Low-Latency Computation Offloading based on 5G Edge Computing Systems

Zhen-Yuan Pan*, Jiann-Liang Chen* and Yao-Chung Chang**
*Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan.
**Department of Computer Science and Information Engineering, National Taitung University, Taitung, Taiwan.
Lchen@mail.ntust.edu.tw, ycc@nttu.edu.tw

***Abstract*- This study proposed a Low-latency Services Offloading Policy based on a Greedy algorithm, called LSOPG, for 5G edge computing. LSOPG algorithm inherits the advantages of low complexity and high efficiency of the greedy algorithm and improves the frequency of congestion caused by queuing between users, finds the best balance between latency and load balancing. Compared with the previous study, when there are 20 users and the service type is1080P@60fps video streaming, the proposed LSOPG policy can improve about 1.92% latency and 22.25% packet loss rate. When the number of users is increased to 50, it improves about 2.41% latency and 35.32% packet loss rate. This experimental result confirms that the LSOPG policy can provide well service quality in 5G networking.**

Keywords- Multi-access Edge Computing, Edge Network, Offloading Policy, 5G, Games as a Service

I. INTRODUCTION

With the rapid growth of Information and Communications Technology (ICT) and IoT devices, the 2018 Google survey report shows that users of the 4th generation mobile networks (4G) service have continuously increased their latency requirements. If the waiting time exceeds 3 seconds, most users will choose to abandon the service[1]. In addition, when using Virtual Reality (VR), if the image cannot keep up with the user to change the direction of view, it is easy to cause dizziness. A 2018 Universitas Ouluensis research found that the delay rate needs to be less than 10 milliseconds to achieve an ideal VR experience [2]. To respond to the needs of users for mobile communication networks, the International Telecommunication Union (ITU) defined the 5th generation mobile networks (5G) specification, and the 3rd Generation Partnership Project (3GPP) proposed the 5G standard (3GPP Release-15) [3]. 5G and edge computing reduce the time for data to and from the cloud, significantly improve the service quality of the virtual experience, and improve the problem of delay affecting the experience.

Compared with traditional services, users of GaaS cannot obtain all the data of the application, so it can effectively reduce the problems of cracked version, data modification, and hardware update mentioned above [5]. However, GaaS has very high requirements for network connection speed and latency. On the other hand, 4G networks cannot meet users' needs for bandwidth, many connections, and latency. Therefore, 5G networks have been proposed to meet various needs. Type of service requirement. 5G adopts the Mobile Edge Computing (MEC) architecture, which can effectively reduce the time for data to enter and exit the cloud, significantly improve the service quality of GaaS and improve the problem of latency affecting the experience. However, the edge server does not have the substantial computing resources of the cloud server. Therefore, this study proposed the low-latency computation offloading based on a 5G edge computing system, dynamically configuring the network service's execution environment, improving the overall operating efficiency of the network service.

This study proposed the LSOPG algorithm to improve the quality of GaaS. In this study, Data Collection mainly collected UE (User Equipment), MEC, and cloud resource information. According to the Data Collection report, Services Simulation Model can simulate the service transmission and execution latency. According to simulation results, Offload Model can avoid service congestion caused by queuing and waiting and get the best offloading policy.

II. RELATED WORK

GaaS was proposed in 2000. It connects to a GaaS server and remotely controls video games. Traditional online gaming clients need to be responsible for application storing, rendering, and action capturing, while Game logic operates on the online gaming server, as shown in Figure 1. Since rendering works on the client, it requires higher hardware specifications, such as Graphics Processing Unit (GPU) and Central Processing Unit (CPU); GaaS moves application storing and rendering to a dedicated server for processing. The client only needs to have multimedia decoding. Since the computing work is not processed on the client, there is no need to worry about incompatibility between the operating system. The application or the problem of hardware upgrades is also since the type of data transmitted by the application has changed from metadata to multimedia, which has dramatically increased the application's network requirements [6]. Currently, well-known open-source GaaS architectures are Gaming Anywhere [7], Rainway [8], and Moonlight-Stream [9]. To measure and improve the GaaS as mentioned above architecture, many researchers have proposed various measurement methods in the past [10, 11].

