ML-Driven DASH Content Pre-Fetching in MEC-Enabled Mobile Networks

Rasoul Behravesh*[†], Daniel F. Perez-Ramirez[‡], Akhila Rao[‡],
Davit Harutyunyan*, Roberto Riggio*, Rebecca Steinert[‡]
*Fondazione Bruno Kessler, Trento, Italy
[†]Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy
[‡]RISE Research Institutes of Sweden AB, Stockholm, Sweden
email: {rbehravesh, d.harutyunyan, rriggio}@fbk.eu, {daniel.felipe.perez.ramirez, akhila.rao, rebecca.steinert}@ri.se

***Abstract*—Streaming high-quality video over dynamic radio networks is challenging. Dynamic adaptive streaming over HTTP (DASH) is a standard for delivering video in segments, and adapting its quality to adjust to a changing and limited network bandwidth. We present a machine learning-based predictive pre-fetching and caching approach for DASH video streaming, implemented at the multi-access edge computing server. We use ensemble methods for machine learning (ML) based segment request prediction and an integer linear programming (ILP) technique for pre-fetching decisions. Our approach reduces video segment access delay with a cache-hit ratio of 60% and alleviates transport network load by reducing the backhaul link utilization by 69%. We validate the ML model and the pre-fetching algorithm, and present the trade-offs involved in pre-fetching and caching for resource-constrained scenarios.**

***Index Terms*—video streaming, DASH, caching, pre-fetching, machine learning, MEC, mobile edge, 5G.**

I. Introduction

Video content is the dominant traffic in terms of volume on the current Internet. Cisco forecasts that video will constitute roughly 79% of the total Internet traffic by the end of 2022 [1]. With the ongoing deployment of 5G high-speed networks, the expectation for high-quality 2K/4K video streaming over mobile networks has also increased. The emerging multi-access edge computing (MEC) technology enables mobile network operators (MNO) to provide network services at the mobile edge [2]. The effective handling of video traffic at the edge by MNOs will become increasingly important to satisfy customers with guaranteed video quality of service (QoS), as they stream higher volumes at higher qualities. A video streaming use-case, such as live streaming events at a sports festival, is an example of a challenging scenario wherein a large number of users stream videos in a small area, streaming instant replays, live streams etc. of other games at the sports festival. In the future, this could even include live-VR with stricter QoS requirements.

Currently, DASH [3] is the dominant video delivery standard that has been adopted by most content providers, e.g., YouTube, and Netflix [4]. It dictates that each video is split into equally-sized segments available at multiple video qualities, or *bitrates*. High variability of the bandwidth available to a user in dynamic mobile networks requires an adjustment of the video bitrate based on the current state of the network and playback buffer. This is done by the adaptive bitrate (ABR) algorithm, which, using monitoring information adjusts the bitrate of the next segment request, to maintain the highest possible quality of experience (QoE) for the users [5].

978-3-903176-31-7 © 2020 IFIP

Caching video content closer to the users, at the edge MEC servers, yields benefits both for the users as well as for the MNOs. It reduces the content access delay for the users, improving their QoE [6] while also alleviating the backhaul (BH) transport network load for the MNOs. The limited capacity of the edge MEC servers, however, calls for intelligent decisions on what content to cache and where to cache it, so as to improve QoE while also using the network resources (e.g., storage, bandwidth) in an efficient manner. In this context, the prediction, anticipatory pre-fetching, and caching of video segments of the right quality during streaming, at the MEC servers, plays a pivotal role in MEC-enabled DASH video streaming.

In this work, we employ machine learning (ML) algorithms to predict the number and bitrate of the video segments expected to be requested based on current network bandwidth, playback buffer conditions, and radio network metrics made available by the radio network information service (RNIS) at the MEC. We also predict user base station association to know where to pre-fetch a predicted segment when the user associations are changing. We then formulate an ILP based problem for jointly optimizing video segment pre-fetching, transcoding, and resource allocation, with the objectives of maximizing cache-hit and byte-hit ratios.

II. Related Work

A. Pre-fetching and Caching

The problem of DASH video caching has been well studied in various network deployment scenarios [6–12]. Algorithms for pre-fetching and caching under the constraints of limited BH bandwidth [6, 7] have been proposed and evaluated for maximizing the byte-hit ratio objective. Kumar et al. [12] propose a method that stores the highest bitrate of each segment on the MEC and employs the processing power at the MEC server to transcode the video segment on demand. Liang et al. [6] demonstrate and motivate the benefits of predictive pre-fetching. They consider streaming over a wired network wherein the rate of change of segment bitrate is very low,

