PAPER Special Section on the Architectures, Protocols, and Applications for the Future Internet

Simulation Study of Low Latency Network Architecture Using Mobile Edge Computing*

Krittin INTHARAWIJITR[†a)], ***Student Member***, **Katsuyoshi IIDA**[††b)], ***Senior Member***, ***and*** **Hiroyuki KOGA**[†††c)], ***Member***

SUMMARY Attaining extremely low latency service in 5G cellular networks is an important challenge in the communication research field. A higher QoS in the next-generation network could enable several unprecedented services, such as Tactile Internet, Augmented Reality, and Virtual Reality. However, these services will all need support from powerful computational resources provided through cloud computing. Unfortunately, the geolocation of cloud data centers could be insufficient to satisfy the latency aimed for in 5G networks. The physical distance between servers and users will sometimes be too great to enable quick reaction within the service time boundary. The problem of long latency resulting from long communication distances can be solved by Mobile Edge Computing (MEC), though, which places many servers along the edges of networks. MEC can provide shorter communication latency, but total latency consists of both the transmission and the processing times. Always selecting the closest edge server will lead to a longer computing latency in many cases, especially when there is a mass of users around particular edge servers. Therefore, the research studies the effects of both latencies. The communication latency is represented by hop count, and the computation latency is modeled by processor sharing (PS). An optimization model and selection policies are also proposed. Quantitative evaluations using simulations show that selecting a server according to the lowest total latency leads to the best performance, and permitting an over-latency barrier would further improve results.

key words: *mobile edge computing, 5G, low latency, network architecture, processor sharing, and performance evaluation*

1. Introduction

Latency is a critical requirement for many mobile applications. As technologies have become more refined, lower service latency has been more strongly desired. Latency-sensitive services (e.g., real-time applications, streaming video, autonomous driving, and machine-to-machine communication) are of wide interest and demand is growing strongly [2]. Such services promise to improve the quality of life in many fields of human society; for example, we could control automobile traffic and prevent accidents through vehicle-to-machine communication [3], or we could use augmented reality (AR) to provide virtual information on a real environment in real time [4].

Achieving extremely low service latency will lead to many new mobile services. *Tactile Internet* [5]–[7] is one example. It will allow an application to interact with objects in either a real or a virtual environment remotely through the Internet, and do this with a very short-reaction time. Such quick interaction will need to be consistent with the natural sensory perceptions of humans; for example, the virtual reaction will have to be completed within about 10 ms, so a real-time service based on vision will require that each new frame be rendered within 10 ms [8] to avoid any lack of perceived continuous vision.

To satisfy the latency qualification, the fifth generation (5G) mobile network will need to provide a low-latency network architecture for the Tactile Internet [2], [6]. The 5G network is now being standardized to attain a higher level of QoS, and it should be able to deliver 1-ms round-trip latency as described in [7], [9], [10]. Such an ambition is a considerable challenge for network developers and researchers, but if a 5G architecture achieves the latency requirement, mobile users will be able to easily access Tactile Internet services.

To enable such low-latency applications, however, the 5G network must deal with the communication distance when serving the applications. Most applications in the Tactile Internet rely on cloud computing services to complete their heavy tasks in terms of computational complexity. For example, a mobile-AR application needs to send processing-heavy jobs (such as object identification or image processing) to cloud-based servers [11]–[13]. Nonetheless, as shown in Fig. 1, a cloud data center may sometimes be distant from a mobile user (e.g., in a far-away country) so

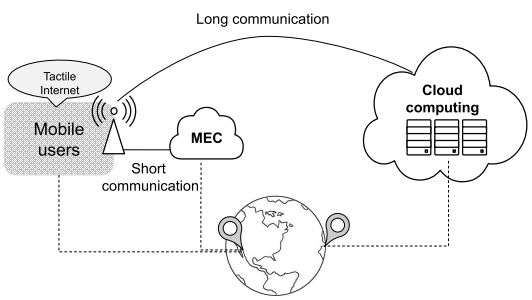


Fig. 1 Illustration of cloud and MEC

Manuscript received July 20, 2016.
Manuscript revised December 12, 2016.
Manuscript publicized February 8, 2017.

[†]The author is with the Dept. of Information and Communications Engineering, Tokyo Institute of Technology, Tokyo, 152–8852 Japan.
[††]The author is with the Information Initiative Center, Hokkaido University, Sapporo-shi, 060–0811 Japan.
[†††]The author is with the Dept. of Information and Media Engineering, University of Kitakyushu, Kitakyushu-shi, 080–0135 Japan.

*Earlier version of this paper has been presented in [1].

a) E-mail: intharawijitr@net.ict.e.titech.ac.jp
b) E-mail: iida@iic.hokudai.ac.jp
c) E-mail: h.koga@kitakyu-u.ac.jp

DOI: 10.1587/transinf.2016NTP0003

there could be a long transmission distance and high communication latency.

A novel technology has been introduced to solve this problem. As illustrated in Fig. 1, Mobile Edge Computing (MEC) [14], also known as Edge Cloud Computing (ECC) [15], has been proposed. The MEC concept is to bring a service to the edge of networks where users can access the nearest network component. The MEC enhances capability at the edge through a form of computational resources, called edge servers. A packet from clients will be sent to the mobile edge, but not passed through the core networks as the cloud architecture. This approach has the potential to provide very short communication between mobile users and service providers within the acceptable time interval.

Although, MEC can overcome distance issues, and unlock many amazing services, surveys [16], [17] have found that users and developers must still confront latency issues in MEC. Latency remains the critical metric with regard to latency-sensitive applications, and total latency is mainly composed of two kinds of latency. One is *communication latency*, the time needed for a workload to travel from a source to a destination (and in the opposite direction) via the wired and wireless communication in networks. The other is *computing latency*, the time spent on the computation in a server. To realize the latency needed to enable, for example, Tactile Internet, both types of latency must be considered to satisfy the service-time boundary.