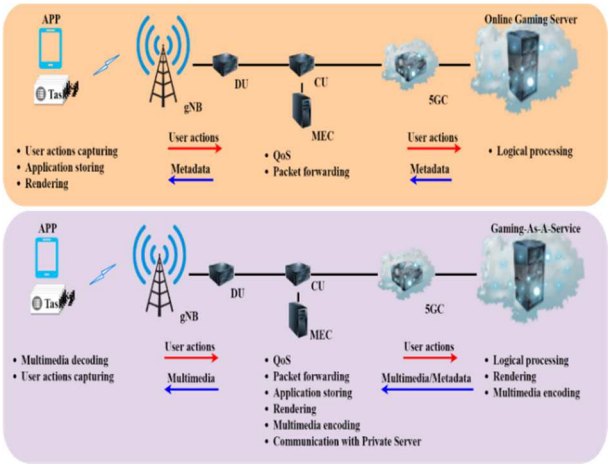

Figure 1. Online gaming and GaaS

Optimized for UE energy consumption, such as Ref. [12], minimize the average energy consumption of the system for computing-intensive and data-intensive multimedia services. In Ref. [13], proposed a cooperative offloading technique based on the Lagrangian Suboptimal Convergent Computation Offloading

Algorithm for multi-access MEC in a distributed Internet of Things network, find out smart communicating devices' optimal computational velocity and transmit power allocation. In Ref. [14], proposed a Multi-BS Computing Cost Optimization based on a Genetic Algorithm.

Optimized for E2E latency, such as Ref. [15] proposed a low-complexity PSO-Based Algorithm and Greedy Algorithm based on the number of UEs and the requirements of each application for communication, computing, and storage resources. Ref. [16] uses an efficient Lagrangian Relaxation heuristic algorithm to reduce the complexity of the model. Ref. [17] uses a fuzzy decision to select the best target node for offloading. Ref. [18,19] reduces application offloading latency, offloading costs and achieves load balancing of edge nodes. Ref. [20,21] enable nearby MECs to share resources, reducing E2E latency. Ref. [22] proposed an efficient collaborative task offloading scheme. There are two schemes, one can reduce the proportion of task failures, and the other can reduce the latency and execute many tasks.

To solve the joint optimization problem of task offloading and resource allocation in a short time, Ref. [23] uses a deep neural network, and Ref. [24] uses a deep reinforcement learning algorithm, predict the optimal computation offloading decision and resource allocation. Ref. [25-28] use different algorithms to optimize the task completion time and UE energy consumption. Ref. [29] considers MEC computing resource allocation, channel allocation, and the user uploads power control. Ref. [30] formulated the task scheduling and resource allocation problem as an NP-hard non-convex mixed-integer problem and proposed a hyper-graph-based channel pre-allocation algorithm to find the best solution and save bandwidth resources. Ref. [31] used a Decision Tree to classify highly sensitive and Low-sensitive latency cloud gaming. Ref. [32] used Random Forest to classify 6 different types of game videos.

III. PROPOSED LSOPG POLICY

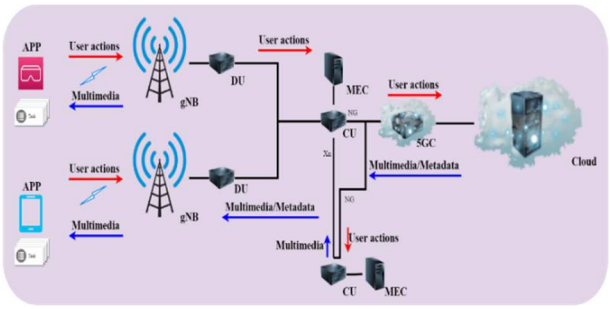

Figure 2. Proposed LSOPG policy with 5G edge computing system

This study mainly finds the low-latency computation offloading decision among the limited MEC and cloud computing resources. Figure 2 describes the Low-Latency Computation Offloading based on the 5G Edge Computing System. First, the UE establishes a connection with the GaaS server, and the MEC obtains resource information of the UE, the MEC of the collaboration space, and the cloud. Then the UE sends the action to the GaaS server for logical processing. After processing, a set of T of computation tasks will be generated and returned to the MEC for simulation and find a low latency offloading policy. The computation task will be sent to the location designated by the offloading decision for execution. The processed data will return to the MEC and encoded into a real-time video stream to the UE. Finally, the UE decodes and displays the real-time video stream. UE and gNB are connected by 5G NR, and Xn converges the CU (MEC) in the collaborative space. The cloud can provide more resources with the CU (MEC) through 5GC is connected by fiber. The architecture of this study is divided into three parts: Data Collection, Services Simulation Model, and Offload Model.

A. Data Collection

This section introduces the 3 collectors for Data Collection. It collects UE, MEC, and cloud resource information, including CPU, GPU, RAM, computation workload. The collected information will be used as the input data of the Services Simulation Model.

