Segment Prefetching at the Edge for Adaptive Video Streaming

Jesús Aguilar-Armijo, Christian Timmerer, and Hermann Hellwagner
Christian Doppler Laboratory ATHENA, Institute of Information Technology, Alpen-Adria-Universität Klagenfurt, Austria
Email: {*firstname.lastname*}@aau.at

***Abstract*—Segment prefetching is a technique that transmits the next video segments in advance closer to the user to serve content with reduced latency. Due to its location and capabilities, an edge computing node is an ideal component for executing segment prefetching policies and storing/caching the prefetched segments. In this work, we study segment prefetching techniques deployed at the edge computing node for adaptive video streaming. We propose different types of segment prefetching policies and study their costs and benefits, including segment prefetching based on past segment requests, transrating, a Markov prediction model and machine learning. Besides, we analyze and discuss which segment prefetching policy is better under which circumstances and the influence of the ABR algorithm and the bitrate ladder on segment prefetching.**

***Index Terms*—Edge computing, MEC, content delivery, adaptive video streaming, HAS, segment prefetching**

I. Introduction

According to the Ericsson Mobility Report [1], video streaming accounts for the majority of mobile network traffic. One of the main reasons for this is the wide use of HTTP Adaptive Streaming (HAS) techniques and protocols such as *HTTP Live Streaming (HLS)* [2] and *Dynamic Adaptive Streaming over HTTP (MPEG-DASH)* [3]. HAS techniques divide the video into different segments or chunks encoded in different qualities, enabling Adaptive Bit Rate (ABR) algorithms to request the segment quality that best fits the current network conditions. Multi-access Edge Computing (MEC) allows for computing, storage and some traffic control capabilities at the network edge, *i.e.*, closer to the end users. Video streaming can be improved by deploying mechanisms at the edge that leverage edge capabilities, achieving a better final Quality of Experience (QoE) for the users.

Segment prefetching is one of the key mechanisms that can be deployed at the edge computing node and support the video streaming process. This mechanism transmits future video segments closer to the user before they are requested, reducing serving latency. We aim to study the benefits of this technique and how it can be executed. Moreover, as this technique can highly improve the final QoE of the users, we propose different segment prefetching policies and examine which policy is better under which conditions. Content Delivery Networks (CDNs) and caching are out of the scope of this paper—we aim to study segment prefetching policies deployed at the edge computing node co-located with a base station of a mobile (*e.g.*, 5G) network cell. There are two main reasons for considering this type of edge computing: This location, closer to the user than regular CDN edges, offers access to Radio Access Network (RAN) metrics using the Radio Network Information Service (RNIS), which can be used to predict future network conditions with machine learning techniques and, therefore, future segment requests. The second reason is that the latency can be critical in some segment prefetching policies, and the edge computing node serves the users from a closer location than CDNs, reducing latency.

The contributions of this paper are as follows:

- Proposal, analysis and evaluation of segment prefetching policies based on *(i)* the last segment request, *(ii)* transrating, *(iii)* a Markov prediction model, and *(iv)* machine learning techniques, w.r.t. their costs and benefits.
- Studies of which of these segment prefetching policies are better under which circumstances, *i.e.*, toward the goals of maximizing users' QoE and minimizing bandwidth consumption and computing power requirements.
- Analysis of the influence of the client-side ABR algorithm as well as the bitrate ladder on segment prefetching.

The remainder of the paper is structured as follows. Section II reviews the related work. In Section III we present the different segment prefetching policies. Section IV evaluates the performance of the different policies. Section V presents the discussion where we attempt to answer key questions related to segment prefetching, as indicated previously. Section VI concludes the paper and presents future work.

II. Related Work

In [4], Chen *et al.* address the problem of proxy jitter, *i.e.*, delayed fetching of uncached segments. The authors propose an active prefetching method that offers a good tradeoff between improving the byte hit ratio and reducing proxy jitter. Moreover, the authors present *Hyper Proxy*, a streaming proxy system that generates minimum proxy jitter with a low delayed startup ratio at the expense of a slight decrease of byte hit ratio compared with similar mechanisms. While it is interesting to reduce proxy jitter, we prefer to focus the comparison on QoE-related metrics such as mean bitrate, mean segment switches, and stalls. In [5], Cao *et al.* propose an MEC-based prefetching and caching framework to improve the users' QoE in high-speed train use cases. The high-speed train can act itself as a moving content proxy, prefetching the next contents when the throughput is high, *i.e.*, when it is close to a base station, storing the segments in the local cache and serving them when the connectivity deteriorates,