Figure 2 illustrates the latencies in MEC networks. MEC shortens the communication route and decreases transmission latency. However, in contrast with cloud services, it can probably serve only a limited number of users. Numerous flows within the same area can affect one particular edge server and place a heavy computing load on that server. When that happens, MEC lengthens the processing time for all current approaches. Excessive computing latency is definitely unacceptable for low-latency service. However, in such cases there are often other edge servers supporting only a few users in surrounding areas. Taking a slightly longer path to a lightly loaded server could lessen the computing time, as shown in Fig. 2.

As a result, in this research, we have studied how the impact of computing time and communication time in the MEC architecture with regard to the demand from mobile users and the supply of edge servers in the network system under the latency constraint. This paper is organized as follows. In Sect. 2, we describe related work in terms of MEC to consider the state of the art and explain our motivations. In Sect. 3, we propose a model to analyze both types of latency: communication and computation. We then formulate an optimization model. In Sect. 4, we explain simulation experiments that we conducted as part of this work. In Sects. 5 and 6, we present and discuss numerical results from the simulation. We conclude and discuss our future work in Sect. 7.

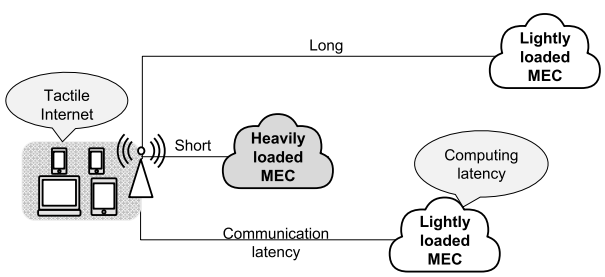


Fig. 2 Latency with MECs

2. Mobile Edge Computing (MEC) & Related Work

The definition and architecture of MEC are explained in this section. Additionally, existing research relating to this study is discussed briefly to show the difference between the existing research and this work.

2.1 Mobile Edge Computing (MEC)

The concept of *Mobile Edge Computing* is to distribute cloud capabilities to the edge of networks where they will be very close to local mobile users. Its main purpose is to reduce the network latency in a Radio Access Network (RAN). In addition, it can decrease the traffic in the core architecture by not forwarding packets into the core networks. At present, MEC is being discussed regarding the next evolution of cellular networks. As in [14], the European Telecommunications Standards Institute (ETSI) is standardizing MEC to fulfill the needs of both consumers and enterprises.

Several ideas are similar to MEC's underlying principle. *Cloudlets* [18] is a low latency network architecture supporting cognitive assistance; e.g., AR and image processing. Cloudlets is a virtual platform in the middle of networks (between the data center and users). *Edge Cloud Composites or Edge Cloud computing* (ECC) is also related to MEC [15]. It provides services at the edge, but is described from a perspective of software and supporting multiple users. Lastly, *fog computing* or *foggy* has been introduced by Cisco Systems [19], [20]. It is defined as *"an extended cloud computing at the edge of networks"*. The fog computing concept is to work through the cooperation of powerful edge devices in networks and is widely used in enterprises. An overview of MEC is shown in Fig. 3. The MEC architecture can be divided into three main groups

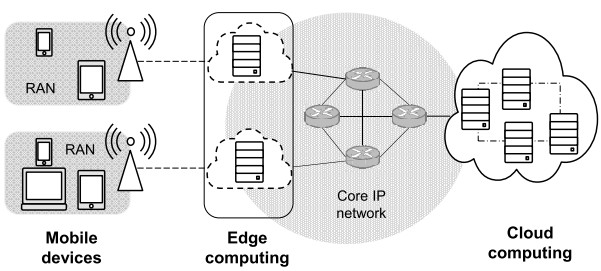


Fig. 3 Illustration of MEC architecture

for individual purposes and has different properties as described.

1. Mobile devices: All terminal devices, such as tablets, laptops, smart phones, which are connecting to a base station (BS) or an access point (AP) over the RAN will generate abundant tasks sent through the mobile network to gain service resources.
2. Mobile edge computing: A cluster of edge components – i.e., routers, switches, and gateways – is formed to perform as a virtual server node at the edge of networks by using a concept of network virtualization. Such a node can provide services as cloud computing in both computation and storage supply. Any request processed via the edge servers is not transmitted through a core network to reach a far away data center. However, a task really needing extensive cloud services should be forwarded to the destination as usual.
3. Cloud computing: A huge task desiring higher performance will be sent through the Internet to a cloud center although it might be located at a greater distance. Such a center comprises many ingenious servers to deal with such a global problem and big data analysis. The computation performance will be excellent, but a long transporting delay may be necessary.

2.2 Related Work

MEC will be a key technology in 5G cellular networks for the provision of a low latency network architecture. Work has been done to investigate the use of MEC in 5G networks using mathematical models [21]–[24]. Barbarossa *et al.* in [21] examine offloading computation in distributed mobile cloud computing over a 5G network and construct an optimal model of the transmit power consumption in MEC networks. Chen *et al.* in [22] also offer an algorithm of offloading distribution in MEC using game theory. They apply an optimization model to minimize overhead in the distribution. In [23], Sarkar *et al.* mathematically model MEC in the context of the Internet of Things for a lower-pollution system. Deng *et al.* in [24] design a mathematical model to analyze the trade-off between a conventional and an extended cloud by examining power consumption.

