IEEE P1935 Edge/Fog Manageability and Orchestration: Standard and Usage Example

Tse-Yu Chen, Yao Chiang, Jian-Han Wu, Huan-Ting Chen, Chiao-Cheng Chen and Hung-Yu Wei
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan

***Abstract*—With the innovation of mobile applications and the arrival of the fifth generation of telecommunication, edge computing has become a popular scheme due to its geographical proximity to end users; thus, the overall architecture has the advantage of lower latency and higher user experience. However, because of the physical limitation, management and orchestration in the edge system are necessary to maintain operations, standard functionalities and application lifecycle. Accordingly, the IEEE P1935 working group introduced a standardized, orchestration-wise design to simplify the process and offer better system performance and user experience. In this work, we first give an overview of the three-level Edge/Fog architecture in the P1935 standard and briefly show the components in each level. Moreover, we describe the mechanisms for managing and orchestrating resources and applications. Last but not least, we implement an actual testbed following the standard. To show that the P1935 standard can benefit the Edge/Fog systems, we deploy an edge-based live-streaming service and a real-time transcoding mechanism on the testbed. The evaluation results show that the P1935-aligned system performs well regarding the overall Quality of Experience (QoE).**

***Index Terms*—edge computing, management and orchestration, standardization, live streaming, application lifecycle management, resource lifecycle management**

I. Introduction

Edge computing has become a sensation in fifth-generation (5G) telecommunication technologies. With the emergence of innovative mobile applications, the call for low latency, high efficient bandwidth, and better Quality of Experience (QoE) cannot be overemphasized. Traditional cloud-based network architecture fails to fulfill these requirements. Fortunately, the Edge system [1] meets these requirements by providing computing capabilities proximate to end-users.

Nevertheless, owing to limited computing and storage capabilities and the volatile environment at the network edge, management and orchestration on the edge/fog system has become an essential topic [2]. Some related works focus on task offloading and resource allocation in the edge system. In [3], the authors formulated a dependency-aware task offloading decisions in the Multi-access Edge Computing (MEC) system and aimed to minimize the execution time of tasks. [4] jointly optimized the offloading decision and computational resources to reduce the delay and energy consumption for User Equipments (UEs). [5] proposed a Deep-Q-Learning (DQL)-based scenario to dynamically reallocate resources of the network slices and addressed the task offloading problem on a three-tiered edge computing architecture. [6] extended its previous work [7] to a multi-server scenario and introduced load distribution to leverage the computation resources of MEC servers effectively. An intelligent QoS-aware resource management framework mentioned in [8] is designed for millimeter wave (mmWave) communication in the Radio Access Network (RAN). On the other hand, some works focused on network management. For instance, the authors presented the concept of virtual link migration in [9]. The substrate network utilization can be increased by re-optimizing the embedded paths periodically. [10] proposed a hierarchical edge-cloud architecture to enable 5G network slicing and support various QoE requirements.

Achieving an efficient management and orchestration framework that works well with typical applications and use cases raises its importance as well. For example, M. Zang et al. proposed an LTE-based MEC caching system in [11] that focuses on moving caching entities closer to the users. [12] proposed 2-tiered MEC collaborative video caching architecture and presented a QoE-driven video adaptation algorithm to dynamically transcode the cached videos into appropriate resolution on edge servers. The authors in [13] proposed an edge-assisted system to provide Content Delivery Network (CDN) functionality as a service. The ETSI standard team proposed specifications for MEC [14] to allow various designs to meet general requirements and communicate with each other practically. Then, the IEEE P1935 working group [15] aimed to introduce a standardized, orchestration-wise design to simplify the processes and offer better system performance and user experience. With this unified management and orchestration, the P1935 standard can ensure better availability, flexibility, reliability, scalability, stability, service mobility, and performance in the Edge/Fog systems. In advance of the release of the complete document, this paper provides a brief P1935 introduction and shows our testbed implementation with

Tse-Yu Chen, Yao Chiang, Jian-Han Wu, Huan-Ting Chen, Chiao-Cheng Chen and Hung-Yu Wei are with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan. (email: gpoint.ghdkghdk@gmail.com, owen12300@gmail.com, f10921065@ntu.edu.tw, b06901023@ntu.edu.tw, cccheng0313@gmail.com and hywei@ntu.edu.tw)

The authors have made major contributions to the IEEE P1935 standardization. Hung-Yu Wei is the current P1935 working group chair.