e.g., when the high-speed train is moving away from the base station. Nevertheless, this solution focuses on high-speed train scenarios with specific conditions *e.g.*, path loss and Doppler effect. It is preferable to present different segment prefetching policies that can be adapted to diverse scenarios with different restrictions in storage, bandwidth, and computing power. Ge *et al.* [6] present a content delivery system named MVP (Mobile edge Virtualization with adaptive Prefetching) that achieves seamless playback of 4K Video-on-Demand (VoD) by using segment prefetching at the edge computing node. The authors study the tradeoff between prefetching too few segments, which may cause rebuffering, and prefetching too many segments, which can lead to more bandwidth consumption and edge storage usage. Finally, the MVP scheme is evaluated in a real LTE-A network infrastructure under realistic scenarios. The proposed method achieves the goal of guaranteeing 4K VoD content in LTE networks. However, we believe it is interesting to investigate several prefetching methods with different tradeoffs between resource usage and performance, so the service provider can adapt better to its requirements.

While the existing literature covers different aspects of segment prefetching in the context of adaptive video streaming, it lacks a thorough analysis of different segment prefetching policies and the possibilities edge computing offers to improve its performance. Thus, this paper aims to propose and investigate in depth various segment prefetching policies at the edge with respect to their costs and benefits under different circumstances.

III. Study of Segment Prefetching Policies

This section presents several segment prefetching policies, each with different resource usage and performance. We study four types of segment prefetching policies: *(i)* based on the last segment quality, *(ii)* based on transrating, *(iii)* based on a Markov prediction model, and *(iv)* based on machine learning.

A. Prefetching Based on Last Segment Quality

We first study segment prefetching based on the last segment quality. In particular, we analyze three different policies:

- **Last segment quality (LSQ)**: LSQ prefetches the same quality level as in the last segment request. This policy leverages an ABR algorithm's tendency to reduce the number of quality switches as they deteriorate the QoE.
- **Last segment quality plus (LSQ+):** Similar to LSQ, in the LSQ+ policy the edge prefetches: *(1)* the same quality level as in the last segment request; *(2)* one quality above; and *(3)* one quality below.
- **All segment qualities (ASQ):** This policy prefetches all available quality levels for the next segment.

The LSQ policy is expected to behave better in stable network situations with only a few quality switches. The fewer quality switches, the fewer times LSQ will result in prefetching misses, and the more efficient this policy will be. The ABR algorithms play an essential role here, as ABR algorithms that are conservative on quality switches will perform better with LSQ than aggressive ones. The LSQ+ policy is based on the hypothesis that most quality switches are within one quality level higher or lower. Furthermore, this policy balances serving segments directly from the edge and bandwidth consumption in the network's backhaul. Nevertheless, LSQ+ may not be sufficient for unstable networking conditions with a high amplitude of segment switches or when the bitrate ladder contains many different representations. Finally, the ASQ policy assures that the next segment will be served directly from the edge computing node. On the downside, it drastically increases the backhaul network bandwidth and edge storage requirements. However, if multiple users with different network conditions consume the same content, ASQ can be appropriate.

TABLE I
Measured transrating times for 4-sec. segments [ms]

		To					
		Q0	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
From	**Q0**	0	-	-	-	-	-
	Q1	286	0	-	-	-	-
	Q2	538	852	0	-	-	-
	Q3	735	1060	1993	0	-	-
	Q4	914	1202	2124	3856	0	-
	Q5	1731	2008	3081	4519	6819	0

B. Prefetching Based on Transrating

The next segment prefetching policy we analyze is prefetching based on transrating, *i.e.*, the process of converting multimedia files to a reduced bit rate (and resolution, possibly). In this policy, the edge computing node always requests the highest quality level from the server. The main reason is that transrating to a lower quality guarantees an excellent visual quality of the segment as opposed to transrating to a higher quality, where the visual quality might be degraded. When the highest quality segment arrives at the edge node, it is transrated to the quality requested by the user. The extra delay caused by the live transrating step can be compensated by *(i)* the delay reduced by serving the segment directly from the edge node and not from the video server, and *(ii)* the player buffer.