Much of the research into MEC has relied on mathematical models. However, in most cases the focus has been on energy consumption due to growth in the number of mobile devices and there has been less concern about latency. Some that consider latency as one factor in the model [23], [24] do not look at it as a major part and treat it rather simplistically. Martini *et al.* in [25] examines the latency of virtualization nodes in networks through an optimization model, but does not focus on the edge computing architecture.

In this paper, we also apply a mathematical model of MEC. Our goal is to illustrate how the network-edge can attain ultra-low latency under constraints of the time-service boundary and a 5G environment, but without considering energy utilization. We accordingly consider two types of network latency: *the computing latency and the communication latency.* Our previous model of the MEC concept, presented in [1], used a linear function to determine the processing time. This work extends the model by including Processor Sharing (PS) to demonstrate the computation time within a more realistic framework.

3. Mathematical Model

As we explained, the purpose of our paper is to develop a mathematical model that includes both communication latency and computing latency. In this section, we first define the problem, as well as communication and computing latency. After that, we introduce latency optimization models: strict and permissive underestimation systems as explained in Sect. 3.5. Finally, we introduce policies on how we select mobile edge servers if we have multiple candidates.

3.1 Problem Definition

To design a low latency network architecture, we define a problem model as shown in Fig. 4. A group of mobile devices connecting to the same base station (BS) is represented as one *source node*. We specify a set of source nodes, $\mathcal{S} = \{S_1, S_2, \ldots, S_i, \ldots, S_N\}$. Each S_i produces workloads according to a Poisson process with rate λ_i ($\lambda_i > 0$). Here, we regard *a workload* as a computational task, which should be performed by a mobile edge node in Fig. 4. Our latency of interest is the sum of 1) round-trip communication latency between the source node and the selected mobile edge node, and 2) computational time required by the mobile edge node. Note that we do not directly consider mobile devices themselves, but since they connect to a source node with very small latency this is not a serious omission.

For the service provider, mobile edge servers considered as *edge nodes* are denoted by a set $\mathcal{E} = \{E_1, E_2, \ldots, E_j, \ldots, E_M\}$. Furthermore, the amount of service demand (or workload size) in edge node E_j is assumed to be independent and identically distributed according to an exponential distribution with mean μ_j^{-1}. Since we assume a PS queue (described later in Sect. 3.3), the actual service time is determined by the number of workloads currently accepted in the queue.

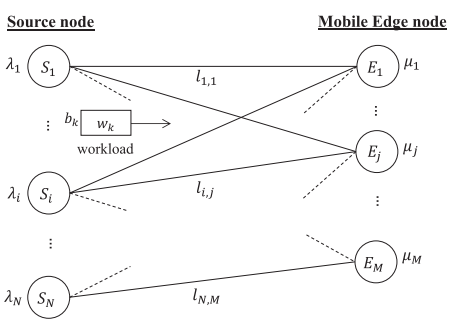


Fig. 4 Problem model

Workloads are also defined as a set $\mathcal{W} = \{w_1, w_2, \ldots, w_k, \ldots, w_K\}$, where w_k obtains b_k as a size. Here, w_1 is the first workload which can be generated from one of the source nodes in $\mathcal{S}$ independently with ordering, while w_K is the last workload pushed into the system.

3.2 Communication Latency

According to Fig. 4, a source node S_i that acquires services from the edge node E_j will transmit its workload w_k by a link having propagation delay $l_{i,j}$. This delay stands for the round-trip time, excluding the computation time in an edge node, and is determined by the hop count between S_i and E_j.

The hop count, corresponding to the communication distance, is given by the difference between the IDs of nodes. For example, E_j is located one hop away from S_i when $i = j$, and 2 hops away from $E_{j\pm1}$, where $E_{j\pm1} \in \mathcal{E}$. Given that E_1 and edge node E_M are one hop apart, S_1 and E_M consequently have a two-hop distance. In a case where $N > M$, S_i (where $i > M$) is considered to have a hop count like a source $S_{(i \mod M)+1}$. We always determine the shortest distance from the source to the edge node.

In addition, if we let l_h be the unit of hop delay and $H(i, j)$ be the hop count from S_i to edge E_j, we can determine $l_{i,j}$ from the following:

$$l_{i,j} = l_h \times H(i, j), \quad S_i \in \mathcal{S},\ E_j \in \mathcal{E}. \tag{1}$$

3.3 Computation Latency

The computation time is influenced by the number of workloads in the real server, so we need to make some realistic assumptions to model the actual computation time. We follow the concept of processor sharing (PS), also called time sharing, [26]–[28] to represent the service time. With a single server, PS characterizes the amount of service time according to the actual load. The capability of a server is fixed. Each present workload in the system shares resources controlled by weights [26]. In this study, we consider all workloads to have identical weights. That causes each workload to take an equal fraction of service. Additionally, when a workload arrives at a server, it can be started without any queuing time, and the server will allocate its capacity to a newly arriving workload equally with respect to other current processes.

Following the PS principle, let workloads that are accepted and served by E_j at time $t \geq 0$ be in a set $\mathcal{A}_j^t$, and the number of workloads in edge E_j at time t be $n_j(t) = |\mathcal{A}_j^t|$. All workloads $w_k \in \mathcal{A}_j^t$ will be individually processed with $\mu_j / n_j(t)$ instructions per time unit. Also, we define that $\mathcal{A}_j = \bigcup_{t=0}^{\infty} \mathcal{A}_j^t$ for a set of all accepted workloads at edge node E_j, and $\mathcal{A} = \bigcup_{j\in\mathcal{E}} \mathcal{A}_j$ for the total of accepted workloads in the system.

