5G Edge Computing Experiments with Intelligent Resource Allocation for Multi-Application Video Analytics

Tzu-Hsuan Chao, Jian-Han Wu, Yao Chiang, Hung-Yu Wei
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan

***Abstract*—The fifth-generation mobile network is characterized as the edge of wireless connectivity for all intelligent automation. Technically, the services' requirements for Quality of Service (QoS) have become more strict on latency and throughput. As a result, the concept of Mobile Edge Computing (MEC) has become promising. By placing servers close to the user-equipment (UE), the paradigm enables much lower data transmission time compared to the cloud-based scenario. With this advantage, MEC reaches the requirements of low-latency. Moreover, recognition and detection technology can be thus implemented in several live video analytics scenarios. However, due to the limited physical size on the edge server, resource allocation becomes a crucial issue. In this paper, we proposed a Resource Management method with Multiple Applications in Edge architecture (RMMAE) to intelligently reallocate computing tasks in the heterogeneous network. We design an algorithm to allocate computing resources to applications such as facial detection, object detection and pose estimation in our Edge testbed, and we prove impressive improvement and performance on our testbed with multiple applications.**

***Index Terms*—Edge Computing, Resource Allocation, Live Video Analytics**

I. Introduction

With the rapid rise of the fifth-generation mobile communications era, artificial intelligence and Deep Learning (DL), the efficiency and speed of network transmission and powerful computing capabilities are becoming increasingly important. Therefore, cloud computing is also popular because of its sufficient data storage and excellent computing power without direct active management by the user. Nowadays, several providers have paid service of the cloud server, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, etc. Users may choose suitable services to solve the problems. Nevertheless, the weakness of cloud servers that the transmission delay between end-users and cloud servers may keep the cost increasing considerably. Video analytics services in real-time are also harsh constraints on network bandwidth and congestion. Therefore, the purpose of our works is to provide a high-quality and low-latency computing system. Base on the ETSI Mobile Edge Computing (MEC) in 5G white paper [1], we realize the edge computing is an important milestone in the 5G era. Furthermore, live video analytics can be a common application in the MEC system because of low-latency demand. For example, surveillance cameras play an essential role in maintaining public safety. In addition, the widespread development of traffic monitors, self-driving, Virtual Reality (VR), and Augmented Reality (AR) related to video analytics. Object detection and recognition apply in many areas of computer vision, including image retrieval, security, surveillance, automated vehicle systems, and machine inspection.

However, limited computing resources of edge servers may not afford multiple complexity and huge DL models. In brief, it is a trade-off problem between low latency and high quality. Delivering all data to the cloud server and performing high-quality video analytics come with exceptionally high transmission costs. If we only process the data to the edge server, the server near users, the computing power may constrain the result. So, we choose edge computing of cloud computing on demand. Some works proposed the idea that determines whether the service is quality-oriented or speed-oriented by scoring users' needs, such as [2] and [3]. The distributed computing for the edge server and the cloud server is another choice. The concept is similar to federated learning [4]. The authors applied a decentralized approach to make the system more efficient. Both [5] and [6] deployed the work through distributed deep learning models deployed on the MEC nodes for better performance. We can divide one task into sub-tasks and distribute them to an edge server and a cloud server for computing in an Edge-Cloud system. On the other hand, resource allocation is also an essential issue for the edge-cloud system. Hence the restriction of the computing resource on the edge server, we need to allocate limited resources effectively and use them where it is most need. [7] proposed a block-chain-based video streaming system, and they formulate the problem of optimizing the offload scheduling to optimize. [8] provided a promoted QoS method to offload tasks.

The DICE-IoT system in [9] is our previous work. It provides a facial recognition app for video analytics and formulates the management to incentivize cooperative computing provision between the Edge and the Cloud. Inspired by this work, we will continue the concept of resource allocation and implement more applications on our testbed to deal with

The authors are grateful for the funding support from Ministry of Science and Technology (MOST) of Taiwan under grant 109-2221-E-002-148-MY2 and 109-2218-E-002-018-.

