Improving UE Energy Efficiency Through Network-Aware Video Streaming Over 5G

Basabdatta Palit, Argha Sen, Abhijit Mondal, *Member, IEEE*, Ayan Zunaid, Jay Jayatheerthan, and Sandip Chakraborty, *Member, IEEE*

***Abstract*—Adaptive Bitrate (ABR) Streaming over cellular networks has been well studied in the literature; however, existing ABR algorithms primarily focus on improving the end-user Quality of Experience (QoE) while ignoring the resource consumption aspect of the underlying device. Consequently, proactive attempts to download video data to maintain the user's QoE often impact the battery life of the underlying device unless the download attempts are synchronized with the network's channel condition. In this work, we develop EnDASH-5G – a wrapper over the popular DASH-based ABR streaming algorithm, which establishes this synchronization by utilizing a network-aware video data download mechanism. EnDASH-5G utilizes a novel throughput prediction mechanism for 5G mmWave networks by upgrading the existing throughput prediction models with a transfer learning-based approach, leveraging publicly available 5G datasets. It then exploits deep reinforcement learning to dynamically decide the playback buffer length and the video bitrate using the predicted throughput. This ensures that the data download attempts get synchronized with the underlying network condition, thus saving the device's battery power. From a thorough evaluation of EnDASH-5G, we observe that it achieves a near 30.5% decrease in the maximum energy consumption than the state-of-the-art Pensieve ABR algorithm while performing almost at par in term of QoE.**

***Index Terms*—4G LTE, 5G, mmWave, energy efficiency, ABR video streaming, cellular networks, mobility, transfer learning, QoE.**

I. Introduction

SUSTENANCE in the new normal way of life stimulated by the global pandemic has been made possible by the pervasive usage of online video streaming services. These services range from official meetings and classes to the increase in screen time over *Over the Top* (OTT) media services, such as Netflix, Amazon Prime, etc. [1]. The quality of the streaming sessions depends on the underlying network quality. Mobile Internet uses cellular connectivity, which may display unwarranted fluctuations and may have unidentified blind spots with little to no reception. In such cases, the users experience a very low throughput, which affects the quality of the streaming sessions. As cellular networks prepare to migrate to mmWave communication in the 5^{th} Generation (5G), fluctuations are going to become more incessant [2] due to the increased susceptibility of the high-frequency communication to path loss and scattering.

Such fluctuations in the channel quality not only impact the Quality of Experience (QoE) of video streaming users but also affect the devices' energy consumption. Existing adaptive bitrate (ABR) video streaming protocols are generously biased towards user-experienced QoE and minimizing data wastage than saving device energy consumption. To minimize data wastage during the playback, the existing video players typically use *fixed-size buffers* to download and store video chunks. The download of future video chunks is synchronized with the video playback – the download is stalled until the user finishes watching a portion of the already downloaded video chunks. These approaches assume that the user may skip watching a part of the video, and, thus, such schemes can reduce data wastage. However, these impose an additional constraint that the next download session should initiate immediately after the available data in the playback buffer is depleted beyond a threshold. Consequently, the player is bound to initiate a download request even if the underlying network's signal quality is poor, thereby resulting in higher energy consumption at the User Equipments (UEs). A recent measurement study on 5G device energy consumption [3] shows that services such as ABR video streaming are more susceptible to increased energy usage in 5G mmWave networks. It is, therefore, imperative to design energy-efficient video streaming algorithms for 5G mmWave networks [4], [5], [6], especially to cater to the ubiquitous QoE demand of vehicular UEs.

Supporting QoE for video streaming and the device's energy efficiency simultaneously needs a strong *cross-layer* integration between the streaming application and the underlying network, as the download sessions need to be synchronized with the network quality. Existing streaming applications predominantly use Dynamic Adaptive Streaming over HTTP (DASH) [7], an ABR streaming mechanism over HTTP. In DASH, a target video is broken into chunks of fixed duration

Manuscript received 24 April 2022; revised 7 August 2022, 8 December 2022, and 11 February 2023; accepted 14 February 2023. Date of publication 28 February 2023; date of current version 9 October 2023. The funding for this work has been provided by Intel Technology Pvt. Ltd., India as part of the project "Traffic Engineering for Enabling Energy-aware Design in Next Generation Cellular Networks". The associate editor coordinating the review of this article and approving it for publication was H. Lutfiyya. *(Corresponding author: Sandip Chakraborty.)*

Basabdatta Palit is with the Department of Electronics and Telecommunication Engineering, Indian Institute of Engineering, Science and Technology Shibpur, Howrah 711103, India (e-mail: basabdatta.iitkgp@gmail.com).

Argha Sen and Sandip Chakraborty are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India (e-mail: arghasen10@gmail.com; sandipc@cse.iitkgp.ac.in).

Abhijit Mondal is with Adweb, VDX.tv, Noida 201305, India (e-mail: am@abhijitmondal.in).

Ayan Zunaid is with Flipkart Pvt. Ltd, Bengaluru 560103, India (e-mail: ayanzunaid10@gmail.com).

Jay Jayatheerthan is with Intel Technology Pvt. Ltd., Bengaluru 560103, India (e-mail: jay.jayatheerthan@intel.com).

Digital Object Identifier 10.1109/TNSM.2023.3250520

and stored at a Content Distribution Network (CDN) server at different qualities. During a video session, the DASH video player at the end-user device uses the estimated network throughput and the application layer playback buffer length to decide the quality (bitrate) at which the next chunk is to be fetched. However, to ensure that the device's energy efficiency improves, we argue that not only the bitrate for the next chunk but also the maximum playback buffer length needs to be tuned adaptively to the *predicted network condition* for the near future. Thus, in line with our earlier work in [8], we propose *EnDASH-5G*, a client-side energy-efficient video streaming algorithm that acts as a wrapper over DASH – custom-made for 5G mmWave communication. EnDASH-5G is essentially a rate adaptation mechanism, which pursues opportune fetching of video chunks tuned to the wireless link condition through the following operations – (i) It first predicts the cellular throughput from radio-related parameters. (ii) It then uses the predicted throughput to estimate the maximum playback buffer length such that the energy usage is minimized. (iii) Finally, it chooses the optimal bitrate for the next video chunk to be downloaded using the estimated playback buffer length. The earlier version of EnDASH in [8] was designed to suit the network quality variations of 4^{th} Generation (4G) only. The corresponding throughput prediction, as well as the buffer length and bitrate prediction, are not directly extensible to 5G mmWave networks. The reasons are as follows.

1) The throughput prediction mechanism used in EnDASH depends on 4G network parameters, such as received signal strength, base station characteristics, handover history, etc. These parameters are drastically different in a typical 5G network. In addition, 5G being a highly directional communication technology due to its high frequency of operation, other factors, such as the number of antennas, the direction of antenna beams, etc., also impact the received throughput [3]. This inevitably implies that the throughput prediction model will need a complete redesign. Furthermore, many of the input parameters to the throughput prediction engine of EnDASH are specific to a service provider or locality of operation. As the availability of large-scale commercial deployments of 5G networks is still scarce, it is difficult to have sufficient data for training a supervised learning model. This also calls for a new throughput prediction engine.
2) The cross-layer integration of the video streaming at the application layer with the network needs a complete redesign as the network connectivity model is different in 5G compared to a 4G network. For example, 5G uses a more dense deployment of gNodeBs (gNBs) compared to 4G evolved NodeBs (eNBs). Such an underlying deployment directly impacts the communication environment that needs to be considered during model development.
3) The 5G NR UE energy consumption model is also significantly different from 4G, which impacts the analysis and measurement of the device energy consumption during data communication. Hence, the model for energy analysis needs to be upgraded considering the 5G UE energy states.

Considering these factors, in this paper, we modify our earlier model in [8] to EnDASH-5G – an energy-efficient adaptive bitrate streaming protocol for 5G mmWave networks. To understand the impact of 5G network variability on the QoE and device energy consumption, we perform a thorough pilot study (Section III) over a simulated 5G mmWave network using the `ns3 – mmWave` [9] simulator framework. The current protocols for adaptive bitrate streaming attribute a higher weight to the application layer playback buffer length than the user's instantaneous received signal strength. This inevitably manifests in the increased energy consumption by accessing the network under poor channel conditions (Section III), thereby asserting that there is room for improved energy efficiency by tuning the playback buffer length as well as the bitrates of the video downloads to network variations in the underlying 5G mmWave networks. The network throughput needs to be predicted for synchronizing the video downloads to the network changes. To address the issue related to the model training for throughput prediction, we use an innovative approach in this paper by applying the *Transfer Learning* [10] technique. Transfer learning is used to compensate for the limited availability of the 5G network throughput traces. For this, we first train a recurrent neural network with real-world 5G datasets [3], [11], [12] and then retrain the learned model with data collected from extensive simulations of a 5G mmWave network. The simulations help us capture every little nuance associated with the functioning of the 5G mmWave network while also allowing us to vary the number of active users in the network and observe the corresponding performance over the video streaming application. Based on the predicted future throughput, we design a reinforcement learning (RL) mechanism which dynamically tunes the maximum playback buffer length and the bitrate for the next video chunk to the network throughput variation, intending to maximize the QoE and minimize the UE's energy consumption. The buffer-length and bitrate adaptation engines operate using $A3C$ [13], a state-of-the-art actor-critic RL algorithm, and run asynchronously concerning one another. We train individual prediction modules over a large corpus of collected data traces and evaluate EnDASH-5G using an emulation environment. The source code of EnDASH-5G has been made publicly available.[1]

EnDASH-5G has been compared in terms of QoE and energy consumption with state-of-the-art ABR schemes, such as BOLA [14], Fast-MPC & Robust-MPC [15], and Pensieve [13]. Our evaluation shows that in comparison to existing ABR algorithms, EnDASH-5G significantly improves the energy savings in 5G mmWave users by 30.5% (Section VI). Energy saved from playing a 100 second video using EnDASH-5G can be used to gain an additional 36 seconds of video playback time compared to Pensieve, which delivers the highest QoE among all ABR algorithms.