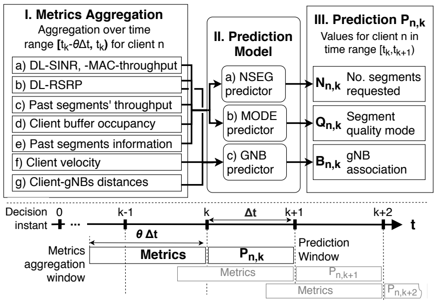

I. Metrics Aggregation
Aggregation over time range [t_k-$\theta\Delta$t, t_k) for client n
a) DL-SINR, -MAC-throughput
b) DL-RSRP
c) Past segments' throughput
d) Client buffer occupancy
e) Past segments information
f) Client velocity
g) Client-gNBs distances
II. Prediction Model
a) NSEG predictor
b) MODE predictor
c) GNB predictor
III. Prediction $P_{n,k}$
Values for client n in time range [t_k,t_{k+1})
$N_{n,k}$ No. segments requested
$Q_{n,k}$ Segment quality mode
$B_{n,k}$ gNB association
Decision instant
0
k-1
k
Δt
k+1
k+2
t
θ Δt
Metrics aggregation window
Metrics
$P_{n,k}$
Prediction Window
Metrics
$P_{n,k+1}$
Metrics
$P_{n,k+2}$

Fig. 1: Prediction inputs and outputs at a decision instant.

justifying their assumption that the next bitrate requested can be assumed to be the same as the previous bitrate requested.

In contrast to existing literature, our work considers the trade-offs that exist between pre-fetching, transcoding higher bitrate video segments, MEC resource allocation, and jointly optimizes them. Our formulation also incorporates using Xn links between base stations (gNBs), making it possible to further improve performance by fetching requested segments cached on adjacent gNBs. With input from our ML prediction, we can evaluate scenarios that are more dynamic and realistic in 5G mobile networks.

B. ML-Driven DASH

Performing predictions in radio access networks (RANs) is especially challenging due to a large number of metrics involved, non-linear relationships between them, and fast-changing network conditions [13]. Using ML for RAN state prediction has hence gained importance in the past years [13–16]. Within the context of DASH, previous work employing ML focus mainly on throughput prediction at the client , which constitutes an input to most ABR algorithms. In this context, random forests (RF) [17], and reinforcement learning with Actor-Critic neural networks [18] have been evaluated. Raca et al. [19] demonstrate that integrating throughput prediction in the client can increase QoE regardless of the employed ABR algorithm.

In summary, compared to the literature, we consider a scenario that specifically addresses the challenges of pre-fetching in a highly dynamic wireless environment by using ML-driven segment prediction, and not throughput prediction, make it agnostic to the underlying ABR.

III. Proposed Method

A. ML Prediction Model

The prediction algorithm runs periodically over *prediction windows* to predict the requests expected in the next window. The ILP decides which of these requests to pre-fetch at the beginning of each window at the *decision instant*. The prediction model's goal is to provide the ILP with information at each decision instant about what segment requests are to be expected from the clients in the next window. For each client, the ILP requires a prediction of the segment requests and their bitrates, as well as a prediction of gNB association to know which MEC server node in the network to cache the content at. This prediction is made for each DASH client at the respective MEC server, at each decision instant before the ILP is run. As shown in Fig. 1, the prediction window size is denoted by Δt. At any given decision instant k, RAN and DASH client-related metrics are collected over the metrics aggregation window, i.e., over the time interval $[t_k - \theta\Delta t, t_k)$, $\theta \in \mathbb{R}^+$. These metrics are fed to the predictor model (see Fig. 1). Since the ILP runs synchronously, the prediction output $P_{n,k}$ for each client n at the decision instant k must include (i) the number of segments expected to be requested by each client over the prediction window, (ii) the expected bitrate of these segments, and (iii) the expected gNB association to place the content. In this work, we devise a ML system composed of these three individual predictors, as we describe below.

Number of Segments: The *NSEG-Predictor* returns the expected number of segments requested by the client n ($N_{n,k}$ in Fig. 1-III). Configurations at the DASH client and the playback buffer's current state determine the client's expected number of requests (which could also be zero). Hence, $N_{n,k}$ is an integer number greater than or equal to 0 ($N_{n,k} \in \mathbb{N}_0$).

Bitrate mode: The *MODE-Predictor* returns the most frequent bitrate of the requested segments for client n ($Q_{n,k}$ in Fig. 1–III). This bitrate $Q_{n,k}$ is predicted and assigned to the $N_{n,k}$ segments expected to be requested. While the bitrate of requested segments within a window can be different, we assign them all the bitrate of the mode since the rate of change of this bitrate is low in reference to our window size. In case there are multiple modes, the highest bitrate is selected since this can be transcoded to a lower bitrate at the MEC. MODE-Predictor performs predictions only on instances with $N_{n,k} \geq 1$.

gNB association: The *GNB-Predictor* returns the expected gNB association for client n ($B_{n,k}$ in Fig. 1–III). gNB association is relevant when considering client mobility since it determines where to place the pre-fetched content. Wrongly allocating MEC capacity to content will negatively affect the cache-hit ratio.

