QoE Performance for DASH Videos in a Smart Cache Environment

Sheyda Kiani Mehr and Deep Medhi
University of Missouri–Kansas City, USA

***Abstract*—During the past decade, Internet has seen dramatic increase in video traffic. Users expect a high quality of experience with online video streaming. For video delivery, the DASH (Dynamic Adaptive Streaming over HTTP) standard is one of the common approaches for streaming used by content providers. In order to give users a higher quality of experience, in-network caching and prefetching are useful to reduce video delivery latency with DASH-generated videos. In this paper, we present a Smart Cache framework that uses a cache prefetching scheme that prefetches segment bitrate based on forecasted throughput at the cache entity by using previous throughput values from clients. For our study, we have implemented our framework on the GENI testbed, and our results for single-client and two-client interactions show that Smart Cache increases the byte-hitrate and reduces the number of unused prefetches for cache. We also consider the impact on Quality of Experience (QoE) for each client during contention.**

keywords: Cache, DASH, forecasting, prefetching, QoE

I. Introduction

Video streaming traffic accounts for a significant amount of traffic on the Internet; it has seen a sustained growth due to the increase in user demand. Cisco estimates that video streaming and downloads will make up for 80% of the world's Internet traffic by 2019 [1]. It is expected that the online video streaming service could eventually replace the standard television sets. Users of online video services expect optimum quality, regardless of network or server conditions.

Dynamic Adaptive Streaming over HTTP (DASH) has become the de facto standard for video streaming. Video content providers such as YouTube, Hulu, and Netflix use DASH for streaming video contents. A video that uses the DASH standard is available with multiple resolutions and each video resolution is divided into segments with constant playback duration; this is done in order to enable different resolutions during the playback of the video at the user end. In DASH, each video title is associated with a metadata file called Media Presentation Description (MPD), wherein each segment for a particular resolution is associated with a Universal Resource Locator (URL). Before playback begins, the client requests and fetches the MPD file and parses it to determine the available bitrates and URLs for all segments. A DASH client starts by downloading the first segment, usually with the lowest bitrate representation, and uses an adaptation bitrate algorithm (ABR) to determine the most suitable bitrate to be requested for each of the subsequent segments. This decision is based on two factors: the network conditions represented by the throughput measurements for the downloaded segments and the current status of the client buffer.

978-3-903176-15-7

Providers may use in-network caches with prefetching to improve both byte latency and throughput of subsequent video segments to the end users. In-network caches have a smaller storage size than that of servers. Enhancing the prefetch hitrate decreases the number of unused prefetch segments, and improves the bandwidth use of a link that is connected from cache to video content servers. Furthermore, when multiple clients simultaneously contend for a particular video from the cache, the overall performance and QoE may be impacted negatively. There are two prefetching scheme modes, cache-driven or a client-driven DASH prefetching that each has its own drawbacks. As a consequence, service providers may not be able to guarantee a premium quality of service with DASH [2].

In this work, we present a Smart Cache framework that enhances [3], along with a comprehensive set of studies on cache, the cache's interaction with two clients, and overall QoE performance. The Smart Cache framework is proposed to predict future requests based on the throughput measurements. Based on our study, we observe that implementing the Smart Cache prefetch scheme at the cache increases the cache performance by reducing the number of unused prefetches and improving QoE metrics, such as reduced segment fetch delay and higher bitrate gain for the user.

The rest of this paper is organized as follows: Section II presents related work on DASH QoE and prefetching and cache performances. We describe the Smart Cache framework along with its algorithmic details in Section III. In Section IV, we present our study and the results. We end with a concluding remark in Section V.

II. Related Work

Some works has been done in order to improve cache performance and QoE. Liang et al. [4] assume that there is high probability that a client requests the same bitrate each time, so prefetching the next segments is based on the current bitrate. The issue with this basic method is that it does not utilize the available resources, such as bandwidth of the link, and the user is not able to receive the best possible bitrate for a segment. Rejaie et al. [5] designed a cache server that prefetches the segments based on the average bandwidth between the origin server and the cache. This is not comparable with a typical DASH adaptation scheme that is based on the throughput of the link between the client and the cache.