II. Background and Related Work

In this section, we discuss the energy consumption in 5G New Radio (NR) UEs and subsequently discuss the existing ABR video streaming algorithms.

[1] https://github.com/arghasen10/endash (Accessed: February 26, 2023)

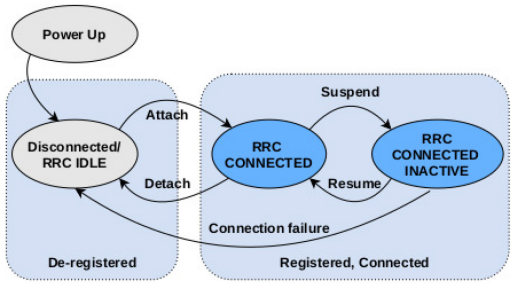


Fig. 1. User-Equipment (Mobile Phone) RRC State Machine for 5G NR [16].

A. UE Energy Consumption Model in 5G

Energy management in 5G NR user equipments is governed by a three-state Radio Resource Control (RRC) state machine as shown in Fig. 1. The three states are: 1) an RRC CONNECTED state in which there is an active data transmission or reception, 2) an RRC IDLE state with no data activity, and 3) an RRC CONNECTED INACTIVE state between two successive active transfers. The RRC CONNECTED INACTIVE state was introduced in 5G to reduce the control plane latency and the energy required for transitioning from RRC IDLE to RRC CONNECTED state. So, in the RRC CONNECTED INACTIVE state, the UE context is maintained both by the network and the UE. Simultaneously, the connections between the access and core networks are also maintained.

B. Related Work

The existing works on ABR streaming can be explored from two different directions – works that focus on improving the video QoE and works that target reducing the on-device energy consumption during video streaming.

1) Improving QoE: In [17], the authors propose to improve the QoE of video streaming through DASH-aware bandwidth allocation. References [18], [19] use client-server cooperation to select video streaming bitrates and improve the QoE. In contrast, our work assumes that no server or Base Station (BS) side information is available at the client. So, the objective is to optimize the ABR streaming at the client using only the client-side information.

Buffer-based ABR algorithms like BOLA select optimal download bitrates as a function of buffer occupancy [14], [20]. While MPC [15], Pensieve [13], Oboe [21], etc., select bitrates as a function of buffer length and network throughput [13], [15], [21], [22], [23], [24], [25], [26]. Of these, Pensieve [13], Oboe [21], and HotDash [23] use deep Reinforcement Learning (RL) algorithm for optimal bitrate selection to maximize over a QoE metric. Several works have used bandwidth prediction [27], [28], [29], [30], [31] or predicted cellular network throughput for improved bitrate selection [28], [29], [30], [31], [32], [33]. The ABR streaming algorithm in [26] has been trained using the combined data generated from several well-known ABR algorithms and, therefore, combines the characteristics of all these algorithms. These works [13], [15], [22], [23], [24], [25], [26], [30], [32], [33], [34] improve the QoE of users by optimizing the chunk bitrates while paying little or no attention to improving device energy consumption.

2) Reducing Energy Consumption: Several works have investigated the energy consumption of mobile phones independently of QoE or ABR streaming algorithms. GreenTube in [35] proposes to tune cache management to user behavior and network conditions. The energy-aware QoE-driven bitrate adaptation algorithms in [5], [36] use the remaining energy of the smartphone. In [6], authors show the importance of the user's context in QoE estimation and the battery life and have designed context-aware and energy-aware video streaming. Reference [4] leverages Deep Reinforcement Learning-based approaches for finding the optimum policy to minimize the accumulated energy consumption of each BS with minimized video playback interruptions. EVSO in [37] proposes a new perceptual similarity calculation method that uses the information generated by the video encoder and adjusts the frame rate according to the degree of motion intensity. This implementation is more focused on the upstream service. A popular method to reduce energy consumption in mobile phones is to optimize the tail energy, which is achieved in [38] by either prefetching or delaying packets. The authors in [3] proposed an energy-saving mechanism that opportunistically switches the network interface from 5G to 4G when the 5G mmWave network condition worsens. Such vertical handovers incur increased signaling message exchanges, as well as increased energy consumption [8]. It can also cause ping-pong vertical handovers between 4G and 5G during incessant network quality fluctuations. The consequent impact on energy consumption may be non-negligible.

The question that remains to be answered is – How can the QoE of users and the corresponding energy consumption be jointly optimized? A potential answer to this question has been provided in our previous work [8], which proposed a method and system to improve the energy consumption of 4G smartphones during video streaming by tuning the video download sessions to the 4G network connection quality fluctuations. Tuning the bitrate to connection fluctuations reduces the CONNECTED state dwell time and, hence, the energy consumption while the bitrates are selected to maximize the QoE of the user. Although [8] achieves energy savings, it does not investigate the implication of using the EnDASH algorithm in 5G mmWave networks, which is addressed in this work.

III. Pilot Study

To analyze the impact of network parameters on video streaming applications over 5G mmWave networks, it is essential to understand the complex interactions among different network-related parameters (such as received signal strength, handovers, etc.) on one hand and device/application-related parameters (such as UE speed, application throughput, device's energy consumption, video playback buffer length, video playback quality, etc.) on the other hand. In this work, we have used extensive system-level simulation-based studies to interpret the relationship between network parameters and device/application performance counters.

TABLE I
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Bandwidth of mmWave eNBs	1 GHz	mmWave carrier frequency	28 GHz
mmWave transmission power	30 dBm	Bandwidth of the LTE eNB	20 MHz
LTE carrier frequency	2.1 GHz	LTE Downlink transmission power	30 dBm
LTE Uplink transmission power	25 dBm	Noise Figure	5 dB
eNB Uniform Planar MIMO array size	8× 8	UE UPA MIMO array size	4 × 4
eNB scanning directions	8	UE scanning directions	16
Sounding Reference Signal (SRS) duration	10 μs	Period between SRSs	200 μs
Radio Link Control (RLC)buffer size	10 MB	One-way delay on X2 links	1 ms
UDP payload size	1024 bytes	UDP packet inter-arrival time	20, 80 μs
One-way Mobility Management Entity (MME) delay	10 ms	ABR Algorithms for DASH application	BOLA [14], Pensieve [13], Robust-MPC [15], Fast-MPC [15]

Why simulation? Ideally, in order to understand the effect of network behavior on the performance of mobile devices, extensive measurement-based studies on real 5G mmWave networks are desirable. However, the 5G technology remains within the development phase, and thus, the availability of commercially deployed 5G networks is limited, particularly in middle and low-economy countries. Although some existing works present real network 5G mmWave datasets [3], [11], [12], these have been collected using one or two smartphones, i.e., users. As the user-experienced datarate is a function of the network load, i.e., the number of users associated with a BS, hence, it is significant to test the efficacy of any new algorithm for different network loads. However, the information associated with the network load is proprietary and is available only at the BS.

Using a simulation platform helps to overcome this issue wherein datasets can be generated, and algorithms can be tested for any number of active users. An important advantage is that it can be executed multiple times under the same network configuration and helps regenerate the traces with different levels of detail. One can tune the network configuration and change the deployment scenario and topology. Furthermore, 1s run with a sampling frequency of 100 (or 10 ms time interval for generating logs) can provide 100 entries of throughput, thus producing adequate data for accurate throughput prediction. Many researchers have studied the performance of 5G in simulated platforms, such as ns-3 [9], [39], [40].

A. Experimental Setup

In this section, we explain the simulation setup for a DASH-based video client player running in a 5G UE, which is connected to a 5G mmwave network. The setup can be broadly divided into three parts - a) a 5G mmWave network, for which we have used the `ns3 – mmWave` module in [9], b) a video client player, which we have implemented in a Python setup, and c) an HTTP based DASH video streaming CDN server that hosts the video segments.

1) ns-3 Implementation of 5G Network: 5G mmWave communication has two levels of access points - a) low power 5G gNBs, which operate at the higher 28 GHz frequency range, and execute the data exchange between 5G NR devices, and b) 4G Long Term Evolution (LTE) eNBs, which operate in sub-7GHz range, and oversee the communication between the 5G gNBs. Accordingly, in our simulation scenario, we have deployed a single LTE eNB at the center while 10 gNBs are distributed uniformly within a 1km × 1Km square area. The detailed simulation parameters are provided in Table I. To deploy the UEs, we have first collected a real-world map of a one square kilometer area of Minneapolis, USA, using OpenStreet Map.[2] We have then converted it into a native `xml` format using Simulation of Urban MObility (SUMO) and imported it into our ns-3 simulation. SUMO captures the real-world traffic pattern, which helps in analyzing the effect of vehicular mobility on 5G user throughput. The average velocity of the UEs has been varied from the pedestrian speed of 5km/h to that of high-speed vehicles of 65km/h. A snapshot of the SUMO generated simulation map is shown in Fig. 2.

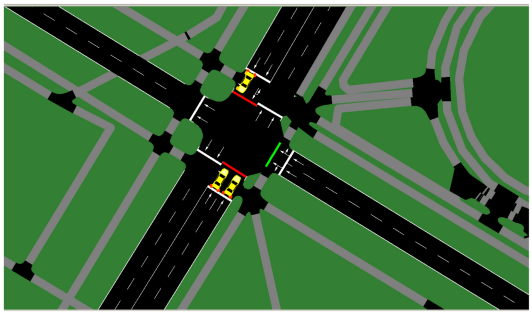


Fig. 2. Snapshot of a SUMO simulation setup of Minneapolis, USA; taken from OpenStreetMap.

To capture the variability in network conditions, we have identified three parameters - 1) the number of UEs, 2) the UE speed, and 3) the number of buildings deployed inside the simulation area. The number of UEs is used to vary the network load. Variation in the UE speed models the frequency dispersion of the wireless channel. Changes in the number of buildings in a given area cause the multipath fading to vary. We have generated 27 scenarios considering different values of these parameters as shown in Table II. We have run several simulation instances or drops of each of these scenarios to ensure a faithful averaging over a wide range of variations in network conditions and wireless channel behavior.

