A Deep Reinforcement Learning-based Resource Management Scheme for SDN-MEC-supported XR Applications

Bao Trinh
Insight SFI Research Centre for Data Analytics
Dublin City University
Dublin, Republic of Ireland
bao.trinh@insight-centre.org

Gabriel-Miro Muntean
School of Electronic Engineering
Dublin City University
Dublin, Republic of Ireland
gabriel.muntean@dcu.ie

***Abstract*—The Multi-Access Edge Computing (MEC) paradigm provides a promising solution for efficient computing services at edge nodes, such as base stations (BS), access points (AP), etc. By offloading highly intensive computational tasks to MEC servers, critical benefits in terms of reducing energy consumption at mobile devices and lowering processing latency can be achieved to support high Quality of Service (QoS) to many applications. Among the services which would benefit from MEC deployments are eXtended Reality (XR) applications which are receiving increasing attention from both academia and industry. XR applications have high resource requirements, mostly in terms of network bandwidth, computation and storage. Often these resources are not available in classic network architectures and especially not when XR applications are run by mobile devices. This paper leverages the concepts of Software Defined Networking (SDN) and Network Function Virtualization (NFV) to propose an innovative resource management scheme considering heterogeneous QoS requirements at the MEC server level. The resource assignment is formulated by employing a Deep Reinforcement Learning (DRL) technique to support high quality of XR services. The simulation results show how our proposed solution outperforms other state-of-the-art resource management-based schemes.**

***Index Terms*—SDN, NFV, Edge computing, QoS, extended Reality**

I. Introduction

The fast pace of technology advancements in both hardware and software [1] [2] has provided support for many innovative applications, including eXtended Reality (XR) ones. XR applications can be deployed on devices such as mobile phones, smart glasses, etc., but they require high computing power, storage capacity, and network bandwidth in order to guarantee good Quality of Service (QoS) and consequently high user Quality of Experience (QoE). For example, in Augmented Reality (AR) applications, digital information such as text or virtual objects are seamlessly integrated with real world objects to achieve immerse experiences. Such applications have employed deep learning-based object detection algorithms, such as: Fast R-CNN [3], SSD [4], YOLO [5], [6] that need high intensive computing and storage capabilities. Such demands can overwhelm the mobile devices with limited processing power, storage and battery capacity. A possible solution to address this issue is to offload the intensive computing tasks to cloud servers, conserving energy and reducing the processing needs at devices. Another example includes Virtual Reality (VR) applications with 360° video streaming services. Such applications require both high bandwidth and low latency network connectivity for streaming rich video content [7].

The Mobile Cloud Computing (MCC) paradigm [8] was introduced to help in these situations. However, despite their many advantages in terms of large amount of storage capacity and processing power, MCC systems suffer from excessively long latency for mobile applications. Also, network bottleneck can occur at the backhaul due to the high bandwidth traffic required by users.

To address these MCC limitations, Multi-Access Edge Computing (MEC) [9] was introduced to provide computing capabilities at network edge, within the Radio Access Network (RAN), in close proximity to the mobile user and achieve latency and backhaul network traffic reduction. Various applications can benefit from deploying MEC, such as: Internet of Things (IoT) system [10], [11], increasing QoS/QoE for video delivery [12], [13]. However, in comparison to MCC, MEC servers have limited resources in terms of storage and computational power. There is a need for MEC servers to manage these resources and solutions are needed, including for resource-hungry XR applications.

The European Telecommunication Standards Institute (ETSI) has established an industry specifications group (ISG) to define MEC features to support a wide range of applications [9]. However, the existing work is dedicated to non-computation-intensive applications, mostly bandwidth resource demanding, and to applications with homogeneous QoS requirements. There is a gap for solutions to optimize the resource utilization for high computation demanding applications such as XR with different QoS levels.