We measure the transrating times of 4-second segments from higher quality levels to lower ones and show the results in Table I. We use the *Big Buck Bunny* video encoded using H.264/AVC [7], in the six different representations of bitrate ladder (C) shown in Table III. The resolution ranges from 320×180 for a bitrate of 200 kbps (quality 0) to 3840×2160 for a bitrate of 17000 kbps (quality 5). We use *FFmpeg* [8] to perform the transrating operations on a regular PC with Intel Core i7-9750H 2.60GHz, 16 GB RAM, 64-bit Windows 10. The transrating times from the maximum quality level to qualities 0, 1 and 2 are lower than the segment duration, but to qualities 3 and 4 they are higher. We can anticipate that when qualities 3 and 4 are requested, the added latency due to transrating might reduce the buffer occupancy at the client, potentially causing stalls. However, it is expected that hardware with higher computing power would be deployed in a real edge node, resulting in transrating times lower than the segment duration. Furthermore, since this policy always requests the highest quality, it will consume more bandwidth than other segment prefetching policies such as LSQ. However, this policy assures a 100% hit rate, *i.e.*, the requested quality from the user will always be served directly from the edge node.

C. Prefetching Based on a Markov Model

We design a segment prefetching policy based on a Markov finite state machine deployed at the edge computing node. During the video streaming session, each user has its own Markov state matrix continuously updated with each request, *i.e.*, the probability of requesting each quality is adjusted considering the previous quality. This policy prefetches the quality with the highest probability of being requested. To perform the prediction, we use a Markov prediction model, where the states represent the different segment qualities of the video, and the state change probability is the probability of switching from one quality to another. Our Markov prediction model consists of an N×N matrix, where N is the number of segment quality levels. This policy has a simple implementation but requires edge capabilities to store the Markov matrix and find the quality with the highest probability of being requested for each user request. Furthermore, the prediction in the early phases might be problematic as the matrix does not have enough information. A possible solution can be to re-use the Markov state matrix of clients with similar network conditions.

D. Prefetching Based on Machine Learning

Machine learning (ML) techniques can accurately predict future segment requests of ABR algorithms. Supervised learning algorithms learn the relation between inputs and outputs using training datasets and can predict future outputs with new inputs achieving excellent accuracy with enough training. This approach is suitable for predicting the behavior of ABR algorithms, as they use input metrics such as throughput, bandwidth status or current buffer size to choose the content quality best suited to the current conditions. Each edge node continuously collects data that can be used to improve its ML model's training and, therefore, the predictions' accuracy. This way, the training data can be adapted for the specific traffic conditions of each edge node.

We use such a policy employing ML techniques to predict the next quality request and prefetch it to the edge. To perform the prediction, the ML model needs some metrics as input. ABR algorithms employ different metrics to perform their decisions, such as throughput-based or buffer-based metrics. Similarly, we select four different input metrics for our prefetching policy:

- Buffer size (in ms): This metric indicates the buffer occupancy in milliseconds when the segment request is made.
- Bandwidth (in Mbps): The perceived bandwidth for the player, obtained by dividing the segment size by the segment downloading time.
- Previous quality: The quality level of the previous request.
- Previous bandwidth (in Mbps): The perceived bandwidth for the player at the last segment request.

The performance of the segment prefetching policy based on ML depends on the accuracy of their segment quality predictions. ABR algorithms with simpler logic are expected to be easier to predict, while more complex ones could lead to mispredictions that deteriorate the performance. The ML-based prefetching policy has minimum bandwidth and storage consumption but requires computing power at the edge to run the ML predictions. We study the accuracy of the predictions, the processing time and the overall performance of this policy.

TABLE II
RANDOM FOREST PREDICTION EVALUATION

	Bitrate ladder (A)			Bitrate ladder (B)			Bitrate ladder (C)		
	TBA	BBA	SARA	TBA	BBA	SARA	TBA	BBA	SARA
Avg. error of LSQ (qualities)	0.27	0.03	0.22	0.2	0.04	0.16	0.36	0.08	0.44
Avg. error of RF prediction (qualities)	≈0	≈0	0.14	≈0	≈0	0.16	≈0	≈0	0.25
Accuracy of RF prediction (%)	99.99	99.96	92.78	99.99	99.99	91.4	99.99	99.98	90.86