Each UE in our simulation has an ongoing video streaming application session. The throughput data of the mobile devices have been collected for these sessions only. Now throughput varies with the network quality variations. Correspondingly, the RRC CONNECTED state dwell time of the devices and, hence,

[2]https://www.openstreetmap.org

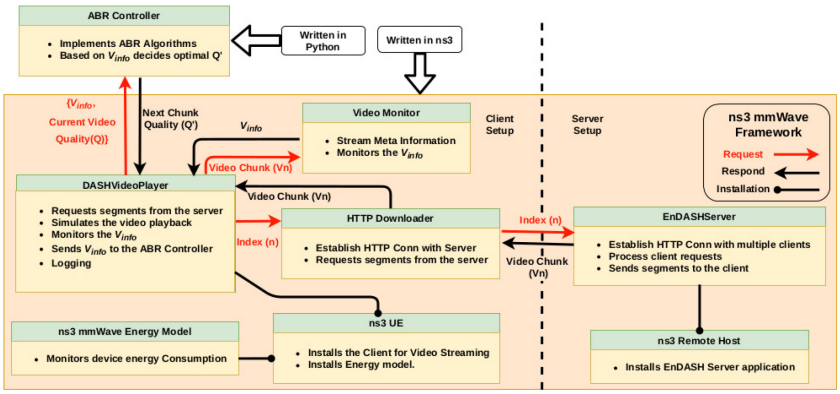


Fig. 3. Client Server setup for video streaming simulation.

TABLE II
SIMULATION SCENARIOS

Number of UE	Speed (km/h)	Buildings Deployed
1	5	0
10	30	10
20	65	20

their energy consumption also changes, as shown in Fig. 4. To measure the energy consumed by the 5G NR devices, we have implemented an energy module in ns-3, which has been integrated with the `ns3 – mmWave` module. The implementation details of this energy module are available in [41]. During an ongoing video session, the data recorded at each UE includes - (a) network-related parameters, such as the network throughput, Received Signal Strength Indicator (RSSI), Reference Signal Received Quality (RSRQ). Reference Signal Received Power (RSRP), Modulation Coding Scheme (MCS), number of handovers, and data state (CONNECTED or IDLE), and (b) UE parameters, such as distance from the gNBs, device speed, energy consumption per bit.

2) Video Streaming Setup: To interpret how throughput fluctuation affects video streaming in 5G mmWave communication, we have implemented a DASH *server-client* system operating in the downlink. The video streaming setup is shown in Fig. 3. The client module, which executes the majority of the functions in the setup, has been designed partially in both `ns3 – mmWave` and using a Python setup. The server, on the other hand, developed in `ns3 – mmWave`, implements an HTTP server to emulate the functions of a CDN server.

The details of the client-server setup follow.

Video Streaming Client: The `DashVideoPlayer` in Fig. 3 is the `ns3 – mmWave`-based application that executes the client functionalities as explained next.

1) The client first initiates a streaming session by establishing an HTTP connection with the video server, using the **HTTP downloader** sub-module, which has been implemented in `ns3 – mmWave`.
2) The client simulates the video playback by monitoring the playback buffer status and the throughput. This is done by the **Video Monitor** sub-module, which monitors and stores video session-related information, such as playback buffer occupancy, playback buffer length, rebuffer time, last chunk quality, and average throughput, etc. in the metadata object V_{info}.
3) The DASH player sends the V_{info} to the **ABR controller**. The controller runs an ABR algorithm, such as Pensieve or BOLA, which determines the optimum quality for the next video chunk and sends it back to the DASH video player. The ABR Controller module is not implemented in ns-3. It is a separate Python server hosted on the same local machine where the ns-3 simulation runs.
4) The client maps the next chunk download quality to the corresponding bitrate and requests the streaming server for the video segment through the **HTTP Downloader** module. It stops sending requests to the server when the streaming session is over.

Video Streaming Server: Videos are "Dashified" and stored in a streaming server. The EnDASH Server manages connections with clients and processes client requests. Upon receiving the client's request, the server responds with the requested video chunk in the requested quality. We first explain the video dashification and then the EnDASH server implementation.

1) *Dashifying videos:* In this work, we have used 56 DASHified videos [8] and streamed them over the simulation setup. Dashifying implies the conversion of the video to a form suitable for playing it using a DASH player. Essentially, the videos have been split into segments with fixed duration, and then each segment is stored at multiple qualities. The length of each video chunk is kept as 8sec, while the chunk size, in terms of bytes, depends on the encoding bitrate value. Each segment has been stored in the CDN server in the following encoding - [144p, 270p, 360p, 480p, 570p, 720p]. We have used the open-source 'ffmpeg' non-interactive video manipulation tool to dashify the videos. All the videos are stored at the server and have been streamed over the different simulation scenarios outlined in Table II.
2) *Implementing the server:* To implement the server, we have developed `EnDASHServer` – an ns-3 based application that is installed over a remote host. In each incoming request, the server responds with the requested segment based on the information it receives about the segment length or the number of bytes it has to send.

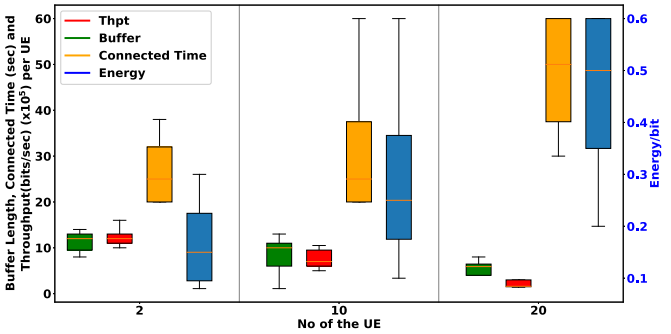


Fig. 4. Variation of Buffer Length, Connected state dwell time, Throughput, Energy/bit, with the number of UEs; ABR streaming algorithm - BOLA.

B. Results and Observations

Using the setup in Section III-A, we have made a detailed analysis of the effect of 5G network quality fluctuations on video streaming using the log generated by the `DashVideoPlayer`.

Fig. 4 shows the variation in the network throughput, the available playback buffer length, time spent in the connected state, and the average energy consumption per bit of an individual user as a function of the total number of UEs in the network. The average speed of each UE is 30km/h, and the number of buildings deployed in the one square kilometer simulation area is 10. We have simulated three scenarios with two, ten, and twenty users, respectively, keeping the speed, the number of buildings, and network parameters unchanged. We have run five simulation drops of each of these scenarios, each with a duration of 100 seconds. Averaging over the simulation drops yields Fig. 4. Each whisker plot shows the maximum and the minimum on the two extreme sides. The upper and lower boundaries of the box represent the 75^{th} and 25^{th} percentile points. The line inside the box represents the median, i.e. the 50^{th} percentile point.[3] It is observed that the network throughput and the buffer length decrease with the increase in the number of UEs. At the same time, there is an increase in the energy consumption of the individual UEs.

An in-depth study of the plots reveals that the per-user throughput is high with fewer users in the network, which allows the application layer to download a higher number of video segments at a higher rate. This, in turn, increases the playback buffer length. **A higher rate of download implies a lower CONNECTED state dwell time**, which manifests in the lower energy consumption as seen from Fig. 4. It may be inferred that an increase in the number of users increases the network load, thereby dividing the network resources among more users. This causes lower throughput, shorter playback buffer length, and higher energy consumption per user.

Of all the simulation drops, we have shown in Fig. 5 the snapshot of the worst-case scenario with 20 UEs, each moving with an average speed of 65km/h, in a scenario with 20 buildings deployed. This snapshot has been generated with BOLA [14] as the ABR streaming algorithm. It shows the variation of energy consumption per bit, video segment download throughput, video chunk bitrate, playback buffer length, received signal strength (in blue), and the handover events (shown in red dots) of a typical UE. In line with [8], the trace in Fig. 5 shows that the volume of video segments downloaded is not always commensurate with the connection quality, which in this work has been quantified by the RSSI. As can be seen in Fig. 5(a), in the event where there are frequent handovers between 11 seconds and 14 seconds, the RSSI shows high variability, i.e., the channel quality worsens. This is also observed in Fig. 5(b). Nevertheless, Fig. 5(c) shows that even at the time of handover, the UE commits to packet download at high bit rates. This attempt to download video segments at a higher bitrate, even in fluctuating throughput, causes the device to download the chunk for a longer time. Hence, it stays in the high-power RRC CONNECTED state for longer. Consequently, the energy consumption increases as is reflected in Fig. 5(a). On the other hand, even when the channel quality is good between 4 seconds and 8 seconds, the player downloads the video segments at a lower rate, as observed in Fig. 5(c).

We may, therefore, infer from Fig. 5 that some parameters other than the channel quality also drive the QoE during video streaming. The rate at which the video buffer is filled depends on a) the available bandwidth and b) the quality of the video segments downloaded. Once the buffer length exceeds a threshold, the download stops and restarts only when the buffer length goes below the threshold. So, if the buffer is full when signal quality improves, then the UE does not download any video packet. We can, therefore, summarize our takeaway from this pilot study below.

Takeaway: *The current protocol of video download always attributes a higher weight to the playback-buffer length than the user's instantaneous received signal strength or throughput, ensuing a significantly high energy consumption.*

This inference from the pilot study provides the design criteria for developing an energy-efficient video streaming mechanism for 5G systems, which is discussed next.

IV. EnDASH-5G System

EnDASH-5G acts as a wrapper over DASH, enabling energy-efficient video streaming over HTTP in 5G mmWave networks. Based on the observations from the pilot study, EnDASH-5G engages in the opportune fetching of video chunks during good channel conditions, which is facilitated by the prediction of the channel quality over a finite future time window. This reduces the CONNECTED state dwell time and eventually economizes the energy expenditure. To accommodate the higher volume of downloads, the core idea of EnDASH-5G is to dynamically tune the playback buffer length to the current network and wireless channel conditions, unlike existing players that use a static buffer. Once the playback buffer length is adjusted, EnDASH-5G, an ABR video streaming algorithm, decides the video chunk bitrates as a function of this decided playback buffer length and the current channel condition.

EnDASH-5G operates in a time-slotted manner, i.e., it divides the timeline into time-slots of length '$\mathcal{W}$' [8].