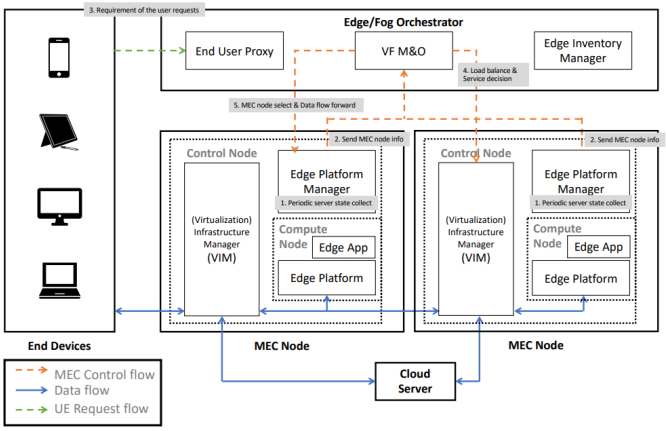


Fig. 1. Hierarchical architecture of two-tier Edge-Cloud in our testbed

more complex problems. Furthermore, we consider a model-level pipeline composition that can decompose the model on the different servers to cooperate for one application. The architecture of our MEC system is a two-tiered edge cloud computing with real-time video analytics. On top of that, we provide an improved method RMMAE for resource load balancing and implement the complete MEC system in the actual environment.

The main contribution of our work is as follows:

- We reduce the transmitted latency by allocating resources appropriately according to the system's current situation.
- This is the Edge system with implied reasonable algorithms to discuss and realize the situation where multiple applications are in the system at the same time.
- We distribute the resources by the detection accuracy request, current video transmission quality and GPU status of edge server.

II. Proposed Scheme Design

In this section, we introduce the three-layered architecture of the system, including the Cloud Layer, the MEC Layer and the User Application Layer. The interaction between these layers allows better allocation and improves the system performance. Also, we design the data flow and the control flow with our management mechanism.

A. Edge-Cloud Cooperative Computing Architecture

Edge and Fog Management and Orchestration is essential in the MEC Layer, which plays a critical role in coordination between components. According to hierarchical management, we can master and integrate all the MEC nodes easily and orderly. Figure 1 shows the architecture of the Edge system in our testbed. First, the Edge server includes several MEC nodes and the Edge Orchestrator; then, the Edge Orchestrator will manage and coordinate MEC nodes to work in the MEC server. As for a single MEC node, it consists of a control node and a compute node. The control node may receive the control signals sent by the orchestrator. It also collects and sends the information about GPU status to the orchestrator periodically. The compute node in the MEC node provides low latency computing, but with limited resource constraint while the cloud server is assumed to have unlimited resource budget.

TABLE I
Parameters of Configuration

Notation	Definition
fr^{ϕ}	Frame sampling rate (in fps)
w^{ϕ}, h^{ϕ}	Frame {width, height} (in pixels)
rs^{ϕ}	Video resolution $rs^{\phi} = w^{\phi} \times h^{\phi}$
b_{edge}^{ϕ}	Data size uploaded from the device to the edge node per frame (in MB)
b_{cloud}^{ϕ}	Data size output from the edge node to the cloud per frame (in MB)
m_{edge}^{ϕ}	GPU memory usage at the edge node (in MB)
m_{cloud}^{ϕ}	GPU memory usage at the cloud (in MB)
u_{edge}^{ϕ}	GPU utilization at the edge node (in $[0, 1]$)
u_{cloud}^{ϕ}	GPU utilization at the cloud (in $[0, 1]$)
c_{edge}^{ϕ}	Energy consumption at the edge node (in Watts)
c_{cloud}^{ϕ}	Energy consumption at the cloud (in Watts)
p_{edge}^{ϕ}	Per frame processing time at the edge node (in ms)
p_{cloud}^{ϕ}	Per frame processing time at the cloud (in ms)
Q^{ϕ}	Quality of detection and recognition (in $[0, 1]$)
z^{ϕ}	Partial offloading indicator to the cloud

We considered three types of end devices: mobile phones connected to a 4G/5G base station, those connected to a WiFi access point, and PCs connected with the Ethernet; each of them has a camera. The solid line is the data flow, and the dashed line means is the control flow. The camera records live video and streaming it to the MEC server for real-time analytics. The control nodes and the orchestrator in the MEC server will then allocate computing resources and forward the streaming request to either an appropriate MEC node or the cloud server, based on the end users' demand and the GPU status of the edge server. Generally, the edge server aims to provide the service with high quality and low latency.

B. System Model and Configurations

Our video programs run in both Cloud and MEC Layers, but the resource management is only considered in MEC due to the infinite resource assumption in Cloud. The devices in the User Application Layer capture video and streaming it to the MEC node for real-time video analytics, such as object detection, facial detection, and pose estimation. Suppose that each device has a specific latency requirement L^{req} and a quality requirement, Q^{req}. We also define configuration ϕ as a specific decomposable pipeline and Φ as the collection of all configurations.

