Human-centric Computing and Information Sciences

July 2021 | Volume 11

www.hcisjournal.com

KIPS
Korea Information Processing Society

KIPS CSWRG
Korea Information Processing Society
Computer Software Research Group

HCIS | Human-centric Computing and Information Sciences

RESEARCH(Research Manuscript) **Open Access**

Hum. Cent. Comput. Inf. Sci. (2021) 11:28
https://doi.org/10.22967/HCIS.2021.11.028
Received May25, 2021; accepted July 2, 2021; published July15, 2021.

An SDN-Based Packet Scheduling Scheme for Transmitting Emergency Data in Mobile Edge Computing Environments

Chan Haeng Lee[1] andJiSu Park[2,*]

Abstract

With the Internet of Things (IoT) sensor and smart devices gradually increasing nowadays due to the fourth industrial revolution, the information collected from the sensors is very diverse, and the amount of information transmitted is also massive. Since the data collected from a lot of devices can generate excessive traffic, several researchers have tried to solve this issue using mobile edge computing (MEC). To cope with the numerous and diverse IoT devices, the MEC infrastructure has to support a lot of connected devicesand the processing of the massive data collected and complex applications. However, the MEC edge server has limited computational and processing resources compared to high-end servers in the cloud server. In this paper, we propose a Software-Defined Networking (SDN)-based packet scheduling scheme for quickly transmitting emergency data in MEC environments. In the proposed scheme, the MEC includes the OpenFlow function to communicate with the SDN controller. In the OpenFlow-enabled MEC host, we adopt two queues for emergency data and normal data. For the performance evaluation of the proposed scheme, a comparative analysis was performed with the existing priority-based scheduling model, and the analysis result shows that the proposed scheme has better performance than the existing priority-based scheduling method.

Keywords

SDN, Priority, Emergency Data, Packet Scheduling, Internet of Things, Edge Computing, Mobile Edge Computing, MEC

1. Introduction

Since the 2000s, the growth of the Internet and network technologies has been rapidly developing with the advent of various devices such as smartphones, tablets, and wearable devices. According to the demands of users who need an Internet connection using various devices, a lot of data on the Internet can be collected, processed, and transferred to other devices as necessary, and this has become a foothold for leading the era of the 4th Industrial Revolution. The Internet of Things (IoT), one of the

***Corresponding Author:**Ji Su Park(jisupark@jj.ac.kr)
[1]Research Institute of Engineering & Technology, Jeonju University, Jeonju, Korea
[2]Department of Computer Science and Engineering, Jeonju University, Jeonju, Korea

technologies attracting attention as a technology in the era of the 4th Industrial Revolution, is growing rapidly due to the rapid development of such Internet and network technologies and the advent of various smart devices. With sensors and mobile devices increasing rapidly, people believe that the huge data generated by various smart devices and IoT devices will growexplosively in the near future.

Nowadays, IoT is used in not only home IoT that can remotely control various objects in the housebut also in various fields such as logistics management through environmental sensors such as temperature and humidity, facility security such as fire detection and external intrusion detection, smart city construction, etc. [1, 2]. The information collected from the sensors is very diverse, and the amount of information transmitted is also massive. In order to extract or analyze meaningful data from a lot of information, the collected data must be transmitted to a data center. Especially, emergency data that needs to be processed quickly, such as fire or intrusion detection, should be processed ahead of other data. However, processing a large volume of rapidly occurring data in realtime is very difficult due to problems such as limitation of transmission bandwidth, transmission delay of network, and data processing time.

Since the data collected from a lot of devices can generate excessive traffic, several researchers have tried to solve this issue usingcloud computing, fog computing, andmobile edge computing, also called multi-access edge computing (MEC) [2–10]. MEC is a new technology that is currently being standardized in an Industry Specification Group (ISG) of the European Telecommunications Standards Institute (ETSI) [6,9, 10]. MEC is a new paradigm that applies distributed cloud computing technology to the radio access network (RAN) to move network traffic and computing service from a centralized cloud to the edge of the network to provide services at a location close to the users. MEC enables analyzing, processing, and storing data at the edge of the network, instead of sending all data to the cloud for processing. By deploying various services and caching contents close to the users and IoT devices, congestion can be eased in the core network, and new local services can be created. It also supports real-time data processing with low latency. Therefore, MEC is able to provide real-time services with faster response and which are efficient and secure for lots of end-users, mobile devices, and IoT devices.

MEC supports mobile computing and IoT technologies by providing distributed processing ability with a distributed open architecture. To cope with the numerous and diverse IoT devices, the MEC infrastructure has to support a lot of connected devices and the processing of the huge data collected and complex applications. However, the MEC edge server has limited computational and processing resources compared to high-end servers in the cloud server. Therefore, to support high scalability, ultra-low latency, high throughput, and reliable transmission of data, the Software-Defined Networking (SDN) paradigm is regarded as one of the suitable solutions [11–13]. SDN has a structure wherein the network control function is separated from the physical network [11]. Considering the characteristics of the SDN technology that enables dynamic, programmatically efficient network configuration by decoupling the network into a control plane and a data plane, it is possible to improve the performance and low latency of the edge server.

We focus on the programmable and decoupled features of SDNand propose an SDN-based packet scheduling scheme for processing emergency data quickly in MEC environments. In the proposed scheme, the MEC server includes the OpenFlow protocol function to communicate with the SDN controller. In the OpenFlow-enabled MEC, we adopt two queues for emergency data transfer called emergency queue (EQ) and for general packet transfer called normal queue (NQ). The EQ is used for transmitting emergency data, and the NQ is for the other data.

The main contributions of this paper are summarized as follows: (1) unlike the existing priority-based technologies, the proposed scheme uses two queuesinstead of multiple queues for resource management of the MEC server; (2) we propose a method of processingan emergencypacket first when itcomes into the queues; and (3) to reduce transmission delay, the SDN concepts are adopted for the MEC server and MEC hosts.