Each predictor is designed as a multi-class classification model. The number of possible classes for NSEG- and MODE-predictor depends on the choice of Δt, which is chosen in the order of seconds. For the GNB-Predictor, it depends on the number of MEC edge servers in the network over which we are performing the ILP optimization. Fig. 1 summarizes the metrics used for the three prediction tasks. The input metrics used to train the NSEG- and MODE-predictors either directly or indirectly influence the bitrate chosen by ABR algorithms (a–e from Fig. 1–I). The input metrics to the GNB-Predictor influence the clients gNB association (b,g,f from Fig. 1–I). Since only averages over windows do not capture the distribution of some fast changing metrics, we also use the 25th, 50th, 75th, and 90th quantiles for metrics a, b, and d.

These metrics can be obtained at the MEC server through the RNIS and a monitoring component that logs the current state of each DASH client based on the segment requests and responses that pass through the MEC. The overall complexity of the prediction procedure scales linearly with the number of clients in the network.

B. ILP Model

Problem Statement: The envisioned mobile network is composed of a 5GC, and gNBs, with MEC servers co-located with gNBs. The MEC servers are characterized by processing, memory, and storage resources. While these resources are limited for the MEC servers at the network edge, they are abundant at the 5GC. Therefore, the resource usage at the 5GC is much cheaper than that of the network edges, though the former also imposes transport network usage costs.

We assume that at any given time, a set of requests will be issued from UEs for a set of video segments, possibly in different bitrates. The ML model proposed in Section III-A is responsible for predicting the requests, the bitrate, and the gNB association of each UE. After obtaining the prediction outcome from the ML model, the ILP model decides whether to pre-fetch the requested video segment(s) in specific bitrates to the network edge, or, if available, to transcode higher bitrate segments available at the network edge to make sure that the bitrate requirements of UEs are satisfied while the network resources are used efficiently.

Depending on the segment duration, segment bitrate, and availability of the substrate network resources, there might be multiple pre-fetching options, each in favor of optimizing certain aspects of the network. The problem of joint video segment pre-fetching, transcoding, and resource allocation can be formally stated as follows.

We consider a 5G network composed of MEC servers co-located with gNBs and the 5GC, which are interconnected via transport network links. A set of UEs that are connected to these gNBs make video segment requests with specific bitrates. Our task is to find a joint video segment pre-fetching, transcoding, and resource allocation solution, with the objective to maximize (i) the cache-hit ratio and (ii) the byte-hit ratio. We define the cache-hit ratio as the number of requests served from the edge (whether directly from the same gNB, from neighbor gNBs, or using transcoding) to the number of requests issued to the network. Similarly, the byte-hit ratio is defined as the number of bytes served from the edge to the number of bytes requested by the UEs.

Mobile Network Model: Let $G = (N, E)$ be an undirected graph modeling the mobile network, where N represents the computing nodes, which are the union of the set of gNBs N_{gnb} and the 5GC N_{5gc}, $N = N_{gnb} \cup N_{5gc}$. E represents the set of BH and Xn links, interconnecting the gNBs with the 5GC and gNBs with each other, respectively. As already mentioned, each node $n \in N$ has a collocated MEC server that is characterized with a storage $C_{stg}(n)$ and processing capacity $C_{cpu}(n)$. While the former is used to cache video segments, the latter, if needed, is used to transcode video segments from a high bitrate h to a lower bitrate q, which is the one predicted to be requested by the UE. There is a link $e^{m,n} \in E$ between the nodes $m, n \in N$ if they are directly connected, which has a certain amount of bandwidth denoted by $C_{bwt}(e)$. N_{vid} represents the set of videos available to the UEs. Each video $v \in N_{vid}$ is divided into multiple segments N^v_{seg}, each of which $s \in N^v_{seg}$ is available in multiple bitrates $N^{v,s}_{br}$. Table I summarizes the parameters of the mobile network.

TABLE I: Mobile network parameters

Parameters.	**Description**
$G = (N, E)$	Graph representing the mobile network.
N	Set of nodes that can host videos $N = N_{gnb} \cup N_{5gc}$
E	Set of links connecting the nodes in G.
N_{gnb}	Set of gNBs in G.
N_{5gc}	Set of core nodes/servers in G.
N_{vid}	Set of videos.
N^v_{seg}	Set of segments of each video $v \in N_{vid}$.
$N^{v,s}_{br}$	Set of available bitrates for each segment $s \in N^v_{seg}$ of video $v \in N_{vid}$. Bitrates are in order from the lowest to the highest $q_1 < ... < q_5$.
$\omega^{h,q}_{cpu}$	Number of CPU cores required for transacting a segment from bitrate h to the desired bitrate q of the user.
$\mathcal{C}_{cpu}(n)$	Number of CPU cores available on node $n \in N$.
$\mathcal{C}_{stg}(n)$	Caching storage of node $n \in N$ in Megabytes.
$\mathcal{C}_{bwt}(e)$	The bandwidth capacity of the substrate link $e \in E$.
τ^s	Segment time duration. All the segments are considered to be in the same duration.
α	The weight factor to prioritize the embedding options.