Corresponding author: Hung-Yu Wei(email: hywei@ntu.edu.tw).

an onboard video streaming application. The main contributions of our work are listed as follows:

- We propose the architecture and components of the P1935 standard. Besides, we interpret its thorough lifecycle process and mechanisms for applications and resource management and orchestration.
- We design a lightweight testbed based on the standard and verify the scheme's effectiveness by deploying a P1935-aligned live streaming service with a real-time transcoding algorithm. The results show that the P1935 standard can enhance service performance.

To specify the necessary parts of the management and orchestration in the P1935-aligned Edge/Fog system, Section II introduces the framework, components and functionalities of the Edge/Fog system. Section III describes the procedures for managing the applications and the methods to orchestrate the Edge/Fog system resources. Based on this, we showcase our P1935-aligned testbed implementation with the live streaming application and propose a real-time transcoding decision algorithm in Section IV. Then, the experimental evaluation results are shown in V to show the effectiveness of the standard and our testbed. Finally, we conclude the paper in Section VI.

II. P1935 Edge Management and Orchestration Framework

The P1935 standard defines a three-level architecture for different functions to enable the management and orchestration of the Edge/Fog system. Due to the page limitation, we only cover the fundamentals of this standard, and the complete version can be referred to documents in [16]. Fig. 1 shows the overall architecture of the Edge/Fog system, containing entities and components inside and outside the system, such as 3rd-party software. The following subsections specify the system's three levels: Orchestrator Level, Controller Level and Computer Level.

A. Orchestrator Level Entities

The Edge/Fog Orchestrator (EFO) is designed to manage and control the Edge system. It is responsible for interacting with users, handling applications properly and providing system functionalities and infrastructures.

The **Edge App Designer** handles application design and onboarding. It provides the management functionality, simulates and certifies system assets, processes and policies. **Lifecycle Management Enabler** (LCM Enabler) is responsible for designing and managing the application lifecycle. **Rule Framework** addresses the conditions, requirements, constraints, attributes, and needs that must be provided, maintained and enforced. The **EFO Controllers** configures and coordinates VNFs and related resources. **End User Proxy** (EUP) interacts with end-user applications and allows them to request onboarding, instantiation, and termination of the Edge applications and authorizes these requests. **Edge Inventory Manager** (EIM) component provides a real-time view of system's various inventories, including resources, services, products and their mutual relationships. **Virtual Function Management and Orchestration** (VF M&O) assures the start-up and operation of applications, and operates on the models distributed from Edge App Designer. **External API** is provided for the Edge Service Operators or other entities to to execute specific operations and access the functionalities of the EFO.

B. Controller Level

The Controller Level includes one or more Edge/Fog control nodes, overseeing the Edge/Fog compute nodes and managing related resources. Entities located at Controller Level manage and orchestrate the Edge compute nodes. The entities can be divided into two classes according to their management targets: Resource Management Elements (RME) and Application Management Elements (AME). The function and responsibility of each class are described in the following subsections.

1) Resource Management Elements: Resource management elements focus on managing and storing network infrastructure. They can control and schedule the usage of the resources in the Edge system. **(Virtualization) Infrastructure Manager** (VIM), the primary component for resource management, manages both virtualized and non-virtualized infrastructures and resources, including resource allocation, infrastructure preparation, and system information report.

2) Application Management Elements: Application management elements focus on the lifecycle management of the Edge Platform and the Edge applications running on the platform. **Edge Platform Manager** is the main element for application management, which manages the rules, requirements and lifecycle of applications, and provides element management functions to the Edge Platform. The Edge Platform Manager receives virtualized resource fault reports and performance measurements from the VIM for further processing. It also includes the functional blocks responsible for managing the Edge Platform and applications with standard LCM procedures.

C. Computer Level

The Computer Level includes one or more Edge/Fog compute nodes and is responsible for the practical computing tasks. These compute nodes can also be called "edge nodes" as an abbreviation. The Edge computer entities are the functional elements involved in the computing tasks, including the Edge Platform and a virtualization infrastructure providing compute, storage, and network resources for the Edge applications. **Edge Applications** Edge applications are executed as virtual machines based on the virtualization infrastructure provided by the Edge host. They interact with the Edge Platform to provide Edge services. An Edge Application comprises single or multiple virtual function(s), which can be managed and orchestrated by the VF M&O component in EFO. **Edge Platform** The Edge Platform provides Edge functions to run Edge applications. It serves as a platform where Edge applications can provide, discover, and consume certain Edge services. The Edge Platform also instructs the data plane and configures