There are a few works where the client chooses the bitrates based on cache information (cache-driven), or the client receives assistance from the cache. Pham et al. [6] propose a client ABR that is assisted by a cache server in order to get better QoE. The cache monitors the bandwidth from the origin server and sends it to the client. Mok et al. [7] propose QDASH, a cache that measures the available bandwidth and is responsible for helping clients to select the most suitable video quality level. Krishnamoorthi et al. [8] suggest cooperative buffer-aware prefetching in which a client continually shares its buffer occupancy with the cache, and the cache shares its fragments and segment fetch timing with the client. The client can thereby give preference to downloading fragments that are already stored within the cache. Liu et al. [9] propose a joint client-driven prefetching and rate adaptation algorithm (CLICRA), in which the cache affects the client decision for the next segment(s). We propose a smart client-driven cache to forecast the most accurate segment for clients.

III. Methodology

The Smart Cache framework has two components: *DASH Request Handler* (DRH) and *Cache Manager* (CM). In the original framework, CM simply included the *Prefetch Manager* (PM). In the Smart Cache framework, CM also includes a new module: *Replacement Manager* (RM) (Fig. 1).

DRH serves the requests from the DASH clients and gives back the requested segments to the client; this component also receives all of the HTTP header information, including the client's smooth throughput value, and stores it for later use. The initial MPD file is transferred to the MPD Parser module in DRH, which parses the data to store them in the Segment-Request Index module. The client's ID, Session-ID, requested video ID, and its related list of bitrates are stored in the Segment-Request Index. Bandwidth Estimator module in DRH receives the average throughputs, A_j, for j-th request from the client, which is computed based on the previous m segments:

$$A_j = \frac{\sum_{k=j-m}^{j-1} S_k/m}{\sum_{k=j-m}^{j-1} T_k/m} = \frac{\sum_{k=j-m}^{j-1} S_k}{\sum_{k=j-m}^{j-1} T_k}, \quad j \geq 2, \qquad (1)$$

where S_k is the segment size of the k-th segment, and T_k is the total time taken for the client to receive kth segment from cache.

CM is the functioning brain of the Smart Cache. The PM module in CM uses the throughput stored in Segment-Request Index to forecast the throughput for prefetching the next segment that will be requested by the client. Forecasting is based on an exponential smoothing approach. Consider that F_{j+1} is the future throughput for the $(j+1)$-th segment given that F_j is the currently forecasted throughput, and A_j is the current actual throughput from the client for the j-th segment. For the smoothing parameter, α $(0 \leq \alpha \leq 1)$, this relationship is given by

$$F_{j+1} = \alpha A_j + (1-\alpha)F_j, \quad j \geq 2. \qquad (2)$$

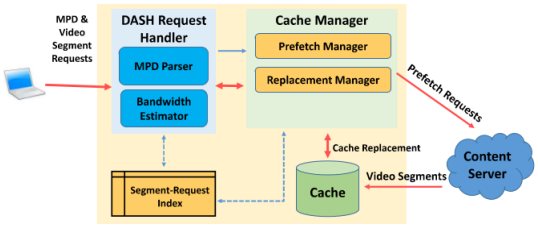


Fig. 1: Smart Cache Framework.

However, for the very first segment request, the cache serves the first request with the lowest bitrate from the origin server. When the client asks for the second segment, The PM needs a throughput value in order for its ABR to prefetch the second segment. In this case, the cache's throughput value for the second segment is the size of the segment divided by the time that it takes for the request to transfer from the client to the cache plus the time that it takes for the cache to fetch the segment from origin server. Therefore, the second segment's estimated throughput value is calculated as follows:

$$\begin{aligned} \text{Throughput\_forecast} &= \frac{S_1}{\Delta t_1 + \Delta t_2}, \\ \text{where } \Delta t_1 = t_2 - t_1, &\quad \Delta t_2 = t_3 - t_2. \end{aligned} \qquad (3)$$

Here, t_1 is the moment that the client sends the request, t_2 is the instant that the cache receives the request from the client, and t_3 is the instant that the cache sends the segment to the client.