The rest of this paper is organized as follows: Section 2 introduces related works for MEC and SDN;

Section 3 discusses the proposed architecture and scheduling method model; Section 4 shows the scheduling model analysis and analytical results of the simulation; Section 5 presents the conclusion and future works.

2. Related Work

Cloud computing is a technology that provides computing services such as server, storage, and software analysis through the Internet [13–15]. As a kind of computing technologies based on the Internet, cloud computing enables data storage, processing, and fast processing by using other computing devices or storage connected to the Internet. Due to the exponential increase of users who use the existing cloud services, however, vast amounts of data are concentrated on the central server and data center through the network, giving rise to problems such as data transmission delay, throughput decrease, and security problems.

Edge computing has emerged as a solution to these problems [16, 17]. As a distributed open architecture that enables processing such as aggregation, storage, and processing of data collected at the edge of a network, edge computing enables the efficient processing of large amounts of processable data around the source, which significantly reduces the data processing time and Internet bandwidth usage. Thus, edge computing is used primarily for applications that require massive data collection/processing and rapid data analysis for real-time response [17]. Such edge computing can be applied to a distributed computing model because it is designed considering network connection and latency, bandwidth constraints, and various functions embedded in a terminal device.

MEC is a kind of edge computing that brings cloud computing to the network edge and extends cloud computing capabilities[7,10,18, 19]. In addition, MEC is able to support IT and cloud computing functions at the edge of a cellular network or a legacy network and is designed to solve the problems of real-time processing and transmission delays caused by data transmission to remote servers far away from users and devices as the disadvantages of traditional cloud computing in general [15]. The computing resources are located at the base station (BS) of mobile networks called evolved Node Bs (eNodeBs), so it is possible to provide real-time, high-bandwidth, low-latency data processing. For this reason, it can be utilized as one of the methods that can distribute the operation and processing process for data without transmitting all data to the data center, central offices, and other branches of the network. It can play an important role in reducing network congestion and improving applications by performing related processing tasks closer to end-users or IoT sensor devices. Nowadays, standardization and discussion on the standardization of technical requirements and frameworks related to MEC are ongoing [3–6,9].

SDN is a conceptual architecture that decouples the control plane and data plane of the network and enables network partitioning [11]. SDN architecture is divided into three categories: the physical infrastructure layer, the controllable control layer, and the application layer. In other words, in SDN, the network control function should be separated from the hardware such as the existing switch or router, and it has a network structure that can be developed and executed separately from the data transfer function. The ability to control the network is concentrated on the SDN controller. As a result, network operators and administrators can simplify and manage their networks programmatically instead of entering and managing manually from a variety of distributed network devices [11]. Network control functions are separated from hardware such as switches and routers, and control-related programming such as network topology configuration and routing is possible. This lets SDN not only create complex paths that cannot be configured in existing networks but also effectively cope with changing traffic patterns and quickly configure the virtual networks required in cloud environments.

The separation of the control plane and the data plane of the network has the advantage of being able to respond more quickly to a malfunction caused by a problemand increase the flexibility and availability of the network. These characteristics of SDN can lead to the implementation and

performance improvement of various additional functions by using together with MEC and edge computing. The OpenFlow protocol, standardized by the Open Networking Foundation (ONF), is used between the control plane and data plane as a communication interface in the SDN environment [20, 21]. Using the OpenFlow protocol, the SDN controller can control the underlying network components.

Nowadays, several studies on the convergence of MEC and networking functions such as network virtualization and SDN are actively underway. In [22], the authorsproposed the integration of SDN and cloud-native virtualization schemes with the MEC to facilitate orchestration and management and to support the mobility and QoS management of mobile edge hosts (MEH). They focused on seamless end-to-end mobility of the vehicle networks and customized specific functions and architectural components of MEC to their proposed architecture. In [23], the authorsproposed an MEC framework for SDN-based LTE/LTE-A networks. They implemented an SDN-based MEC framework with ETSI and 3GPP compliance and focused on how the LTE/LTEA network can fully cooperate with MEC and SDN.

3. Proposed Architecture and Scheduling Scheme

In this section, we introduce the reference architecture and the proposed architecture first, followed by the priority-based scheduling method. In addition, the scheduling model for reducing transmission delay is illustrated.

3.1 Proposed Architecture

According to [5] from ETSI, MEC enables the implementation of MEC applications as software-only entities that run on top of a virtualization infrastructurelocated in or close to the network edge.

In the illustration of the reference architecture of [5], the MEC system consists of MEC hosts and MEC manager to run MEC applications within an operator network or its subnet. In the MEC system, the MEC host means an entity that contains an MEC platform and a virtualization infrastructure to run MEC applications, and the MEC platform indicates the collection of essential functions to run MEC applications on the virtualization infrastructure. The document shows the variant version of the reference architecture for MEC by using network function virtualization (NFV). We focuson this feature and try to integrate the SDN features by alternating the virtualization infrastructure to the MEC architecture.

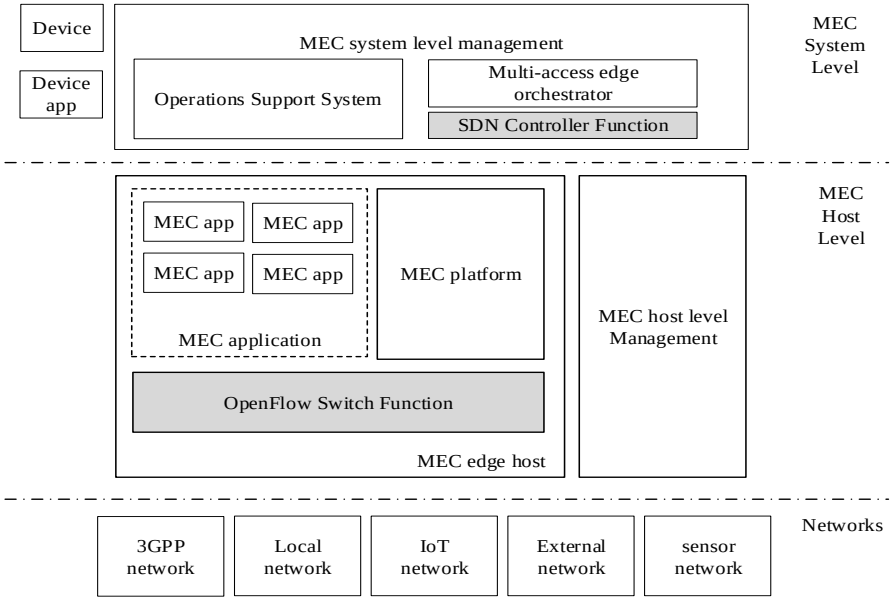


Fig. 1. Proposed architecture for SDN-enabled MEC.