This paper proposes a novel Deep Reinforcement Learning-based Resource Management scheme (DRL-RM) for SDN-enhanced MEC architectures to support high quality XR applications at mobile devices (MD). DRL-RM addresses

978-1-6654-3161-3/22/$31.00 ©2022 IEEE

challenges in terms of both network communication and computing while guaranteeing heterogeneous application QoS requirements. The main contributions of our proposed DLR-RM are as follows:

- We employ SDN and NFV technologies to orchestrate multiple wireless access technologies when inter-working with the aim to increase the data traffic volume. Also, by integrating the SDN controller with MEC servers, unified control plane interfaces are provided by decoupling the control plane from the data plane without placing new infrastructures. With global network control, intelligent traffic steering and efficient resource management can be achieved to improve the overall resource utilization. Furthermore, by enabling NFV in could-computing and MEC servers, different network functions supporting XR applications can be programmed on servers flexibly with reduced provisioning cost.
- We formulate the resource utility optimization problem in the context of a Markov Decision Process (MDP) framework. The optimization problem considers both overall resource utilization with heterogeneous QoS guarantee via computing task migration and radio bandwidth slicing. Then the Deep Reinforcement Learning technique is employed to derive the close-to-optimal resource assignment for each incoming requests from XR users with regards to the heterogeneous QoS requirements.
- The proposed scheme is evaluated in a simulation environment and benchmarked against other state-of-the-art MEC resource management algorithms.

The rest of this paper is organized as follows: Section II discusses some relevant works in MEC resource management. The overall system architecture is introduced in section III. The proposed DRL-based algorithm is described in section IV and the simulations and results are discussed in section V. Conclusions and future works end the paper.

II. Related Works

This section discusses some state-of-the-art resource management schemes for various use cases and employing a MEC-based architecture.

The MEC paradigm has received much attention over last several years in both academia and industry. As already mentioned, a MEC Industry Specifications Group (ISG) was established to standardize the adoption of MEC into RANs [14]. The authors in [15] proposed a system architecture for MEC server based on SDN and NFV concepts. However, none of these proposals introduced any mechanism for supporting heterogeneous resource requirements.

The authors of [16] proposed a SDN-based resource management scheme for connected and autonomous vehicles (CAV) networks. The computing and storing resource management is formulated as an optimization problem that maximizes an objective network utility. The simulation results show how the proposed scheme outperforms alternative solutions in terms of network throughput. A similar solution that supports vehicular networking by cooperating with MEC is proposed in [17]. This a scheme, based on the SDN concept, mainly focuses on improving the mobility management of nodes, whereas also guaranteeing QoS expectation. A SDN-based radio resource management framework for eNodeB is proposed in [18]. In this scheme, the management scheme that is executed at the SDN controller is responsible for coordinating distributed resources over multiple MEC servers.

The researchers authors of [19] proposed a joint solution that considers both an offloading scheme at mobile devices and a resource allocation algorithm at MEC server. The numerical simulation results show how such proposed scheme performs closely to the optimal solution in terms of both offloading decision and resource allocation. [20] considered the multicast-aware resource allocation for MEC system that optmize both computing and caching scenarios.

In summary, most of the existing works mainly focus on managing computing, storage, and bandwidth resources at MEC server separately. However, XR applications require a combination of these three resource types at different levels. This article bridges this gap by proposing a resource management scheme that takes into account all of these three resource types for XR users.

III. DRL based resource management for SDN-based MEC systems

A. System Model

We consider a number of base stations (BS) connecting with a single MEC server that is located at the edge of core network as illustrated in Figure 1. Each BS is directly connected to the MEC server through a well-provisioned wired network such as we can ignore the communication latency between them. The serving MEC server also has connections with the central cloud and another MEC server in order to migrate the computation tasks. Although the radio resource management is conducted at BS, we assume that the MEC system might be able to monitor the radio resource status via the interface between MEC server and BS.

The MEC server is able to provide computation offloading service to multiple users simultaneously. Different from other scenarios, XR applications normally require high computation and bandwidth resources for running highly computational tasks and/or streaming video content to multiple devices. So, the computing resources made available at each MEC server includes computing/storing resources plus the bandwidth assigned for a specific request. After receiving the offloaded task from a XR user, it will execute the task on the behalf of the user, and finally return the output result back to the user. The MEC server is also responsible for streaming the video to VR users.