TABLE II: UE request parameters

Parameters	**Description**
$\bar{G}(\bar{N}, \bar{E})$	Video request graph.
$\bar{N}$	Set of requests in $\bar{G}$.
$\bar{N}^r_{vid}$	The video in the request $r \in \bar{N}$.
$\bar{N}^{r,v}_{seg}$	The segment of video $v \in N_{vid}$ in the request $r \in \bar{N}$.
$\bar{N}^r_{br}$	The bitrate of the request r for its video segment.
$\omega_{br}(r)$	The bitrate of a video segment for the UE's request $r \in \bar{N}$.
$\bar{E}$	Set of links connecting UEs to the requested bitrate in $\bar{G}$.

UE Request Model: The UE requests are modeled as a directed graph $\bar{G} = (\bar{N}, \bar{E})$, where $\bar{N}$ is the union of UEs and their requested bitrate of a specific video and segment, $\bar{N} = \bar{N}_{ue} \cup \bar{N}^{v,s}_{br}$, and $\bar{E}$ represents the virtual links between UEs and their requested bitrate. Moreover, $\omega_{br}(r)$ represents the bitrate of the given requested video segment by the UE. It is possible to have multiple requests from the same UE in any given time. Table II summarizes the notations used for the service requests.

Problem Formulation: The joint video segment pre-fetching, transcoding, and resource allocation problem is modeled as a virtual network embedding (VNE) problem, which is proven to be NP-hard [20]. The embedding process is performed in two steps, including the node embedding and the link embedding step. In the node embedding step, each virtual node (e.g., UEs and video segments) in the request is mapped to a substrate node (e.g., gNBs). In the link embedding instead, each virtual link is mapped to a single substrate path.

1) ILP Formulation: ILP techniques are employed to formulate the described VNE problem that has two objective functions. While the first objective (1) tends to maximize the cache-hit ratio, the second (2) maximizes the byte-hit ratio. Table III summarizes the variables used in the ILP model.

There are multiple ways to increase the cache-hit ratio. The UE requested video segment(s) with a specific bitrate, if already cached/available, can be served either from the host gNB or from a neighbor gNB leveraging the Xn interface. If the requested bitrate of the desired segment is not already available at any of the gNBs/edge sites while higher bitrates of the same video segments are available, then they can be transcoded to the desired bitrate/quality. Note that the video segment pre-fetching is based on the prediction of the UE-requested video segment along with its bitrate for the subsequent timeslot.

$$\boldsymbol{CacheHit:} \quad max \sum_{r\in\bar{N}} \sum_{n\in N_{gnb}} \sum_{\substack{h\in N_{br}^{r} \\ h\geq q}} \alpha\chi_n^{r,h} \tag{1}$$

The second objective (2) has the goal of maximizing the byte-hit ratio, employing the same weighting factors. This objective is particularly beneficial for storing videos with high storage demand at the edge resulting in the BH load alleviation.

$$\boldsymbol{ByteHit:} \quad max \sum_{r\in\bar{N}} \sum_{n\in N_{gnb}} \sum_{\substack{h\in N_{br}(r) \\ h\geq q}} \alpha\chi_n^{r,h}\omega_{br}(r) \tag{2}$$

In the following, we present the constraints that, regardless of the objective function, have to be satisfied for a valid solution. The first constraint ensures that the storage used for storing the videos does not exceed the maximum storage capacity of the edge nodes.

$$\forall n \in N_{gnb}: \sum_{v\in N_{vid}} \sum_{s\in N_{seg}^{v}} \sum_{q\in N_{br}^{v,s}} \chi_n^{v,s,q}\omega_{br}^{v,s}\tau^{s} \leq \mathcal{C}_{stg}(n) \tag{3}$$

At any given time, each UE should be provided with one video segment in the requested bitrate.

$$\forall r \in \bar{N}: \sum_{n\in N} \sum_{\substack{h\in N_{br}^{r} \\ h\geq q}} \chi_n^{r,h} = 1 \tag{4}$$

The following constraint guarantees that a video segment is available at the edge only if at least there is one UE requesting that video segment.

$$\begin{aligned} &\forall n \in N, \forall v \in N_{vid}, \forall s \in N_{seg}^{v}, h \in N_{br}^{v,s}: \\ &\sum_{r\in\bar{N}} \chi_n^{r,h} - \mu\chi_n^{v,s,h} \leq 0 \end{aligned} \tag{5}$$

where s and v are the segment and the video of request r, while μ is a big number. Constraint (6) ensures that the virtual links are mapped on a substrate link as long as the link has sufficient capacity.

$$\forall e \in E: \sum_{\bar{e}\in\bar{E}} \chi_e^{\bar{e}}\omega_{br}(\bar{e}) \leq \mathcal{C}_{bwt}(e) \tag{6}$$

Constraint (7) enforces a continuous path established between the UE and the bitrate of the video segment in the virtual request $r \in \bar{N}$.