First, we consider the OpenFlow switch function to be installed in the MEC edge, which is responsible for collecting, storing, filtering, and transmitting data from the MEC hosts to the MEC system. In addition, it should be possible to process a large amount of data in the vicinity of the source, enabling efficient data processing; thus reducing Internet bandwidth usage. In case of the emergency data, the data is also processed at the MEC edge host first. The controller function of the SDN network is located in the MEC orchestrator to support the transmission of commands through a secure channel between the controller and the MEC edge host using the OpenFlow protocol. By applying this characteristic of the SDN network to the MEC reference architecture, we proposed a modified version of the reference architecture that delivers data processed from the MEC edge hosts to the controller through the MEC system. Fig. 1 shows the proposed architecture for the SDN-enabled MEC system.

3.2 Operating Process

We consider the terminal devices to be variable sensing devices and IoT devices, transferring periodic data and emergency data according to their condition. Therefore, the MEC edges have to process data aggregation and fast transmission to reduce bandwidth and process data transmission. The operation processes are divided into two categories: one is for the normal operation for periodic data, and the other is for emergency data. Each operation is introduced below.

3.2.1 Normal operation

Data generated by variable terminal devices such as sensors and IoT devices can be transmitted to the MEC edge, which includes computing resources and storage devices for the direct processing of data collection, data aggregation, and data transmission. The MEC edge allocates resources for the analysis of periodic data collected from a number of devices; after performing a process of aggregating periodic data delivered to the same destination, the collected information can be delivered to a data center or an analysis server. The collected data is transmitted to the destination address through the data plane through encapsulation as in normal data transmission, and the source address at this time can be the address of the MEC edge. In the aggregation process, each data from the sensing device is aggregated with their device identifier (ID).

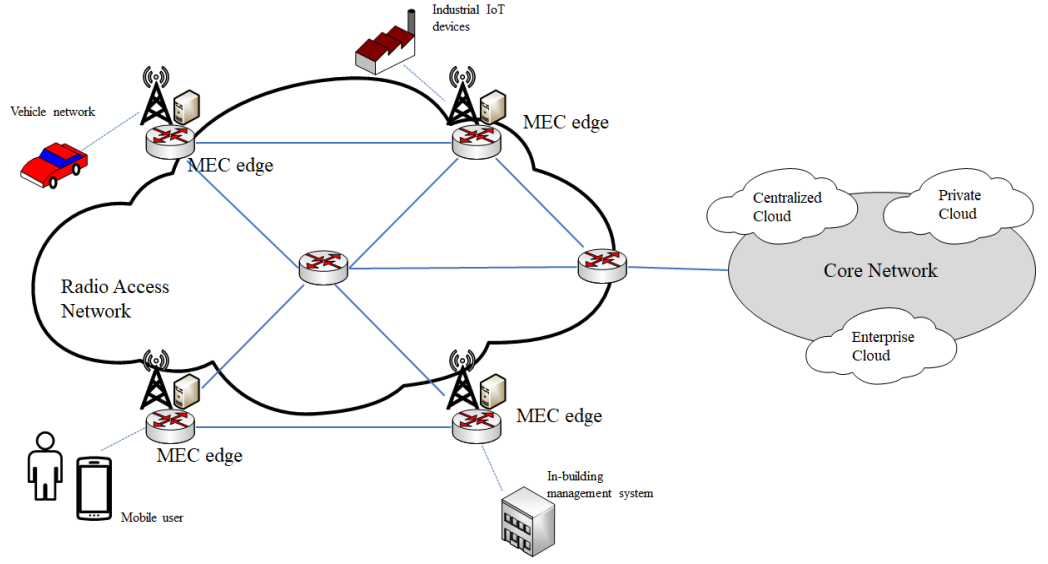


Fig. 2. General infrastructure of basic MEC system.

In the MEC edge, if the received data is normal data, it is transmitted through the data plane

according to the internal flow table. As mentioned before, the MEC edge has the SDN function of OpenFlow switch, and the MEC can transmit data through the data plane. After the data arrives at the MEC edge, the MEC edge looks up its flow tables for processing the data. If a table entry matches, the MEC edge forwards the data to the data plane according to the rule. If it does not match the flow table entry, a command for the data is requested by sending a packet-in message to the controller of the MEC system. After receiving the packet-in message, the controller sends a rule to the MEC edge through a packet-out message according to the classified data, and the MEC edge then adds a new entry to its flow table and processes the data according to the rule. When the data arrives at the destination side, the data center or analysis server takes care of the data. The collected data according to the device ID extracted from the data arriving at the destination may be stored in a database or reflected to a situation monitoring system. Fig. 2 shows the basic infrastructure of the proposed scheme, and Fig. 3 depicts the operation process.