1) UE Resource Information Collector

The UE Resource Information Collector module mainly collects UE resources information, including the CPU, GPU, RAM, and computation workload. The resources information is used to simulate the processing time required to execute the services in the UE. The time sequence of the UE Resource Information Collector is shown in Figure 3.

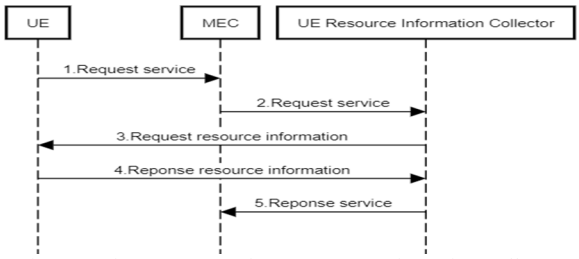

Figure 3. Time sequence of UE Resource Information Collector

2) MEC Resource Information Collector

MEC Resource Information Collector module mainly sends a resource information request to MEC Hosting Infrastructure Management System to real-time collect all MEC resources information, including the MEC-ID, CPU, GPU, RAM, computation workload, location, and resource utilization status [33]. The resource information is used to simulate the processing time required to execute the service in the MEC and the resource usage rate of each MEC, the time sequence of MEC Resource Information Collector as shown in Figure 4.

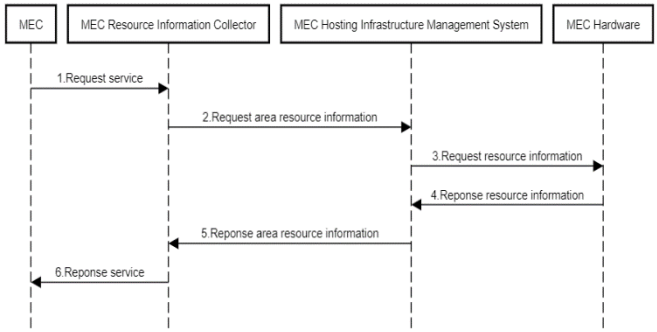

Figure 4. Time sequence of MEC Resource Information Collector

3) Cloud Resource Information Collector

Cloud Resource Information Collector module mainly sends a resource information request to Cloud Management Platform, then requests information about computing resources from Cloud Resource Management to real-time collect Cloud resources information, including the CPU, GPU, RAM, Computation workload, and location. The time sequence of Cloud Resource Information Collector is shown in Figure 5.

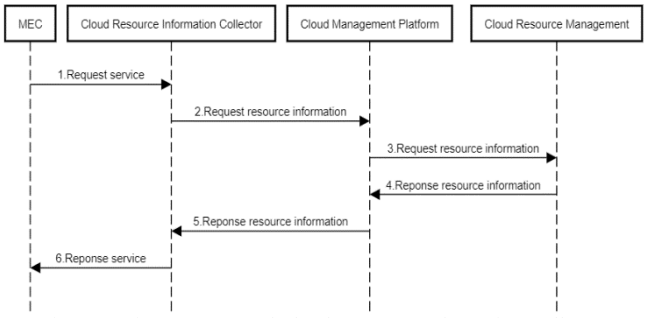

Figure 5. Time sequence of Cloud Resource Information Collector

B. *Services Simulation Model*

This section introduces 2 simulators of the Service Simulation Model. Transmission Latency Simulator is used to simulate the transmission latency. And the Execution Latency Simulator is used to simulate the service execution latency of the service in UE, MEC, and cloud. The parameters used in this study are summarized in Table 1.