[3] All whisker plots in the paper follow this same convention here.

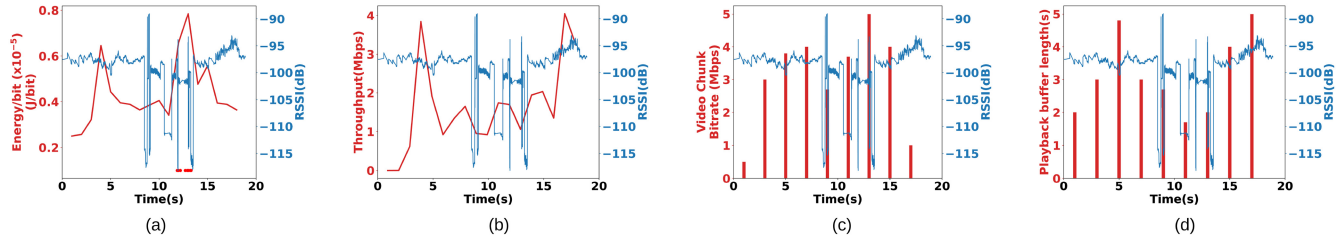


Fig. 5. Pilot Study: Fig.(a) Variation of Energy per bit with time, Fig.(b) Variation of Throughput with time, Fig.(c) Variation of Bitrate with time, Fig.(d) Variation of Playback buffer length with time. The variations of all the metrics are shown against the temporal variations of RSSI.

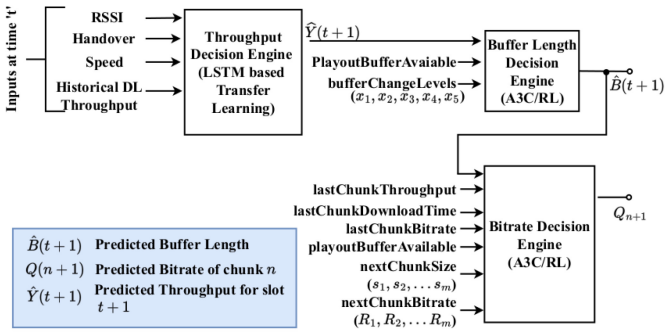


Fig. 6. Composite Representation of the EnDASH-5G model; a cascaded model where the predicted throughput acts as an input network state to the Actor-Critic RL based decision engine.

Its dynamics move through a cascade of three engines: (i) throughput decision, (ii) buffer length decision, and (iii) bitrate decision, as shown in Fig. 6.

1) At the beginning of any time slot 't', EnDASH-5G executes the following:
   a) In the *throughput decision engine*, it first *predicts the average UE throughput* in the current time slot, i.e., for the next '$t + \mathcal{W}$' seconds [8] by exploiting the network parameter and the downlink throughput information of the previous '$t-H$' seconds.
   b) Next, in the *buffer length decision engine*, it uses this predicted throughput to *estimate the optimal average playback-buffer length* for the duration of the current slot such that the energy consumption is minimized, using the Actor-Critic (A3C) deep RL algorithm.
2) Finally, within a time slot, just before downloading each video chunk, EnDASH-5G enters the *bitrate decision engine* where it uses the predicted optimal buffer length as an input to another A3C deep RL engine to predict the optimal download bitrate of the video chunk.

The detailed functioning of these decision engines follows.

A. The Throughput Prediction Engine

The first step towards the implementation of EnDASH-5G is the design of an efficient throughput prediction engine for 5G cellular networks. As discussed in [8], [29], the cellular network throughput varies with a wide range of factors like – radio signal strength at the user devices, the technology of the associated and neighboring BSs, UE speed, the handovers between BSs of same or different technologies, the number of users being served by a BS as well as the radio resource allocation algorithms. It is, thus, evident that there exists a non-trivial relation between the cellular network throughput perceived by a running application and the corresponding network parameters. So, traditional averaging mechanisms such as arithmetic, exponential, or harmonic moving averages used by most ABR streaming algorithms do not perform well for predicting cellular network throughput in practice [29].

1) Existing Works on Throughput Prediction and Their Limitations in the Context of 5G Networks: Several existing works in the literature [8], [28], [29], [31], [32], [42] have explored machine learning algorithms for predicting network throughput perceived by a running application. Works like [8], [29], [31], [32] train an Random Forest (RF)-based supervised learning algorithm for predicting 4G network throughput using real-network throughput traces. Deep Learning models [28] perform significantly better than the RF-based models; however, these models need a huge training dataset to train the model properly.

As mentioned earlier, the commercial deployment of 5G is still in a nascent stage. Besides, most existing 5G throughput traces do not contain information associated with the variability in the network load, i.e., the number of active users. Hence, in this work, we have used synthetic 5G network throughput traces generated as in Section III-A. Nevertheless, there is a significant departure in the statistical characterization of simulated traces from real network traces. To accommodate for this, i.e., to imbibe the salient features of real-network throughput prediction in our model, we propose to use *transfer learning* [10]. In transfer learning, the knowledge gained from previously learned tasks is applied to a target domain of related tasks. Interestingly, the publicly available 5G datasets [3], [11], [12] provide enough throughput traces that can be used as a source domain for the transfer learning and can be applied to a target 5G domain with a simulated dataset. To the best of our knowledge, this is the first work that uses transfer learning to predict 5G throughput using simulated traces.

2) Transfer Learning: In transfer learning, given a source domain D_S with a significant amount of data and a target domain D_T with a limited amount of data, a deep neural network model M_S is first trained for learning a task ζ_S in D_S. Subsequently, the model M_S is retrained with the dataset of D_T for learning a task ζ_T in D_T. The knowledge, i.e., the features, weights, etc., gained while training M_S in D_S enhances the learning of the target predictive function

Algorithm 1 Throughput Prediction Algorithm

1: **procedure** TRAIN_LSTM($\mathcal{D}_S$)
2: **for** $i \in \tau_S$ **do**
3: $\{\psi^i_{X,S}, \psi^i_{Y,S}\}$ = create_samples($\mathcal{D}_S, H, \mathcal{W}$).
4: Find $\theta_S = \arg\min_{\theta_S}$ loss($\hat{Y_S}^{(i+\mathcal{W})}, \overline{Y_S}^{(i+\mathcal{W})}$).
return θ_S
5: $\mathcal{D} = \text{preprocess}(\mathcal{D}_T)$.
6: $\mathcal{D}_{norm} = \frac{\mathcal{D}-min(\mathcal{D})}{max(\mathcal{D})-min(\mathcal{D})}$.
7: **procedure** FINE_TUNE_LSTM($M_S, \theta_S, \mathcal{D}_{norm}$)
8: **for** $i \in \tau_T$ **do**
9: $\{\psi^i_{X,T}, \psi^i_{Y,T}\}$ = create_samples ($\mathcal{D}_{norm}$, H, $\mathcal{W}$).
10: Find $\Delta\theta = \arg\min_{\Delta\theta}$ loss($\hat{Y_T}^{(i+\mathcal{W})}, \overline{Y_T}^{(i+\mathcal{W})}$).
return $\Delta\theta$.
11: Generate final weights $\theta_T = \theta_S + \Delta\theta$.

$\mathcal{F}_T(.)$. Here, $D_S \neq D_T$, and ζ_S may or may not be the same as ζ_T. Transfer learning effectuates a shorter training time while improving the accuracy even with a smaller dataset size in the target domain. In our work, the source domain D_S corresponds to real 5G network throughput data, while the target domain D_T corresponds to the simulated 5G network. The source and target tasks ζ_S and ζ_T both correspond to the prediction of throughput. So, we have $D_S \neq D_T$ but $\zeta_S = \zeta_T$.

3) Model Description and Transfer Learning: The proposed throughput prediction engine, outlined in Algorithm 1, is designed to predict the network throughput $\mathcal{W}$ seconds into the future based on the information available in the network and user-related parameters as well as that of the downlink throughput of the previous *H* seconds. At the heart of this prediction engine is a recurrent neural network with **two layers** (128 units each) of Long Short Term Memory (LSTM) cells followed by a **fully connected layer** of 1 node. We have also added a dropout of 0.2 after every LSTM layer as a form of regularization. LSTM is often chosen as the preferred model for data training when the corresponding dataset is quite large. In this work, we have a significantly extensive real 5G network as well as simulated 5G network throughput traces available. Moreover, throughput data is in time-series format [28]. Hence, we have used LSTM. The details follow.

a) The source domain: The first step is training the LSTM model M_S using the source domain dataset in order to learn the weights θ_S. This dataset is a collection of data points $\{\mathbf{X}^i_S, Y^i_S\}, \forall i \in \tau_S$, where τ_S is the observation time. The set $\mathbf{X}^i_S = \{X^i_{(S,1)}, X^i_{(S,2)}, \ldots, X^i_{(S,n)}\}$ is the set of '*n*' Radio Channel and Location Metrics while Y^i_S is the downlink throughput, at the time step *i*.