The role of the new module, RM, in the Smart Cache framework is to determine the segments that are to be replaced in order to accommodate new segments, especially during contention from multiple clients. If the number of segments stored in the cache becomes greater than an initial number, n, then the cache makes the decision to replace segments using the RM. In our approach, we replace the Least Recently Used (LRU) segment.

In the Smart Cache framework, we also designed a multi-processing cache to handle multiple client connections simultaneously. Each connection is served as a separate process. As a process, client connection has multiple threads, such as prefetching thread (prefetches a segment from origin server) and current thread (calculates throughput and bitrate of the prefetching segment). After the cache calculates the forecasted throughput of the next segment, the prefetching method in the cache uses the same ABR that is used by the client.

The client uses a hybrid ABR that is throughput-based and buffer-based, with a simple average value for throughput. The cache entity only uses the throughput-based version of the client's ABR. So, each time the cache calculates the next throughput, this throughput is given to the throughput-based ABR in the PM module. The PM module prefetches the next segment based on the calculated forecasted throughput. Our entire approach is presented in Algorithm 1.

IV. Evaluation and Results

The Smart Cache framework is implemented on the GENI research testbed [10]. There are three entities in our imple-

Algorithm 1 Smart Cache Procedure

R: **Request from client,** α**: smoothing constant,** n**: cache size,** Q**: caches segement queue**

Function MAIN(R, α, n, Q)
  $bitrates[]$
  $URLs[]$
  $j \leftarrow 2$
  $forecast\_throughput \leftarrow 0$
  While R do
    if R==MPD
      $bitrates, URLs \leftarrow$ PARSE(R)
      STORE$(bitrates, URLs)$
    else if R==S_1
      $t_1 \leftarrow$ HEADER(R)
      $t_2 \leftarrow$ MEASURE_TIME()
      RECIEVE_FROM _ORIGIN_SERVER(S_1)
      $t_3 \leftarrow$ MEASURE_TIME()
      SEND_TO _CLIENT(S_1)
      $\Delta t_1 \leftarrow t_2 - t_1$
      $\Delta t_2 \leftarrow t_3 - t_2$
      $forecast\_throughput \leftarrow$ SIZE$(S_1)/(\Delta t_1 + \Delta t_2)$
      $S_2 \leftarrow$ PREFETCH$(forecast_throughput)$
      RM(S_2)
      $F_2 \leftarrow 0$
      STORE(F_2)
    else
      SEND_TO_CLIENT(S_j)
      $A_j \leftarrow$ HEADER(R)
      $forecast\_throughput \leftarrow$ FORECAST(A_j)
      $S_{j+1} \leftarrow$ PREFETCH$(forecast_throughput)$
      RM(S_{j+1})
      $j++$
    end if
  end while
End Function

Function FORECAST(A_j)
  RETRIEVE(F_j)
  $F_{j+1} \leftarrow \alpha * A_j + (1-\alpha) * F_j$
  STORE(F_{j+1})
  $return\ F_{j+1}$
End Function
Function PREFETCH $(forecast\_throughput)$
  Use ABR with all available bitrates from MPD
  $return\ S_{j+1}$
End Function
Function RM (S_{j+1})
  if $Size(Q) > n$
    Pop(Q)
  end if
  $Q \leftarrow S_{j+1}$
End Function

mentation: clients, cache, and DASH Server. In the topology, two clients are connected to a LAN switch. The in-network cache is connected to the same LAN switch. The cache is then directly connected to the video content server. The GENI platform allows us to regulate link bandwidths. Smart Cache and DASH player were written in Python [11]. For calculating the smooth throughput, Client's ABR uses $m = 10$ previous segments. Apache2 web-server is used as the DASH content server. For our study, we used the publicly available Big Buck Bunny video from the ITEC dataset [12]. In particular, we used the 10 sec and 15 sec segment durations (which we refer them as 10s and 15s in the rest of the paper) while the entire video has 60 segments and 40 segments, respectively (for a total video length of 600 sec). Both datasets have 20 different bitrates (resolutions) for each segment to choose from. All the possible scenarios are replicated 5 times.