Table 1. Parameters

Parameter	Description
U	Number of UE
i	Denote the ith UE
T	Number of computation tasks
j	Denote the jth computation task
$D_{i,j}^{in}$	The Input data size of the $T_{i,j}$
$D_{i,j}^{p}$	The Logically processed data size of the $T_{i,j}$
$D_{i,j}^{\text{out}}$	The Output data size of the $T_{i,j}$
$D_{i,j}^{len}$	The data length of the $T_{i,j}$
F_i^{UE}	Computation resource of the ith UE
F_i^{MEC}	Computation resource of the MEC assigned to the ith UE
F_i^{OMEC}	Computation resource of the other MEC assigned to the ith UE
F_i^{Cloud}	Computation resource of the cloud assigned to the ith UE
R_i^{U}	The Uplink data rate of the ith UE
R_i^{D}	The Downlink data rate of the ith UE
R^{Xn}	Xn link data rate between the MEC and the other MEC
R^{Fiber}	Link data rate between MEC and cloud
B_i^{U}	Uplink channel bandwidth allocated to the ith UE
B_i^{D}	Downlink channel bandwidth allocated to the ith UE
$EL_{i,j}^{UE}$	Execution latency for $D_{i,j}^{p}$ at the ith UE
$EL_{i,j}^{MEC}$	Execution latency for $D_{i,j}^{p}$ at the MEC
$EL_{i,j}^{OMEC}$	Execution latency for $D_{i,j}^{p}$ at the other MEC
$EL_{i,j}^{Cloud}$	Execution latency for the $D_{i,j}^{p}$ at the cloud
$TL_{i,j}^{UE2MEC}$	Transmission latency for $D_{i,j}^{in}$ from the ith UE to the MEC
$TL_{i,j}^{MEC2UE}$	Transmission latency for $D_{i,j}^{\text{out}}$ from the MEC to the ith UE
$TL_{i,j}^{MEC2OMEC}$	Transmission latency for $D_{i,j}^{p}$ from the MEC to the other MEC
$TL_{i,j}^{OMEC2MEC}$	Transmission latency for $D_{i,j}^{\text{out}}$ from the other MEC to the MEC
$TL_{i,j}^{MEC2Cloud}$	Transmission latency for $D_{i,j}^{p}$ from the MEC to the cloud
$TL_{i,j}^{Cloud2MEC}$	Transmission latency for $D_{i,j}^{\text{out}}$ from the cloud to the MEC
$E2E_{i,j}^{MEC}$	End to End latency for finish the $T_{i,j}$ at the MEC
$E2E_{i,j}^{OMEC}$	End to End latency for finish the $T_{i,j}$ at the other MEC
$E2E_{i,j}^{Cloud}$	End to End latency for finish the $T_{i,j}$ at the cloud
$\tau_{i,j}$	The Completion deadline for finish the $T_{i,j}$
U^{MEC}	Utilization of the MEC
U^{OMEC}	Utilization of the other MEC
$\alpha_{i,j}$	Offloading decision value denotes $D_{i,j}^{p}$ offloading to the MEC
$\beta_{i,j}$	Offloading decision value denotes $D_{i,j}^{p}$ offloading to the other MEC
$\gamma_{i,j}$	Offloading decision value denotes $D_{i,j}^{p}$ offloading to the cloud

1) Transmission Latency Simulator

Offloading computing tasks from the MEC to other servers, the network will incur a communication cost. When it is offloading to different locations, the calculation method of its communication cost is also different. This section will introduce three offloading scenarios, and then transmitted to the UE is connected by 5G NR, as shown in Figure 6 (Refer to 3GPP TS 38.306 V15.13.0).

- Scenario 1 processing at the MEC_A is the lowest latency, and after the tasks are processed, the data are transmitted to UE is connected by 5G NR.
- Scenario 2 processing at the MEC_B is the lowest latency, the tasks are transmitted from the MEC_A to the MEC_B are connected by Xn for processing. After the tasks are processed, the data is returned from the MEC_B to the MEC_A is connected by Xn, and then transmitted to UE is connected by 5G NR.
- Scenario 3 processing at cloud is the lowest latency, the tasks are transmitted from the MEC_A to the cloud are connected by fiber for processing. After the tasks are processed, the data is returned from the cloud to the MEC_A is connected by fiber.

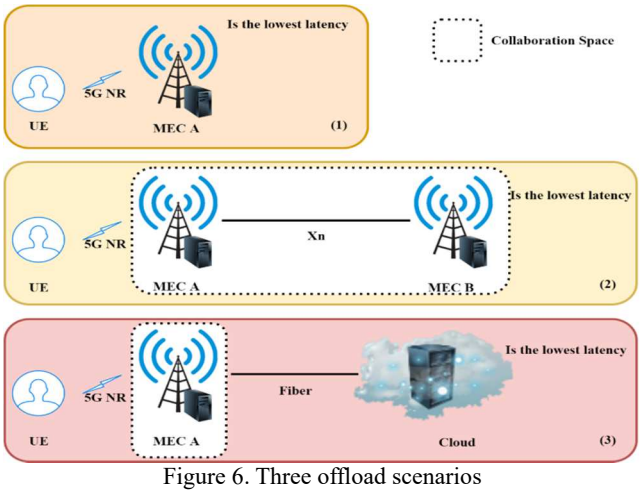

Figure 6. Three offload scenarios

2) Execution Latency Simulator

In this study, it was assumed that each UE application i has a task j that needs to use the computation resource. Therefore, the computation of task j requires execution time $EL_{i,j}^{UE}$, the execution latency at UE is expressed as:

$$EL_{i,j}^{UE} = \frac{D_{i,j}^{len}}{F_i^{UE}} \quad (1)$$

Where F_i^{UE} denotes the computation workload of the i^{th} UE, $D_{i,j}^{len}$ denote the data length of the $T_{i,j}$, F_i^{UE} denote the computation resource of the i^{th} UE.