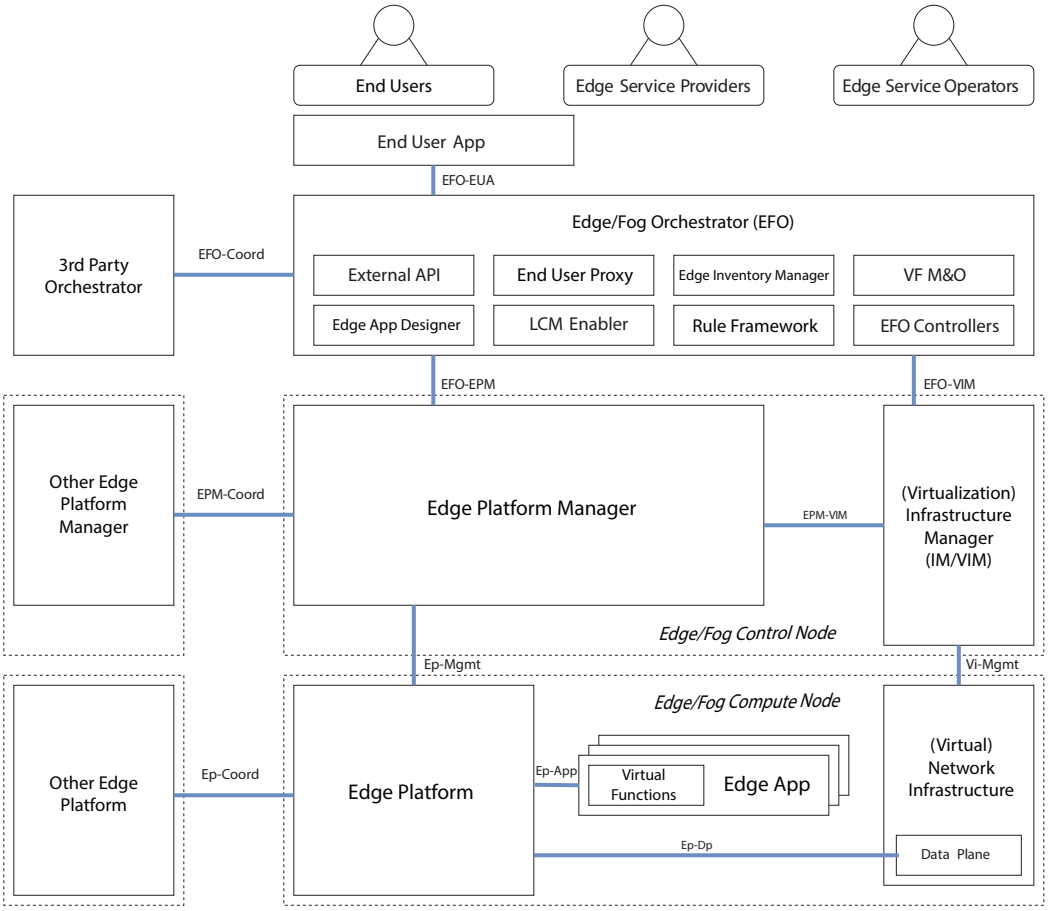


Fig. 1: The framework of P1935. This figure includes the three-layered architecture and their inner components.

DNS proxy/server according to the traffic rules prescribed by the Edge Platform Manager. **Data Plane** The Data Plane in the Edge host deals with the traffic rules, including routing the traffic among entities, task offloading, and application installation from VIM. **(Virtual) Network Infrastructure** The (Virtual) Network Infrastructure (VNI), managed by the VIM, is the totality of all hardware and software that build up the entire environment where VNFs are deployed.

D. External Entities

External entities connect to the external environment. **End User Applications** (EUA) can interact with the End User App LCM component in EFO at the Orchestrator level via the standard API. The users, including end users, Edge Service Operators, and Edge Service Providers, can access the EFO services through the EUA.

III. P1935 Application and Resource Management

To ensure the Edge system functioning properly, the P1935 standard defines a comprehensive management and orchestration scheme. In this section, we give details of management and orchestration of both application and resource.

A. Application Management and Orchestration

Application management and orchestration is the workflow that defines how edge systems interact with edge applications. To provide the lifecycle of edge applications, the EFO has to manage and coordinate both networking and computing resources properly. With such management, Edge Service Providers (ESPs) and Edge Service Operators (ESOs) can provide and manage their edge applications, and End Users (EUs) can access the Edge applications on the Edge Platform.