We consider two general scenarios: In the first scenario there is one client, and in the second scenario there are two clients. Within the first scenario, we tested once with no background traffic on the links (10Mbps for each link), and then with UDP background traffic on the link between the cache and the origin server. The double-client scenario is studied only with background traffic between the cache and the origin server. The choice of the bottleneck link through background traffic is motivated by prior work. For example, it has been reported that the cache-server link with limited bandwidth could cause a bottleneck in the network [4] and [13]. Based on [8], a simple cache solution is potentially beneficial when the bandwidth bottleneck is between the client and the cache. However, caching can be more effective if the bottleneck is between the cache and the origin server. A common issue that arises when the DASH technology applies cache is that the connectivity between the client and the cache will have a higher bitrate than that of the connectivity between the cache and the server [14]. This causes a cache to consume more time fetching uncached segments from the original server [15]. Thus, the ABR in the client incorrectly estimates network conditions, which in turn causes wrong quality decisions, ultimately resulting in a negative effect on QoE [16]. When there are multiple clients contending at the same cache, this issue is exacerbated. Thus, in this work, we consider the case of a network bottleneck on a cache-server bandwidth backbone link.

The background UDP traffic was created by transferring a very large file (1.6 GB) using UDP socket python code on each of the cache and the server machines 40 sec after the first client began its requests. 40-second interval, since by this time, the throughput value has already surpassed the peak threshold for initial bitrate ramp-up phase. On the other hand, if we generate the traffic before 40 sec, then the client would stay on the requests with lower bitrate for a greater number of segments. Therefore, implementing background traffic has no significant effect, as the requests would have remained at the low bitrates for a longer period before the throughput hits the threshold. We conducted our study based on the condition that when a second client initiates a connection, the cache is empty. Thus, in the double-client scenario, the cache always has to prefetch the segments for both clients, because the required segments from the second client that the first client also requests are all removed. To capture the performance penalty associated with the cache misses, it is necessary to consider a cache policy in which the cache is cleared for each new client [8]. We consider a 10-segment cache size in our study with the LRU method in the Replacement Manager. In the double-client scenario, the second client starts with a short delay of 20 sec. We observe that when both clients run simultaneously, the second client's segment requests are mostly cache hits. This occurs because the second client requests the same segments as the first client, especially for the first few segments. In this case, there is not much difference in the results of single-client and double-

TABLE I: Exponential smoothing error ratio for double-client scenarios with 10s and 15s

client (playback duration)	α=0.5	α=0.7	α=0.9
client1(10s)	13.42	10.74	9.35
client2(10s)	15.83	12.31	10.25
client1(15s)	20.36	15.79	14.22
client2(15s)	17.51	14.13	13.14

client scenarios. However, by injecting a delay for the second client prior to initiating its connection, we were able to see the effect of cache size on our final results. From our initial experimentations, we found that the first segment is deleted in less than 20 sec when the cache size is 10 segments.

We compared the Smart Cache with other works such as [4] and [8], in which the basic duty of the cache is to prefetch the next segment of the same bitrate as the current segment [8]. Although the cache performs a basic function in these works, some may implement extra cache functions to improve the prefetching scheme, while still maintaining the same basic function. Overall, in this study we consider the basic concept that is common to most of the works mentioned in our related work, and we refer to it as a basic prefetching scheme. After showing the result of throughput accuracy with exponential smoothing, we study the result of cache performance metrics such as cache hitrate and its effect on client throughput. Finally, since various metrics are attributed to QoE [17], we did a QoE study considering the following metrics: average latency, average bitrate gain, and average of bitrate switches or oscillations.

A. Throughput Accuracy

The accuracy of the exponential smoothing forecasting method (2) is dependent on the value used for α. To determine the best values for α, we evaluate the forecasting method under the most tense network background traffic, by applying UDP traffic on the cache-server link for the double-client scenario, with 10s and 15s using α = 0.5, 0.7, 0.9 (See Table I). The accuracy increases as we increase α for all clients in both the datasets. Based on this initial assessment, we chose α = 0.9 for the rest of our study.

It is important to note that the second client has higher error ratio than the first client. The cause for this difference is most likely due to the second client experiencing more traffic on the bottleneck link, which is caused by the presence of the first client's requests. The other observation is that the 15s segment duration video is found to have a higher error ratio compared to the 10s segment duration video.