A new workload produced from a source node must look for an edge node to get a service. The processing needed to accept new workloads will increase the computing latency. However, we cannot determine an exact value for this latency because of the PS discipline. Defining $P_{k,j}(t)$ as the estimated computing latency of workload w_k at time t served by E_j, we propose an estimation method by considering the latency based on the value of $n_j(t)$:

$$P_{k,j}(t) = b_k \left(\frac{n_j(t) + 1}{\mu_j} \right), \quad w_k \in \mathcal{W},\ E_j \in \mathcal{E}. \tag{2}$$

3.4 Total Latency

After the communication and computing latency are designated, the following can be stated:

$$L_{i,j,k}(t) = \begin{cases} l_{i,j} + P_{k,j}(t), & w_k \in \mathcal{A}_j^t \quad \text{(3a)} \\ 0, & \text{otherwise} \quad \text{(3b)} \end{cases}$$

$$S_i \in \mathcal{S},\ E_j \in \mathcal{E},\ t \geq 0.$$

The absolute latency can be determined by Eq. (3). It becomes equal to the aggregate of both the communication and computing latency in the case of (3a), if w_k can be executed by edge E_j. In addition, if w_k is not received by any nodes, it will be abandoned without latency (3b).

3.5 Optimization Model

To enable extremely low latency in a 5G network with MEC, we have formulated an optimization model to evaluate the system performance and system conditions. First, we define a set $\mathcal{R} = \mathcal{W} - \mathcal{A}$, collecting all rejected workloads which no edge node can support.

In addition, each mobile device prescribes its service-latency tolerance. We accordingly consider that value as the maximum allowance θ of total latency. This means if the estimated total latency $L_{i,j,k}(t)$ will exceed θ, we should reject the workload.

Regarding the computing latency with PS, estimation leads to overestimation or underestimation in the system. In the case of overestimated latency, the server finishes the workload before the estimated time and does not violate the latency maximum allowance of total latency. In contrast, underestimation can result in the actual latency being greater than the service tolerance. Consequently, we reconsider

$$\begin{aligned} &\delta_{a,j}(t) = P_{a,j}(t) - P_{a,j}(t'); \\ &\quad t' < t,\ w_a \in \mathcal{A}_j^t,\ E_j \in \mathcal{E}, \end{aligned} \tag{4}$$

where $\delta_{a,j}(t)$ denotes the time gap between estimations in the previous period t' and at the current time t, and w_a is the current workload processed in a server at time t.

3.5.1 Strict Underestimation System

To protect against underestimation in the computing latency estimation, the optimization model must check all processing workloads in the edge node to ensure that an incoming

workload will not cause the maximum allowance (θ) of all processing workloads in the server to be exceeded. Thus, we propose the following objective function and constraints:

$$\min_{b_r, b_k} \quad P_b = \frac{\sum_{w_r \in \mathcal{R}} b_r}{\sum_{w_k \in \mathcal{W}} b_k} \tag{5}$$

$$\text{subject to} \quad L_{i,j,a}(t) \le \theta, \quad \forall w_a \in \mathcal{A}_j \tag{6}$$

$$L_{i,j,e}(t') + \delta_{e,j}(t) \le \theta, \ \forall w_e \in \mathcal{A}_j^t \tag{7}$$

$$n_j(t) \ge 0 \tag{8}$$

$$l_h, \ \forall \lambda_i, \ \forall \mu_j, \ \forall b_k > 0 \tag{9}$$

$$t > t' \ge 0 \tag{10}$$

$$\forall S_i \in \mathcal{S}, \ \forall E_j \in \mathcal{E}, \ \forall w_k \in \mathcal{W}, \ \forall w_r \in \mathcal{R}.$$

This objective function (5) tries to achieve the minimum *blocking probability* (P_b) defined as a fraction of the total size of rejected workloads in all workloads. Before meeting the objective, a system is required to first satisfy all constraints (6), (7), (8), (9), and (10).

Here, (6) requires that the total latency of an accepted workload w_a, composed of the computing delay and the communication delay, must not exceed the maximum allowance θ. However, (7) requires that a newly arriving workload w_a not affect any already accepted workload w_e currently being processed by the edge node. Restrictions (8), (9), and (10) require that all parameters have non-negative values as shown in the constraints.

3.5.2 Permissive Underestimation System

The strict system in Sect. 3.5.1 has very tight constraints in that it does not allow errors with respect to going beyond the bounds of the maximum allowance θ. Checking all existing workloads in the system, though, would require considerable system effort. Loosening some restrictions could reduce the amount of constraint checking. A system that ignores constraint (7) is called a permissive underestimation system. Such a system will accept and process all workloads satisfying the remaining conditions, though other current workloads could be outside of the settled maximum allowance. Thus, the optimization model can be given as

$$\begin{aligned} &(5) \\ &\text{subject to} \quad (6), (8), (9). \end{aligned} \tag{11}$$

A permissive underestimation system may improve the blocking probability, but at the cost of underestimation error.

3.6 Policies

When the new workload w_k arrives, we need to select one edge node to accommodate the workload if we do not reject it. For this purpose, we create a list of candidate edge node, each of which meets the constraints stated in Sects.3.5.1 and 3.5.2. If we have multiple candidate edge node, we need to select one. We therefore have to consider a policy stating how we should select it. In this study, we have considered the three policies given below.

- *Random policy* (Random): A simple way to choose the accepting edge node from the candidate list. One edge is randomly selected according to a uniform distribution to execute a workload.
- *Lowest latency policy* (Low. latency): Comparing the absolute latency in (3a) for each candidate, the edge node providing the lowest total latency given the current state of the system is selected.
- *Minimum remaining time policy* (Min. time): The remaining time is given by the definition of how long before the workload will be completed. Giving ζ_k^t as an expected finishing time of w_k at time t, the total remaining time of E_j at time t is denoted by

$$T(j,t) = \sum_{w_k \in \mathcal{A}_j^t} \left(\zeta_k^t - t\right), \quad t \ge 0, \ E_j \in \mathcal{E}. \tag{12}$$

This policy will select an edge node with the minimum $T(j,t)$. The remaining time can indicate the amount of available server resources.