B. Problem Formulation

Denote B_{max}, C_{max}, and S_{max} as maximum network bandwidth, computing and resource capacity at MEC server, respectively. Figure 2 illustrates the network resource management architecture based on the concept of SDN/NFV that includes the following components:

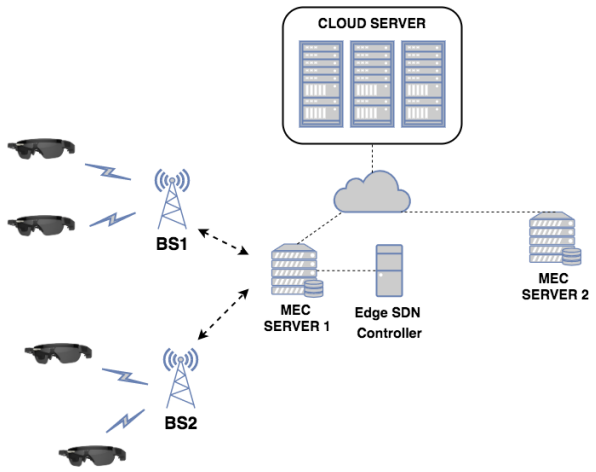

Fig. 1. Example of general architecture of an MEC system

TABLE I
NOTATIONS & DEFINITIONS

B_{max}	Maximum bandwidth at MEC server
C_{max}	Maximum computing capacity at MEC server
S_{max}	Maximum Storage capacity at MEC server
$\{b_i, c_i, s_i\}$	Resource required for request i in terms of bandwidth, computing, storage capacity
$\hat{L}^i$	Latency required for request i
θ	Actor network parameter
$\mathbf{w}$	Critic network parameter
U	Resource utility value of MEC server

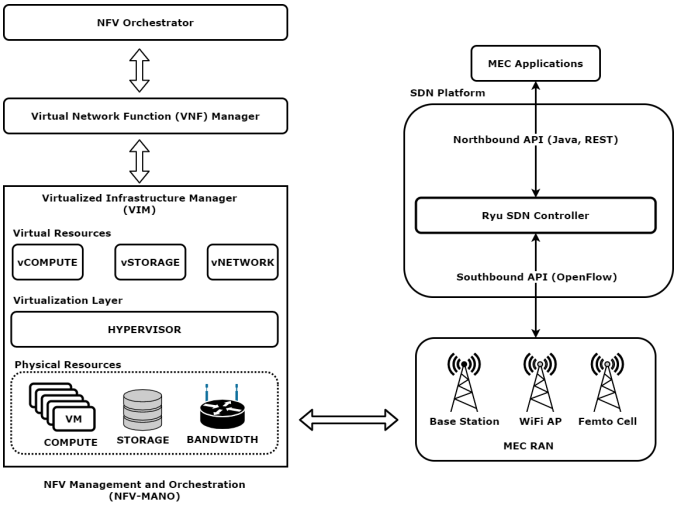

Fig. 2. SDN/NFV based resource management at MEC server

- *Physical resources* consists of bandwidth, computing, and storage modules. These are the physical resources related to virtual machine, storage and network bandwidth, respectively.
- *Virtualization layer* refers to the abstraction of the physical resources and runs on top of a hypervisor platform.
- *Virtual resources* includes vCompute, vStorage, and vNetwork modules. The Data Plane includes the VNFs which are controlled and managed by the SDN-controller via Southbound API using the OpenFlow protocol. MEC Applications communicates with SDN-Controller via Northbound API to monitor and manage the flow tables in the data plane.
- *Virtual network function (VNF) manager* module manages the life-cycle of VNFs including instantiation, upgrading, termination.
- *NFV Orchestrator* module is responsible for coordinating the network bandwidth, computing, and storage resource to provide cloud-based services and applications.