B. Cache Hitrates

For all the scenarios, the Smart Cache scheme's hitrate is significantly higher than that of the basic scheme's hitrate (see Table II). Furthermore, for the single-client and double-client scenarios, the difference in the hitrate for the Smart Cache compared to the basic scheme increases more for videos with 15s than it does for videos with 10s. On the other hand, the throughput accuracy error from the Smart Cache throughput accuracy section shows that 15s has more error than that of the 10s. The cause of this higher error ratio may be attributed

TABLE II: Cache hitrate for single client client and double-client scenarios

scenarios	basic	Smart	increase
10s-notraffic	66.67%	97.33%	45.99%
15s-notraffic	50.00%	97.5%	95.00%
10s-traffic	41.00%	80.00%	95.12%
15s-traffic	27.00%	70.00%	159.26%
10s-Double-Client	52.17%	79.67%	52.71%
15s-Double-Client	41.5%	73.25%	76.51%

to the different features of data, such as the fact that there are fewer segments with larger size and higher bitrates for the 15s than there are for the 10s.

For the single-client scenario, we also report results without and with background traffic to understand how background traffic affects performance. We observe the highest increase in Smart Cache hitrate as compared to the basic scheme in the single-client scenario when the traffic is added to the cache-server link.

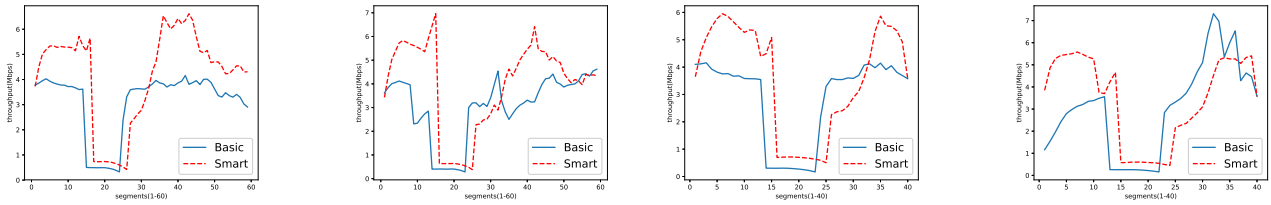
The Smart Cache double-client scenario with a 15s has same hitrate compared to a 10s. Overall, the Smart Cache hitrate increases less when the second client is added to the video fetching process, as compared to Smart Cache with a single client. The reason could be that the cache takes care of two clients at same time and there is more traffic with the presence of the second client.

C. Client Throughput Measurements

In order to understand the client throughput, it is important to understand that the increase in the number of misses in the cache server causes the cache to request the missed segment from the original server, thereby consuming more time and resources for the client to receive the requested segment. Since there is a small number of misses in the Smart Cache scheme, each time that the client requests a segment from the cache, it will be served by the cache itself. This effect will be more prevalent when there is background traffic on the outgoing link of the cache.

We found that the throughput measured by the DASH player is higher with the Smart Cache scheme than with the basic scheme. In Table III, the measured throughput for Smart Cache and basic for single-client with 10s is almost equal, because only the first few segments are cache misses, which have smaller sizes. In the double-client scenarios, the clients that request 15s experience less throughput increase when using Smart Cache than for 10s. The second client for 15s shows the least increase with Smart Cache compared to basic (see Table IV). Due to a higher number of misses in the cache, the difference in throughput increases for the double-client scenario is less when compared to the single-client scenario, specifically for the second client. Note that, in the basic scheme, the client must wait for the cache to fetch all of the missed segments from the origin server.

In Fig. 2, we present throughput behavior with timeline shown based on the segment number in the x-axis. The first peak shows that the throughput measurement is increasing to a threshold while the two clients are requesting segments and background traffic is running; however, it suddenly drops after



(a) First client using 10s (b) Second client using 10s (c) First client using 15s (d) Second client using 15s

Fig. 2: Throughput measurement for clients: Smart Cache vs. Basic.