We also consider the resolution and the video frame sampling rate for all devices and the GPU resources as the computing power on the cloud server and the edge server for each configuration ϕ. The partial offloading indicator φ^{ϕ} is a binary indicator and is defined as follows: If $\varphi^{\phi} = 1$ means

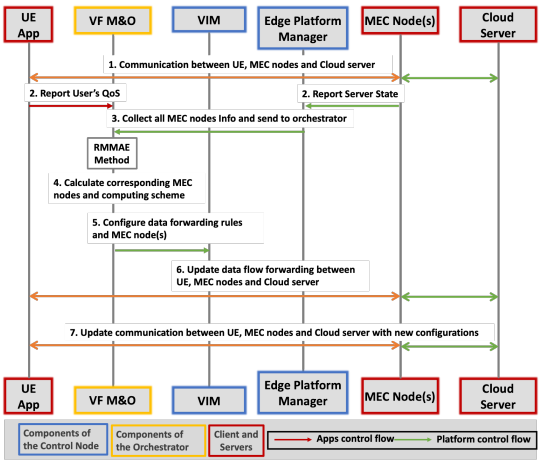


Fig. 2. Workflow of periodic resource allocation control signals

the video is uploaded to Cloud Layer for serving. Otherwise, MEC Layer processes the task.

To sum up, we denote the complete configuration set $\Phi^{\phi}(rs^{\phi}, fr^{\phi}, \varphi^{\phi})$, which consists of all available configurations for the proposed DICE scheme. On the other hand, we also denote the configuration set, which only consists of the MEC stratum configuration $\Phi^{mec} = \left\{\phi : \varphi^{\phi} = 0,\ \forall \phi \in \Phi^{U}\right\}$ We may select the configuration according to the resource status of the MEC node, latency requirement, and accuracy requirement. Table 1 organize the parameters list which we introduce in our problem formulations.

C. Work flow for control signals

Figure 2 presents the periodic control signals between the devices, the MEC nodes, and the cloud server. The period starts with communication between the servers and the user device. After that, the user QoS report and the server state are sent to the orchestrator. Next, the orchestrator will apply our RAMME method to balance the load between the edge server and cloud server, and to select proper MEC nodes and the corresponding computing scheme. In addition, it will configure data forwarding rules for data managers. Therefore, the system updates data flow and the connection between MEC nodes, cloud servers, and user devices. In conclusion, the orchestrator will manage the edge server and also communicate with the UE apps while the system will update the computing configurations and offloading scheme in a more suitable way. We can then control and adjust the system's status dynamically to make services more advanced and efficient.

III. Algorithm and Problem Formulation

We apply the intelligent algorithm RAMME for joint latency- and accuracy-aware live video analytic services with multiple applications in our actual testbed. Referring to the previous work [9], this paper not only considers several types of communication connected to our edge server but implements kinds of applications for users to choose. The problem we've solved is to balance the QoS and the service latency. Therefore, we develop a scoring mechanism considering both social welfare to measure the revenue, and the cost on the Edge and Cloud, to judge the performance of the offloading scheme. The expected social welfare is maximized by selecting optimal $\mathbb{n}, \mathbb{x}$, where $\mathbb{n} = \left\{n_j^{\phi}\ \ \forall j, \phi\right\}$ represents the configuration selections of MEC nodes. The binary variables $\mathbb{x}$ stand for user association decision, where $\mathbb{x} = \left\{x_{ij}^{\phi}\ ,\ \forall\ i, j, \phi\right\}$. The i and j are the number of devices and MEC nodes, respectively. $x_{ij}^{\phi} = 1$ means the device i is associated with the MEC node j with configuration ϕ for video analytics; otherwise $x_{ij}^{\phi} = 0$. We analogize the network to a directed graph $G\left(\mathcal{M},\ \mathcal{L}\right)$, where $\mathcal{M}$ means the set of MEC nodes and $\mathcal{L}$ indicates the communication between devices and the servers.