To enable the training and testing using the LSTM cells, the collected data in time series format is mapped to two data matrices $(\psi^i_{X,S}, \psi^i_{Y,S}), \forall i$ (Line 3). While $\psi^i_{X,S}$ represents the matrix of network parameter and location-related features, $\psi^i_{Y,S}$ represents the vector of downlink throughput from $i-H$ seconds to $i-1$ seconds, i.e.,

$$\psi^i_{X,S} = \begin{pmatrix} X^{(i-H)}_{S,1} & X^{(i-H)}_{S,2} & \cdots & X^{(i-H)}_{S,n} \\ X^{(i-H-1)}_{S,1} & X^{(i-H-1)}_{S,2} & \cdots & X^{(i-H-1)}_{S,n} \\ \vdots & \vdots & \vdots & \vdots \\ X^{(i-1)}_{S,1} & X^{(i-1)}_{S,2} & \cdots & X^{(i-1)}_{S,n} \end{pmatrix}. \quad (1)$$

$$\psi^i_{Y,S} = \begin{bmatrix} Y^{i-H}_S & Y^{i-H-1}_S & \cdots & Y^{i-1}_S \end{bmatrix}. \quad (2)$$

The LSTM training procedure learns the weights θ_S by minimizing the loss between the predicted and the actual average throughput, $\hat{Y}^{(i+\mathcal{W})}$ and $\overline{Y}^{(i+\mathcal{W})}$ (Line 4), respectively, i.e.,

$$\theta_{\mathbf{S}} = \arg\min_{\theta_S} loss\Big(\hat{Y}^{(i+\mathcal{W})}_S, \overline{Y}^{(i+\mathcal{W})}_S\Big), \forall i \in \tau_{\mathcal{S}}. \quad (3)$$

Here, the predicted throughput $\hat{Y}^{(i+\mathcal{W})}$ is obtained as,

$$\hat{Y}^{(i+\mathcal{W})}_S = \mathcal{F}_{\mathcal{S}}\Big(\Big(\psi^i_{X,S}, \psi^i_{Y,S}\Big), \overline{Y}^{(i+\mathcal{W})}_S, \theta_{\mathbf{S}}\Big), \quad (4)$$

and the actual average throughput $\overline{Y}^{(i+\mathcal{W})}_S$ is given by:

$$\overline{Y}^{(i+\mathcal{W})}_S = \frac{1}{\mathcal{W}} \sum_{j=i}^{i+\mathcal{W}} Y^{(j)}_S. \quad (5)$$

Here, learning the predictive function $\mathcal{F}_{\mathcal{S}}$ is equivalent to learning the weights θ_S for the model M_S. Once the source domain trained LSTM model M_S is available, the next step is the retraining of the layers of the neural network for the target domain D_T.

b) The target domain: The target domain dataset $\mathcal{D}_T$ is first preprocessed to extract the features (Section V-A3) which are common with the source domain dataset $\mathcal{D}_s$ (Line 5). After the feature selection, the target domain dataset $\mathcal{D}_T$ is normalized using Max-Min normalization techniques (Line 6). The normalized dataset $\mathcal{D}_{norm}$ is a collection of data points $\{\mathbf{X}^i_T, Y^i_T\}$, over an observation time τ_T. $\mathbf{X}^i_T$ corresponds to only those network parameter and location-related features as $\mathbf{X}_S$. Matrices $(\psi^i_{X,T}, \psi^i_{Y,T})$ are then created in the same way as Equation (1) and Equation (2) to override the time-series format (Line 9). Finally, we fine-tune $\mathcal{F}_{\mathcal{S}}$ using the dataset $\mathcal{D}_T$ in the target domain such that the error in predicting the throughput in the target domain is minimized (Line 10), i.e.,

$$\mathbf{\Delta}\theta^* = \arg\min_{\Delta\theta} \text{loss}\Big(\hat{Y_T}^{(i+\mathcal{W})}, \overline{Y_T}^{(i+\mathcal{W})}\Big), \forall i \in \tau_T, \quad (6)$$

where $\hat{Y}^{(i+\mathcal{W})}_T$ is the predicted average throughput over a finite future time window of length T in the target domain.

$$\hat{Y}^{(i+\mathcal{W})}_T = \mathcal{F}_T\Big(\Big(\psi^{\mathbf{i}}_{\mathbf{X},\mathbf{T}}, \psi^i_{Y,\mathcal{W}}\Big), \overline{Y_T}^{(i+\mathcal{W})}, \theta_{\mathbf{S}} + \Delta\theta\Big), \quad (7)$$

and $\overline{Y_T}^{(i+\mathcal{W})}$ is the actual average throughput over $\mathcal{W}$ in the target domain.

Fine Tuning: In this work, the new weights θ_T have been obtained from θ_S as $\theta_T = \theta_S + \Delta\theta$ by retraining the source domain LSTM model using the following inductive bias transfer techniques - (i) weight transfer from source

domain [43], (ii) partial retraining of layers [44], (iii) retraining all layers [45].

Only the last layer of the LSTM model M_S has been retrained for partial retraining. Further, details are provided in Section VI.

B. The Buffer Length Decision Engine

In addition to the throughput prediction engine, the Buffer Length Decision engine also runs at the beginning of each time slot. Once the network throughput is predicted, it is used to predict the optimal playback buffer length so that the energy consumption is minimized. However, the relationship between the predicted throughput and the buffer length is not straightforward and is not analytically tractable. So, we employ an A3C RL based deep neural network to determine the optimal buffer length.

The components of the RL algorithm corresponding to the buffer decision engine are as follows:

(i) Environment E - video player.

(ii) Input state at timeslot 't' $\Omega_t = (\hat{Y}_t,\ B_t^{av},\ \mathcal{X})$, where $\hat{Y}_t$ is the average predicted cellular network throughput, B_t^{av} is the current playback buffer capacity available in seconds, and $\mathcal{X}$ is the set of possible changes in buffer length. As the current buffer length is B_t^{av}, the next buffer length can be predicted to be $\hat{B}_{t+\mathcal{W}} = B_t^{av} + x$ where $x \in \mathcal{X} := \{-2, -1, 0, +1, +2\}$. $x = 1$ implies an increase in buffer length by a single chunk and each chunk of eight seconds.

(iii) Action A_t - Decisions on the increase or decrease of buffer length at timeslot *t*.

(iv) Reward - The reward function is designed in such a manner that the QoE is maximized, and the energy consumption is minimized as shown in (8).

$$\Xi_{\text{bufflen}} = w_1.QoE + w_2.\left(\left|E_{\text{EnDASH-5G}_t} - E_{\text{old}_t}\right|\right) \quad (8)$$

The first term in (8) is the **QoE** metric [15]:

$$\text{QoE} = \frac{1}{N}\sum_{i=1}^{N} q(R_i) - \frac{1}{N}\mu\sum_{i=1}^{N}\delta_i - \frac{1}{N-1}\sum_{i=1}^{N-1}|q(R_{i+1}) - q(R_i)| \quad (9)$$

The **QoE** metric is defined for a video with *N* chunks. R_i is the bitrate of chunk '*i*' in bits per second, and $q(R_i)$ maps the bitrate to a quantity representing the quality perceived by the user. We have taken $q(R_i) = R_i$ [13]. δ_i is the rebuffering time in seconds incurred while downloading the chunk '*i*' at rate R_i. μ is the degree of penalty associated with δ_i. We have taken $\mu = 4.3$ [13]. The last term represents the playback smoothness. The QoE increases when the streamed video is smooth, i.e., there is less variation in the bitrate of subsequent chunks. Smoothness is quantified by its inverse, which is the throughput variation having a unit of bits per second. QoE, thus, increases with bitrate and reduces with rebuffering time and throughput variability.

The second term in (8) gives energy savings with respect to a baseline ABR algorithm. In this work, we have chosen BOLA [14] as the baseline since its energy consumption is the lowest among existing algorithms (excluding EnDASH-5G, as seen in Section VI-A). E_{old_t} and $E_{\text{EnDASH-5G}_t}$ represent the energy consumption of BOLA and of EnDASH-5G at time slot '*t*', respectively. When the energy consumed by EnDASH-5G is minimized, the difference in the energy consumed by EnDASH-5G and BOLA increases. Thus, the second term in (8) also increases. Consequently, overall we attempt to maximize the reward, which is a weighted combination of the QoE and the energy difference. We have introduced the two weights for generalization. For the current work, we have taken $w_1 = w_2 = 1$. These may be controlled to handle the tradeoff according to the user's requirement.

We choose this energy minimization method in order to compensate for the lack of ground truth on the energy consumption of the ABR streaming algorithms. By using this method, pre-trained models can be loaded into smartphones. The energy consumed is obtained as follows: the RRC states are first identified from the download packet capture of a video trace. Next, the dwell time in each RRC state is multiplied by the corresponding power consumption (obtained from the RRC state machine) to get the energy quantities.

C. The Bitrate Decision Engine

The third component of EnDASH-5G is the bitrate decision engine which runs within a time slot and is used to predict the bitrate of the next video chunk immediately before it is downloaded. The buffer length and bitrate decision engines run at two different time scales- the buffer length decision engine at the beginning of a time and the bitrate decision engine within a time. So, time slot indices for the variables related to the buffer length decision engine and the bitrate decision engine are denoted by '*t*' and '*n*', respectively. The bitrate decision engine is also an A3C-RL-based deep neural network whose components are as follows:

(i) Environment E – video player.

(ii) Input state before downloading the chunk 'n',

$\Omega_n = (\hat{\mathcal{B}}_t,\ y_c(n-1),\ d_{n-1},\ l_{n-1},\ B_n,\ \beta_n,\ r_n)$, where $\hat{\mathcal{B}}_t$ is the predicted playback buffer length for current slot '*t*', $\tau_c(n-1)$ is the throughput of the last chunk, d_{n-1} is the time taken to download last chunk, l_{n-1} is the bitrate of the last chunk, B_n is the available playback buffer length, $\beta_n \in (b_1, b_2, \ldots, b_m)$ is the possible size of next video chunk (*m* available sizes), $r_{n+1} \in (R_1, R_2, \ldots, R_6)$ is the possible bitrate levels for next video chunk.

(iii) Action A_n – Optimal bitrate decision for the next chunk.

(iv) Reward – QoE score obtained from Equation (9).

V. Implementation and Training of ML Models

In this section, we explain the implementation details of the three decision engines of EnDASH-5G.