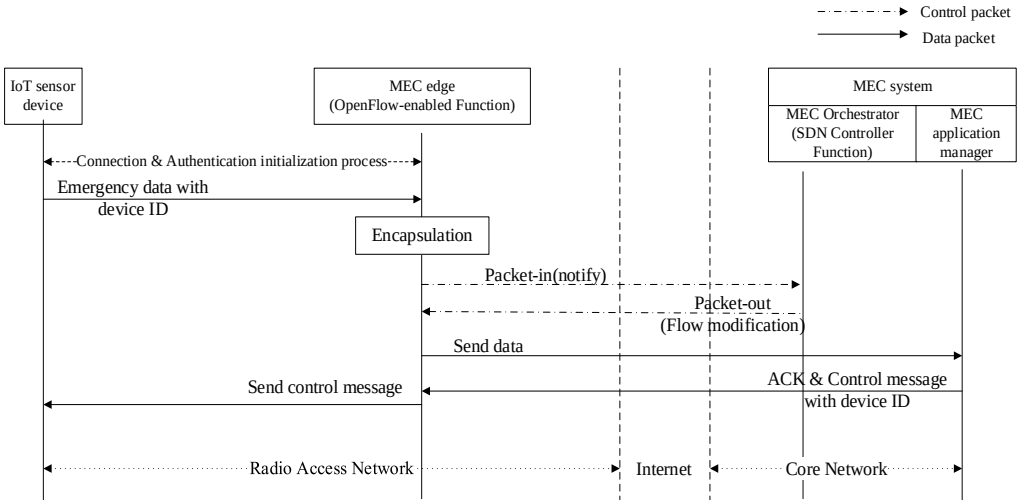


Fig. 3. Data transmitting operation process using OpenFlow function in MEC edge.

3.2.2 Emergency data processing operation

If data arrives at the MEC edge, the MEC edge has to check a specific field labeled emergency to separate emergency data from normal periodic data. The way of handling or representing emergency data may vary depending on the using applications on each device. Therefore, this paper does not discuss how to mark emergency on data because it is outside of the scope.

In the case of emergency data, it should be processed as fast as possible unlike periodic data. Emergency data is also collected from sensing devices, but the processing operation differs from that for periodic data. Periodic data is collected periodically during a certain time interval, whereas emergency datacan occurrandomly but must be transmitted immediately. Therefore, the MEC edge must be able to respond as quickly as the emergency data is received. If the MEC edge receives emergency data, it follows the general OpenFlow operation like the other normal packets do. The MEC edge packetizes the emergency data with its address as source address and prepares to send it to the destination. Simultaneously, it transmits a packet-in message to notify the controller of the occurrence of an emergency packet; after the controller receives the packet-in message, it checks and transmits a packet-out message containing information on the fastest path among its cache table entry. The MEC edge receiving the packet-out message changes or adds its flow table from the received message and delivers to the suitable output port.After the OpenFlow-based function, the MEC edge checks the data to determine whether to transfer the data enqueue to the priority queue or normal queue.

3.3 Proposed Scheduling SchemeMethodology

The queue configuration takes place outside the OpenFlow function; therefore, the output packets after flow table matching are queued in front of the MEC edge output port. In the OpenFlow protocol specification, a switch can optionally have one or more queues attached to a specific output port, and those queues can be used to schedule packets exiting the data path on such output port. Therefore, it is reasonable for the MEC edge to have a function similar to the OpenFlow switch.

When the MEC edge finds out that the received data is emergency data, a scheduling method is required to transmit emergency data ahead of the periodic data. When the data collected from the sensor arrives at the MEC edge, the MEC edge performs the aggregation process with the received data; after the aggregation processing, data are sequentially accumulated in an internal queue for transmission.

Generally, regular data is processed in order of arrival in the queue, called first-come first-served (FCFS) manner, whereas the emergency or urgent data received from the sensor devices should be processed preferentially compared to the other data. If the data to be processed first is included, the FCFS method is not suitable.

Priority queue (PQ) is a method of solving the shortcomings of FIFO (first-in first-out) so that such data to be processed prior to processing can be included. The general PQ method uses four queues divided into four classes: “high,”“medium,”“normal,” and “low.” It follows the way that processes all queues corresponding to the high-priority class first, and then processes the next lower class until the queues are empty. In this case, the PQ method needs at least four queues for the classes. Of course, the MEC edge is equipped with resources such as hardware that can perform calculations and processing; for stable and efficient operation, however, wastage of resources must be prevented as much as possible. Therefore, in this paper, we try to usetwokinds of queues, EQ and NQ, for periodic and emergency data transmission.Among the data generated in the industrial IoT environment, it is assumed that emergency data is only processed with EQ, and that all other cases such as periodic sensing dataare processed with NQ. Fig. 4 shows the operation of the management of the data in queue at the MEC edge.

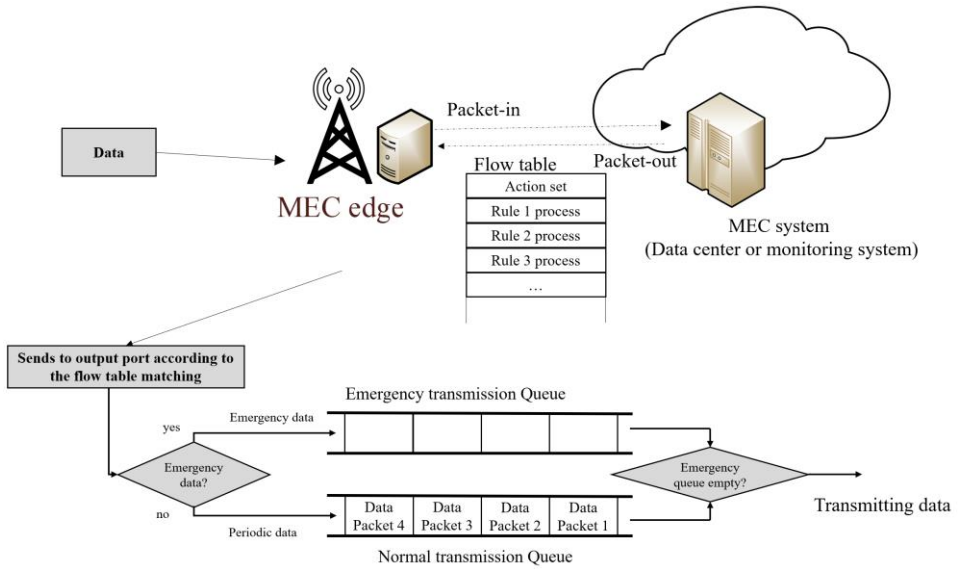


Fig. 4. Enqueue and dequeue operation at the MEC-edge.