A. Social Welfare derivation

$$
\begin{aligned}
&\max W\left(\mathbb{x}, \mathbb{n}\right) = U_{mec} + \ U_{cld} \\
&s.t.\ C1: \begin{cases} \sum\limits_{j \in \mathcal{M}_i^c} \sum\limits_{\phi \in \Phi_j^c} x_{ij}^{\phi} \leq 1\ \forall i \\ \sum\limits_{j \in \mathcal{M} \setminus \mathcal{M}_i^c} \sum\limits_{\phi \in \Phi} x_{ij}^{\phi} = 0\ \forall i \\ \sum\limits_{j \in \mathcal{M}} \sum\limits_{\phi \in \Phi \setminus \Phi_j^c} x_{ij}^{\phi} = 0\ \forall i \end{cases} \\
&C2: \begin{cases} n_j^{\phi} \leq N_j^{\phi}\ \forall j, \phi \\ n_j^{\phi} \leq \sum\limits_{i \in \mathcal{N}} x_{ij}^{\phi}\ \forall j, \phi \\ \sum\limits_{\phi \in \Phi} n_j^{\phi} u_j^{\phi} \leq\ u_j^{avl}\ \forall j \\ \sum\limits_{\phi \in \Phi} n_j^{\phi} m_j^{\phi} \leq\ m_j^{avl}\ \forall j \end{cases} \\
&C3: fr^{\phi} \sum_{i \in \mathcal{N}} x_{ij}^{\phi}\ \leq \frac{n_j^{\phi}}{t_j^{\phi}} \Longrightarrow\ \ fr^{\phi} t_j^{\phi} \sum_{i \in \mathcal{N}} x_{ij}^{\phi} \leq\ n_j^{\phi}, \forall j, \phi \\
&C4:\ x_{ij}^{\phi}\left(L_i^{req} - L_{ij}^{\phi}\right) \geq 0\ \forall i, j, \phi. \\
&C5:\ \sum_{j \in \mathcal{M}} \sum_{\phi \in \Phi} x_{ij}^{\phi} fr^{\phi} b^{\phi}\ \leq\ r_{ij},\ \forall l_{ij} \in \mathcal{L} \\
&C6: \begin{cases} \sum\limits_{l_{ij} \in \mathcal{L}} \sum\limits_{\phi \in \Phi} x_{ij}^{\phi} fr^{\phi} b^{\phi}\ \leq\ r_{ij},\ \forall l_{ij} \in \mathcal{L} \\ \sum\limits_{i \in \mathcal{N}} \sum\limits_{\phi \in \Phi} x_{ij}^{\phi} fr^{\phi} b_{cld}^{\phi}\ \leq\ r_{j,cld},\ \forall j \end{cases} \\
&x_{ij}^{\phi} \in \{0, 1\}, \quad n_j^{\phi} \in \left\{0, 1, \cdots, N_j^{\phi}\right\}\ \forall i, j, \phi
\end{aligned}
\tag{1}
$$

where the U_{mec} and U_{cld} are the utility of MEC Layer and utility of Cloud Layer. Constraints C1 and C2 represent the user association between UEs and MEC nodes, and they are bounded by the device requirements and GPU status. C3 considers the frame rate as the constraint to maintain stability of service. Since the latency requirement is asked by the device i, C4 shows the latency constraint. Furthermore, the data rate

of each link needs to satisfy its transmission capacity by C5 and C6.

B. Utility of Cloud layer

Since the MEC stratum pays π to compensate the per unit effort to the cloud for the optimal computing provision, the utility function of the cloud may like:

$$U_{cld} = \ R_{cld} - \ C_{cld} \tag{2}$$

C_{cld} means the operation cost of the cloud per second:

$$C_{cld} = \ \gamma_{cld} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{\phi\in\Phi} x_{ij}^{\phi} \ \ price_{cld}^{\phi} \ \ t_{cld}^{\phi} \tag{3}$$

The γ_{cld} is a constant that denote the cost of converting power consumption at the cloud server. We regard the total GPU utilization as the effort per second of the cloud.

$$e_{cld} = \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{\phi\in\Phi} x_{ij}^{\phi} t_{cld}^{\phi} u_{cld}^{\phi} {fr^{\phi}} \tag{4}$$

And the revenue of the cloud present as:

$$R_{cld} = \ \pi e_{cld} \tag{5}$$

C. Utility of MEC layer

The expected utility function of the MEC stratum is:

$$U_{mec} = \ R_{mec} - \ C_{mec} - P_{mec} \tag{6}$$

And $P_{mec} = R_{cld}$ represents the total service fee.