- Scenario 1 execution latency at the $\mathrm{MEC_A}$ is expressed as:

$$EL_{i,j}^{MEC} = \frac{D_{i,j}^{len}}{F_i^{MEC}} \quad (2)$$

Where F_i^{MEC} denotes the computation resource of the MEC assigned to the i^{th} UE.

- Scenario 2 execution latency at the $\mathrm{MEC_B}$ is expressed as:

$$EL_{i,j}^{OMEC} = \frac{D_{i,j}^{len}}{F_i^{OMEC}} \quad (3)$$

Where F_i^{OMEC} denotes the computation resource of the other MEC assigned to the i^{th} UE.

- Scenario 3 execution latency at cloud is expressed as:

$$EL_{i,j}^{Cloud} = \frac{D_{i,j}^{len}}{F_i^{Cloud}} \quad (4)$$

Where F_i^{Cloud} denotes the computation resource of the cloud assigned to the i^{th} UE.

C. Offload Model

This section introduces the design of the low latency services offloading policy based on the greedy algorithm.

- Scenario 1 total latency is expressed as:

$$E2E_{i,j}^{MEC} = TL_{i,j}^{UE2MEC} + EL_{i,j}^{MEC} + TL_{i,j}^{MEC2UE} \quad (5)$$

- Scenario 2 total latency for offloading a task from the $\mathrm{MEC_A}$ to the $\mathrm{MEC_B}$ is expressed as:

$$E2E_{i,j}^{OMEC} = TL_{i,j}^{UE2MEC} + TL_{i,j}^{MEC2OMEC} + EL_{i,j}^{OMEC} + TL_{i,j}^{OMEC2MEC} + TL_{i,j}^{MEC2UE} \quad (6)$$

- Scenario 3 total latency for offloading a task from the $\mathrm{MEC_A}$ to the cloud is expressed as:

$$E2E_{i,j}^{Cloud} = TL_{i,j}^{UE2MEC} + TL_{i,j}^{MEC2Cloud} + EL_{i,j}^{Cloud} + TL_{i,j}^{Cloud2MEC} TL_{i,j}^{MEC2UE} \quad (7)$$

The computation offloading problem is formulated as the following constrained optimization formulation problem:

$$Latency_{i,j} = \alpha_{i,j} \times E2E_{i,j}^{MEC} + \beta_{i,j} \times E2E_{i,j}^{OMEC} + \gamma_{i,j} \times E2E_{i,j}^{Cloud} \quad (8)$$
$$Latency_{i,j} \leq \tau_{i,j} \; Latency_{i,j} \leq EL_{i,j}^{UE}$$

The objective function of the optimization problem is to minimize the cost of the entire system in terms of time through the deployment of task offloading. Figure 7 presents the design of the low latency services offloading policy based on the greedy algorithm.

Algorithm Low Latency Services Offloading Policy based on Greedy Algorithm

Input: UE actions
Output: offloading decision values $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}$

Initialization: $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}$
For *UE* = 1 to *i* **do**
  Collect the UE, MEC, and cloud resource information;
  Calculate the uplink and downlink data rate R_i^U and R_i^D according to Equation (1);
  Generate the data of computation task after GaaS server logical processing;
  For task = 1 to *j* **do**
    Simulate the transmission latency $TL_{i,j}^{UE2MEC}$, $TL_{i,j}^{MEC2UE}$, $TL_{i,j}^{MEC2OMEC}$, $TL_{i,j}^{OMEC2MEC}$, $TL_{i,j}^{MEC2Cloud}$, and $TL_{i,j}^{Cloud2MEC}$ according to Equation (2), (3), (4), (5), (6), and (7);
    Simulate the execution latency $EL_{i,j}^{UE}$, $EL_{i,j}^{MEC}$, $EL_{i,j}^{OMEC}$, and $EL_{i,j}^{Cloud}$ according to Equation (8), (9), (10), and (11);
    Solve the problem in Equation (15) and update the low-latency computation offloading decision values.
  end For
end For

Figure 7. Pseudocode of the proposed LSOPG policy

IV. SYSTEM PERFORMANCE ANALYSIS

The system environment, performance analysis, and summary of performance are described as follows. In this experiment, the results of the LSOPG algorithm are analyzed. Finally, the simulation results are compared based on the LSOPG algorithm, the default algorithm, and the greedy algorithm.