4. Simulation

Our objective in this paper is to analyze the computing and communication latency in the MEC architecture. In this section, we discuss the simulation experiments based on the mathematical model developed in the previous section that were done to obtain numerical results.

4.1 Work Flow of Simulation

We used C++ programming language to develop the simulation used to demonstrate a model with variable parameters. In the program, we classify events into three types.

Event I: The first type of event is to produce a workload. A source node indicated in this event will push its new workload into a system. Every edge node satisfying all restrictions will be considered as a candidate for processing a task. The program selects the target node with regard to the defined policies.

Event II: The second type of event is where the selected edge node receives an incoming workload. The edge node starts a new task immediately by fairly sharing resources from other existing processes. All remaining workloads will extend their expected finishing time because PS provides them with a lower service rate proportion.

Event III: In the third type of event, the edge node releases a workload that is fully processed. The resultant free resource will be used to compute a new service rate for individual current processes. Then, each remaining workload can adjust its expected finishing time to an earlier one.

Each source generates its own first event (Event I) by random time according to a Poisson process. The simulation gets a running event with time priority. For any Event I, the

edge node creates the next Event I. It also produces Event II if a new workload is accepted by some edges. For Event II, Event III always follows automatically with the expected time to finish a workload. However, the Event III finishing time varies depending on arriving and finished workloads.

Eventually, we end the simulation when the system stays in the steady state, where the result remains unchanged even as the simulation time passes. To ensure the steady state, we run the simulation for 3000 s. Furthermore, we ignore results from the first 200 s to avoid a transient state in the warm-up time which can bias the final result from the steady state.

4.2 Evaluation Metrics

Besides simulating MEC, we collected further data for evaluation purposes while running the simulation. Metrics obtained while measuring the model can be used to represent system performance. In this study, we evaluate our proposed model by the following three metrics.

- *Blocking probability* (P_b): This probability is defined in Eq. (5). We measure this value in order to show how effective the system can support demand from users. Both strict and permissive systems need to use this metric to evaluate performance.
- *Decision error* (ϵ): According to Sect. 3.5.2, PS that allows underestimated time can induce an actual latency over the maximum allowance for latency. Measurement of these faulty latencies, as a result, is necessary for analysis of the model. Considering that all accepted workloads are labeled as either suitable and unsuitable acceptance, we define a set $\mathcal{V}$ gathering proper workloads, and a set $\mathcal{I}$ gathering improper workloads; in other words, we can say $\mathcal{A} = \mathcal{V} \cup \mathcal{I}$. Here, we express the decision error as

$$\epsilon = \frac{\sum_{w_u \in \mathcal{I}} b_u}{\sum_{w_a \in \mathcal{A}} b_a}. \tag{13}$$

- *Modified blocking probability* (P'_b): This is the probability unlike the pure blocking probability defined in Eq. (5). However, we cannot fairly compare system performance in both strict and permissive underestimation. Therefore, an alternative metric must be considered. The permissive system has an wrong decision that accepts an unsuitable workload. If we reconsider all rejected and unsuitable workloads as the new metric of the system, we get

$$\begin{aligned} P'_b &= \frac{\sum_{w_r \in \mathcal{R}} b_r + \sum_{w_u \in \mathcal{I}} b_u}{\sum_{w_k \in \mathcal{W}} b_k} \\ &= P_b + \epsilon - P_b \cdot \epsilon. \end{aligned} \tag{14}$$

Eventually, the strict underestimation system has $\epsilon = 0$ ($P_b = P'_b$), so we can correlate both systems with the modified blocking probability of the permissive system and the pure blocking probability of the strict system.

We note that decision error (ϵ) and modified blocking probability (P'_b) can exist in the permissive system only.

5. Numerical Results

In this section, we show numerical results for the strict and permissive underestimation systems with respect to the evaluation metrics. In particular, we concentrate on the effects when we allow a longer maximum allowance for service latency and more edge nodes.

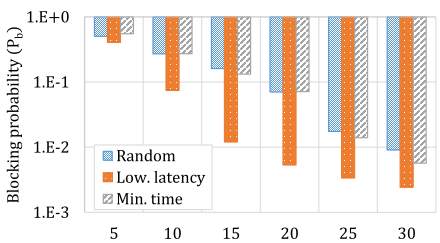
Parameters and default values used in the simulation are shown in Table 1. We assume that each workload has an equal size, every source node produces workloads at the same rate, and all edge nodes provide an identical service rate. We fix 10 source nodes in the simulation for all results. We vary the number of edge nodes for diffusion of supplies. Moreover, the maximum service latency allowance is also determined for various values.