A. Dataset Used for Throughput Prediction

1) Source Domain Datasets: The source domain 5G datasets include three datasets as explained next-

- *Lumos-5G dataset* [11] - The Lumos-5G dataset from [11] contains a measurement study of commercial 5G mmWave services in Minneapolis, USA, and has recorded the downlink throughput as perceived by applications running on UE. It contains information on the UE's geographical coordinates, moving speed, compass directions, downlink throughput (reported using *iperf 3.7*), radio type (4G/5G), signal strength (LTE – RSRP, RSRQ, RSSI & 5G – SSRSRP, SSRSRQ, SSRSSI), handover events, etc. With Verizon's 5G Ultra Wideband network, the measurement study has been conducted over a period of 6 months, using 4 Samsung Galaxy S10 5G phones.
- *Irish dataset* - In [12], the authors have collected 5G trace datasets from a major **Irish** mobile operator. They have considered two application patterns (video streaming and file download). The dataset consists of (1) channel-related metrics such as signal strength (RSRP, RSRQ, SNR, CQI), neighboring cell RSRP, RSRQ, (2) context-related metrics (e.g., GPS of the device, device velocity), (3) cell-related metrics such as eNBs ID, and (4) throughput information(both uplink and downlink). The dataset consists of 83 traces, with a total duration of 3142 minutes, using a Samsung S10 5G Android device.
- *MN-Wild* dataset - In [3], authors have carried out an in-depth measurement study of the performance, power consumption, and application QoE of commercial 5G networks in the wild. They have examined different 5G carriers (Verizon and T-Mobile), deployment schemes (Non-Standalone, NSA vs. Standalone, SA), radio bands (mmWave and sub-6-GHz), Radio Resource Control state transitions for power modeling, mobility patterns (stationary, walking, driving), client devices (such as Samsung Galaxy S20 Ultra 5G (S20U) and Samsung Galaxy S10 5G (S10)), and upper-layer applications (file download, video streaming, and Web browsing). The measurement studies were conducted in two U.S. cities (Minneapolis, MN, and Ann Arbor, MI), where both carriers have deployed 5G services. Of the datasets belonging to the two cities, we have found that the dataset corresponding to Minneapolis city (MN) contains throughput and other important UE information, such as the speed of the UE, which is not available for the Ann Arbor (MI) dataset. Therefore, in this paper, we have selected the Minneapolis dataset and referred to it as MN-Wild.

2) Target Domain Datasets: As mentioned earlier, the **target domain dataset** is the **simulated 5G network**. The corresponding dataset is the synthetic throughput data collected from the `ns3 – mmWave` simulation outlined in Section III. The target domain dataset has 23778 datapoints, captured at the granularity of 0.44 seconds on average.[4]

3) Identifying the Feature Space: As identified in [8] the cellular network throughput depends on several parameters - the UE parameters such as user speed and distance from BS, as well as the network related parameters such as RSSI, RSRP, RSRQ, number of handovers, technology of associated and neighbouring BSs, MCS, data state (CONNECTED or IDLE). In this work, we have first identified a standard set of features that are common to both the source and the target domain datasets. These are user speed, distance from BS, RSSI, RSRP, RSRQ, number of handovers, and data state.

TABLE III
PREDICTION SCORE WITH NS-3 SIMULATED 5G TRACE DATASETS

Technique	Simulated 5G	
	R^2 score	Pearson Corr. Coeff.
Weight Transfer	82.17%	93.53%
Partial Layers	89.32%	95.39%
All Layers	91.27%	95.91%

We train the LSTM model of the throughput decision engine with the public-5G datasets [3], [11], [12] with a train-validation split of 80%-20%. The trained model is then retrained by means of Transfer Learning using the ns-3 simulated 5G dataset. For the retraining, we have used a train-validation-test split of $72\%-8\%-20\%$. This tunes the hyperparameters of the LSTM model. We have provided the prediction R^2 score on the test set in our evaluation (Table III). The trained model is then deployed in the ns-3 simulation setup for testing with new throughput data generated in runtime.

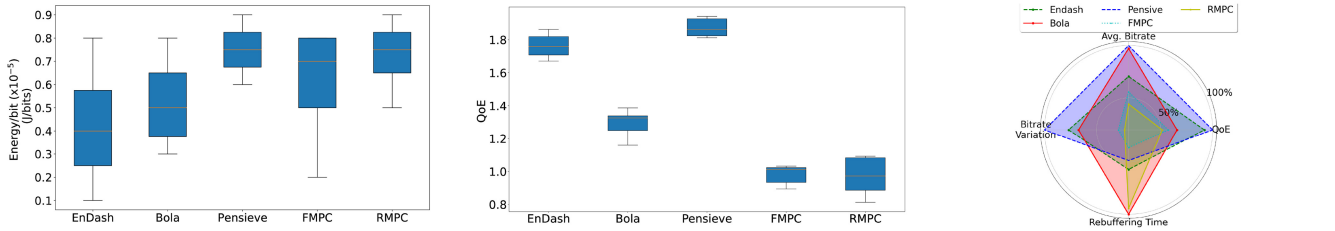
B. Emulation Platform for RL Training

In the training phase, the RL agent of the buffer and the bitrate decision engines of EnDASH-5G should ideally be trained using real video downloads at actual video streaming clients. Emulating a standard video streaming session demands that for each training data point, all the video chunks are downloaded over a Web browser before training can start. This unnecessarily lengthens the training time. As downloads over cellular networks can be pretty slow due to network fluctuations, the training time can become longer. Hence, to save training time, we train EnDASH-5G and its competing ABR algorithms using a python-based video emulation setup that closely mimics a real video client application as in [8], [13], [23]. Our source code is publicly available in GitHub.[5]

The emulator maintains its own representation of a real client's playback buffer compatible with the Mahimahi [46] network emulation tool. At the beginning of a time slot, the emulator first predicts the average throughput and then the playback buffer length for the slot. When a chunk is to be downloaded within the slot, the emulator first assigns a download time to the chunk based on - a) its bitrate and b) the internal network throughput derived from previous chunks. It then depletes the playback buffer by the current chunk's download time to emulate the playback buffer drainage during an ongoing chunk download. Finally, it adds the playback duration of the downloaded chunk to the playback buffer. The emulator sleeps temporarily once the playback buffer is full. Rebuffering occurs if the playback buffer is completely drained before the next chunk download. The emulator keeps track of the rebuffering event and the rebuffering time. At the end of each slot and chunk download, the emulator prepares the state

[4]In ns-3, a datapoint is recorded at each event trigger. Hence, the granularity of the datapoints is not constant.

[5]https://github.com/arghasen10/endash/tree/main/rl_train (Accessed: February 26, 2023)



(a) Energy Consumption (b) QoE score (equation 9) (c) Scaled Spider chart across QoE components

Fig. 7. Performance comparison of EnDASH-5G with baseline ABR streaming algorithms, BOLA [14], Pensieve [13], FMPC [15], RMPC [15]. The 100% in Fig. 7(c) corresponds to the maximum value of the respective parameter.

Ω_t and Ω_n for the RL training algorithm of the buffer and bitrate decision engines, respectively.

C. The RL Training Algorithm

Both the buffer length and bitrate decision engines of EnDASH-5G are trained using the state-of-the-art actor-critic method, which involves training two neural networks [13]: an actor-network and a critic network. A3C is a policy gradient RL algorithm that estimates the gradient of the total reward from the trajectories of the execution followed by a policy. The action is chosen based on the policy by the actor network, while the critic network estimates the advantage of selecting the action by returning a value function. Both networks update their weights in each time step. Each parameter of the state Ω of both the buffer length and bitrate decision engines are passed on to their respective actor and critic networks to enable learning of optimal buffer lengths and bitrates by the corresponding RL modules [13]. As in [13], we initiate multiple learning agents to accelerate the training. By default, there are 16 agents, each of which is designed to encounter a different set of input parameters, e.g., network traces. The learning agents continuously report their individual tuples of (state, action, reward) to a central model, aggregating them to generate a single model.

D. Implementation of the RL Modules

The *bitrate selection engine* of EnDASH-5G is adopted from the Pensieve algorithm [13], which also uses A3C to learn optimal bitrates. We have used a pre-trained model provided by Pensieve, albeit with different input states. The *buffer length decision engine* has been implemented in TensorFlow. The engine passes five previous buffer length values to a 1D convolution layer (CNN) with 128 filters, each of size 4 with stride 1. The remaining inputs, i.e., predicted link throughput and currently available buffer length, are passed onto another 1D-CNN having the same shape. Results obtained from these layers are combined into a hidden layer having 128 neurons and use the softmax function. The critic network uses the same inputs with the same neural network, but its final output is a linear neuron. During the algorithm's training, we set the discount factor to 0.9, i.e., current actions can influence by 100 future steps. The learning rates for the actor and the critic networks are 0.0001 and 0.001, respectively. Over iterations, the entropy factor is gradually decreased from 1 to 0.1. An important point to note is that the buffer length and bitrate decision engine does not focus on hyperparameter tuning. We have used the hyperparameters from [13].

E. Simulation Environment for EnDASH-5G

We have implemented EnDASH-5G ABR Controller over the ns-3 simulation framework outlined in the description of the Video streaming server in Section III-A2. The models corresponding to the three decision engines, pre-trained on the training dataset, have been ported to the designed ABR Controller. In runtime, the RF decision engines predict the buffer length and video bitrate based on the playback buffer information, and the pre-trained throughput predictor predicts the network throughput based on the runtime throughput traces.

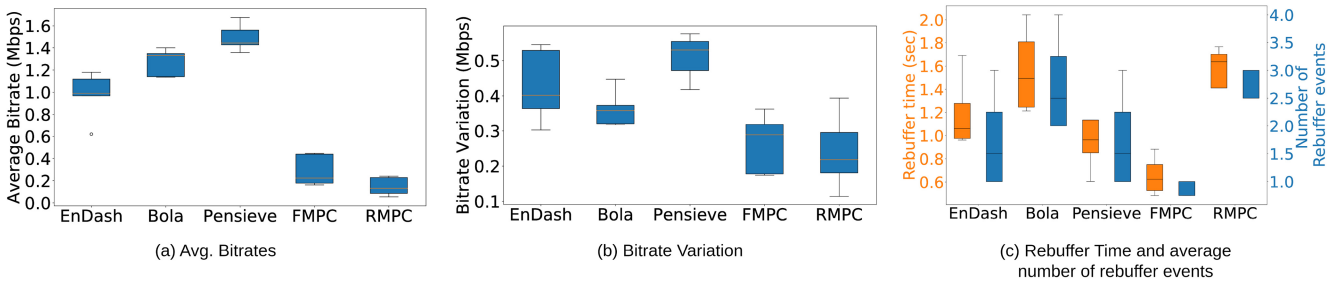
VI. Results

We first discuss the overall energy and QoE performance of EnDASH-5G and then dig into its individual modules.