We define the QoS requirement for an incoming request i as a set of $Q_i = \{b_i, s_i, c_i\}$ where b_i, s_i, and c_i denotes downlink bandwidth, storage, and computing, respectively. Then the queue at MEC server with heterogeneous QoS levels is specified by $Q = Q_1, Q_2, \ldots Q_T$ where T is the queue length. For each QoS level request, the MEC server decides a corresponding set of resources in terms of computing, storing and bandwidth. A computing unit can be a set of containers or a virtual machine (VM) with specified Central Processing Unit (CPU) and memory. For generalization, we denote C_{max} as the maximum unit of computing resource at the MEC server. For storing the offloaded data for further processing or video streaming, the MEC server needs to allocate some storage capacity. We denote S_{max} as the maximum storage that MEC server is capable of allocating. Finally, we assume that the bandwidth is reused and sliced among the BSs connected to the same MEC server. Then, the maximum bandwidth that MEC server can allocate to users is B_{max}.

MDP-based Problem Formulation

In order to obtain optimal computing and storing resource allocation while balancing the trade-off between increasing the computing/storing resource utilization and reducing the task migration cost, we formulate the problem of resource management by using the MDP framework as follows:

- **State space:** including QoS set of incoming request $Q_i = \{b_i, c_i, s_i\}$ and $\hat{L}_i$ as the latency required by such request before it moves to the next BS.
- **Action space:** optimal corresponding resource allocation set including $\{\hat{b_i}, \hat{c_i}, \hat{s_i}\}$ OR migrate the incoming request to central cloud / other MEC server.
- **Reward function:** the reward value is used as feedback of the quality of chosen action. In this paper, we define reward value as resource utility U_i minus the task migration cost M_i as follows:

$$r_i = U_i - M_i \tag{1}$$

The formula to of U_i can be derived as:

$$U_i = \frac{B_i^{remain}}{B_{max}} + \frac{C_i^{remain}}{C_{max}} + \frac{S_i^{remain}}{S_{max}} \tag{2}$$

where B_i^{remain}, C_i^{remain}, and S_i^{remain} refer to current remaining resources managed at the MEC server at time step i. For generalization, we consider M_i as a scalar

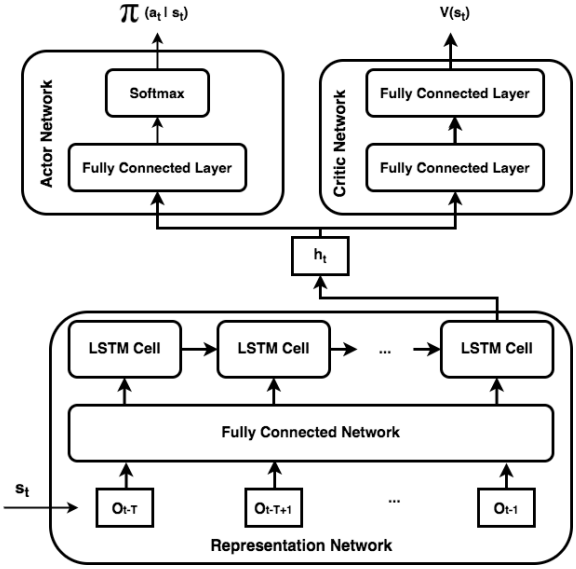

Fig. 3. The LSTM based representation network

value referring to the migration cost, that can be extra latency or bandwidth consumption.

C. DRL-RM Algorithm

Figure 3 illustrates the Long Short Term Memory (LSTM) Actor Critic (AC) based architecture for solving the MDP. LSTM is a powerful artificial neural network architecture that is widely used in prediction and classification, such as in time series data [21]. We employ LSTM in this paper to learn the temporal regularity of continuous state space in terms of bandwidth, computing, storage, and required latency of users. The details of LSTM AC are as follows:

- *Representation network* incorporates a fully connected (FC) layer and an LSTM layer. This network is responsible for detecting the temporal correlation of states. The FC layer takes the buffer $\mathbf{B}$ as input and then feeds the extracted feature tensor to the LSTM layer. The output of the LSTM layer is the variation regularity of states from the last T observation vectors in the buffer. After T updates, last LSTM cell outputs a completed representation of the environment h_t that is then used as input for both Actor and Critic networks.
- *Actor network* comprises one FC layer that takes the output from the *representation network* and generates actions for the current states that is specified by a Softmax function. The output of Softmax function is a probability of different available actions $\pi(a_t|s_t)$. Then, the taken action is sampled following $\pi(a_t|s_t)$.
- *Critic network* estimates value of current state and incorporates two FC layers. The first FC layer takes the h_t from representation network and extract value-related features. Then, the second FC layer output the estimated state value $V(s_t)$

Algorithm 1 presents the details of the DRL-RM scheme. Denote θ and $\mathbf{w}$ as Actor and Critic network parameters, respectively. We use a buffer $\mathbf{B}$ with length T to concatenate a series of states to feed into LSTM layer. We initialize the buffer via running a loop with T iterations to take a series of states in to $\mathbf{B}$. From the beginning of each loop, all states in the buffer $\mathbf{B}$ is concatenated and fed into the representation network. The output h_t is then considered as input of both critic and actor network. The action a_t is taken via sampling from the output of actor network and the next state s_t is then appended into the buffer $\mathbf{B}$. The output of critic network is the estimated value of $V(s_t)$. Next, the agent continues concatenating the data from buffer $\mathbf{B}$ to form another input s_{t+1} to representation network. The value $V(s_t)$ is then estimated from the output of critic network. We calculate the Temporal Difference (TD) error δ via using equation $\delta = r_t + \gamma V(s_{t+1}) - V(s_t)$. α_A and α_C are learning rate of actor and critic network, the values of θ and $\mathbf{w}$ are updated according to equation (3) and equation (4), respectively.

Algorithm 1 DRL based Resource Allocation Algorithm

Input: Incoming request

Output: Optimal computing/storage/bandwidth resources, B

Initialisation

1: Actor network parameters θ
2: Critic network parameters $\mathbf{w}$
3: An empty replay buffer B of length T
4: **for** $i = 1$ to T **do**
5: Randomly choose an action $a_i \in A$ and perform a_i
6: The agent takes the next state O_i
7: Append O_i to buffer B
8: **end for**
9: **while** True **do**
10: Concatenate states in the buffer $\mathbf{B}$ to form $s_t = \{O_{t-T}, \ldots, O_{t-1}\}$
11: Feed O_t to the representation network and take the output h_t
12: Feed h_t to Critic network and calculate $V(s_t)$
13: Feed h_t to Actor network and take $\pi(a_t|s_t)$ and perform a_t
14: The agent receives the reward r_t and gets the new observation O_t
15: Append O_t to the buffer $\mathbf{B}$
16: Concatenate observations in the buffer $\mathbf{B}$ to form $s_{t+1} = \{O_{t-T+1}, \ldots, O_t\}$
17: Feed O_{t+1} to the representation network and take the output h_{t+1}
18: Feed h_{t+1} to Critic network and calculate $V(s_{t+1})$
19: Calculate Temporal Difference (TD) error $\delta = r_t + \gamma V(s_{t+1}) - V(s_t)$
20: Update θ of the Actor network following equation (3)
21: Update $\mathbf{w}$ of the Critic network following equation (4)
22: **end while**
23: **return** P

The parameters θ for Actor network and $\mathbf{w}$ Critic network can be updated with the following equations:

$$\theta \leftarrow \theta + \alpha_A \delta \nabla ln \pi(a_t|s_t, \theta) \tag{3}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_C \delta \nabla \hat{v}(s_t, \mathbf{w}) \quad (4)$$

IV. Simulation Testing

This section describes how the proposed DRL-RM was evaluated in comparison with alternative approaches.

A. Simulation Settings

We have implemented the proposed DRL-RM solution by using Mininet version 2.2.2 [22] with Ryu-SDN controller [23] on Ubuntu 18.0 LTS. The DRL-RM algorithm is built upon using the Tensorflow library [24] for Python.