5.1 Strict Underestimation

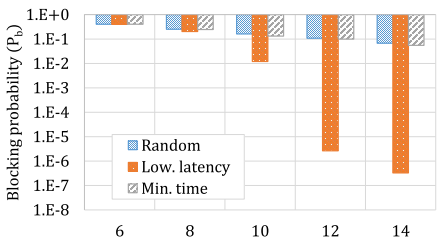
Results for the strict system are shown in Fig. 5. A bar chart in Fig. 5 (a) shows the impact of maximum allowance

Table 1 Simulation parameters

Parameter	Description	Value
N	Number of source nodes	10
M	Number of edge nodes	6-14
λ	Producing work rate of a source node	2 workloads/ms
μ	Processing rate of an edge node	32 billion instructions/s
b	Size of a workload	16 million instructions
l_h	Hop delay	2 ms
θ	the maximum service latency allowance	5–30 ms

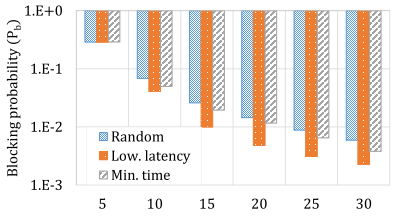


(a) Impact of maximum allowance θ ($M = 10$)

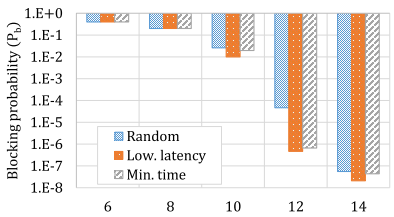


(b) Impact of number of edges M ($\theta = 15$)

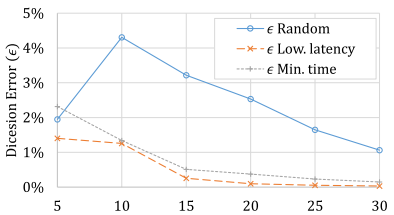
Fig. 5 Blocking probability P_b of strict underestimation



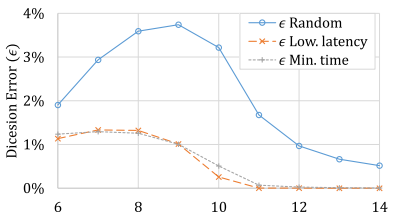
(a) Impact of maximum allowance θ on blocking probability P_b (M = 10)



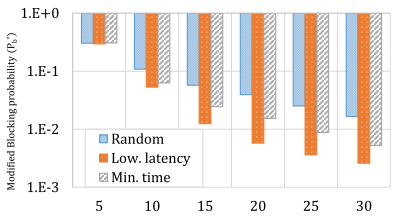
(b) Impact of number of edges M on blocking probability P_b (θ = 15)



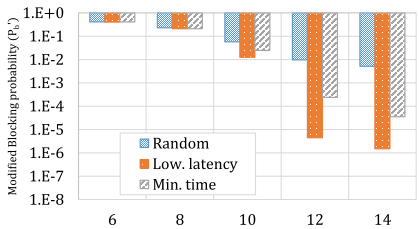
(c) Impact of maximum allowance θ on decision error ϵ (M = 10)



(d) Impact of number of edges M on decision error ϵ (θ = 15)



(e) Impact of maximum allowance θ on modified blocking probability P_b' (M = 10)



(f) Impact of number of edges M on modified blocking probability P_b' (θ = 15)

Fig. 6 Numerical results for permissive underestimation

θ over a wide range when we set $N = M = 10$. The Y-axis represents blocking probability (P_b) on a logarithmic scale. When we increase the maximum allowance, the system can support more workloads from sources and lower the blocking probability. A higher maximum allowance for latency permits a workload to be processed and transmitted within a longer time. The workload can then access farther edge nodes if near nodes cannot accept it. Additionally, the edge node can hold each workload longer and collect more workloads.

Comparing the three policies, the lowest latency policy provides the least blocking in all cases, especially with a higher maximum allowance. This is because the lowest total time consists of low communication delay and low computing delay. The short transmission restricts a source node from sending its workload to near edge nodes. In addition, quicker processing in the edge node releases workloads so quickly that the node can take on more workloads. The other policies, random and minimum remaining time, do not perform as well as the lowest latency policy. Both policies obviously require more time with respect to total latency per workload. With the restriction of underestimated error, workloads are always rejected when there are too many workloads held in all edge nodes, because if the system accepts more workloads, the computing time of existing workloads will grow longer and the delay will become excessive.

In addition, the random and minimum remaining time policies give very similar results. They all have the ability to distribute workloads to edge nodes equitably according to a different principle. The random policy selects a target node randomly by uniform distribution while the minimum remaining time policy instead prefers the highest available resource. Eventually, both policies lead to all edges nodes having a similar workload amount on average.

The blocking probability with various numbers of edge nodes when θ = 15 ms is shown in Fig. 5 (b). More edge nodes lead to a lower blocking probability due to the larger supply of available nodes for a fixed amount of demand. Furthermore, the impact of more edge nodes in the networks shows that the lowest latency policy dominates the other policies. Clearly, adding more edge nodes means we add not only more resources, but also more distant edge nodes. The lowest latency is attained by using the extended nodes as auxiliaries. In this case, more distant nodes obtain workloads when the near edge nodes are highly overloaded, but the workload amount on the more distant nodes is less than on the near edge nodes. In contrast, the two other policies try to balance workloads between the near and the added nodes without considering the transmission time.

5.2 Permissive Underestimation

Figure 6 shows numerical results for the permissive system. As for the strict system, the blocking probability (on the Y-axis with a logarithmic scale) is shown as a function of the maximum allowance of latency in Fig. 6 (a) and of the number of edge nodes in Fig. 6 (b). Comparing the results from both charts shows that the blocking probabilities of the random and minimum remaining time policies are improved by a high maximum allowance and by more edge nodes, while the performance of the lowest latency policy changes little. Without checking all present workloads before acceptance, with either of these policies the system can process more tasks. The quick response of the lowest latency policy, however, enables acceptance of a new workload without greatly affecting existing workloads. That is why the performance of the permissive system does not significantly change.