TABLE III: Binary decision variables

Variables	Description
$\chi_n^{v,s,h}$	Indicates if the bitrate $h \in N_{br}^{v,s}$ of the segment $s \in N_{seg}^{v}$ of video $v \in N_{vid}$ has been mapped on the edge node $n \in N_{gnb}$.
$\chi_n^{r,h}$	Indicates if the request $r \in \bar{N}$ of the video segment bitrate h has been mapped on the node $n \in N$.
$\chi_e^{\bar{e}}$	Indicates if the virtual link $\bar{e} \in \bar{E}$ is mapped on the substrate link $e \in E$.

$$\begin{aligned} &\forall i \in N, \forall e^{m,n} \in \bar{E}: \\ &\sum_{e\in E^{i\rightarrow}} \chi_e^{e^{m,n}} - \sum_{e\in E^{\rightarrow i}} \chi_e^{e^{m,n}} = \begin{cases} -1 & \text{if } i = m \\ 1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{7}$$

where $E^{i\rightarrow}$ represents the links originating from node $i \in N$, while $E^{\rightarrow i}$ represents all the links entering node $i \in N$.

Finally constraint (8) makes sure that the number of CPU cores utilized for transcoding a video segment in bitrate $h \in N_{br}^{v,s}$ to the lower desired bitrate $q \in \bar{N}_{br}^{r}$ are not higher than the number of CPU cores available at the edge.

$$\forall n \in N_{gnb}: \sum_{r\in\bar{N}} \sum_{\substack{h\in N_{br}^{r} \\ h>q}} \chi_n^{r,h} * \omega_{cpu}^{h,q} \leq \mathcal{C}_{cpu}(n) \tag{8}$$

IV. Evaluation

Here we begin by describing the process of training the ML predictors and their evaluation. We then present an evaluation of the proposed ILP algorithms (with cache-hit and byte-hit objectives) in comparison to a baseline.

A. Data collection for prediction and evaluation

We use an ns-3 [21] simulated network to generate data to train the prediction algorithms and to evaluate the ILP algorithms. The ILP evaluation is done offline by processing the output from the predictor, and the ground-truth from the ns-3 generated data. Two separate datasets [22] were generated from a simulated urban mobile network deployment scenario with the same setup parameters. We use the ns-3 DASH implementation by Vergados et al. [23] to simulate client-server interaction of segment requests and response.

The setup consists of 12 gNBs and uses carrier aggregation with three 20 Mhz component carriers to provide a maximum downlink bandwidth of 60 Mhz, which can reach an aggregated downlink bitrate of up to 225 Mbps. A variable number of UEs (between 27-68) move between these gNBs with velocities ranging between 1.4 - 5.0 m/s (walking/cycling speeds expected in the considered deployment scenario). UEs use DASH to stream video content. Monitoring and application level data is collected over 27 runs of 1000 seconds each. Of these, 23 of them were used to training the ML predictors, and 4 of them for the evaluation of predictive pre-fetching as described in Sec. IV-C.

The UEs in the network request segments from one of 10 live stream videos. The UE's video stream play times are within

10 seconds of each other, representing expected behavior of live streams that are viewed within small delays of the actual event (this delay could be due to replays or delayed requests). The videos consist of 8 second segments and are streamed at 50 frames per second. The set of available bitrates are $\{1, 2.5, 5, 8, 16, 35\}$ Mpbs (higher rates correspond to HD, 2K, and 4K video qualities).

B. ML Prediction

We employ random forest (RF) for the predictor models since it has the flexibility of a non-parametric method and has been demonstrated to perform well for similar tasks [16, 17]. Additionally, we explore gradient boosting trees for classification (XGB) [24] due to prior evidence of boosting trees outperforming RF [25].

1) Pre-Processing: This step consists of reading and parsing the log files generated by various monitoring components in the network. Metrics are aggregated over the metrics aggregation window and the data is parsed and structured over prediction windows for each client's data logs.