The algorithm of enqueue and dequeue is presented in Fig. 5. Emergency data and normal data are stored in different queues, but enqueue and dequeue operations of each queue follow the general FCFS method. Since the types of IoT devices are very diverse, emergency data and normal data are extracted

from the header information of the sensing data packet. In the process of putting data in the queue for passing on to the output port, the queue with the field value for priority or emergency processing and the queue without the field value are distinguished, stored separately in two queues, and transmitted.

Algorithm. Queueing algorithm in the output port

[in output port queue]
Receive output packet(*new_pkt*) after the OpenFlow function processing :

[enqueue process]

```
if new_pkt.emergency_check is TRUE then
    if isfull(EQ) is FALSE then
        enqueue.EQ(new_pkt)
    else
        drop.EQ(old_pkt)
        enqueue.EQ(new_pkt)
else
    if isfull(NQ) is FALSE then
        enqueue.NQ(new_pkt)
    else
        drop.NQ(old_pkt)
        enqueue.NQ(new_pkt)
```

[dequeue process]

```
while (!isempty(queue.EQ) is TRUE or !isempty(queue.NQ) is TRUE)
{
    if isempty(queue.EQ) is TRUE then
        dequeue.NQ(pkt)
    else
        dequeue.EQ(pkt)
}
```

Fig. 5. Queueing algorithm in the MEC-edge.

3.4 SchedulingModel Analysis

In general, network delay refers to the time taken in the process of transmitting one data packet from the source to the destination. Therefore, the calculation of network delay can generally be calculated as the total sum of the transmission delay, propagation delay, processing delay, and queuing delay. Many recent network devices are equipped with high-performance processors and vast amounts of memory and storage devices to speed up the networking process. Moreover, the transmission technique using 5G supports transmission speeds of up to 1 Gbps and an average of 400–500 Mbps due to bandwidth expansion. In case the same devices are used,or the performance of devices is similar, the distance to the destination, the medium that delivers the packet, and the queuing delay—which is responsible for processing in the queue—are factors affecting the delay. Therefore, in this section, the proposed scheduling model focuses on the queuing delay of the network delay.First, we discuss the mean waiting time of the basic FCFS manner and analyze the mean waiting time of the proposed scheduling model. It is assumed that the controller has infinite buffer space, and that the MECedge has finite buffer space. We also assumed that the transmission process of a data packet is completed without interruption, even if an emergency packet arrives during the transmission process.The incoming packets are handled by the M/G/1 queuing model considering the MECedge [24–26].

3.4.1 Mean waiting time

For the mean waiting time of FCFS and the proposed scheme, the non-Markovian queueing model is selected. In the FCFS system, the packet arrival rate follows the Poison distribution, and the service time follows the geometric distribution. In M/G/1-FCFS, the mean waiting time of an arriving data has two components: the mean remaining service time of the job in service, and the sum of the mean service times of the data in the queue at the time of data arrival. In the case of emergency data, it should be processed faster than normal data. Therefore, M/G/1-FCFS scheduling is applied for emergency data.We consider the non-preemptive prioritythat a queue in the MECedge can complete the processing of ongoing data without interruption even if an emergency packet arrives.In addition, a logically separate queue is maintained for normal data and emergency data. In other words, each time the system becomes idle, the first data in the emergency queue are processed first. Several notations are used in

each equation as presented in Table 1.

Table 1. Notations

Notation	Meaning
$\bar{W}$	Mean waiting time
N_{EQ}	Number of packets belonging to the emergency queue
N_{NQ}	Number of packets belonging to the normal queue
W_{EQ}	Waiting time of emergency packets
W_{NQ}	Waiting time of normal packets
ρ	The probability of $server\ is\ busy$. The utilization of the server process.
$\bar{R}$	Mean remaining service time in the server
$\bar{S}$	Mean service time for a packet
μ	Service rate
λ	Arrival rate

With the M/G/1 queue, mean waiting time $\bar{W}$ is calculated as follows according to the Pollaczek-Khinchin(P-K) formula:

$$\bar{W} = \bar{N} * \bar{S} + \bar{R} \tag{1}$$

In the equation, $\bar{N}$means the number of waiting packets, and it is also represented as queue length. $\bar{S}$means the service time needed to service a packet. $\bar{R}$ indicates the mean residual service time of the MECedge, which is processing data. In the case of idle state, the residual service time is obviously 0. When a packet arrives in the queue, the job in service needs $\bar{R}$ time units on the average to be finished. In the proposed scheme, two queues are maintained for the emergency packet and non-emergency packet as the normal packet. In general, the probability of utilization of the queueing process for an ongoing service is represented as ρ. Therefore, the utilization of each queue is represented as $\rho_{EQ}$and ρ_{NQ}. When the utilization rate in a single server is less than 1, it is called the stable state. Therefore, it is reasonable that $\rho_{EQ}+ \ \rho_{NQ} < 1$ for the stable state. For the general holding time distribution, mean service time $\bar{S}$can be represented as $\frac{1}{\mu}$. Due to the non-preemptive policy, it is reasonable for all packets to have the same service time distribution, and for residual service time $\bar{R}$to be the same as well. Similar to the P-K formula, the mean waiting time of emergency packet $\overline{W_{EQ}}$is calculated as follows:

$$\overline{W_{EQ}} = \bar{R} + \bar{S} \cdot \overline{N_{EQ}} = \bar{R} + \frac{1}{\mu}\lambda_{EQ}\overline{W_{EQ}} \tag{2}$$

From Little's equation $\bar{N} = \lambda\bar{W}$, equation (2) is expressed as:

$$\overline{W_{EQ}} = \frac{\bar{R}}{1-\rho_{EQ}} \tag{3}$$

In the case of $\overline{W_{NQ}}$,the waiting time differs from $\overline{W_{EQ}}$because its priority is always lower than emergency packets. Therefore, $\overline{W_{NQ}}$can be expressed as:

$$\overline{W_{NQ}} = \bar{R} + \bar{S} \cdot \overline{N_{EQ}} + \bar{S} \cdot \overline{N_{NQ}} + \frac{1}{\mu}\lambda_{EQ}\overline{W_{NQ}} \tag{4}$$