A. EnDASH-5G Versus Baseline ABR Algorithms

In this work, we have considered four state-of-the-art ABR schemes as baselines and compared their performances with EnDash-5G. These algorithms include - (i) BOLA [14] - a **buffer based** algorithm that tunes the playback buffer length, (ii) FMPC & RMPC [15] - **control theoretic** schemes that maximize QoE based on upcoming chunk sizes and predicted future throughput, and (iii) Pensieve [13] - a **learning-based** approach that use deep neural network decision engine to decide the optimal bitrate.

Fig. 7 compares the average energy consumption and the average QoE score of EnDASH-5G with the aforementioned baseline ABR algorithms over the 27 different scenarios outlined in Table II. Pensieve [13] has been reported to generate optimal chunk bitrates and hence, delivers a high QoE to users. Fig. 7(a) and 7(b) show that while Pensieve indeed delivers the highest QoE among the different ABR streaming algorithms, it is quite expensive in terms of energy expenditure. EnDASH-5G reduces the mean energy consumption of Pensieve by nearly 30.5% in 5G-mmWave networks. This reduction, however, comes at the cost of QoE as the QoE of EnDASH-5G is 4.17% less than Pensieve's, as seen in Fig. 7(b). The bitrate decision algorithm of Pensieve is oblivious to the underlying channel condition. As a result, if it decides upon a higher bitrate during poor link conditions, the algorithm tries to download chunks at the high bitrate only, thereby getting stuck in the high power CONNECTED state of the RRC state machine (Fig. 1). In a similar situation, when



(a) Avg. Bitrates (b) Bitrate Variation (c) Rebuffer Time and average number of rebuffer events

Fig. 8. Comparison of different components of QoE score (average bitrate, bitrate variation, rebuffering time) of EnDASH-5G with baseline ABR streaming algorithms, BOLA [14], Pensieve [13], FMPC [15], RMPC [15].

EnDASH-5G anticipates the increased energy consumption, it immediately switches to a lower bitrate – commensurate with the link condition. Thus, it may be inferred that EnDASH-5G trades off the energy usage with the delivered QoE of ABR streaming algorithms. In the following subsection, we review the individual components of the QoE metric.

B. QoE Performance Analysis

The individual components of the QoE metric corresponding to Fig. 7(a) have been outlined in eq. (9), and explained in Section IV-B. These are plotted in Fig. 8, and their spider chart is shown in Fig. 7(c). Fig. 7(c) and Fig. 8 also summarize the QoE behavior across all the 27 scenarios of Table II. It is observed from Fig. 7(c) and Fig. 8(a) that the average bitrate of EnDASH-5G is lower than BOLA and Pensieve. This is because the buffer length in EnDASH-5G is tuned to the average predicted throughput. On the other hand, BOLA predicts bitrates using playback buffer length only, which manifests in the lower bitrate variation (Fig. 8(b)) but the high rebuffering time (Fig. 8(c)). Pensieve uses the previous chunk download rates to predict future bitrates. As this captures the instantaneous throughput variation, it reduces the rebuffering time but increases the bitrate variation.

In contrast, EnDASH-5G tunes the buffer length to the average throughput of the underlying channel. This may lead to lower bitrates if the average throughput predicted at the beginning of any time slot is low. In such cases, the system fails to exploit the instantaneous high throughput events within the timeslot. However, tuning buffer length to throughput yields lower bitrate variation than Pensieve. The number of rebuffering events and rebuffering time of EnDASH-5G is also comparable to Pensieve. This compensates for the reduced bitrates of EnDASH-5G, and in combination with its reduced energy consumption, it emerges as a viable candidate for delivering appreciable QoE to streaming videos.

C. Energy Performance Analysis

Our objective has been to improve energy consumption even when the network load increases. An increased network load implies fewer radio resources would be available per UE, which will decrease individual user throughput. To evaluate how EnDASH-5G performs under different network loads, we have plotted the energy consumption of EnDASH-5G and other ABR algorithms for a different number of UEs (2, 10, and 20 users) in the system. Further, for each network load and each UE speed, we have varied the number of buildings in the one square kilometer area as 0, 10, and 20. Averaging over the network conditions and the UE speed for a given network load, we have plotted the energy consumption of the different ABR streaming algorithms under different network loads in Fig. 9. It is observed that EnDASH-5G keeps the energy consumption significantly lower than other algorithms, even at a higher network load.

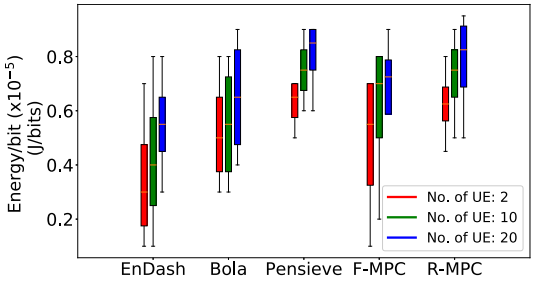


Fig. 9. Energy consumption of EnDASH-5G and other baseline algorithms under different network load.

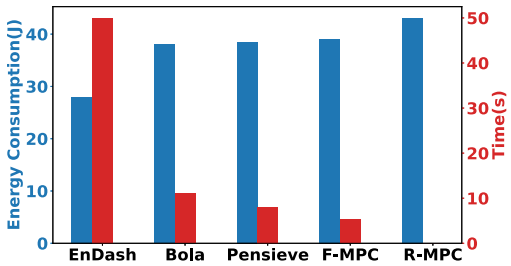


Fig. 10. Energy Consumption and Extra Playtime obtained w.r.t. RMPC, which has the highest energy consumption.

D. Gain From Energy Savings

This section analyzes the gain in video playback time achieved by EnDASH-5G. Fig. 10 shows the energy consumed by the ABR algorithms, averaged across all network loads, UE speed, and the number of buildings in the simulation area. It is seen that RMPC consumes the highest energy. Hence, we calculate the gain in playback time with respect to RMPC. Let $\mathcal{E}_a$ denote the energy consumed by an ABR algorithm, where $a \in \{\texttt{FMPC}, \texttt{RMPC}, \texttt{BOLA}, \text{and } \texttt{Pensieve}\}$. As playback time is inversely proportional to energy consumption, for any ABR algorithm a, the playback time is $T_a = \frac{k}{\mathcal{E}_a}$. We may assume that k is a constant specific to a UE handset-service provider-mobile network combination. Therefore, for algorithm a, extra playback time w.r.t. RMPC is, $T_a - T_{\texttt{RMPC}}$, where $T_{\texttt{RMPC}} = \frac{k}{\mathcal{E}_{\texttt{RMPC}}} = \frac{T_a \times \mathcal{E}_a}{\mathcal{E}_{\texttt{RMPC}}}$. Thus, the extra playback time

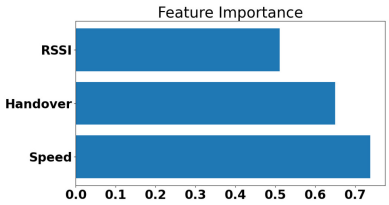


Fig. 11. Feature importance in SHAP (SHAPley Additive Explanations) values of the input parameters for deciding throughput; Number of Handovers, signal strength(RSSI), and speed of the User Equipment.

w.r.t. RMPC is given by

$$\text{Extra play time} = \frac{\mathcal{E}_{\texttt{RMPC}} - \mathcal{E}_a}{\mathcal{E}_{\texttt{RMPC}}} \times T_a \tag{10}$$

The extra time gained by each algorithm over RMPC when playing a 100-second video is also shown in Fig. 10. It is observed that while EnDASH-5G gains 42 seconds, BOLA and Pensieve gain 9 and 6 seconds of extra playback time than RMPC. The increase in playback time will also increase with the length of the video. Thus, one may infer that while streaming videos, EnDASH-5G can be used to improve the battery backup of 5G smartphones.

E. Performance of Throughput Prediction Engine

This section evaluates the performance of the proposed transfer learning-based throughput prediction algorithm for 5G mmWave networks. The throughput prediction engine is the heart of EnDASH-5G. The results in this section have been generated for a historical window size of $H = 30$s and a future window size of $\mathcal{W} = 30s$. To evaluate the goodness of prediction, we use R^2 score and **correlation coefficient** as evaluation metrics. We have explored all three induction bias transfer techniques as discussed in Section IV-A3.

The prediction accuracy for the simulated 5G mmWave dataset is tabulated in Table III. It is observed that the method of weight transfer performs reasonably well, with an R^2 score of 82.17%. The performance improves further by 7% by partial retraining of the layers. Complete retraining of all layers improves the R^2 score compared to partial retraining by 2%. In this work, we have partially retrained the LSTM layers for 5G throughput prediction in the final implementation of the proposed system.

Fig. 11 shows the feature importance, in SHAP (SHAPley Additive Explanations) values [47], of different input parameters, such as the UE speed, number of handovers, and RSSI when predicting the throughput. It may be noted that speed has the highest importance as a feature. Vehicular speed directly impacts the link condition of the channel, which in turn affects the throughput. This points to the absolute necessity of taking mobility into account for predicting throughput.

VII. Conclusion

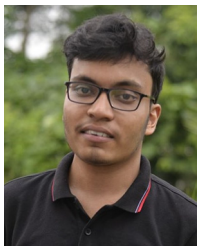
This paper proposed EnDASH-5G – an energy-aware ABR video streaming algorithm that minimizes energy consumption while not compromising the QoE of users under mobility in 5G mmWave networks. It exploits the high throughput regions in the user's trajectory for aggressive fetching of video chunks, thereby reducing energy consumption even in regions with limited network coverage. To achieve this, it intelligently tunes the playback buffer length to the average predicted network throughput and then resorts to optimal bitrate selection for video chunks. As a result, the buffer length no longer remains fixed. The network throughput is predicted by EnDASH-5G using a novel transfer learning approach. EnDASH-5G offers an additional 36 seconds of playback buffer time when playing a 100-second video, compared to the popular Pensieve algorithm, albeit with a marginal reduction in QoE. EnDASH-5G being tunable, can be designed to adapt to a specific requirement, such as energy or QoE maximization. This, in addition to a real-life implementation of EnDASH-5G and investigating the corresponding improvement in energy efficiency, will be our immediate future work.