As shown in Fig. 2, a complete application lifecycle consists of six stages: on-boarding, instantiation, context creation, reconfiguration, context deletion, and termination.

1) On-boarding: The purpose of the onboarding procedure is to prepare the edge application for the Edge platform. The application onboarding procedure consists of two stages. First, The ESP uploads the application software package and the corresponding blueprint to EFO. The blueprint is the description file to deploy the application appropriately. Second, ESO checks the validity of these files and distribute them to the Edge Platform.

2) Instantiation: Based on the package and blueprint file uploaded during the onboarding procedure, ESO launches the application on a specific Edge Platform. After this procedure, EUs can access Edge applications from the Edge Platform.

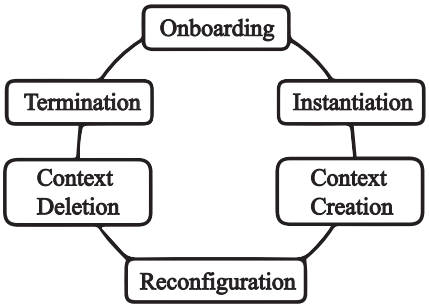


Fig. 2: The lifecycle of application management and orchestration.

3) Context Creation: The application context is the information that helps the ESO make decisions. EUs and ESO can perform application context creation. EUs upload their data regularly, and the data is used to make better decisions by the system. ESOs can update or add the information according to their own needs.

4) Reconfiguration: Based on the application context and information, ESOs can reconfigure their Edge applications. For example, it is possible to increase the accuracy of the service at the cost of latency to meet the requirements of the specific EU.

5) Context Deletion: The purpose of application context deletion is to delete the application context generated in the application context creation procedure. Both EUs and ESO can delete the application context. EUs can delete their data from the Edge application, and ESO can clean up outdated data to free up storage space.

6) Termination: If an edge application is no longer needed in the edge system, the ESO can terminate the application from both the Edge Platform and the Edge Platform Manager at Controller Level.

After defining application management and orchestration procedure, the Edge system can administer the applications.

B. Resource Management and Orchestration

To manage resources properly, the P1935 standard defines a service API that follows Representational State Transfer (REST) style to communicate between the components in the Edge system [17]. The resources mentioned here include both physical and virtual resources. The physical resources are the hardware equipped on the Edge system, including physical servers, facilities, and infrastructure. The virtual resources can be considered as the software, computing, networking, and storage resources in the system and the virtual infrastructure.

Service API includes five actions on resources: creation, status query, discovery, reconfiguration, and deletion.

1) Resource Creation: New resources can be created by sending an HTTP POST request to the Edge system. The POST request should contain information about the new creating resources. The new resources will be given a unique resource URL as an identifier. After creation, these Edge resources will also be configured automatically based on the request payload. Then, the resources can be split into several pieces or slices, called a "resource quota," providing higher flexibility and system utilization. Resource creation is the necessary procedure for the ESOs and ESPs to register their resources into the system, or the system will not be able to recognize them and perform other operations.

2) Resource Status Query: The resource information can be obtained by an HTTP GET request with empty payload that is available for most resources. This operation is usually used when the entity already knows the resource URL, which can be obtained via the resource discovery operation. In all cases, this operation returns the information of the target resource, and it will give corresponding error messages if the requested resource is not found in the system.

3) Resource Discovery: The resource searching with the filtering rules can be done by an HTTP GET method with query parameters that can be used to control the content of the query result. The filtering rules are the restrictions and conditions to narrow the search results. An empty list should be returned if no resource matches the filtering rules. The users and the system can use this operation to find the necessary resources.

4) Resource Reconfiguration: Resource reconfiguration means the modification of the properties or attributes of the target resource. These properties and attributes are generated based on the payload of the resource creation operation request. There are two methods to reconfigure the resource: the HTTP PUT method can overwrite the resources for resource replacement. All the old resource content will be omitted after resource replacement. On the other hand, the HTTP PATCH method is used to update a resource by only changing the part described in the payload from the client. Before the modification in both cases, the resource status query operation is performed to ensure the version of the resource to avoid race conditions.

5) Resource Deletion: The resources can be deleted by sending an HTTP DELETE request with an empty payload. After deletion, all the information about the deleted resource will be removed, and the related resources will be updated and rearranged. The system can delete the target resource based on their needs and feedback.

The Edge system can easily organize the Edge Platform with both application and resource management and orchestration procedures. According to the P1935 standard, we implement a testbed in the real world and deploy a live streaming service to evaluate its performance. Descriptions are illustrated as follows.