As mentioned, the permissive underestimation leads to

decision error (ϵ). Figures 6 (c) and 6 (d) respectively show the decision error as a function of the maximum allowance and of the number of edge nodes. Both figures show that the random policy has a very high error rate, exceeding that of the other policies in most cases. The random policy can lead to a wrong decision because of the high variance caused by non-pattern selection. The lowest latency policy always holds any workload for a very short time. This helps to ensure that a new decision has little impact on other workloads. Surprisingly, the error of the minimum remaining time policy is as low as that of the lowest latency policy. This is because the policy keeps the number of workloads in each edge node quite constant through more relaxed constraints. Hence, it can always get a precise estimated value with minor error.

Figures 6 (c) and 6 (d) show a peak for the random policy. In case of insufficient resource (like $M \leq 10$ or $\theta \leq 10$), the random policy leads higher decision error when the system provides more candidates because it randomly selects a target without logical strategy. Thus, when we increase either the number of workloads or extend the maximum allowance, the system can get more candidates that induce more wrong decisions. On the other hand, if there are sufficient resources, the candidates mostly have potential to complete a workload in time so the random policy can get more accurate decision. Besides, the lowest latency and minimum remaining time policies could meet the same situation as the random policy if we consider Fig. 6 (d) but the number of candidates dose not impact the decision as the random policy.

The results for modified blocking probability (P'_b) are shown in Fig. 6 (e) and Fig. 6 (f) as a function of the maximum allowance and the number of edge nodes, respectively. Here, the differences between the strict and permissive systems are explained and analyzed. Since the both systems cannot be fairly compared with using the blocking probability because of the decision error in the permissive system, we have to compare the blocking of the strict system in Fig. 5 and the modified blocking of the permissive system in Figs. 6 (e) and 6 (f) to see the true performance and effectiveness of each system. Figures 5 (a) and 6 (e) show that the permissive system does not impressively improve the performance of the strict system in case of increasing the maximum allowance of latency. Furthermore, the performance of permissive system with the random policy probably deteriorates from the strict system due to a high error rate in some cases ($\theta \geq 25$). As shown in Fig. 6 (f), permissive underestimation can enhance system efficiency for two policies when we increase the number of edge nodes. Unfortunately, performance with the lowest latency deteriorates. Because its true blocking probability is very low, when the system allows some errors its effectiveness should also deteriorate. In any case, the lowest latency policy still dominates the others.

6. Discussion

In the previous section, we have shown numerical results from measurements of the model in both a strict and a permissive system. Adding edge nodes can more effectively influence the probability of blocking than increasing the maximum allowance. When we select policies, the lowest latency can usually produce the best results in all situations. However, with permissive underestimation, the system will have some decision errors that might lower the quality of service. The lowest latency still provides the lowest blocking probability.

In terms of the implementation cost, the strict system will conceivably have a higher cost than the permissive system. The strict constraint that requires examination of all existing workloads puts a great deal of load on the system. To decrease the cost and system load, a permissive method can instead be developed. Additionally, with regard to selecting policies, random selection is obviously a simple approach, but at the cost of high error rates and poor performance. Examining the remaining time needed for every workload in the server seems excessive. Calculation of the communication and computing time should require less system effort.

The results indicate that the best edge node to support a workload from a mobile user is the closest one. This enables the shortest propagation delay, and leaves more time for computation in a server. However, if there are other edge nodes in the network that have lighter loads, we can instead select these nodes if they will provide lower total latency.

The combination of determining total latency and relaxing some restrictions looks like an attractive way to provide services at the edge of a network. For example, visual services can access local services rapidly. However, performance also depends on the client application. If a source cannot tolerate any errors, we need to reconsider the strict condition as an alternative.

A final point to note is that our proposed system has to rely on a centralized controller, which collects all necessary information from sources and edges through probe messages. The controller then uses the gathered data to decide which edge node is best to assign to each workload. We plan to investigate this consideration in detail in our future work.

7. Conclusion

MEC is needed to enable a low-latency network architecture. In this research, we have proposed a mathematical model of MEC using the PS concept to estimate the computing time in edge nodes. Strict and permissive underestimation conditions in the optimization model were tested to analyze the system. We then developed three policies for selecting an edge node when several nodes satisfy the system constraints.

The numerical results show that determining the lowest latency policy provides the best performance even though there might some decision errors. Furthermore, a permissive system could decrease the implementation cost. In our work, we intend to extend this research to implementation in a real environment to test the practicality of our concept.

Acknowledgements

This work was supported in part by JSPS KAKENHI Grant-in-Aid for Scientific Research (B) Number 16H02806.

References

[1] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5G cellular networks," Proc. IEEE Int'l Conference on Pervasive Computing and Communication Workshops (PerCom Workshop 2016), Sydney, NSW, Australia, pp.1–4, March 2016.

[2] A. Aijaz, M. Dohler, A.H. Aghvami, V. Friderikos, and M. Frodigh, "Realizing the tactile internet: Haptic communications over next generation 5G cellular networks," IEEE Wireless Commun., pp.2–9, 2016.

[3] A.F. Cattoni, D. Chandramouli, C. Sartori, R. Stademann, and P. Zanier, "Mobile low latency services in 5G," Proc. IEEE Vehicular Technology Conference (VTC Spring 2015), Glasgow, UK, pp.1–6, May 2015.

[4] K.Č. Pucihar and P. Coulton, "Exploring the evolution of mobile augmented reality for future entertainment systems," ACM Computers in Entertaint, vol.11, no.2, pp.1–16, Jan. 2015.

[5] G.P. Fettweis, "The tactile internet: Applications and challenges," IEEE Vehicular Technology Mag., vol.9, no.1, pp.64–70, March 2014.

[6] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-enabled tactile internet," IEEE J. Selected Areas in Communications, vol.34, no.3, pp.460–473, March 2016.

