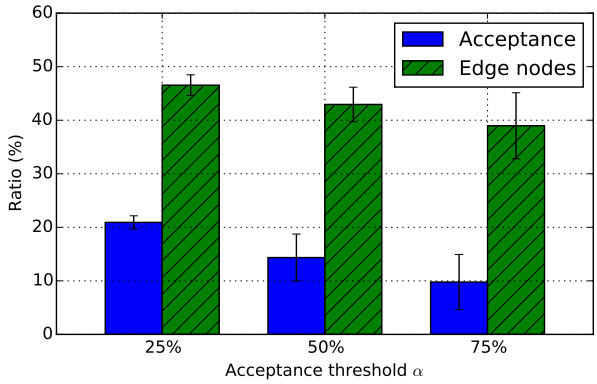

Figure 7. Impact of the deadline acceptance threshold α on the deadline acceptance in the network, averaged over all scenarios.

Table II. AVERAGE SESSION DURATION FOR THE CONSIDERED VALUES OF λ.

λ	Avg. session duration
-∞	100%
-7	85.81%
-3	71.91%
0.05	49.58%
3	28.10%
7	14.19%

in the network is shown in Fig. 7, averaged over all deadline announcement and user behavior scenarios. The results for $\alpha = 0\%$ are omitted for visibility reasons, as in this case all deadlines get accepted at the edge nodes. As expected, the number of accepted deadlines decreases with higher values of α. Furthermore, when α increases, a bigger part of the deadlines is accepted in the network core nodes. This can again be explained by the fact that core nodes simultaneously perceive requests of multiple edge nodes, such that cached segments can be reused across geographical locations.

B. Impact of deadline announcement timeframe

The timeframe in which deadlines are announced prior to playout defines the time window the caches can use to base the replacement decisions on. The impact of this timeframe on the performance of the proposed approach with an acceptance threshold $\alpha = 25\%$ in a scenario where all sessions are viewed in totality ($\lambda = -\infty$) is shown in Fig. 8. As could be expected, a higher performance increase can be achieved when deadlines are announced more in advance.

In a realistic scenario where deadlines are announced 20 to 50 minutes prior to playout (e.g., when announcing the playout of a consecutive episode of the same TV show), announcing 60% of the deadlines yields a significant performance increase of 69.61% compared to LRU and 8.82% compared to a reuse time based replacement strategy. When deadlines are announced closer to the session start, the gain compared to a reuse time based replacement strategy is limited.

C. Impact of session duration model

Up to now, users were considered to always stream a video entirely. However, as session churn limits the accuracy of the deadlines, the user behavior will likely impact the performance

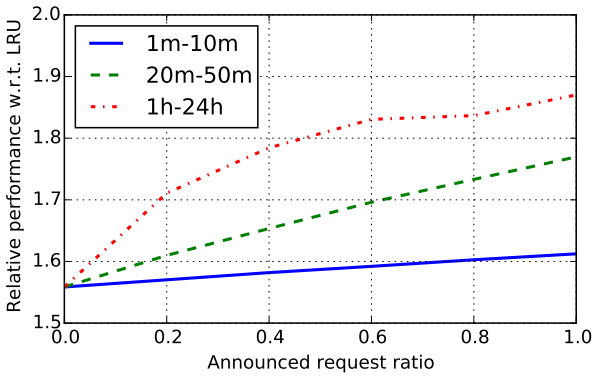

Figure 8. Impact of the deadline announcement timeframe on the relative hit ratio compared to LRU with a deadline acceptance threshold $\alpha = 25\%$.

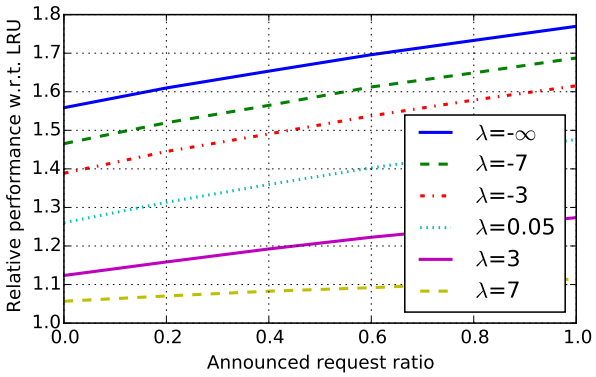

Figure 9. Impact of the session duration model parameter λ on the relative hit ratio compared to LRU with a deadline acceptance threshold $\alpha = 25\%$ and deadline announcements 20-50 minutes in advance.

of the proposed deadline based approach. As described in Section IV-B, a normalized exponential session duration model has been applied. The average session durations for the considered values of λ are presented in Table II, while the impact of this parameter on the performance of the approach is shown in Fig. 9. As expected, the performance gain increases with a growing average session duration. Furthermore, when the average session duration increases, the gain of taking into account the announced deadlines is higher. This can also be seen in Fig. 10, showing the impact of the average session duration on the relative hit ratio for various session announcement ratios. For each announcement ratio, the performance gain grows with an increasing average session duration, but a steeper growth can be seen when more deadlines are announced.