A. System Environment

This work performance is simulated on NS-3 and an ETSI-MEC compliant NS-3 module. The simulation parameters are shown in Table 2. The time UE sends service requests and uses resources is modeled through memory-less Poisson. The simulation time is equal to 0.5 hours.

B. Performance Analysis

This section introduces the low latency services offloading policy based on the greedy algorithm (LSOPGA) and the greedy algorithm (GA) proposed by A. Mseddi et al. [15], using 720P@60fps, 1080P@60fps, and 4K@60fps, 3 types of service cases to compare the effectiveness of each algorithm. This experimental case mainly compares MEC performance with the LSOPGA and the GA. This testing environment has a cloud server, the main MEC, 3 regionals cooperative MECs, and 20 UEs.

Table 2. Simulation parameters

Parameters		Value
F_i^{UE}	CPU	50,000 MIPS
	RAM	8 GB
	Storage	32 GB
F_i^{MEC}	CPU	400,000 MIPS × 4
	RAM	64 GB
	Storage	1024 GB
F_i^{OMEC}	CPU	400,000 MIPS × 4
	RAM	64 GB
	Storage	1024 GB
F_i^{Cloud}	CPU	500,000 MIPS × 20
	RAM	1024 GB
	Storage	1024 × 1024 GB
J		1
$v_{Layers}^{(j)}$	Maximum number of supported MIMO layers	8
	Number of Beam with MU–MIMO Users	4
$Q_m^{(j)}$		8
$f^{(j)}$		1
R_{max}		$\frac{948}{1024}$
$N_{PRB}^{BW(j),\mu}$		78
μ		1
T_s^{μ}		0.00003571428572
$OH^{(j)}$		0.14

Table 3 shows the comparison results of the LSOPGA and GA when the number of UEs is fixed. The average marks of 10 experiments show that the LSOPGA is suitable for 720P@60fps services. To use 1080P@60fps services, it is necessary to increase the hardware resources of MEC and increase the upper limit of the resources that each service can use. 4K@60fps services are entirely inapplicable.

Table 3. Performance analysis (UEs are fixed)

	720P@60fps (deadline:10ms)	1080P@60fps (deadline:10ms)	4K@60fps (deadline:20ms)
Average E2E Latency (With LSOPGA)	5.3ms	7.65ms	13.201ms
Average E2E Latency (With GA)	5.7ms	7.8ms	13.218ms
Comparison	**7%**	**1.92%**	**0.12%**
Average Packet Loss Rate (With LSOPGA)	9%	41.39%	43.163%
Average Packet Loss Rate(With GA)	35%	63.64%	43.046%
Comparison	**26%**	**22.25%**	**-0.117%**

Table 4 shows the analysis results of the LSOPGA and the GA when the service is fixed. The average marks of 10 experiments show that when UEs are 10 or 50, it only slightly affects the packet loss rate. When the priority offloading order of UE services is MEC, and MEC calculation resources can process UE services immediately, the LSOPGA can effectively improve the E2E latency and packet loss rate.

Table 4. Performance analysis (service is fixed)

	10 UEs (deadline:10ms)	20 UEs (deadline:10ms)	50 UEs (deadline:10ms)
Average E2E Latency (With LSOPGA)	7.6ms	7.65ms	7.68ms
Average E2E Latency (With GA)	7.64ms	7.8ms	7.87ms
Comparison	**0.52%**	**1.92%**	**2.41%**
Average Packet Loss Rate (With LSOPGA)	22.5%	41.39%	43.67%
Average Packet Loss Rate (With GA)	45.06%	63.64%	78.99%
Comparison	**22.56%**	**22.25%**	**35.32%**

V. CONCLUSIONS

This study proposed the low-latency computation offloading based on a 5G edge computing system. First, the system collects UE, MEC, and Cloud resources information through UE Resource Information Collector, MEC Resource Information Collector, and Cloud Resource Information Collector. The second module, through Transmission Latency Simulator and Execution Latency Simulator, simulates the service transmission and execution latency. Finally, the simulation results use an intelligent algorithm to avoid service congestion caused by queuing and waiting and get the best offloading policy. To verify the reliability of the LSOPG algorithm, compared with the previous study, when there are 20 users, the proposed method can improve about 7% E2E latency and 26% packet loss rate when the service type is 720P@60fps video streaming. When the service type is 1080P@60fps video streaming, it improves about 1.92% E2E latency and 22.25% packet loss rate. When the number of users is increased to 50, and the service type is 1080P@60fps video streaming, it can improve about 2.41% E2E latency and 35.32% packet loss rate. It means the services which execute at MEC can get better performance after with the LSOPG algorithm.