IV. Usage Example: Video Streaming

The P1935 standard provides a comprehensive description of application management and orchestration, which enables services to be assisted and enhanced by intelligent algorithms. To show the usage example, we deploy a live streaming service on the testbed, which is one of the promising use cases that benefit from the P1935 standard. In this section, we first

introduce our testbed design. Then, an overview of the live streaming service and a live streaming system aligned with the testbed is illustrated. Next, a real-time transcoding decision policy for edge-based live streaming services is presented.

A. Testbed Implementation

This subsection will introduce our testbed edge computing structure aligned with the P1935 standard. The system consists of three layers corresponding to the standard design: Orchestrator Level, Controller Level, and Computer Level. We use Python script to implement the Orchestrator Level with a web-based user interface and an external application programming interface (API). With these, the EFO can manage nodes in the Controller Level and record data and metadata from users, nodes and applications with an external database. On the other hand, we adopt Kubernetes and Prometheus at the Controller and Computer Levels. Kubernetes [18] has a control plane and multiple worker nodes within a cluster and has a similar structure to our standardized architecture. It manages resources and containerized services with high scalability and fault tolerance. Therefore, the Controller Level can reconfigure resources to a service when it requires more or resume a service as soon as possible when any fault happens. Besides, we define several P1935 testbed APIs that coincide with those in Kubernetes to ensure functionality and availability. Prometheus [19] is an open-source project that monitors a system and stores its parameters in a time-series database. Operators can query these time-variant data and obtain the generated ad-hoc graphs, tables and alerts with high visualization. Among these components in the system, all message flows and links follow the procedures specified in the standard.

B. Use Case Scenario

There are two main challenges for the existing live-streaming services, where the stream segments are transcoded into dynamic adaptive streaming over HTTP (DASH) or HTTP live streaming (HLS) files of multiple resolutions. First, the current mechanism transmits stream segments of different resolutions and consumes plenty of network resources. Second, streamers and viewers may have a bad user experience in a region without caching server because of poor channel quality. To this end, edge computing technology has been viewed as a promising solution to reduce backhual network traffic and provide better channel quality by placing video content on edge servers. What's more, we deploy our real-time dynamic transcoding decision on the system to solve the issue of limited computing power inherent in edge servers, which thanks to the standard providing a well-established management and orchestration framework for edge computing.

C. System Structure

The design of the live streaming services that cope with the standard differs from the existing ones. There is no central media center in the edge-based live streaming system. As shown in Fig. 3, the system is composed of several characters: The

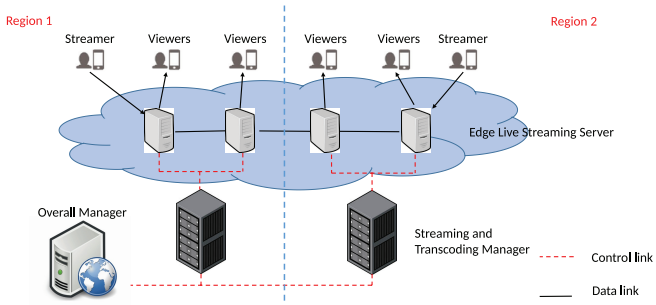


Fig. 3: The illustration of edge computing based live streaming system.

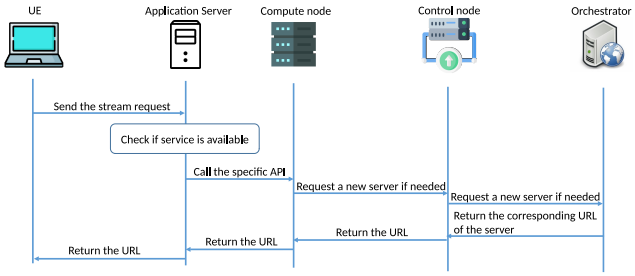


Fig. 4: The example workflow of starting a stream in the standardized edge system.

Edge Live Streaming Servers (ELSS) are deployed on the network edge at Computer Level, responsible for transcoding live streams from streamers into DASH format and caching video segments for viewers. An ELSS can be considered as being composed of an RTMP server, which facilitates communication with other entities, and a transcoder function for transcoding purposes. The **Streaming and Transcoding Manager (STM)** in Controller Level organizes all ELSSs in the region, managing streaming flows between the ELSSs and deciding the transcoding rate for each live stream. An **Overall Manager (OM)** in Orchestrator Level records the location and live streams of all ELSSs. Streamers push their live streams to the ELSS; viewers watch the specific one on demand.