D. Performance comparison

Finally, the performance of the proposed approach can be evaluated in a realistic scenario where users watch 72% of a video on average ($\lambda = -3$) and deadlines are announced 20 to 50 minutes in advance of the playout. Fig. 11 provides a comparison in terms of hit ratio between the proposed approach, the LRU approach and a purely reuse time based approach, comparable to the approach proposed by *Wu et al.* [16]. It can be seen that taking into account the temporal structure of the content in the replacement strategy leads to a significant performance increase of 38.86% compared to the

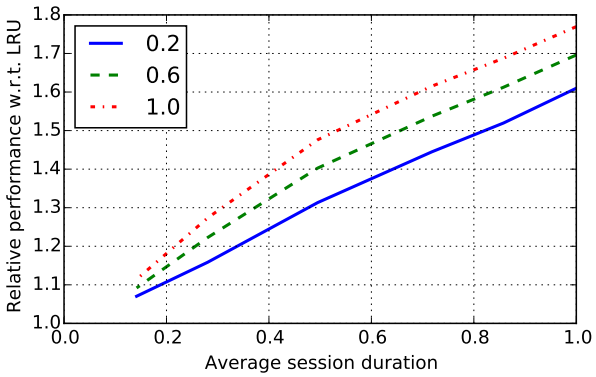

Figure 10. Impact of the average session duration on the relative hit ratio compared to LRU with a deadline acceptance threshold $\alpha = 25\%$ and deadline announcements 20-50 minutes in advance.

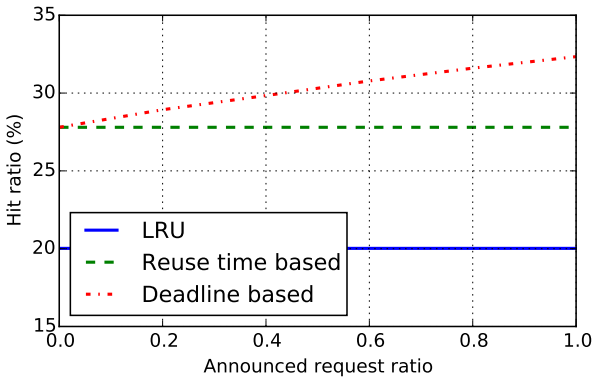

Figure 11. Performance comparison in terms of hit ratio in a realistic scenario where users watch 72% of a video on average and deadlines are announced 20-50 minutes in advance.

LRU strategy. Taking into account the announced deadlines results in an additional performance increase of up to 16.33%, depending on the amount of announced deadlines.

Taking into account the user behaviour studies described in Section I, in current OTT streaming services such as Netflix and Hulu, up to 60% of the sessions could be announced in advance. As presented in Table III, in this scenario, the proposed approach yields a hit ratio being 53.75% higher than the hit ratio using the LRU strategy and 10.72% higher compared to a reuse time based strategy. This results in a reduction of average hop count of 9.32% and 2.73% compared to the LRU and reuse time based strategy, respectively. In terms of average bandwidth usage spread across the network, a reduction of 9.66% and 3.06% is achieved respectively.

VI. Conclusion

In this paper, a deadline based cache replacement strategy for Video-on-Demand (VoD) systems was presented. This

Table III. Performance comparison.

Criteria	LRU	Reuse time based	Deadline based
Hit ratio (%)	20.02	27.80	30.78
Avg. hop count	2.36	2.20	2.14
Avg. bandwidth (Mbps)	215.22	200.56	194.43

strategy not only takes into account the temporal structure of videos, but also takes advantage of the phenomenon of *binging* where users watch multiple consecutive episodes of the same TV show. This allows a significant part of the streaming sessions to be announced in advance of the actual video play-out. The deadline based cache replacement approach has been thoroughly evaluated in a wide range of session announcement and user behavior scenarios using a VoD production request trace. It was shown that in a realistic scenario, intelligently taking into account the session announcements in the cache replacement decisions can result in a hit ratio increase of 54% compared to the LRU strategy and 11% compared to the state-of-the-art. Future work will focus on investigating an increased coordination between the caches in the network to further guide caching decisions and avoid duplicated elements.