TABLE III: Client average throughput (Mbps) with a single client scenarios

scenarios	basic	Smart	increase
10s-notraffic	4.17	4.74	13.67%
15s-notraffic	3.73	4.89	31.10%
10s-traffic	3.05	4.13	35.41%
15s-traffic	2.55	3.61	41.57%

TABLE IV: Client average throughput (Mbps) with for doubleclient- scenarios

scenarios	basic	Smart	increase
10s-double client1	2.74	3.65	33.21%
10s-double client2	2.54	3.62	43.70%
15s-double client1	2.45	3.06	24.90%
15s-double client2	2.47	2.78	12.55%

40 sec and stays steady. After the bottleneck is taken away, the trend increases. However, the two clients still compete for resources that results in an unsteady behavior.

D. Analysis of Client QoE

Finally, we focus on client QoE by considering the following metrics: average latency, average bitrate gain, and average of bitrate switches or oscillations.

1) Latency: It has been reported that the time that users wait for the requested content to be downloaded from the server to the local devices can significantly influence user experience [18]. In all the scenarios, Smart Cache displays better QoE than the basic scheme in regards to average latency (see Table V and able VI). For the single-client 10s and 15s without traffic, the delay difference between basic and Smart Cache is not significant. But for the double-client scenarios, we observe that for each client the delay of 15s is more than that of the 10s. In addition, the difference between Smart Cache and basic for 15s is more than 10s. The second client in both datasets experiences less difference between basic and Smart Cache. Thus, an important parameter in QoE experience in terms of average latency is the segment size duration.

Fig. 3 presents trends on latency for 10s and 15s in the double-client scenarios. In the basic scheme, the moment that the traffic runs on the cache-server link occurs at nearly the same time that the client asks for a segment before segment number 15. In the Smart Cache scheme, the moment that the traffic runs on the cache-server link occurs at nearly the same time that the client asks for a segment after segment number 15. This is since in the basic scheme the first few segments are cache missed. When after 40 sec the background traffic is activated on the link, Smart Cache is already a few segments

TABLE V: Average latency and Average Bitrate for a single client in Basic and Smart Cache Schemes with and without traffic

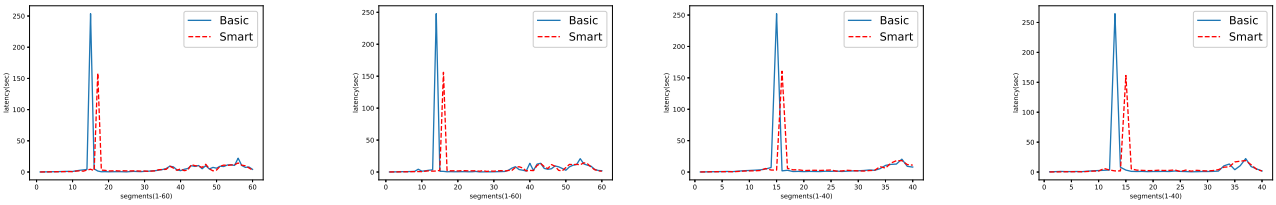
scenarios	Average latency(ms)			Average Bitrate(Mbps)		
	basic	Smart	decrease	basic	Smart	increase
10s-notraffic	5.43	5.20	4.24%	2.79	2.81	0.36%
15s-notraffic	7.63	6.51	14.68%	2.34	2.38	1.71%
10s-traffic	7.58	6.10	19.53%	1.64	1.90	15.85%
15s-traffic	9.41	8.46	10.10%	0.96	1.01	5.21%

ahead of the basic scheme. In the figures, we see that the peak value of latency for the basic scheme is almost 250 sec, while for Smart Cache, this is around 150 sec. This may be attributed to the Smart Cache having accurately prefetched more of the requested segments at that time. This latency can change depending on the size of the segment, which means that if the background traffic happens on higher segment numbers, then we will probably observe more latency and more difference between basic and Smart Cache schemes. If the traffic happens in the beginning of the requests, then the difference will not be significant because the sizes of the segments are smaller.