In our architecture, there is a particular link between each pair of edge servers for communication, enabling edge servers to transmit live streams to others with better channel quality. As only raw RTMP streams will be transmitted to other edge servers, a live stream can be simultaneously transcoded on different ELSSs in different regions with lower backhaul traffic. If there is a need to migrate the live streams or cached segments to another ELSS, the OM will notify the respective ELSSs to establish connections and initiate data transmission. The message flows satisfy the standardized procedure. For

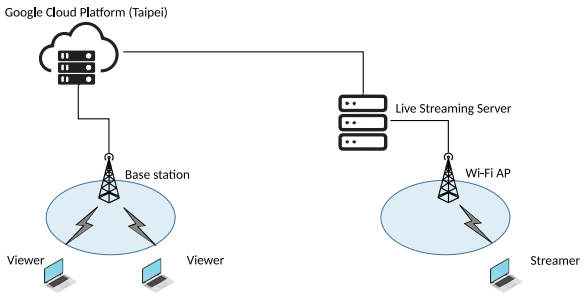


Fig. 5: The illustration of live streaming service on cloud platform.

example, the procedures for a streamer to start a stream in such a scenario are depicted in Fig. 4.

D. Real-time Dynamic Transcoding Decision

The limited computing power of edge servers compared to cloud data centers means that too many transcoding tasks can lead to severe stalls and terrible QoE degradation. To this end, we propose a real-time dynamic transcoding decision mechanism to maximize user satisfaction with the limited computing capacity of the edge servers, which can be implemented as the orchestration algorithm in the EFO. In the proposed scheme, the ELSSs will transcode the video segments into only some of the resolutions according to the transcoding decision assigned by STM if the computational cost of transcoding exceeds its computing capacity. Therefore, the computational burden on the ELSS can be mitigated.

To estimate user satisfaction, we utilize the user experience model derived in [20], which quantitates the perceived quality of video streaming. Accordingly, the user experience model can be expressed as

$$QoE = 100 - I_{ID} - I_{ST} - I_{QV}, \tag{1}$$

where I_{ID} indicates the influence of initial delay; I_{ST} represents the impairment caused by stalling; I_{QV} specifies the impairment caused by quality variation. The complete definitions of these terms are included in [20].

The objective of the real-time dynamic transcoding scheme is to maximize the user experience with the limited resources. Given the user experience model, the problem formulation can be written as

$$\begin{aligned} \max_{\mathbf{D}} \quad & \sum_{i=1}^{|S|} \sum_{j=1}^{|N_i|} \sum_{k=1}^{|R_{i,j}|} QoE_{i,j,k} \cdot d_{i,k} \\ \text{s.t.} \quad & \sum_{k=1}^{|R_{i,j}|} d_{j,k} \leq 1 \ \forall i, j, \\ & \sum_{i=1}^{|S|} \sum_{k=1}^{|R|} c_{i,k} d_{i,k} \leq C, \ d_{i,k} \in \{0,1\} \ \forall i, k. \end{aligned} \tag{2}$$

where $S = \{1, \ldots, |S|\}$ is the set of streams; N_i is the set of the viewers who request for stream i; R is the set of available resolutions in an ascending order; $R_{i,j}$ is the set of candidate resolutions for stream i that is requested by viewer j; $\mathbf{D} = \{d_{i,k} \ \forall i, k\}$ are indicator variables of transcoding decision. $d_{i,k} = 1$ when stream i with resolution k will be transcoded; otherwise, $d_{i,k} = 0$. $c_{i,k}$ is the cost for transcoding stream i with resolution k, and C is the computing capacity of the edge server. Note that the total computational cost can not exceed the computing capacity.

To solve this problem, we adopt exhaustive search to find the optimal solution as the STM will iteratively calculate the overall QoE of all possible decisions and assign the best solution of each stream to the ELSS. Due to the limited options, the algorithm can find the best solution within each streaming segment time, which is set to 4 seconds by default.

V. Experimental Evaluation

In this section, we evaluate the performance of the proposed mechanism on a real testbed with the streaming services mentioned in the previous section.