ACKNOWLEDGMENT

M. Claeys and N. Bouten are funded by a grant of the Agency for Innovation by Science and Technology in Flanders (IWT). The research was performed partially within the ICON SHIFT-TV project (under grant agreement no. 140684). This work was partly funded by FLAMINGO, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

REFERENCES

[1] L. Belady, "A study of replacement algorithms for a virtual-storage computer," *IBM Systems Journal*, vol. 5, no. 2, pp. 78–101, 1966.

[2] N. Megiddo and D. S. Modha, "Outperforming LRU with an adaptive replacement cache algorithm," *Computer*, vol. 37, no. 4, pp. 58–65, 2004.

[3] N. Färber, S. Döhla, and J. Issing, "Adaptive progressive download based on the MPEG-4 file format," *Journal of Zhejiang University SCIENCE A*, vol. 7, no. 1, pp. 106–111, 2006.

[4] K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based Proxy Caching of Multimedia Streams," in *Proceedings of the 10th International Conference on World Wide Web*, 2001, pp. 36–44.

[5] K.-L. Wu, P. Yu, and J. Wolf, "Segmentation of multimedia streams for proxy caching," *IEEE Transactions on Multimedia*, vol. 6, no. 5, pp. 770–780, 2004.

[6] S. Chen, B. Shen, S. Wee, and X. Zhang, "Segment-based streaming media proxy: modeling and optimization," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 243–256, 2006.

[7] ——, "Sproxy: A caching infrastructure to support internet streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 5, pp. 1062–1072, 2007.

[8] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, G. Pavlou, and F. De Turck, "Proactive multi-tenant cache management for virtualized ISP networks," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM)*, 2014, pp. 82–90.

[9] K.-C. Liang and H.-F. Yu, "Adjustable Two-Tier Cache for IPTV Based on Segmented Streaming," *International Journal of Digital Multimedia Broadcasting*, vol. 2012, 2012.

[10] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, and P. Demeester, "Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services," in *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications.*, 2006, pp. 379–386.

[11] D. De Vleeschauwer and K. Laevens, "Performance of Caching Algorithms for IPTV On-Demand Services," *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 491–501, 2009.

[12] D. Marinca, A. Hamieh, D. Barth, K. Khawam, D. De Vleeschauwer, and Y. Lelouedec, "Cache management using temporal pattern based solicitation for content delivery," in *Proceedings of the 19th International Conference on Telecommunications (ICT)*, 2012, pp. 1–6.

[13] Z. Xu, X. Guo, Y. Pang, and Z. Wang, "The Transmitted Strategy of Proxy Cache Based on Segmented Video," in *Network and Parallel Computing*. Springer Berlin Heidelberg, 2004, vol. 3222, pp. 502–507.

[14] D. Hong, D. De Vleeschauwer, and F. Baccelli, "A chunk-based caching algorithm for streaming video," in *Proceedings of the 4th Workshop on Network Control and Optimization*, 2010.

[15] B. Van Roy, "A short proof of optimality for the min cache replacement algorithm," *Information processing letters*, vol. 102, no. 2, pp. 72–73, 2007.

[16] T. Wu, K. De Schepper, W. Van Leekwijck, and D. De Vleeschauwer, "Reuse time based caching policy for video streaming," in *Proceedings of the Consumer Communications and Networking Conference (CCNC)*, 2012, pp. 89–93.

[17] K. De Schepper, B. De Vleeschauwer, C. Hawinkel, W. Van Leekwijck, J. Famaey, W. Van de Meerssche, and F. De Turck, "Shared Content Addressing Protocol (SCAP): Optimizing multimedia content distribution at the transport layer," in *Proceedings of the Network Operations and Management Symposium (NOMS)*, 2012, pp. 302–310.

[18] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, pp. 26–36, 2012.

[19] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *ICN Workshop on Information-centric Networking*, 2012.

[20] I. Ullah, G. Doyen, G. Bonnet, and D. Gaiti, "A Survey and Synthesis of User Behavior Measurements in P2P Streaming Systems," *IEEE Communications Surveys Tutorials*, vol. 14, no. 3, pp. 734–749, 2012.