Using the simulator and other settings described in Section IV-A, we create nine custom datasets, the combination of three bitrate ladders, (A), (B) and (C), and three different ABR algorithms, throughput-based, buffer-based, and hybrid-based. We choose these ABR algorithms because they use different inputs to make their adaptation decisions. Each dataset includes five fields for each segment request: buffer size, bandwidth, previous quality, previous bandwidth, and the final quality requested by the ABR algorithm. The first four fields are the input metrics for the ML algorithm, and the fifth field is the final output, *i.e.*, the ground truth. We have approximately 30,000 segment requests for each dataset. The Random Forest (RF) model is trained one time per dataset. We split each dataset into training and testing data, assigning 90% and 10%, respectively. We performed our experiments using Random Forest as the chosen ML algorithm, since it provides better results for this task according to related work [9]. We implemented the RF model using the Scikit-Learn package [10] on a Windows machine with Intel Core i7-9750H 2.60GHz, 16 GB RAM, 64-bit Windows 10.

We evaluate the prediction accuracy to evaluate the feasibility of RF prediction-based segment prefetching. This metric is relevant since, if the prediction fails, the requested segment is not prefetched; instead, the request has to be sent to the server, or the user has to cope with a different quality level than the one requested. Furthermore, we also evaluate the computing time it takes to predict the next quality. Similar to the segment prefetching policy based on transrating, the time it takes to predict and deliver the prefetched segment must be lower than the segment duration to preserve the buffer occupancy.

Table II shows the performance of our RF model prediction for each dataset. We aim to improve the performance of the LSQ policy, so we use this policy as a baseline here. We measure the average error of LSQ by averaging the difference between the actual quality requested by the ABR algorithm and the quality prefetched following the LSQ policy for all prefetched segments. Similarly, the average error of RF prediction is measured by averaging the difference between the actual quality requested by the ABR algorithm and the quality prefetched following the ML-based policy for all prefetched segments. We use cross-validation to mitigate overfitting. The results show more than 91% of accuracy of RF prediction for all datasets, including close-to-100% accuracy for the TBA and BBA ABR algorithms, which means that the ML model will predict with almost no mistake the next quality. Finally, we measure the computing time to predict the next segment

TABLE III
BITRATE LADDERS UTILIZED TO CONDUCT THE EXPERIMENTS

	Quality levels (kbps)					
Name	**0**	**1**	**2**	**3**	**4**	**5**
(A)	200	750	2300	4300		
(B)	2300	4300	7000	17000		
(C)	200	750	2300	4300	7000	17000

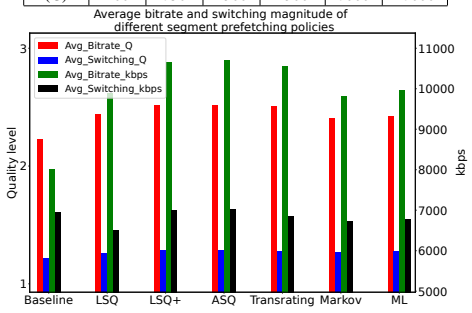

Fig. 1. Average bitrate and switching magnitude of different segment prefetching policies

quality once the model is trained. For that purpose, we tested 100 segment quality predictions and averaged the time, which was ≈75 milliseconds. The added delay is insignificant as compared to the segment duration (*i.e.*, 4s in our evaluation).

IV. EVALUATION

A. Experimental Setup

We use the ANGELA simulator [11] to conduct our experiments. We simulate 100 non-synchronized clients having different video streaming sessions over the same edge computing node, *i.e.*, they share the edge cache, which uses the Least Recently Used (LRU) cache replacement algorithm. The cache size is 1 GB. We use real 4G radio traces [12] with a wide range of radio profiles and mobility patterns such as pedestrian, car, bus, train, and static. We use three types of client-based ABR algorithms: throughput-based [13], buffer-based [14], and hybrid-based (SARA) [15]. We transmit 225 segments with a four-second duration each. We utilize three different bitrate ladders to study the impact of different bitrate ladder configurations on segment prefetching: (A) – from standard resolution up to 1080p; (B) – high resolution, as utilized to broadcast recent events such as the Tokyo Olympic Games [16]; (C) – full bitrate ladder combining bitrate ladders (A) and (B). The three bitrate ladders are shown in Table III. We utilize the representations of a multi-codec DASH dataset [7]. The video popularity follows a Zeta distribution with a shape parameter (α) of 2 [17].

We use six metrics to evaluate and compare the performance of the segment prefetching policies: mean bitrate, mean switching magnitude, mean number of stalls, mean stall duration, prefetching hits percentage, and unused data, *i.e.*, data (measured in size) that was prefetched to the edge node and not used (*i.e.*, not requested by the client).