In equation (4), $\bar{S} \cdot \overline{N_{EQ}} + \bar{S} \cdot \overline{N_{NQ}}$indicates the time needed to serve emergency packets and normal packets ahead in the queue. $\frac{1}{\mu}\lambda_{EQ}\overline{W_{NQ}}$means the time needed to serve emergency packets that arrive

during the waiting time of normal packets. As mentioned earlier, all packets have the same service time distribution, and the mean service times of emergency packets and normal packets areconsidered equivalent.By using Little's equation, equation (4) is expressed as:

$$\overline{W_{NQ}} = \bar{R} + \frac{1}{\mu}\lambda_{EQ}\overline{W_{EQ}} + \frac{1}{\mu}\lambda_{NQ}\overline{W_{NQ}} + \frac{1}{\mu}\lambda_{EQ}\overline{W_{NQ}} = \bar{R} + \rho_{EQ}\overline{W_{EQ}} + \rho_{NQ}\overline{W_{NQ}} + \rho_{EQ}\overline{W_{NQ}} \quad (5)$$

Therefore, equation (5) is solved as follows using equation (3):

$$\overline{W_{NQ}} = \frac{\bar{R}+\rho_{EQ}\overline{W_{EQ}}}{1-\rho_{EQ}-\rho_{NQ}} = \frac{\bar{R}}{(1-\rho_{EQ})(1-\rho_{EQ}-\rho_{NQ})} \quad (6)$$

In the single server, utilization $\rho$is derived as follows:

$$\rho = \lambda * c * \bar{S} \quad (7)$$

In equation (7), c means the number of servers; therefore, itis set to 1. Then the utilization rates of the emergency packet and normal packet are expressed as:

$$\rho_{EQ} = \lambda_{EQ}\bar{S} = \frac{\lambda_{EQ}}{\mu}, \quad \rho_{NQ} = \lambda_{NQ}\bar{S} = \frac{\lambda_{NQ}}{\mu} \quad (8)$$

Mean residual service time $\bar{R}$is derived as follows:

$$\bar{R} = \frac{\overline{S^2}}{2\bar{S}} = \frac{1}{2}\lambda\overline{S^2} \quad (9)$$

$\lambda$is calculated as $\sum_{i=1}^{P}\lambda_i$according to the number of priorities P. In the proposed scheme, the data is divided into normal data and emergency data; therefore,P = 2. Likewise,$\overline{S^2}$is expressed as $\frac{1}{\lambda}\sum_{i=1}^{P}\lambda_i\overline{S_i^2}$. As a result, equation (3) can be expressed as:

$$\overline{W_{EQ}} = \frac{\lambda_{EQ}\overline{S^2}}{2(1-\rho_{EQ})}. \quad (10)$$

In the same way, equation (6) is expressed as:

$$\overline{W_{NQ}} = \frac{\sum_{i=1}^{2}\lambda_i\overline{S_i^2}}{2(1-\rho_{EQ})(1-\rho_{EQ}-\rho_{NQ})} = \frac{\overline{S^2}(\lambda_{EQ}+\lambda_{NQ})}{2(1-\rho_{EQ})(1-\rho_{EQ}-\rho_{NQ})}. \quad (11)$$

4. Simulation Result Analysis

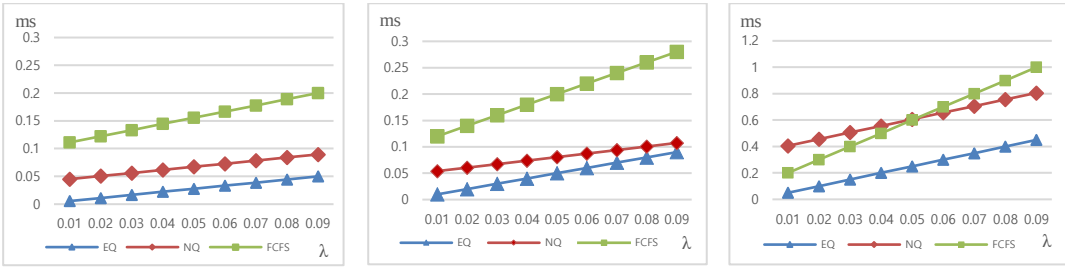
For the comparison between FCFS and the proposed scheduling scheme, the average waiting time and processing delay are analyzed. First, the mean waiting time is analyzed by processing the EQ and NQ data separatelyaccording to both change in system utilization (ρ) and increase of arrival rate (λ).

For the experimental result analysis, the operation according to packet scheduling in one MECedge environment to which the proposed scheme is applied is considered, and the delay according to the mean waiting time and utilization is analyzed. The calculation results of the FCFS scheme were measured considering the operation according to packet scheduling in an environment considered normal MECedge nodes. In calculating the mean waiting time, we set the variances of arrival rate from 0.01 to 0.09 and the mean service time to 1. We consider the number of emergency packets to be 10, and the number of normal packets is set to 90; therefore, the total number of packets is set to 100. In other words, the ratio of emergency packet to normal packet is 1:9. Assuming a general IoT environment, data transmitted from numerous IoT sensors generate more normal data than emergency

data, so the ratio of emergency data to normal data is set to 1:9. Fig. 6 shows the simulation results according to the utilization.