[7] M. Maier, M. Chowdhury, B.P. Rimal, and D.P. Van, "The tactile internet: Vision, recent progress, and open challenges," IEEE Commun. Mag., vol.54, no.5, pp.138–145, May 2016.

[8] E. Azañón and S. Soto-Faraco, "Changing reference frames during the encoding of tactile events," Current Biology, vol.18, no.14, pp.1044–1049, July 2008.

[9] J.G. Andrews, S. Buzzi, W. Choi, S.V. Hanly, A. Lozano, A.C.K. Soong, and J.C. Zhang, "What will 5G be?," IEEE J. Selected Areas in Communications, vol.32, no.6, pp.1065–1082, June 2014.

[10] ITU-R, "IMT vision: Framework and overall objectives of the future development of IMT for 2020 and beyond," Recommendation ITU-R M.2083-0, Sept. 2015.

[11] M. Chen, C. Ling, and W. Zhang, "Analysis of augmented reality application based on cloud computing," Proc. IEEE Int'l Congress on Image Signal Processsing (CISP2011), Shanghai, China, pp.569–572, Oct. 2011.

[12] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," IEEE Communications Surveys & Tutorials, vol.16, no.1, pp.337–368, First Quarter 2014.

[13] Z. Huang, W. Li, P. Hui, and C. Peylo, "CloudRidAR: A cloud-based architecture for mobile augmented reality," Proc. ACM Workshop on Mobile Augmented Reality and Robotic Technology-based Systems (MARS2014), Bretton Woods, NH, USA, pp.29–34, June 2014.

[14] Y.C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," ETSI White Paper, no.11, Sept. 2015.

[15] K. Bhardwaj, S. Sreepathy, A. Gavrilovska, and K. Schwan, "ECC: Edge cloud composites," Proc. IEEE Int'l Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud2014), Oxford, UK, pp.38–47, April 2014.

[16] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," Proc. ACM Workshop on Mobile Big Data (Mobidata2015), Hangzhou, China, pp.37–42, June 2015.

[17] N. Fernando, S.W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," Future Generation Computer Systems, vol.29, no.1, pp.84–106, Jan. 2013.

[18] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: At the leading edge of mobile-cloud convergence," Proc. IEEE Int'l Conference on Mobile Computing, Applications and Services (MobiCASE2014), Austin, TX, USA, 9 pages, Nov. 2014.

[19] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," Proc. IEEE Australasian Telecommunication Networks and Applications Conference (ATNAC2014), Southbank, VIC, Australia, pp.117–122, Nov. 2014.

[20] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," Proc. ACM Workshop on Mobile Cloud Computing (MCC2012), Helsinki, Finland, pp.13–16, Aug. 2012.

[21] S. Barbarossa, S. Sardellitti, and P.D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," IEEE Signal Processing Mag., vol.31, no.6, pp.45–55, Nov. 2014.

[22] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," IEEE/ACM Trans. Networking, vol.24, no.5, pp.2795–2808, 2016.

[23] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," IEEE Trans. Cloud Computing, 14 pages, Oct. 2015.

[24] R. Deng, R. Lu, C. Lai, and T.H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," Proc. IEEE Int'l Conference on Communications (ICC2015), London, UK, pp.3909–3914, June 2015.

[25] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of virtual functions in 5G," Proc. IEEE Conference on Network Softwarization (NetSoft2015), London, UK, pp.1–6, April 2015.

[26] G. Fayolle, I. Mitrani, and R. Iasnogorodski, "Sharing a processor among many job classes," J. ACM, vol.27, no.3, pp.519–532, July 1980.

[27] E.G. Coffman, R.R. Muntz, and H. Trotter, "Waiting time distributions for processor-sharing systems," J. ACM, vol.17, no.1, pp.123–130, Jan. 1970.

[28] S.F. Yashkov, "Processor-sharing queues: Some progress in analysis," Queueing Systems, vol.2, no.1, pp.1–17, March 1987.

Krittin Intharawijitr received the B.E., M.E. degrees in Computer Engineering from Chulalongkorn University, Bangkok, Thailand in 2013, Communications and Computer Engineering from Tokyo Institute of Technology, Tokyo, Japan in 2016 respectively. Presently, he is a Doctoral course student at Tokyo Institute of Technology, Japan. His research interests lie in the fields of network architecture, mobile networks, and cloud computing.

Katsuyoshi Iida received the B.E., M.E. and Ph.D. degrees in Computer Science and Systems Engineering from Kyushu Institute of Technology (KIT), Iizuka, Japan in 1996, in Information Science from Nara Institute of Science and Technology, Ikoma, Japan in 1998, and in Computer Science and Systems Engineering from KIT in 2001, respectively. Currently, he is an Associate Professor in the Information Initiative Center, Hokkaido University, Sapporo, Japan. His research interests include network systems engineering such as network architecture, performance evaluation, QoS, and mobile networks. He is a member of the WIDE project and IEEE. He received the 18th TELECOM System Technology Award, and Tokyo Tech young researcher's award in 2003, and 2010, respectively.

Hiroyuki Koga received the B.E., M.E. and D.E. degrees in computer science and electronics from Kyushu Institute of Technology, Japan, in 1998, 2000, and 2003, respectively. From 2003 to 2004, he was a postdoctoral researcher in the Graduate School of Information Science, Nara Institute of Science and Technology. From 2004 to 2006, he was a researcher in the Kitakyushu JGN2 Research Center, National Institute of Information and Communications Technology. From 2006 to 2009, he was an assistant professor in the Department of Information and Media Engineering, Faculty of Environmental Engineering, University of Kitakyushu, and then has been an associate professor in the same department since April 2009. His research interests include performance evaluation of computer networks, mobile networks, and communication protocols. He is a member of the ACM and IEEE.