2) Prediction Window Size Δt: This parameter directly influences the number of possible classes for the NSEG- and MODE-predictor. Since one bitrate is predicted for all segments in the prediction window, the ideal would be to have only one segment request in each window. This occurs if we use 2 second windows, but results in an imbalanced class distribution with no segments requested $\sim 80\%$ of the time. Additionally, the window size must be significantly larger than the ILP run-time so that there is enough time to pre-fetch the predicted segments before they are requested. We evaluate window sizes of 4, 8, and 16 sec, with number of training samples equal to 211630, 105484, and 52010 respectively, to find the best performing predictors.

3) Training: We employ a train-test set split of 80%-20%. The XGB models use the softmax objective with the default learning-rate and regularization [24]. The RFs perform split decisions based on the Gini impurity, and trees were built using bootstrapping. For both RF and XGB we perform a grid search with the following hyperparameters: i) the number of estimators (number of trees), with possible values $\{5, 15, 50, 100, 200, 350\}$, and ii) the maximum tree depth, with possible values $\{5, 15, 25\}$. Hence, a total of 36 models are trained for each predictor. We employ 10-fold cross-validation, and the best performing model configuration in terms of accuracy on the test set is chosen.

Table IV summarizes the models' performance. The best NSEG-predictor (test accuracy of 83.3%) employs XGB with 350 estimators and 25 max depth using 4-sec for both prediction and metrics aggregation windows. The best MODE-predictor (test accuracy of 89.7%) employs XGB with 350 estimators and 25 maximum depth using a 4-sec prediction and a 10-sec metrics aggregation window. Overall, XGB is generally capable of outperforming RF regardless of the window size combination due to its ability to reduce variance. Moreover, irrespective of the prediction window size, an increase in the metrics aggregation window size decreases the NSEG model's performance. An increase in the prediction window size also reduces overall performance: even matching the number of training samples of the 4-sec prediction window size to that of the 16-sec window yields an 80.7% test accuracy. Since the GNB-predictor problem is simple due to smaller number of metrics with a simple relationship to the target prediction, it obtained a test accuracy close to 100% regardless of the model employed. Overall, the high accuracy score on the training sets suggests that the models are able to learn the separation hyperplanes to classify the data, but the corresponding test accuracy indicates that the models generally overfit, even with a depth of 5. The best performing models were used to provide the required input to the ILP.

C. Pre-fetching

The ILP model that makes the pre-fetching decision was evaluated using the Gurobi mathematical optimization solver [26] in Python. The evaluation was performed on a scenario wherein the 5GC, where the DASH video server runs, is equipped with a 16-core 2.4 GHz processor, and the edge MEC server is equipped with a 4-core 1.6 GHz processor with varying storage of $\{25, 50, 75, 100, 125, 150\}$ MB.

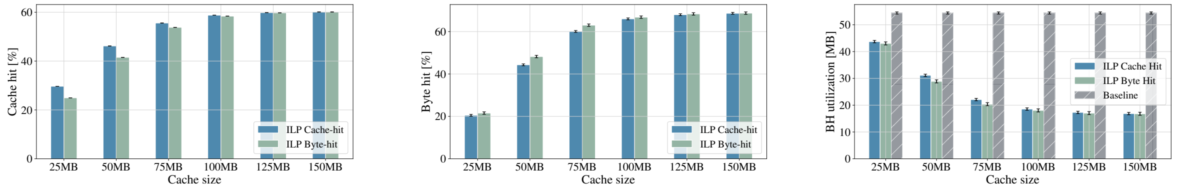
This storage space is shared among all the applications supported by the edge, requiring the use of efficient caching strategies. The edge MEC servers are interconnected over 5 Gbps Xn links, which, along with exchanging control information, are also used to transfer video segment data. The MEC servers are connected to the 5GC over a 20 Gbps BH link.

The pre-fetching ILP algorithm runs periodically every 4 seconds (as was chosen based on evaluations presented in Sec. IV-B), in what we call a pre-fetching time slot. In each pre-fetching time slot, a decision maker at the 5GC obtains the segment/bitrate/association predictions for all UEs from the MEC and puts together a set of pre-fetching requests. Each request is defined by the tuple [destination UE, segment number, bitrate, gNB association of UE]. The ILP tries to find a solution that serves all predicted requests. The pre-fetching time slot matched with the prediction time window is chosen to be always longer than the time to run the pre-fetching algorithm to ensure that the requests from one slot are fetched before the beginning of the next.

The predicted requests can be addressed by either pre-fetching the requested segments to the edge, transcoding an existing higher bitrate segment at the edge, or just waiting for the 5GC to serve the actual request when it occurs. Three cases result in a user being served from the 5GC. (i) the ILP decides that it cannot pre-fetch the request due to resource constraints (ii) due to an error in prediction, the wrong segment bitrate was pre-fetched, and (iii) due to error in gNB association prediction, the pre-fetched segment was placed at the wrong edge MEC server. The cache at the MEC server uses a simple least recently used (LRU) replacement strategy for content management. Data from the ns-3 simulator is processed offline using the output from the ML predictor as input to the ILP and comparing the status of the cache with ground truth values of real requests to

TABLE IV: Accuracy of the trained ML models using different prediction window and metrics aggregation window sizes.