The operation cost per second of the MEC stratum is similar to the Cloud stratum:

$$C_{mec} = \gamma_{mec} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{\phi\in\Phi} x_{ij}^{\phi} p_j^{\phi} {fr^{\phi}} \tag{7}$$

The γ_{mec} is a constant that denote the cost of converting power consumption at the MEC node. We assume the service fee is proportional to the user satisfaction, we have the revenue as follows:

$$R_{mec} = \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{\phi\in\Phi} x_{ij}^{\phi} \mathcal{P}\left(q^{\phi}\right) \tag{8}$$

And according to equation (3) and (7), the social welfare can be derived as:

$$W\left(\mathbb{a},\mathbb{x},\mathbb{m}\right) = W\left(\mathbb{x}\right) = \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{\phi\in\Phi} x_{ij}^{\phi} \left(\mathcal{P}\left(q^{\phi}\right) - \Omega_j^{\phi}\right) \tag{9}$$

The total operation cost

$$\Omega_j^{\phi} = \left(\Upsilon_{mec} p_j^{\phi} + \Upsilon_{cld} p_{cld}^{\phi}\right) {fr^{\phi}}, \forall j, \phi \tag{10}$$

IV. Experimental Results

This chapter set up an implementation platform and applied the RMMAE algorithm. There are multiple video analytic applications on a real testbed to select and prove our MEC system's better performance by the experimental results.

A. Deploy MEC system in our Testbed

1) Testbed Environment: Chunghwa Telecom (CHT) has set up a 5G campus network and MEC server. The edge server of our experimental testbed is point-to-point connected with Chunghwa Telecom's MEC server to develop various scenarios for the main architecture of the system.

MEC server: The server consists of a control node and a compute node. The compute node is equipped with an individual GeForce RTX 3080 GPU instance. As for external connections, there is a CHT SDN switch link between the MEC server and three kinds of UEs.

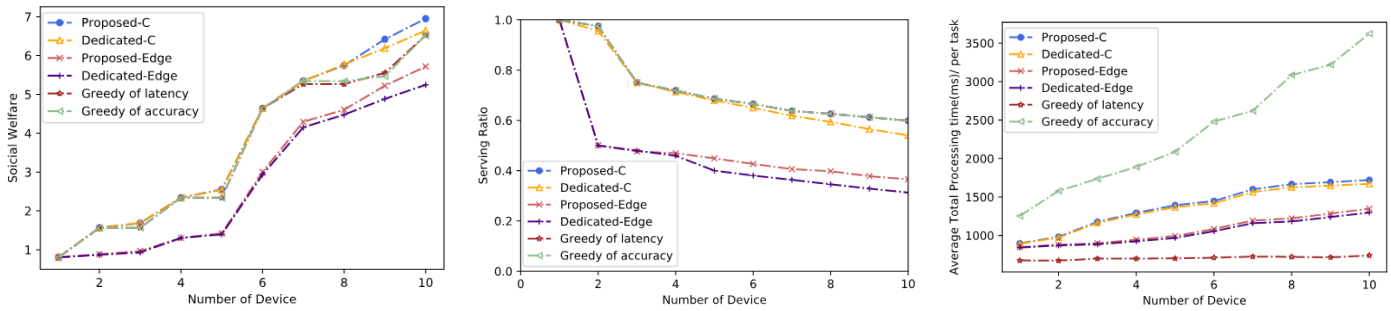
Cloud server: Elastic Compute Cloud (EC2) of Amazon Web Service (AWS) is a cloud-server rental service of Amazon. We rent Amazon EC2 G4 Instances and its type is G4dn.xlarge equipped with NVIDIA T4 GPUs and custom Intel Cascade Lake CPUs.

User Devices: We have three scenarios to simulate the user devices: PCs connected to the Ethernet to surf the Internet, laptops to Wi-Fi, and mobile phones to 4G/5G mobile networks. PCs and laptops connect to the MEC server via the campus internet, while phones connecting via the CHT fiber optic link from the base station to the MEC server.

Applications: Our test platform has three types of applications: facial detection, object detection, and pose estimation. In our previous work [9], we provide the facial detection application. This paper implements object detection and pose estimation with YOLOv4, YOLOv4-tiny AlphaPose, and OpenPose. Also, we combine different kinds of video analytics models above for a decomposable inference pipeline and deploy with three offloading types: Edge only, Cloud only, and Edge-Cloud.

2) Parameter Measurement in Testbed: Taking the actual situation like the scene, we use the mobile phone with CHT 5G network to capture the video stream as input and adjust the relevant camera and transmission information such as frame rate and video resolutions to measure various situations. Both the edge server and cloud server have individual GPU instances as in the previous description. We list 70 kinds of configurations with the different model schemes, and three offloading types: Edge Only means the inference pipeline ran at the MEC layer entirely, Cloud Only means the inference pipeline ran at the Cloud layer entirely, and Edge-Cloud, the inference pipeline is decomposed to run at the MEC layer and Cloud layer.