A. Experimental Environments

On the Computer Level of our architecture, we onboard and instantiate a live streaming service composed of an HTTP server and a real-time messaging protocol (RTMP) server built based on the "rtmp-nginx" module to receive live streaming from streamers. We use a laptop with Open Broadcaster Software (OBS) as a live streamer to push the RTMP stream to the RTMP server, which then adopts FFMPEG to convert the media content into the DASH format and start the process of transcoding. Finally, the viewers will generate and download the DASH segments and the M3U8 files via the HTTP server. For simplicity, the computing capacity is set to be 1, and the computational costs of different resolutions and bitrates are normalized between 0 and 1 as follows based on their transcoding time: a) 1080P with bitrate 2.50 mbps: 0.37, b) 720P with bitrate 1.10 mbps: 0.19, c) 480P with bitrate 0.42 mbps: 0.10, d) 360P with bitrate 0.28 mbps: 0.07, e) 240P with bitrate 0.10 mbps: 0.05.

To consider different loading conditions on the edge server, we consider the following three schemes:

- **Fully transcoding one stream**: We consider only one stream as the situation of low computational load. On the edge server, the stream is transcoded into five resolutions (240P, 360P, 480P, 720P, 1080P).
- **Fully transcoding two streams**: On the edge server, two streams are both transcoded into five resolutions as above, regarded as high computation load situation.
- **Proposed scheme**: To show the effectiveness of the proposed mechanism, the proposed scheme considers two streams, and the edge server transcodes each stream according to its assigned resolution.

Additionally, to demonstrate the effectiveness of an edge-computing-based live streaming system, we also present a scenario involving **streaming from a cloud platform**. This is

achieved by deploying the live streaming server on the Google Cloud Platform, as depicted in Figure 5. By launching the server in Taipei, we ensure its performance is comparable to the traditional CDN-based approach. In our laboratory experiments with other schemes, we restrict the uplink bandwidth to 20 Mbps to mimic real-world conditions where the uplink bandwidth often serves as the bottleneck for streamers.

B. Evaluations

We evaluate the measurement results on our edge computing testbed in terms of the three factors of the impairment and the overall QoE.

As we can observe in Fig. 6a, the three scenarios for streaming from the edge server performed quite similarly and better than the cloud platform in terms of initial delay, where the proposed scheme has a longer initial delay due to the additional time cost of the real-time transcoding decision. In Fig. 6b, the impairment of stalling is over 100 when we choose fully transcode two streams on the edge server. In this case, the video stalls over half of the total experiment time and causes a terrible user experience. There is almost no stalling for the proposed mechanism and fully transcoding one stream, and some stalling for streaming from the cloud platform due to the channel quality differences. In Fig. 6c, since the case of fully transcoding two streams stalls for over half of the experiment time, the impairment of quality variation is over 100. In the case of fully transcoding one stream, the impairment is 0, as the clients can always obtain 1080P video segments since the edge server provides outstanding network quality. In our proposed mechanism, some clients can only get video segments with a slightly worse resolution to provide stable service under limited computing capacity. On the other hand, streaming from the cloud platform causes worse impairment because of the poorer channel quality compared to the edge server. Therefore, the proposed mechanism performs better than other schemes regarding the overall QoE, as shown in Fig. 7. That is to say, the live streaming with the P1935-based edge computing testbed provides almost perfect service quality in the low load situation, and the system with the proposed real-time transcoding decision policy can still provide good service quality even in the high load situation.

VI. Conclusions and Future Work

This paper presents the P1935 Edge/Fog system's architecture and describes the components' functions and relationships. The system is a three-level scheme where the Orchestration Level is responsible for the control of the whole system; the Controller Level aims to manage the resource and the application lifecycle on the edge nodes; the Computer Level handles the computational tasks. Some external entities help the system connect to the external environment. These components indicate a thorough management and orchestration scheme of the system. We show the lifecycle of service orchestration and resource management and their details in each step. Finally, we deploy the live-streaming service to the real P1935-aligned testbed with the real-time transcoding decision. The evaluation results prove that the system can provide stable services even when the system is in a heavy load situation.

With the Internet and telecommunication development, users have higher demands for quality, reliability, and security. Therefore, our desired goal is to provide these functionalities based on the P1935 standard. Although this paper only provides a simple testbed use case, we are currently developing a system that can support higher-resolution videos (e.g., 4K videos) and accommodate more stringent requirements. We also plan to conduct experimental comparisons between other edge systems and our future testbed designs, highlighting the advantages of this system in resource management and application management.