REFERENCES

[1] D. An, "Find out how you stack up to new industry benchmarks for mobile page speed," Retrieved from https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/ (last visited on 2021/04/28)

[2] M.S. Elbamby, C. Perfecto, M. Bennis and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, Vol.32, No.2, pp.78-84, 2018.

[3] 3GPP, "3GPP Release-15," Retrieved from https://www.3gpp.org/(last visited on 2021/05/01)

[4] Ericsson, "Ericsson Mobility Report 2020," Retrieved from https://www.ericsson.com/en/mobility-report (last visited on 2021/04/30)

[5] W. Cai, M. Chen and V.C.M. Leung, "Toward Gaming as a Service," *IEEE Internet Computing*, Vol.18, No.3, pp.12-18, 2014.

[6] J. Roach and K. Parrish, "What is cloud gaming?" Retrieved from https://www.digitaltrends.com/gaming/what-is-cloud-gaming-explained/ (last visited on 2021/05/04)

[7] C.Y. Huang, C.H. Hsu, Y.C. Chang and K.T. Chen, "GamingAnywhere: An open cloud gaming system," *Proc. ACM 4th Conf. Multimedia Syst. (MMSys)*, pp.36-47, 2013.

[8] Rainway, Retrieved from https://rainway.com/ (last visited on 2021/05/04)