3) Testing Scheme: We simulate the user requests and implement the RMMAE system in the actual testbed based on the configurations. The simulation parameters are as follows: the frame rate as 30 fps, the video resolutions are uniform distribution in 1080p, 720p, and 480p, the accuracy requirement is the uniform distribution between 0.5 to 0.8, the



(a) (b) (c)

Fig. 3. Performance versus number of devices. (a) Social Welfare. (b) Serving Ratio. (c) Utility of MEC layer. (d) Utility of Cloud layer. (e) Total Processing Time(ms)

latency requirement is the uniform distribution between 200 to 1200(ms) and the application selection also random in three choices. Further, we compare six schemes for experiment:

(1) Proposed-C: Completely framework with enable pipeline-sharing.
(2) Proposed- Edge: Edge-only system with enable pipeline-sharing.
(3) Dedicated-C: Completely framework with a dedicated pipeline.
(4) Dedicated-Edge: Edge-only system with a dedicated pipeline.
The proposed approaches enable one pipeline to serve several task types. The dedicated serve by dedicated task type.
(5) Greedy for Latency: Greedy for the minimum latency configuration.
(6) Greedy for Accuracy: Greedy for the highest accuracy configuration.

B. Implementation Results

Figure 5 shows the performance with a different number of devices in one request. We make each request with 1 to 10 devices simultaneously, and we will test the same number of device requests 5 times with different user simulation parameters. They will be performed on the above six schemes. Figure 5(a) shows the proposed scheme has better social interest than the other schemes. And compare to the two greedy schemes, the greedy of accuracy is higher than the greedy of latency because, in our calculation, the user quality may impact the revenue and the welfare. However, the greedy of accuracy method is still worse than our completely Proposed scheme. Next, we find out these two greedy schemes can serve most devices, except the requirements exceed the system's limit in Figure 5(b). Even if they can make most requests, they still perform better than our proposed scheme in Figure 5(a). That's because it only considers quality or latency. Yet, there is a trade-off problem between accuracy and latency. Figure 5(c) presents greedy for accuracy scheme costs more latency than the other schemes. It only persuades the better quality but does not care about the processing time anymore. In contrast, greedy for latency only manage the processing time; as a result, it may perform in lower latency than others. The processing time with all of the schemes will increase with the growth in the number of the device.

In conclusion, our Proposed-C scheme performs the better result in the testing demo. It may balance the trade-off between accuracy and latency with our management method and apply it in several applications.

V. Conclusions

We provided multi-application services on the system, such as facial recognition, object detection and pose estimation, and also proposed the RAMME method to deal with more scenarios and deploy with the intelligent MEC orchestrator. We implemented the system with the CHT campus 5G system and run the complete service on our actual testbed. Finally, we proved that the proposed scheme of the RAMME has a better performance than other schemes and with higher social welfare as well.

References

[1] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. L. A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, K.-W. Wen, K. Kim, R. Arora, A. Odgers, L. M. Contreras, and S. Scarpina, "Mec in 5g networks," *ETSI White Paper*, no. 28, 2019.

[2] Y. Chiang, C.-H. Hsu, and H.-Y. Wei, "Collaborative social-aware and qoe-driven video caching and adaptation in edge network," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[3] C. W. Y. Hung and R. Hwang, "Combinatorial clock auction for live video streaming in mobile edge computing," *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 196–201, 2018.

[4] M. Aledhari, R. Razzak, R. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 1–1, 01 2020.

[5] J. Chen, K. Li, Q. Deng, K. Li, and P. S. Yu, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[6] Y. Huang, F. Wang, F. Wang, and J. Liu, "Deepar: A hybrid device-edge-cloud execution framework for mobile deep learning applications," *IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, p. 892–897, 2019.

[7] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *Trans. Wireless. Comm.*, vol. 18, no. 1, p. 695–708, Jan. 2019. [Online]. Available: https://doi.org/10.1109/TWC.2018.2885266

[8] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Task offloading and resource allocation in mobile-edge computing system," in *2018 27th Wireless and Optical Communication Conference (WOCC)*, 2018, pp. 1–4.
[9] Y. Zhang, J.-H. Liu, C.-Y. Wang, and H.-Y. Wei, "Decomposable intelligence on cloud-edge iot framework for live video analytics," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8860–8873, 2020.