References

[1] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[2] Y. Chiang, Y. Zhang, H. Luo, T.-Y. Chen, G.-H. Chen, H.-T. Chen, Y.-J. Wang, H.-Y. Wei, and C.-T. Chou, "Management and orchestration of edge computing for iot: A comprehensive survey," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[3] S. Pan, Z. Zhang, Z. Zhang, and D. Zeng, "Dependency-aware computation offloading in mobile edge computing: A reinforcement learning approach," *IEEE Access*, vol. 7, pp. 134 742–134 753, 2019.

[4] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[5] Y. Chiang, C.-H. Hsu, G.-H. Chen, and H.-Y. Wei, "Deep q-learning based dynamic network slicing and task offloading in edge network," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[6] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Qos-aware mobile edge computing system: Multi-server multi-user scenario," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–6.

[7] ——, "Task offloading and resource allocation in mobile-edge computing system," in *2018 27th Wireless and Optical Communication Conference (WOCC)*, 2018, pp. 1–4.

[8] H. Luo and H.-Y. Wei, "Resource orchestration at the edge: Intelligent management of mmwave ran and gaming application qoe enhancement," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[9] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[10] C. Song, M. Zhang, Y. Zhan, D. Wang, L. Guan, W. Liu, L. Zhang, and S. Xu, "Hierarchical edge cloud enabling network slicing for 5g optical fronthaul," *Journal of Optical Communications and Networking*, vol. 11, no. 4, pp. B60–B70, 2019.

[11] A. Younis, T. X. Tran, and D. Pompili, "On-demand video-streaming quality of experience maximization in mobile edge computing," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019, pp. 1–9.

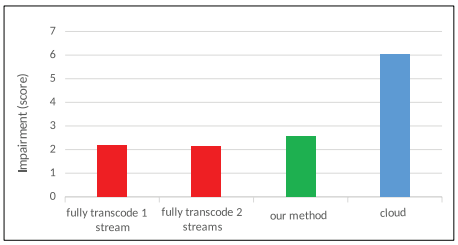
[12] Y. Chiang, C.-H. Hsu, and H.-Y. Wei, "Collaborative social-aware and qoe-driven video caching and adaptation in edge network," *IEEE Transactions on Multimedia*, vol. 23, pp. 4311–4325, 2021.

[13] Y. Tan, C. Han, M. Luo, X. Zhou, and X. Zhang, "Radio network-aware edge caching for video delivery in mec-enabled cellular networks," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018, pp. 179–184.

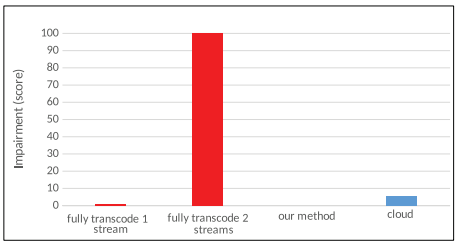
[14] ETSI, "Multi-access edge computing (mec); framework and reference architecture," *ETSI GS MEC 003*, vol. 2.2.1, 2020.

[15] IEEE P1935 WG. Standard for edge/fog manageability and orchestration. [Online]. Available: https://standards.ieee.org/project/1935.html

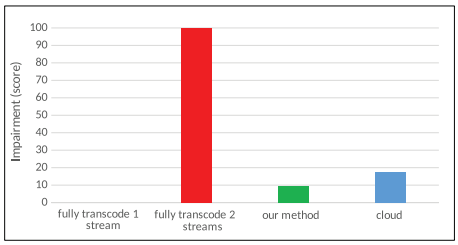
[16] ——. Standards for p1935. [Online]. Available: https://standardscollection.ieee.org/



(a) The impairment of initial delay.



(b) The impairment of stalls.



(c) The impairment of quality variation.

Fig. 6: Measurement results of the three scenarios on our edge computing testbed compared to the cloud platform.

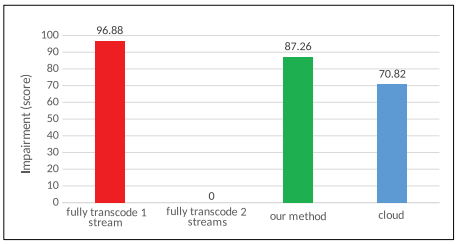


Fig. 7: The values of QoE of the three scenarios compared to the cloud platform.

[17] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[18] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[19] B. Rabenstein and J. Volz, "Prometheus: A next-generation monitoring system (talk)." Dublin: USENIX Association, May 2015.

[20] Y. Liu, S. Dey, D. Gillies, F. Ulupinar, and M. Luby, "User experience modeling for dash video," in *2013 20th International Packet Video Workshop*, 2013, pp. 1–8.