2) Bitrate Gain: The bitrate metric study shows that for all scenarios, Smart Cache shows higher gain than the basic scheme; see Table V and Table VI. Also, the 15s video shows lower bitrate gain than 10s. Fig. 4 shows additional details on the requested bitrate of those clients. The first peaks in each of the plots are related to the moment when the bitrate already hits the link's throughput threshold for ramp-up phase. In a single-client with no background traffic, this peak would be higher than other scenarios, because there is no traffic to stop it and throttle the link throughput.

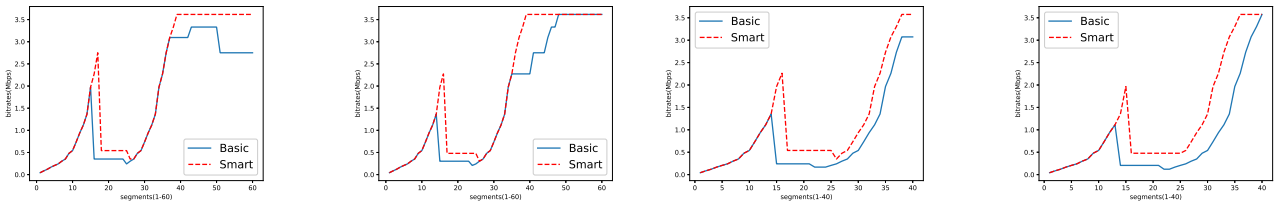
Observe that there is a sudden drop after 40 sec, and the throughput measurements thereafter stay steady on the minimum bitrate. After the bottleneck is taken away, the bitrate values will have sharp upshifts and this trend functions more smoothly as the two clients are still running and competing for link resources. In 10s, the gap between two schemes is more moderate than that of 15s; this relationship is also displayed in numeric values in the tables. Smart Cache compared to basic in the double-client scenario with 15s displays more gain than 10s.

3) Bitrate Switching: In terms of bitrate switching, we did not observe any changes in measurement for the single-client scenario without background traffic. For single-client scenarios with background traffic, the changes are also insignificant. However, there is a small increase in the number of bitrate switching with Smart Cache for 15s (see Table VII). For



(a) First client using 10s (b) Second client using 10s (c) First client using 15s (d) Second client using 15s

Fig. 3: Latency: Smart Cache vs. Basic.



(a) First client using 10s (b) Second client using 10s (c) First client using 15s (d) Second client using 15s

Fig. 4: Bitrate Gain: Smart Cache vs. Basic.

TABLE VI: Average latency and Average Bitrate for double-clinet in Basic and Smart Cache Schemes with a 10s and 15s

	Average latency(ms)			Average Bitrate(Mbps)		
scenarios	basic	Smart	decrease	basic	Smart	increase
client1-10s	7.64	7.10	7.07%	1.63	1.81	11.04%
client2-10s	7.23	6.82	5.67%	1.47	1.78	21.09%
client1-15s	9.83	8.50	13.53%	0.80	1.13	41.25%
client2-15s	9.30	8.58	7.74%	0.75	1.11	48.00%

TABLE VII: Average bitrate switching

scenarios	basic	Smart
Single-10s-traffic	34.4	31.4
Single-15s-traffic	28.2	29.4
Double-10s-client1	31.8	32.4
Double-10s-client2	32.2	32.0
Double-15s-client1	29.0	28.6
Double-15s-client2	30.4	26.4

double-client scenarios, the values for bitrate switching is almost the same except for 15s, where Smart Cache has less bitrate switching compared to the basic scheme. These insignificant changes could be due to the fact that the changes we implement are in the cache itself, and not in the client ABR. When the Smart Cache has higher bitrate switching than the basic scheme, this is related to upshifts. The experience of users is more negatively affected by downshift switching than it is by upshift switching [19]. [20] shows that with larger segment sizes, frequent bitrate switching is not significantly worse than videos with less bitrate switches.