B. Results

We conduct the experiments following the experimental setup described in Section IV-A. Due to lack of space, in this section, we only show results for the hybrid-based ABR algorithm (SARA) and the bitrate ladder (C). However, we will assess the impact of the ABR algorithm and the bitrate ladder in the discussion of Section V.

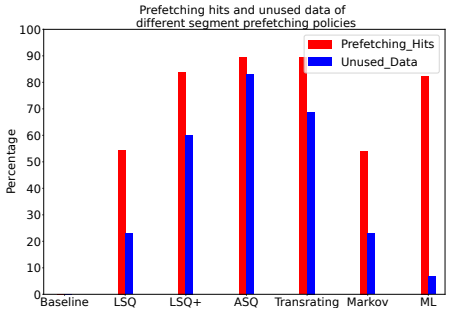

Fig. 2. Prefetching hits and unused data of segment prefetching policies

Figure 1 shows the average bitrate and the average switching magnitude measured in quality levels and kbps for different segment prefetching policies, including baseline results without segment prefetching. The left y-axis shows the average quality level, ranging from 0 to 5, but it is trimmed for better visualization. The right y-axis shows the magnitude expressed in kbps, also trimmed for better visualization of the results. The mean number of stall events and their mean duration turned out to be the same for all segment prefetching policies studied; due to space constraints, these results are not shown. Figure 2 shows the prefetching hits and unused data results for the different segment prefetching policies under study. The prefetching hits are the percentage of segments prefetched at the edge that the client requested regarding the total number of requests (not including the first request as it can not be prefetched). The unused data is the percentage of data sent to the edge node that was not used regarding the total amount of data sent.

The three policies based on the last segment quality, LSQ, LSQ+ and ASQ, achieve a higher average bitrate than the baseline without increasing the switching magnitude. While there is a noticeable improvement in average bitrate from the baseline results to LSQ and from LSQ to LSQ+, there is no big difference between LSQ+ and ASQ. This is because there are not so many switches whose magnitude is more than one quality.

Even though the LSQ policy prefetches just the same segment quality level as the last segment, it achieves hits for more than half (54%) of the prefetched segments. The unused data percentage remains low, close to 23%; the reason is that when it encounters a prefetching miss, the wasted data is just one segment. The LSQ+ policy prefetches the same quality as the last segment, one above, and one below. This increases the unused data percentage up to nearly 60%. On the other hand, it also covers segment switches of the magnitude of one quality level, increasing the prefetching hits percentage up to 84%. The increase in prefetching hits is translated into an increase in QoE but at the expense of more bandwidth consumption. Finally, the ASQ policy prefetches every representation, therefore every segment is served from the edge, either by the cache or by prefetching. For ASQ, the unused data percentage increases up to 83%, the policy with

most backhaul bandwidth consumption, as expected.

Segment prefetching based on transrating has a performance similar to LSQ+ or ASQ, providing a higher average bitrate than the baseline while maintaining the segment switches at the same level. We noticed that the buffer could cope with the extra latency added when transrating to a high quality happens, as the number of stalls is similar to other prefetching mechanisms. We can see a high prefetching hit percentage, the same as for the ASQ policy. Every segment request is served from the edge, either from the cache or by prefetching and transrating. As the maximum quality is sent to the edge for transrating but is not always requested by clients, the unused data percentage in the backhaul is high, up to 69%.

Segment prefetching based on transrating guarantees serving the users' requests directly from the edge computing node, substantially decreasing the latency. However, this policy requires storage and computing power at the edge, and it is not suitable in scenarios with high RAN throughput fluctuation.

The Markov model-based policy improves over the baseline results by having a higher mean bitrate while lower mean segment switching activity. Its performance is similar to LSQ because, in most cases, both policies prefetch the same quality. The prefetching hit and unused data percentages of the Markov policy achieve values similar to the LSQ policy. For most cases, segment prefetching based on the Markov model will request the last segment quality level, usually the most probable one. However, it improves sightly the LSQ performance by adapting smarter to the transitions among different quality representations.

The machine learning policy achieves a higher mean bitrate while reducing the segment switching magnitude. Although the performance of this policy is worse than LSQ+, ASQ and transrating in terms of mean bitrate, the unused data percentage is significantly lower, wasting only 6.7% of data while maintaining a high prefetching hit percentage. This is due to the high accuracy in predicting the next quality level.