Prediction w-size:		4 sec						8 sec						16 sec			
Metrics w-size:		4 sec		10 sec		20 sec		8 sec		10 sec		20 sec		16 sec		20 sec	
Acc. for set [%]:		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
NSEG	RF	97.4	82.8	97.6	79.5	97.3	77.0	96.5	75.2	96.5	74.7	96.1	73.4	95.8	**60.2**	95.8	58.3
	XGB	98.3	**83.3**	98.1	80.7	86.4	79.3	97.6	**76.0**	97.6	75.5	97.4	73.9	95.9	59.5	95.9	57.9
MODE	RF	98.6	87.1	98.8	88.8	98.6	86.3	98.5	85.5	98.5	85.0	98.1	81.9	97.7	77.4	97.5	76.3
	XGB	98.8	88.0	98.9	**89.7**	98.7	87.8	98.6	86.0	98.6	**86.3**	98.3	83.3	97.7	**77.6**	97.6	77.2



(a) Cache-hit ratio.

(b) Byte-hit ratio.

(c) BH link utilization.

Fig. 2: Average cache-hit ratio, byte-hit ratio and BH link utilization for different cache sizes.

obtain the evaluation results. The results are averaged over 4 runs and 95% confidence intervals are obtained.

Cache-hit ratio: Fig. 2a shows the cache-hit ratio under different cache sizes. A higher cache-hit results in the end-users experiencing less delay, resulting in obtaining higher bitrates, and also reducing the MNOs' BH traffic.

Fig. 2a shows that the average cache-hit ratio, for smaller cache sizes, is higher for the cache-hit objective compared to the byte-hit objective. When the cache storage is very limited, all the predicted segments cannot be accommodated at the MEC servers. In such a scenario the ILP cache-hit can make more intelligent decisions compared to the ILP byte-hit algorithm in selecting a set of segments and bitrates to be pre-fetched that leads to the maximum cache-hit. Although, with the increase in the cache size, we encounter a scenario in which all the predicted segments can be stored at the MEC servers; therefore, making the pre-fetching decision straightforward for all of the algorithms to pre-fetch whatever predicted and achieving a high cache-hit ratio.

Byte-hit ratio: Obviously, with more bytes served at the edge, more savings over the BH link can be achieved. This performance metric captures the ratio between the number of bytes served from the edge and the overall number of bytes requested. Therefore, we expect the ILP byte-hit to perform better than the ILP cache-hit algorithm in pre-fetching higher bitrate segments that can be shared among multiple UEs to save BH bandwidth. Another advantage with the pre-fetching of high bitrate segments is that a higher bitrate segment can be transcoded to the lower bitrate segments and avoid the request being re-directing to the core.

As illustrated in Fig. 2b, the number of bytes served from the edge for the ILP byte-hit algorithm is more than the ILP cache-hit algorithm. Similar to the cache-hit evaluation, the algorithm shows a better comparable performance concerning the ILP cache-hit when the cache size is smaller, and the algorithm needs to decide on pre-fetching intelligently video segments to the edge. Here, the ILP cache-hit shows a close performance to ILP byte-hit because it can pre-fetch more number of segments, increasing the number of served bytes, but still being less than the ones achieved by ILP byte-hit.

Link utilization: Figure 2c illustrates the BH link utilization as a function of cache size. This is compared with the baseline in which all the segments are served from the core and no pre-fetching is performed. We can observe that ILP byte-hit achieves the lowest BH link utilization compared with the baseline. This is justified by the fact that ILP byte-hit tends to pre-fetch high bitrate segments that might be shared among multiple requests at the edge. Therefore, a smaller portion of the high bitrate requests will be directed to the core, and the BH link will be conserved. As seen in Fig. 2c, the proposed algorithm can save the BH link up to 69%.

V. Conclusion

In this paper, we proposed a novel method for ML-driven predictive pre-fetching for the problem of DASH video streaming in MEC-enabled mobile networks. The ML model predicts the number of segment requests, bitrate, and the gNB association of the UEs over a prediction time window. An ILP model with two objectives has been proposed for video content pre-fetching and transcoding at the edge. We showed that with an accuracy of (83-88-99%) for the three predictive tasks and using the MEC to pre-fetch (or transcode) segments we gain the cache-hit ratio of 60%, which implies a reduction in access delay for 60% of the requests. We demonstrated that the BH link utilization could be reduced by 69% through caching at the edge using max byte-hit objective in a live streaming scenario.