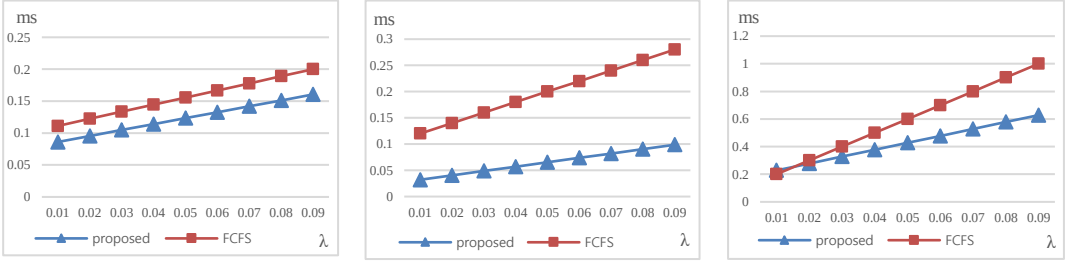
As shown in Fig. 6(a) and 6(b), cases that use the emergency queue or normal queue in the proposed scheme consume less mean waiting time than FCFS. It shows that emergency data can be processed faster when the system utilization rate is the low case (0.1) or normal case (0.5).When the system utilization is high and the arrival rate is lower than 0.05, the waiting time of NQ increases compared to that of FCFS. Even if the usage rate is high, however, it can be seen that, if the arrival rate gradually increases, the waiting time of NQ can show better results than FCFS.

Since the results shown in Fig. 6 are only experimental results for cases using each queue independently, it is necessary to use EQ and NQ together to compare with FCFS in order to compare the proposed technique with the FCFS method. Therefore, other simulations are conducted by increasing the change in arrival rate for low, normal, and high utilization as shown in Fig. 6, and the results of the proposed scheme applying both EQ and NQ and the average waiting time of FCFS are presented in Fig. 7.



(a) (b) (c)

Fig. 6. Mean waiting timeaccording to the variable utilization:
(a) ρ = 0.1, (b) ρ = 0.5, and (c) ρ = 0.9.



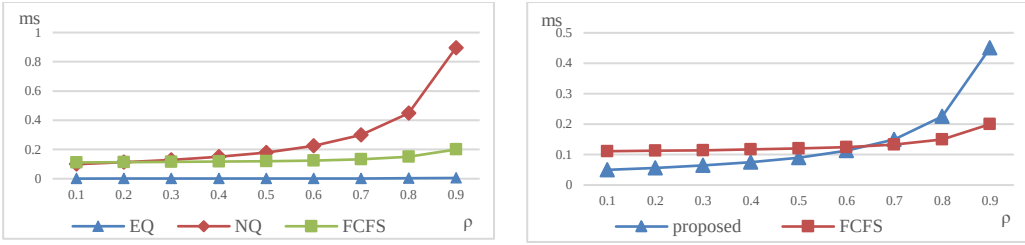
(a) (b) (c)

Fig. 7. Mean waiting time comparison between the proposed scheme and FCFS:
(a) ρ = 0.1, (b) ρ = 0.5, and (c) ρ = 0.9.

In the proposed scheme, each utilization rate was applied differently according to the occurrence rate of the emergency packet and normal packet in order to maintain a stable state. As a result, the comparison of the FCFS and the proposed scheme using EQ and NQ shows results similar to the previous results. Similar to Fig. 6(c), Fig. 7(c) shows that applying the proposed method does not yield much difference from FCFS at low arrival rates. As the arrival rate increases, however, the difference in mean waiting time increases proportionally.

In the case of mean waiting time, the proposed scheme has better performance than the FCFS scheme, but the delay shows different results. In order to compare the queuing delay between the proposed scheme and the FCFS method, the delay according to the increase in the system usage rate was calculated, and the result is shown in Fig. 8. Fig. 8(a) presents the queuing delay of EQ, NQ, and

FCFS, and Fig. 8(b) shows the variation of queuing delay of FCFS and the proposed scheme,which applies EQ and NQ.



(a) (b)

Fig. 8. Queueing delay comparisons between the proposed scheme and FCFS:
(a) each case delay by utilization and (b) delay between the proposed scheme and FCFS.

As shown in Fig. 8(a), the EQ delay is very low compared to the FCFS queuing delay, but the NQdelay gradually increases until the utilization ratereaches50%.Aftertheutilizationrateexceeds60%,the delay increases rapidly.EQisalwaysprocessedbeforetheNQisprocessed.In the case of NQ, the existence of EQ is determined first, and the data of NQ is processed. This is because the determination of the existence of EQ increases as the utilization rate increases.

Fig.8(b)showsthecomparisonof thedelaybetweentheproposedschemeandFCFS, suggesting thattheproposedschemeisslightlybetterthanFCFSat less than 60%utilization; however,FCFSisbetterthantheproposedschemeat over60%utilization.As mentioned earlier, the reason for the sudden increase in delay is that the proposed scheme processes emergency packets first, so there is a delay in the process of processing general packets after the emergency packets are processed.

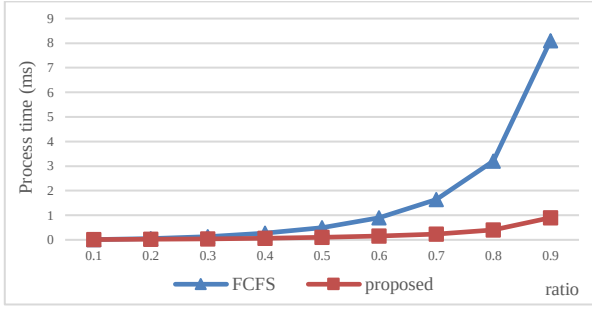


Fig. 9. Processing time comparisons between the proposed scheme and FCFS.

According to the ratio of emergency packet to normal packet, the processing delay was calculated to compare FCFS and the proposed scheme. The processing time was calculated by changing the number of emergency packets from 100 to 900 based on a total of 1,000 packets. The variances of processing time according to the increase in the ratio of emergency packets are shown in Fig. 9. In Fig. 9, the label of the x-axis, “ratio,” represents the ratio of emergency packets to the total number of packets. In the case of FCFS wherein more than half of the entire transmitted data transmits emergency data, it can be seen that the processing time of emergency data increases rapidly.

5. Conclusion and Future Works

In the IoT environment, mobile edge computing can transmit data to destinations faster by reducing transmission delays that may occur due to the transmission of data through several existing stepsand by

giving the MECedge a calculating function. It is a technology for future industrial environments with the advantage of being able to optimize and transmit a large number of data.