V. Discussion

This section aims to answer some key questions related to segment prefetching. First, we analyze which segment prefetching policy is better under which circumstances. Then, the influence of the ABR algorithm is investigated. Finally, we study the influence of the bitrate ladder on segment prefetching.

A. Which Segment Prefetching Policy Is Better?

The actual segment prefetching policy can be characterized based on our preferences and circumstances.

If we want to prioritize the users' QoE, the most evident options are LSQ+, ASQ and transrating, as they provide high mean video bitrates while not increasing mean switching magnitude or stall events. However, these policies imply enormous bandwidth consumption, which can be prohibitive unless the content is very popular, *i.e.*, demanded by many different users under different network conditions.

The best prefetching policies to minimize computing power and storage used at the edge are LSQ, LSQ+, and Markov model-based prediction. These policies have low complexity and do not require high computational power or storage. Only Markov model-based prediction requires preprocessing data to train the model to achieve high accuracy. Although ASQ does not require high computing power as its implementation is simple, the storage costs can be high if many users use this policy and the content popularity is distributed more uniformly.

In case we want to minimize bandwidth utilization while improving the QoE as compared to the baseline (no prefetching), the best segment prefetching policies are LSQ, Markov model-based prediction, and the ML-based policy. These three policies only prefetch one segment, *i.e.*, the one predicted to be requested next by the user. However, if the prediction fails, either the user/player must cope with a different segment quality than requested or the request has to be forwarded to the video server, which increases both bandwidth utilization and delay. Therefore, the better the predictions, the lower the bandwidth consumption.

B. Influence of ABR Algorithm on Segment Prefetching

We conducted the same experiments as shown in Section IV with the TBA and BBA ABR algorithms. We aim to investigate the influence on the segment prefetching of different ABR algorithms that use different metrics to make decisions.

BBA is a smoother algorithm than SARA, with lower segment switching activity and a lower number of stalls, but also a lower average bitrate. Therefore, as there are fewer segment switches, it is easier to predict the next segment, which benefits the policies that rely on predicting the next quality for proper functioning. LSQ improves in terms of prefetching hits, going from 54% with SARA to 82% with BBA while reducing unused data percentage from 23% to roughly 4%. Similar results were obtained with the Markov model policy. The ML-based scheme achieves almost 100% prediction accuracy. Thus, there is little unused data.

TBA offers a higher average bitrate than the BBA algorithm, a similar number of stalls, but also more segment switches. That causes the performance of the different segment prefetching policies to be similar to the SARA algorithm in prefetching hits and unused data transfers. Nevertheless, the ML policy still achieves close to 100% accuracy in predicting the following segment for the TBA algorithm.

In conclusion, smoother ABR algorithms with fewer segment switches will improve the performance of LSQ and of Markov- and ML-based segment prefetching policies. Suppose the ML policy can predict the next segment with approx. 100% accuracy, as for the TBA and BBA algorithms. In that case, it becomes the best segment prefetching policy as it always serves the segment from the edge computing node without additional bandwidth utilization due to mispredictions.

C. Influence of Bitrate Ladder on Segment Prefetching

We performed similar experiments as conducted in Section IV with the bitrate ladders (A) and (B) shown in Table III.

We tested each bitrate ladder with the different segment prefetching policies. As expected, bitrate ladder (A) offers a lower mean bitrate and switching magnitude as the maximum quality is only up to 1080p, but fewer stalls. In contrast, bitrate ladder (B) provides a higher mean bitrate and higher mean switching magnitude, but the number of stalls is also the highest of the three bitrate ladders as it does not have enough representations to cover low-bandwidth scenarios.

The LSQ+, ASQ and ML-based policies have similar performance for all bitrate ladders in terms of prefetching hit and unused data percentages. Even though the number of representations is lower in bitrate ladders (A) and (B) than in bitrate ladder (C), the RF prediction accuracy is similar for each ABR algorithm, resulting in similar performance. The LSQ and Markov-based policies highly benefit from reducing the number of representations, as the prefetching hits increase from 54% for bitrate ladder (C) to 73.8% and 74% for bitrate ladders (A) and (B), respectively. The increase in prefetching hits decreases the unused data sent through the network's backhaul from 23% for bitrate ladder (C) to 8.3% and 12.2% for the bitrate ladder (A) and (B) experiments, respectively. Finally, as bitrate ladder (A) contains representations with lower bitrates and, therefore, lower segment sizes, the transrating policy for bitrate ladder (A) drastically reduces the unused data percentage from 69% to 32.4%.