We assume that the incoming requests from XR devices have heterogeneous QoS requirements in terms of bandwidth, computing and storage. In more details, the bandwidth requirements vary between $1 - 25$ Mbps. The requirements for computing can be specified in terms of Virtual Machine, Container or Central Processing Unit (CPU)-related metrics. We use Gigabyte to describe the storing requirements. We assume that such values are then normalized in the $0-1$ range, so that they can be fed into a neural network as inputs. The number of XR devices is set to 20.

To evaluate the performance of DRL-RM, we take into account two key metrics: the cumulative resource utility over time as described in equation (1) and the average throughput achieved at XR devices.

Finally, the proposed DRL-RM is benchmarked against the following two alternative resource management-based solution for MEC:

- DRL-CAV [16]: A SDN-based approach for resource management for MEC systems. However, DRL-CAV is designed for small data size exchange in Automated Vehicle networks.
- Greedy method: This method always migrate the low QoS requests to the central cloud or other MEC server and only serve high QoS levels. We consider Greedy method as baseline to compare against.

B. Simulation Results

Figure 4 illustrates the average throughput measured at XR devices over time. DRL-RM achieves an average throughput of 16.27Mbps, in comparison to 11.51Mbps and 15.04Mbps of SDN-CAV and greedy methods, respectively. The main reason why the greedy method achieves better performance than SDN-CAV is due to the priority of serving high QoS requests and migrate the low QoS ones to other servers. In contrast, our proposed solution DRL-RM takes into account the wide range of heterogeneous QoS requirements and performs appropriate resource management. Throughout the learning phase, DRL-RM takes a close-to-optimal decision about serving or migrating the request. DRL-RM's average throughput is higher with 41.3% and 8.1% in comparison to those of SDN-CAV and greedy methods, respectively.

Figure 5 illustrates the overall utility value of all schemes. It is observed that, the greedy method has the most unstable results due to the prioritizing high QoS level requests and migrating the low QoS ones to the cloud server or another MEC server. This results in using all resources. On the other hand, SDN-CAV scheme does not take into account the bandwidth resource utilization when allocating computing and storage resources. Neither of these approaches are positive, so DRL-RM achieves better results with 32.7% and 71.12% in terms of resource utility in comparison to SDN-CAV and greedy schemes, respectively.

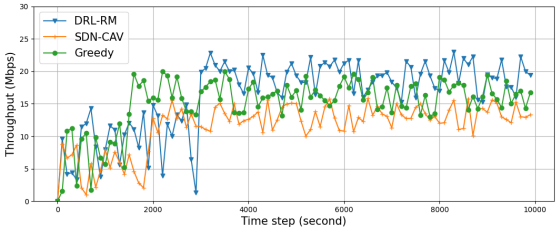

Fig. 4. Average throughput measured at XR devices

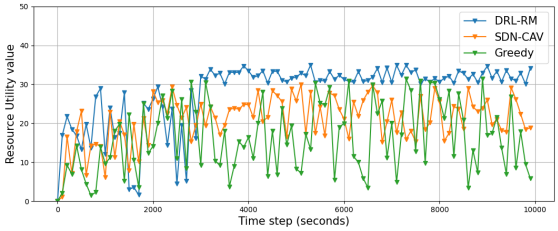

Fig. 5. Resource utility

TABLE II
Simulation Results.

		Average	Improvement
Resource utility	*DRL-RM*	27.74	
	SDN-CAV	20.90	32.7%
	Greedy	16.21	71.12 %
Throughput (Mbps)	*DRL-RM*	16.27	
	SDN-CAV	11.51	41.3%
	Greedy	15.04	8.1%

V. Conclusions and Future Works

In this paper, we proposed a SDN-based resource management solution for MEC systems. The solution specifically addresses the issues of high demanding services such as XR applications with heterogeneous QoS requirements. We formulate the problem of resource management by employing the MDP framework and use the DRL technique to derive the resource assignment decision. The simulation results show how our proposed scheme improves the results of other resource management algorithm in terms of throughput and utility value.

Future works will focus on the issues of resource management in a broader context, with multiple MEC servers. Additionally, we will try to validate the proposed solutions in specific scenarios, such as: education, healthcare, etc.