References

[1] P. Marcos, L. Prehn, L. Leal, A. Dainotti, A. Feldmann, and M. Barcellos, "AS-path prepending: There is no rose without a thorn," in *Proc. ACM IMC*, 2020, pp. 506–520.
[2] X. Liu et al., "Learning to predict the mobility of users in mobile mmWave networks," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 124–131, Feb. 2020.
[3] A. Narayanan et al., "A variegated look at 5G in the wild: Performance, power, and QoE implications," in *Proc. ACM SIGCOMM*, 2021, pp. 610–625.
[4] D. Liu, J. Zhao, C. Yang, and L. Hanzo, "Accelerating deep reinforcement learning with the aid of partial model: Energy-efficient predictive video streaming," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3734–3748, Jun. 2021.
[5] C. Yue, S. Sen, B. Wang, Y. Qin, and F. Qian, "Energy considerations for ABR video streaming to smartphones: Measurements, models and insights," in *Proc. ACM MMSys*, 2020, pp. 153–165.
[6] X. Chen, T. Tan, G. Cao, and T. F. La Porta, "Context-aware and energy-aware video streaming on smartphones," *IEEE Trans. Mobile Comput.*, vol. 21, no. 3, pp. 862–877, Mar. 2022.
[7] T. Stockhammer, "Dynamic adaptive streaming over HTTP–: Standards and design principles," in *Proc. ACM MMSys*, 2011, pp. 133–144.
[8] A. Mondal et al., "EnDASH - A mobility adapted energy efficient ABR video streaming for cellular networks," in *Proc. IFIP Netw.*, 2020, pp. 127–135.
[9] M. Mezzavilla et al., "End-to-end simulation of 5G mmWave networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2237–2263, 3rd Quart., 2018.
[10] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.
[11] A. Narayanan et al., "Lumos5G: Mapping and predicting commercial mmWave 5G throughput," in *Proc. ACM IMC*, 2020, pp. 176–193.
[12] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: A 5G Dataset with channel and context metrics," in *Proc. ACM MMSys*, 2020, pp. 303–308.
[13] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. ACM SIGCOMM*, 2017, pp. 197–210.
[14] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for Online videos," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.
[15] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM*, 2015, pp. 325–338.
[16] "5G NR RRC procedure and its states." Nov. 2017. [Online]. Available: https://www.techplayon.com/5g-nr-rrc-procedure-states/
[17] S. Colonnese, F. Conti, G. Scarano, I. Rubin, and F. Cuomo, "Premium quality or guaranteed fluidity? client-transparent DASH-aware bandwidth allocation at the radio access network," *J. Commun. Netw.*, vol. 24, no. 1, pp. 59–67, Feb. 2022.
[18] I. Triki, R. El-Azouzi, and M. Haddad, "NEWCAST: Joint resource management and QoE-driven optimization for mobile video streaming," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 1054–1067, Jun. 2020.
[19] O. El Marai, T. Taleb, M. Menacer, and M. Koudil, "On improving video streaming efficiency, fairness, stability, and convergence time through client–server cooperation," *IEEE Trans. Broadcast.*, vol. 64, no. 1, pp. 11–25, Mar. 2018.

[20] 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," in *Proc. ACM SIGCOMM*, 2014, pp. 187–198.
[21] Z. Akhtar et al., "Oboe: Auto-tuning video ABR algorithms to network conditions," in *Proc. ACM SIGCOMM*, 2018, pp. 44–58.
[22] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.
[23] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "HotDASH: Hotspot aware adaptive video streaming using deep reinforcement learning," in *Proc. IEEE ICNP*, 2018, pp. 165–175.
[24] T. Xu and L. Ma, "Predictive prefetching for MPEG DASH over LTE networks," in *Proc. IEEE ICIP*, 2015, pp. 3432–3436.
[25] S. K. Mehr and D. Medhi, "QoE performance for DASH videos in a smart cache environment," in *Proc. IFIP/IEEE IM Symp.*, Apr. 2019, pp. 388–394.
[26] Y. Sani, D. Raca, J. J. Quinlan, and C. J. Sreenan, "SMASH: A supervised machine learning approach to adaptive video streaming over HTTP," in *Proc. IEEE QoMEX*, 2020, pp. 1–6.
[27] A. Bentaleb, C. Timmerer, A. C. Begen, and R. Zimmermann, "Bandwidth prediction in low-latency chunked streaming," in *Proc. ACM NOSSDAV*, 2019, pp. 7–13.
[28] D. Raca et al., "On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 11–17, Mar. 2020.
[29] D. Raca et al., "Empowering video players in cellular: Throughput prediction from radio network measurements," in *Proc. ACM MMSys*, 2019, pp. 201–212.
[30] 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. ACM MMSys*, 2018, pp. 460–465.
[31] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei, "Linkforecast: Cellular link bandwidth prediction in LTE networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 7, pp. 1582–1594, Jul. 2018.
[32] D. Raca et al., "Back to the future: Throughput prediction for cellular networks using radio KPIs," in *Proc. ACM HotWireless*, 2017, pp. 37–41.
[33] D. Raca et al., "Incorporating prediction into adaptive streaming algorithms: A QoE perspective," in *Proc. ACM NOSSDAV*, 2018, pp. 49–54.
[34] M. F. Tuysuz and M. E. Aydin, "QoE-based mobility-aware collaborative video streaming on the edge of 5G," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7115–7125, Nov. 2020.
[35] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, "GreenTube: Power optimization for mobile videostreaming via dynamic cache management," in *Proc. ACM Multimedia*, 2012, pp. 279–288.
[36] F. Araújo, D. Rosário, E. Cerqueira, and L. A. Villas, "A hybrid energy-aware video Bitrate adaptation algorithm for mobile networks," in *Proc. IEEE WONS*, 2019, pp. 146–153.
[37] K. Park and M. Kim, "EVSO: Environment-aware video streaming optimization of power consumption," in *Proc. IEEE INFOCOM*, 2019, pp. 973–981.
[38] Y. Yang and G. Cao, "Prefetch-based energy optimization on smartphones," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 693–706, Jan. 2018.
[39] C. R. Storck and F. Duarte-Figueiredo, "A performance analysis of adaptive streaming algorithms in 5G vehicular communications in urban scenarios," in *Proc. IEEE ISCC*, 2020, pp. 1–7.
[40] W. Shi et al., "QoE ready to respond: A QoE-aware MEC selection scheme for DASH-based adaptive video streaming to mobile users," in *Proc. ACM MM*, 2021, pp. 4016–4024.
[41] A. Sen, A. Mondal, B. Palit, J. Jay, K. Paul, and S. Chakraborty, "An ns3-based energy module of 5G NR user equipments for Millimeter wave networks," in *Proc. IEEE INFOCOM WKSHPS*, 2021, pp. 1–2.
[42] A. Samba, Y. Busnel, A. Blanc, P. Dooze, and G. Simon, "Instantaneous throughput prediction in cellular networks: Which information is needed?" in *Proc. IFIP/IEEE Symp. INSM*, 2017, pp. 624–627.
[43] L. Yang, L. Jing, J. Yu, and M. K. Ng, "Learning transferred weights from co-occurrence data for heterogeneous transfer learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 11, pp. 2187–2200, Nov. 2016.
[44] L. Xuhong, Y. Grandvalet, and F. Davoine, "Explicit inductive bias for transfer learning with convolutional networks," in *Proc. ICML*, 2018, pp. 2825–2834.
[45] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" 2014, *arXiv preprint:1411.1792*.
[46] R. Netravali et al., "Mahimahi: Accurate record-and-replay for HTTP," in *Proc. USENIX Conf. UATC*, 2015, pp. 417–429.
[47] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Info. Process. Syst.*, 2017, pp. 4765–4774.

Basabdatta Palit received the B.Tech. degree from the West Bengal University of Technology in 2009, the M.E. degree from Jadavpur University, India, in 2011, and the Ph.D. degree from the Indian Institute of Technology Kharagpur, India, in 2018. She is currently a Faculty Member with the Department of Electronics and Telecommunication Engineering, Indian Institute of Engineering, Science and Technology Shibpur, India. Her research interests include scheduling and resource allocation, vehicular communication, energy efficiency, and URLLC.



Argha Sen received the B.Tech. degree in electronics and communication engineering from the National Institute of Technology Durgapur, India, in 2020. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. His research interests include mobile computing, RF sensing, next-generation energy-efficient cellular networks, and ABR streaming.

Abhijit Mondal (Member, IEEE) received the B.Tech. degree from the Maulana Abul Kalam Azad University of Technology, Kolkata, India, in 2010, the M.Tech. degree from the Indian Institute of Technology Guwahati, India, in 2012, and the Ph.D. degree from the Indian Institute of Technology Kharagpur in 2021. He is currently working as a Senior Engineer with VDX.tv. Previously, he worked with Rovi (currently, TiVo) as a Software Engineer.

Ayan Zunaid received the B.Tech. and M.Tech. degrees in computer science and engineering from the Indian Institute of Technology Kharagpur, India, in 2021. He is currently working with Flipkart as a Software Development Engineer 1. His research interests include machine learning and big data analytics.

Jay Jayatheerthan received the B.Tech. degree from the P.S.G College of Technology, Coimbatore, India, and the master's degree in engineering management from The University of Texas at Austin. He is currently a Technologist with Intel, India, managing a team working in the 5G, Telecom, and Networking industry. His research interest includes optimizing 5G networks for energy efficiency and using Intel Xeon SoCs to newer workloads and market segments.

Sandip Chakraborty (Member, IEEE) received the B.E. degree from Jadavpur University in 2009, and the M.Tech. and Ph.D. degrees from the Indian Institute of Technology Guwahati, India, in 2011 and 2014, respectively. He is working as an Associate Professor with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur. His primary research interest is in the intersection of computer systems, pervasive computing, and human–computer interaction. He is one of the founding members of ACM IMOBILE, the ACM SIGMOBILE chapter in India. He works as an Area Editor of *Ad Hoc Networks* (Elsevier) and *Pervasive and Mobile Computing* (Elsevier).