[9] Moonlight, Retrieved from https://moonlight-stream.org/ (last visited on 2021/05/04)
[10] K.T. Chen, Y.C. Chang, H.J. Hsu, D.Y. Chen, C.Y. Huang and C.H. Hsu, "On the Quality of Service of Cloud Gaming Systems," *IEEE Transactions on Multimedia*, Vol.16, No.2, pp.480-495, 2014.
[11] O.S. Peñaherrera-Pulla, C. Baena, S. Fortes, E. Baena and R.Barco, "Measuring Key Quality Indicators in Cloud Gaming: Framework and Assessment Over Wireless Networks," *Sensors*, Vol.21, No.4, pp.1387, 2021.
[12] Y. Sun, T. Wei, H. Li, Y. Zhang and W. Wu, "Energy-Efficient Multimedia Task Assignment and Computing Offloading for Mobile Edge Computing Networks," IEEE Access, Vol.8, pp.36702-36713, 2020.
[13] J.H. Anajemba, T. Yue, C. Iwendi, M. Alenezi and M. Mittal, "Optimal Cooperative Offloading Scheme for Energy Efficient Multi-Access Edge Computation," IEEE Access, Vol.8, pp.53931-53941, 2020.
[14] W. Li et al., "Computing Cost Optimization for Multi-BS in MEC by Offloading," *Mobile Netw Appl*, 2020.
[15] A. Mseddi, W. Jaafar, H. Elbiaze and W. Ajib, "Joint Container Placement and Task Provisioning in Dynamic Fog Computing," IEEE Internet of Things Journal, Vol.6, No.6, pp.10028-10040, 2019.
[16] M. Amiri, H.A. Osman, S. Shirmohammadi and M. Abdallah, "Toward delay-efficient game-aware data centers for cloud gaming," ACM Trans. Multimedia Comput. Commun. Appl., Vol.12, No.5, pp.1-19, 2016.
[17] D. Hossain, T. Sultana, A. Hossain, I. Hossain, L.N.T. Huynh, J. Park and H. Eui-Nam, "Fuzzy Decision-Based Efficient Task Offloading Management Scheme in Multi-Tier MEC-Enabled Networks," Sensors, Vol.21, No.4, pp.1484, 2021.
[18] D. Hossain, A. Layek, T. Sultana, A. Hossain, I. Hossain, W. Rahman, Y.H. Kim and H. Eui-Nam, "Orchestration-Based Task Offloading for Mobile Edge Computing in Small-Cell Networks," Proceedings of the International Joint Conference on Computational Intelligence, pp.629-641, 2020.
[19] S. Wan, X. Li, Y. Xue, W. Lin and X. Xu, "Efficient computation offloading for Internet of Vehicles in edge computing-assisted 5G networks," J. Supercomput., Vol.76, No.4, pp.2518-2547, 2020.
[20] A. Ndikumana, N.H. Tran, T.M. Ho, Z. Han, W. Saad, D. Niyato,and C.S. Hong "Joint Communication, Computation, Caching, and Control in Big Data Multi-Access Edge Computing," IEEE Transactions on Mobile Computing, Vol.19, No.6, pp.1359-1374, 2020.
[21] P. Zhou et al., "5G MEC computation handoff for mobile augmented reality," *arXiv*, 2021.
[22] D. Hossain, L.N.T. Huynh, T. Sultana, T.D.T. Nguyen, J.H. Park, C.S. Hong and H. Eui-Nam, "Collaborative task offloading for overloaded mobile edge computing in small-cell networks," Proceedings of the 2020 International Conference on Information Networking (ICOIN), 2020.
[23] T. Song, W. Hu, X. Tan, J. Ren, S. Wang and S. Xu, "FAST-RAM: A Fast AI-assistant Solution for Task Offloading and Resource Allocation in MEC," Proceedings of the 2020 IEEE Global Communications Conference, pp.1-6, 2020.
[24] V.D. Tuong, T.P. Truong, T. Anh-Tien, A. Masood, D.S. Lakew, C. Lee, Y. Lee and S. Cho, "Delay-Sensitive Task Offloading for Internet of Things in Nonorthogonal Multiple Access MEC Networks," Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), pp.597-599, 2020.
[25] A. Alshahrani, I.A. Elgendy, A. Muthanna, A.M. Alghamdi and A. Alshamrani, "Efficient multi-player computation offloading for VR edge-cloud computing systems", Applied Sciences, Vol.10, No.16, pp.5515, 2020.
[26] M. Qin, N. Cheng, Z. Jing, T. Yang, W. Xu, Q. Yang and R.R. Rao, "Service-Oriented Energy-Latency Tradeoff for IoT Task Partial Offloading in MEC-Enhanced Multi-RAT Networks," IEEE Internet of Things Journal, Vol.8, No.3, pp.1896-1907, 2021.
[27] L.N.T. Huynh, P. Quoc-Viet, P. Xuan-Qui, T.D.T. Nguyen, D. Hossain and H. Eui-Nam, "Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach," *Applied Sciences*, Vol.10, No.1, pp.1-17, 2020.
[28] Y. Yang, X. Chang, Z. Jia, Zhu Han and Zhen Han, "Processing in Memory Assisted MEC 3C Resource Allocation for Computation Offloading," *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*, 2020.
[29] S. Pang and S. Wang, "Joint Wireless Source Management and Task Offloading in Ultra-Dense Network," *IEEE Access*, Vol.8, pp.52917-52926, 2020.
[30] Y. Liao, L. Shou, Q. Yu, Q. Ai and Q. Liu, "Joint offloading decision and resource allocation for mobile edge computing enabled networks," *Computer Communications*, Vol.154, pp.361–369, 2020.
[31] S.S. Shafiee, S. Schmidt, S. Zadtootaghaj, C. Griwodz and S. Möller, "Delay sensitivity classification of cloud gaming content," *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems*, pp.25-30, 2020.
[32] S. Göring, R. Steger, R.R.R. Rao and A. Raake, "Automated Genre Classification for Gaming Videos," *Proceedings of the IEEE 22nd International Workshop on Multimedia Signal Processing*, pp.1-6, 2020.
[33] ETSI, "Mobile-edge computing introductory technical white paper," *Mobile-edge Computing Industry Initiative*, 2014.

Zhen-Yuan Pan is an undergraduate student at the Department of Electrical Engineering, National Taiwan University of Science & Technology, Taipei, Taiwan. His current research interests are Machine Learning, SDN and AI.

Jiann-Liang Chen was born in Taiwan on December 15, 1963. He received the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan in 1989. Since August 2008, he has been with the Department of Electrical Engineering of National Taiwan University of Science and Technology, where he is a Professor and Dean now. His current research interests are directed at cellular mobility management and personal communication systems.

Yao-Chung Chang [M'03] received his Ph.D. degree from National Dong Hwa University, Hualien, Taiwan, in 2006. He serves as the Chair, Department of CSIE, National Taitung University, Taiwan. He is a recipient of the subsidization program in universities for encouraging exceptional talent, Ministry of Science and Technology, Taiwan, and is the author of more than 60 papers in international journals and conferences. His main research interests include mobile communication networks, AIoT, Cloud Computing and 5G Mobile Computing.