ACKNOWLEDGMENT

This publication has received support from the Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2 for the Insight SFI Research Centre for Data Analytics, co-funded by the European Regional Development Fund.

REFERENCES

[1] Soh K Ong and Andrew Yeh Chris Nee. *Virtual and augmented reality applications in manufacturing*. Springer Science & Business Media, 2013.

[2] SangSu Choi, Kiwook Jung, and Sang Do Noh. Virtual reality applications in manufacturing industries: Past research, present findings, and future directions. *Concurrent Engineering*, 23(1):40–63, 2015.

[3] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[6] John Patrick Sexton, Anderson Augusto Simiscuka, Kevin Mcguinness, and Gabriel-Miro Muntean. Automatic cnn-based enhancement of 360° video experience with multisensorial effects. *IEEE Access*, 9:133156–133169, 2021.

[7] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. A survey on adaptive 360° video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials*, 22(4):2801–2838, 2020.

[8] Mazliza Othman, Sajjad Ahmad Madani, Samee Ullah Khan, et al. A survey of mobile cloud computing application models. *IEEE communications surveys & tutorials*, 16(1):393–413, 2013.

[9] Multi-access edge computing (mec); phase 2: Use cases and requirements. *European Telecommunications Standards Institute*, 2018-10.

[10] Bao-Nguyen Trinh, Liam Murphy, and Gabriel-Miro Muntean. A reinforcement learning-based duty cycle adjustment technique in wireless multimedia sensor networks. *IEEE Access*, 8:58774–58787, 2020.

[11] Bao Trinh Nguyen, Liam Murphy, and Gabriel-Miro Muntean. Energy-efficient qos-based congestion control for reliable communications in wireless multimedia sensor networks. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[12] Wanxin Shi, Qing Li, Ruishan Zhang, Gengbiao Shen, Yong Jiang, Zhenhui Yuan, and Gabriel-Miro Muntean. Qoe ready to respond: A qoe-aware mec selection scheme for dash-based adaptive video streaming to mobile users. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4016–4024, 2021.

[13] Yashar Farzaneh Yeznabad, Markus Helfert, and Gabriel-Miro Muntean. Cross-layer joint optimization algorithm for adaptive video streaming in mec-enabled wireless networks. In *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6. IEEE, 2021.

[14] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[15] Tejas Subramanya, Leonardo Goratti, Shah Nawaz Khan, Emmanouil Kafetzakis, Ioannis Giannoulakis, and Roberto Riggio. Sdec: A platform for software defined mobile edge computing research and experimentation. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–2. ieee, 2017.

[16] Haixia Peng, Qiang Ye, and Xuemin Sherman Shen. Sdn-based resource management for autonomous vehicular networks: A multi-access edge computing approach. *IEEE Wireless Communications*, 26(4):156–162, 2019.

[17] Syed Danial Ali Shah, Mark A Gregory, Shuo Li, and Ramon Dos Reis Fontes. Sdn enhanced multi-access edge computing (mec) for e2e mobility and qos management. *IEEE Access*, 8:77459–77469, 2020.

[18] Prateek Shantharama, Akhilesh S Thyagaturu, Nurullah Karakoc, Lorenzo Ferrari, Martin Reisslein, and Anna Scaglione. Layback: Sdn management of multi-access edge computing (mec) for network access services and radio resource sharing. *IEEE Access*, 6:57545–57561, 2018.

[19] Tuyen X Tran and Dario Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology*, 68(1):856–868, 2018.

[20] Hao Hao, Changqiao Xu, Shujie Yang, Lujie Zhong, and Gabriel-Miro Muntean. Multicast-aware optimization for resource allocation with edge computing and caching. *Journal of Network and Computer Applications*, 193:103195, 2021.

[21] Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang. Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, 57(6):114–119, 2019.

[22] *Mininet, http://mininet.org/download/*. 2019.

[23] *Ryu-SDN, https://ryu-sdn.org/*. 2021.

[24] *Tensorflow, https://www.tensorflow.org/*. 2021.