In conclusion, a bitrate ladder with a high number of representations improves the adaptation of the content to the current network conditions, as there are diverse bitrate options the ABR algorithm can choose. However, it also decreases the accuracy of predicting the next segment for segment prefetching policies. On the other hand, many different representations lead to more segment switches during the video streaming session. More representations also generate more storage and CDN costs.

VI. Conclusions and Future Work

In this work, we present and compare different segment prefetching policies. To perform the evaluation, we use QoE-related metrics, such as average bitrate or switching magnitude, and a cost-related metric, the unused data sent to measure the impact on the network's backhaul.

The best segment prefetching policy depends on our preferences, as the policies offer different performance, bandwidth usage, and computing power utilization characteristics. The LSQ and Markov model-based policies offer a good trade-off between QoE performance and bandwidth usage, with LSQ having a more straightforward implementation than the Markov model but also slightly worse performance. LSQ+ is a more aggressive version of the LSQ policy, prefetching more segments. This more than doubles the costs while assuring more prefetching hits and better performance in terms of QoE. Both ASQ and transrating policies guarantee to serve the users from the edge computing node as the requested segment is always prefetched or created, but they also have a significant bandwidth usage. Finally, ML-based prefetching offers excellent QoE performance with the lowest bandwidth usage. The better the accuracy in predicting the next segment, the lower bandwidth usage this policy will have.

Future work will study different supervised learning algorithms to improve the accuracy of segment prefetching based on machine learning, such as Gradient Boost under different radio conditions and ABR algorithms.

Acknowledgment

The financial support of the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and the Christian Doppler Research Association, is gratefully acknowledged. Christian Doppler Laboratory ATHENA: https://athena.itec.aau.at/.

References

[1] Ericsson, "Ericsson Mobility Report," https://www.ericsson.com/4ad7e9/assets/local/reports-papers/mobility-report/documents/2021/ericsson-mobility-report-november-2021.pdf, Tech. Rep., 2021.

[2] Apple, "HTTP Live Streaming," https://developer.apple.com/streaming/, 2016.

[3] ISO/IEC, "Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," https://www.iso.org/standard/65274.html, Tech. Rep., 2014.

[4] 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.

[5] Y. Cao, N. Wang, C. Wu, X. Zhang, and C. Suthaputchakun, "Enhancing video QoE over high-speed train using segment-based prefetching and caching," *IEEE MultiMedia*, vol. 26, no. 4, pp. 55–66, 2019.

[6] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward QoE-assured 4K video-on-demand delivery through mobile edge virtualization with adaptive prefetching," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2222–2237, 2017.

[7] A. Zabrovskiy, C. Feldmann, and C. Timmerer, "Multi-codec DASH dataset," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 438–443.

[8] FFmpeg, "A complete, cross-platform solution to record, convert and stream audio and video," https://ffmpeg.org/.

[9] H. Yousef, J. Le Feuvre, and A. Storelli, "ABR prediction using supervised learning algorithms," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–6.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] J. Aguilar-Armijo, C. Timmerer, and H. Hellwagner, "ANGELA: HTTP Adaptive Streaming and Edge Computing Simulator," in *2021 10th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN)*. IEEE, 2021, pp. 1–6.

[12] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4G LTE dataset with channel and context metrics," in *Proc. 9th ACM Multimedia Systems Conference*, 2018, pp. 460–465.

[13] D. V. Nguyen, H. T. Le, P. N. Nam, A. T. Pham, and T. C. Thang, "Adaptation method for video streaming over HTTP/2," *IEICE Communications Express*, vol. 5, no. 3, pp. 69–73, 2016.

[14] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2014.

[15] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," in *Proc. IEEE International Conference on Communication Workshops (ICCW)*, 2015, pp. 1765–1770.

[16] D. Yarnell and G. McGilvray, "Behind the scene: delivering 4K olympics games," in *Proceedings of the 1st Conference on Mile-High Video*, 2022, pp. 128–128.

[17] Y. Reznik, T. Teixeira, and R. Peck, "On multiple media representations and CDN performance," in *Proceedings of the 1st Conference on Mile-High Video*, 2022, pp. 56–61.