E. Key Observations

We now summarize key observations with double-clients based on our study:

- Cache Hitrate: For 10s, Smart Cache increases the hitrate by 52% and for 15s Smart Cache increases the hitrate by 76% compared to the basic scheme.
- Client Throughput: With Smart Cache for 10s, the throughput measurement of client1 and client2 increase 33.21% and 43.7%, respectively, compared to the basic scheme. This value for 15s double-client for client1 and client2 increases to 24.9% and 12.55%, respectively.
- Delay: With Smart Cache for 10s, delay for client1 decreases by 7.07% while client1's delay decreases by 5.67% compared to the basic scheme. For 15s with Smart Cache, delay decreases for client1 and client2 by 13.53% and 7.74%, respectively.
- Bitrate Gain: With Smart Cache for 10s, client1 and client2's bitrate gain increases by 11.04% and 21.09%, respectively, over the basic scheme. For 15s, the gain increases by 41.25% and 48% for client1 and client2, respectively.

V. Conclusion

In this work, we presented an Smart Cache framework by adding the Replacement Manager and implemented the framework for multiprocessing clients. We observed that when both clients send the requests at about the same time, the second client receives the segment with the same resolution as the first client. When there is a delay in the start of the second client to flush out the cache, we observe that the number of misses in the cache is reduced with the Smart Cache approach compared to the basic scheme, resulting in the improvement of client throughput measurements and QoE metrics. Smart Cache shows a better average latency with a higher average requested bitrate than the basic prefetching.

References

[1] Cisco Visual Networking Index, "Forecast and methodology, 2014-2019 white paper," *Technical Report, Cisco, Tech. Rep.*, 2015.

[2] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, "Enhancing MPEG DASH performance via server and network assistance," 2015.

[3] S. Kiani Mehr, P. Juluri, M. Maddumala, and D. Medhi, "An adaptation aware hybrid client-cache approach for video delivery with dynamic adaptive streaming over http (short paper)," in *IEEE NOMS*, 2018.

[4] K. Liang, J. Hao, R. Zimmermann, and D. K. Yau, "Integrated prefetching and caching for adaptive video streaming over http: an online approach," in *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, 2015, pp. 142–152.

[5] R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2000, pp. 980–989.

[6] T.-D. Pham, P. L. Vo, and T. C. Thang, "Improving dash performance in a network with caching," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*. ACM, 2017, pp. 255–261.

[7] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "Qdash: a qoe-aware dash system," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 11–22.

[8] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahrmehr, "Helping hand or hidden hurdle: Proxy-assisted http-based adaptive streaming performance," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*. IEEE, 2013, pp. 182–191.

[9] C. Liu, M. M. Hannuksela, and M. Gabbouj, "Client-driven joint cache management and rate adaptation for dynamic adaptive streaming over http," *International Journal of Digital Multimedia Broadcasting*, vol. 2013, p. 16 pages, 2013.

[10] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[11] "Adaptation-aware-hybrid-client-cache." [Online]. Available: https://github.com/Sheydakm/Adaptation-Aware-Hybrid-Client-Cache

[12] "DASH dataset 2014." [Online]. Available: http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/

[13] J.-Y. Kim, K.-W. Cho, and K. Koh, "A proxy server structure and its cache consistency mechanism at the network bottleneck," in *Computer Software and Applications Conference, 1999. COMPSAC'99. Proceedings. The Twenty-Third Annual International*. IEEE, 1999, pp. 278–283.

[14] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in http adaptive streaming: Friend or foe?" in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. ACM, 2014, p. 31.

[15] C. Mueller, S. Lederer, and C. Timmerer, "A proxy effect analyis and fair adatpation algorithm for multiple competing dynamic adaptive streaming over http clients," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*. IEEE, 2012, pp. 1–6.

[16] V. Poliakov, L. Sassatelli, and D. Saucez, "Case for caching and model predictive control quality decision algorithm for http adaptive streaming: is cache-awareness actually needed?" in *Globecom Workshops (GC Wkshps), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[17] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[18] Z. Pang, L. Sun, Z. Wang, Y. Xie, and S. Yang, "Understanding performance of edge prefetching," in *International Conference on Multimedia Modeling*. Springer, 2017, pp. 527–539.

[19] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *International Journal of Human-Computer Studies*, vol. 64, no. 8, pp. 637–647, 2006.

[20] S. Egger, B. Gardlo, M. Seufert, and R. Schatz, "The impact of adaptation strategies on perceived quality of http adaptive streaming," in *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*. ACM, 2014, pp. 31–36.