Acknowledgments

This work has been funded by the EU's Horizon 2020 project 5G-CARMEN (grant no. 825012), by the Celtic Next 5G PERFECTA project (VINNOVA, grant no. 2018-00735), and by the Swedish Foundation for Strategic Research (SSF) Time Critical Clouds project (grant. no. RIT15-0075).

References

[1] G. M. D. T. Forecast, “Cisco visual networking index: global mobile data traffic forecast update, 2017–2022,” *Update*, vol. 2017, p. 2022, 2019.

[2] ETSI, “Etsi gs mec 002 v1.1.1,” *Mobile edge computing (MEC);technical requirements*, 2016.

[3] IEC_MPEG, “Information technology-dynamic adaptive streaming over HTTP (DASH)-part 1: media presentation description and segment formats. ISO,” 2012.

[4] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, “A survey on bitrate adaptation schemes for streaming media over HTTP,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 562–585, 2018.

[5] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, “A survey on quality of experience of HTTP adaptive streaming,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2014.

[6] K. Liang, J. Hao, R. Zimmermann, and D. K. Yau, “Integrated prefetching and caching for adaptive video streaming over HTTP: an online approach,” in *Proc. of ACM MM*, Portland, Oregon, USA, 2015.

[7] P. Juluri and D. Medhi, “Cache’n DASH: Efficient caching for DASH,” *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 599–600, 2015.

[8] A. Araldo, F. Martignon, and D. Rossi, “Representation selection problem: optimizing video delivery through caching,” in *Proc. of IEEE IFIP*, Vienna, Austria, 2016.

[9] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, “A mobile edge computing-based architecture for improved adaptive HTTP video delivery,” in *Proc. of IEEE CSCN*, Berlin, Germany, 2016.

[10] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, “A mobile edge computing-assisted video delivery architecture for wireless heterogeneous networks,” in *Proc. of IEEE ISCC*, Heraklion, Greece, 2017.

[11] Y. Tan, C. Han, M. Luo, X. Zhou, and X. Zhang, “Radio network-aware edge caching for video delivery in MEC-enabled cellular networks,” in *Proc. of IEEE WCNCW*, Barcelona, Spain, 2018.

[12] S. Kumar, D. S. Vineeth *et al.*, “Edge assisted DASH video caching mechanism for multi-access edge computing,” in *Proc. of IEEE ANTS*, Indore, India, 2018.

[13] A. Samba, Y. Busnel, A. Blanc, P. Dooze, and G. Simon, “Instantaneous throughput prediction in cellular networks: Which information is needed?” in *Proc. of IFIP/IEEE IM*, Lisbon, Portugal, 2017.

[14] M. Karimzadeh, Z. Zhao, L. Hendriks, R. D. O. Schmidt, S. La Fleur, H. Van Den Berg, A. Pras, T. Braun, and M. J. Corici, “Mobility and bandwidth prediction as a service in virtualized LTE systems,” in *Proc. of IEEE CloudNet*, Niagara Falls, ON, Canada, 2015.

[15] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, and V. Gopalakrishnan, “Back to the future: Throughput prediction for cellular networks using radio KPIs,” in *Proc. of ACM MOBICOM*, New York, NY, USA, 2017.

[16] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei, “LinkForecast: cellular link bandwidth prediction in LTE networks,” *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1582–1594, 2018.

[17] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello, “Empowering video players in cellular: Throughput prediction from radio network measurements,” in *Proc. of ACM MM*, New York, NY, USA, 2019.

[18] H. Mao, R. Netravali, and M. Alizadeh, “Neural adaptive video streaming with pensieve,” in *Proc. of ACM SIGCOM COMM*, New York, NY, USA, 2017.

[19] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello, “Incorporating prediction into adaptive streaming algorithms: a QoE perspective,” in *Proc. of ACM SIGCOM NOSSDAV*, New York, NY, USA, 2018.

[20] R. Behravesh, E. Coronado, D. Harutyunyan, and R. Riggio, “Joint user association and VNF placement for Latency sensitive applications in 5G networks,” in *Proc. of IEEE CloudNet*, Coimbra, Portugal, 2019.

[21] “Network Simulator ns-3.” [Online]. Available: https://www.nsnam.org/

[22] “ns-3 generated dataset for DASH over dynamic radio access networks.” [Online]. Available: https://github.com/akhila-s-rao/ns3_dash_over_ran/

[23] “An MPEG/DASH client-server ns3 module,” Accessed on 23.07.2020. [Online]. Available: https://github.com/djvergad/dash

[24] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. of ACM SIGKDD*, New York, NY, USA, 2016.

[25] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *Proc. of ACM ICMLC*, New York, NY, USA, 2006.

[26] “Gurobi mathematical optimization solver,” Accessed on 05.07.2020. [Online]. Available: https://www.gurobi.com/