In this study, we suggested an SDN-based packet scheduling method for transmitting emergency data inmobile edge computing environments. We tried to combine SDN into the MECedgeand proposed a new scheduling method by using two kinds of queues in the MECedge. As a result of the simulation, theproposedschemeshowsbetterperformancethanFCFSschedulingfortheemergencypacketprocessing; in the case of normal data, however, the delay time may increase more than that of FCFS as the utilization increases.

In the future, we will make efforts to refine the scheduling algorithm continuously to apply to various environments and try to test in virtual and real environments.

Author's Contributions

Conceptualization, CH, JS. Funding acquisition, JS. Investigation and methodology, CH, JS. Project administration, JS. Resources, CH. Supervision, JS. Writing of the original draft, CH. Writing of the review and editing, CH, JS. Software, CH, JS. Validation, CH, JS. Formal Analysis, CH, JS. Data curation, CH. Visualization, CH.

Funding

This research was supported by the Research Grant of Jeonju University in 2020 and by MSIT(Ministry of Science and ICT), Koreaunder the Information Technology Research Center (ITRC) support program(No. IITP-2021-2020-0-01789) supervised by Institute for Information & Communications Technology Planning & Evaluation (IITP).

Competing Interests

The authors declare that they have no competing interests.

References

[1] I. Jawhar, N. Mohamed, and J. Al-Jaroodi, "Networking architectures and protocols for smart city systems," *Journal of Internet Services and Applications*, vol. 9, article no. 26, 2018. https://doi.org/10.1186/s13174-018-0097-0

[2] G. Li, Y. Yao, J. Wu, X. Liu, X. Sheng, and Q. Lin, "A new load balancing strategy by task allocation in edge computing based on intermediary nodes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, article no. 3, 2020. https://doi.org/10.1186/s13638-019-1624-9

[3] ETSI, "Multi-access Edge Computing (MEC); Study on MEC support for alternative virtualization technologies (ESTI GR MEC 027 V2.1.1)," 2019 [Online]. Available: https://www.etsi.org/deliver/etsi_gr/MEC/001_099/027/02.01.01_60/gr_MEC027v020101p.pdf.

[4] ETSI, "Mobile Edge Computing (MEC); Technical Requirements (ETSI GS MEC 002 V.1.1.1)," 2016 [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf.

[5] ETSI, "Multi-access Edge Computing (MEC); Framework and Reference Architecture (ETSI GS MEC 003 V2.1.1)," 2019 [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf.

[6] ETSI, "Mobile-Edge Computing: introductory technical white paper," 2014 [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf.

[7] M. Mahbub, M. S. Apu Gazi, S. A. Arabi Provar, and M. S. Islam, "Multi-Access Edge Computing-Aware Internet of Things: MEC-IoT," in *Proceedings of 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE)*, Bangladesh, 2020, pp. 1-6. https://doi.org/10.1109/ETCCE51779.2020.9350909

[8] W. B. Daoud, M. S. Obaidat, A. Meddeb-Makhlouf, F. Zarai, and K. F. Hsiao, "TACRM: trust access

control and resource management mechanism in fog computing," *Human-Centric Computing and Information Sciences*, vol. 9, article no. 28, 2019. https://doi.org/10.1186/s13673-019-0188-3
[9] ETSI, "Mobile Edge Computing: a key technology towards 5G (White Paper No. 11)," 2015 [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf.
[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017.
[11] Open Networking Foundation, "Software-Defined Networking (SDN) definition," 2021 [Online]. https://www.opennetworking.org/sdn-definition/.
[12] A. H. Shamsan and A. R. Faridi, "SDN-assisted IoT architecture: a review," in *Proceeding of the 4th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, India, 2018, pp. 1-7. https://doi.org/10.1109/CCAA.2018.8777339
[13] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5706-5712, 2019. https://doi.org/10.1109/JIOT.2019.2942719
[14] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Reliability and high availability in cloud computing environments: a reference roadmap," *Human-Centric Computing and Information Sciences*, vol. 8, article no. 20, 2018. https://doi.org/10.1186/s13673-018-0143-8
[15] S. B. Shaw and A. K. Singh, "A survey on cloud computing," in *Proceedings of the 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, Coimbatore, India, 2014, pp. 1-6.
[16] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: a survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359-2391, 2017.
[17] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900-6919, 2017.
[18] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Mobile Edge Computing Network control: tradeoff between delay and cost," in *Proceedings of 2020 IEEE Global Communications Conference (GLOBECOM)*, Taipei, Taiwan, 2020, pp. 1-6.
[19] J. Liu, G. Shou, Y. Liu, Y. Hu, and Z. Guo, "Performance evaluation of integrated multi-access edge computing and fiber-wireless access networks," *IEEE Access*, vol. 6, pp. 30269-30279, 2018.
[20] Open Networking Foundation, "OpenFlow Switch Specification version 1.5.1," 2015 [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.
[21] H. Farhady, H. Lee, and A. Nakao, "Software-Defined Networking: a survey," *Computer Networks*, vol. 81, pp. 79-95, 2015. https://doi.org/10.1016/j.comnet.2015.02.014
[22] S. D. A. Shah, M. A. Gregory, S. Li, and R. D. R. Fontes, "SDN enhanced multi-access edge computing (MEC) for E2E mobility and QoS management," *IEEE Access*, vol. 8, pp. 77459-77469, 2020.
[23] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency MEC framework for SDN-based LTE/LTE-A networks," in *Proceedings of 2017 IEEE International Conference on Communications (ICC)*, Paris, France, 2017, pp. 1-6.
[24] G. Bolch, S. Greiner, H. Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2006.
[25] P. Kasabai, K. Djemame, and S Puangpronpitag, "Priority-based scheduling policy for OpenFlow control plane," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 2, pp. 733-750, 2019. https://doi.org/10.3837/tiis.2019.02.014
[26] H. Xue, K. T. Kim, and H. Y. Yoon, "Packet scheduling for multiple-switch software-defined networking in edge computing environment," *Wireless Communication and Mobile Computing*, vol. 2018, article no. 7659085, 2018. https://doi.org/10.1155/2018/7659085