**Master's Thesis**

Title

# Adaptive Signaling Control Method for Efficient Resource Utilization of Mobile Core Networks

Supervisor

Professor Morito Matsuoka

Author

Tomoya Adachi

February 5th, 2020

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Adaptive Signaling Control Method
for Efficient Resource Utilization of Mobile Core Networks

Tomoya Adachi

## Abstract

Mobile network operators need to adaptively allocate server resources to mobile core nodes according to the number of accommodating devices and their communication characteristics. On the other hand, it is difficult to predict the number of connected IoT devices that have been increasing in recent years. Such IoT devices have various communication characteristics that cannot also be predicted. In addition, to reduce signaling procedure at the beginning of communication, a state called RRC Connected Inactive is considered, where the information of a device remains stored in mobile core nodes while the device itself is disconnected from the network. Consequently, it is expected that CPU load and memory usage on mobile core nodes would fluctuate greatly, causing the efficient allocation of server resources to be more difficult.

In this thesis, instead of dynamically allocating server resources according to the demand, we consider controlling the demand itself to balance the server resource utilizations and increase the capacity of the mobile core network. For that purpose, a method to adjust control parameters in the signaling procedure is proposed. Specifically, CPU load and memory consumption of mobile core nodes are controlled by adjusting the timeout value for the device's state transition to Idle state. The potential of the proposed method to control CPU and memory resources is evaluated by mathematical analysis. Numerical examples show that the number of accommodated devices could increase up to around 150 % without increasing server resources of mobile core nodes.

However, the appropriate timeout value cannot be determined easily, because the fluctuation of the number of accommodating devices and their communication characteristics brings various changes in the server resource demand. Therefore, we introduce a method to dynamically adjust the timeout value based on CPU load and memory consumption of mobile core nodes, so that the number of devices that can be further accommodated is maximized. A simulation results show

that the method can avoid CPU and memory resources from becoming short when the number of accommodating device changes.

**Keywords**

Mobile Core Network
Internet of Things (IoT)
Resource allocation
RRC Connected Inactive
PID control

# Contents

# List of Figures

# List of Tables

# 1 Introduction

The mobile network operator needs to adaptively allocate server resources to mobile core nodes according to the number of accommodating devices and their communication characteristics. CPU and memory are the main server resources of the mobile core nodes. The CPU handles various signaling procedures such as attach, detach, and authentication. On the other hand, the memory is mainly used to hold session information for devices such as context of devices and corresponding bearers. These server resources are indispensable for providing mobile network services.

On the other hand, the rapid increase of IoT devices has been paid attention in recent years. IoT devices are used in various places and for various applications such as home appliances, automobiles, electric meters, and GPS trackers. The number of IoT devices which connect to mobile core networks varies greatly depending on applications. In addition, most of IoT devices transmit data periodically and intermittently that is different from conventional devices such as smartphones. Therefore, it is difficult for mobile network operators to predict the number of connected IoT devices and their communication characteristics when introducing a mobile core network. Moreover, IoT devices can transit between Idle state and Connected state every time they send data because of their large communication intervals. Because such state transitions require signaling procedure in the mobile core nodes, the control plane, which performs communication and processing related to signaling for the device's state transition, suffers from large load when massive IoT devices are accommodated. For resolving this problem, a state called RRC Connected Inactive is considered, as one possible technique for reducing the signaling procedure, especially for IoT devices [2,3]. In RRC Connected Inactive state, the information of a device remains stored in mobile core nodes while the device itself is disconnected from the network, that would change the utilization of server resources of mobile core nodes.

Because of the above-mentioned issues, it is expected that CPU load and memory usage on mobile core nodes would fluctuate greatly, causing the efficient allocation of server resources to be more difficult. Against such increasing and varying demand, auto scaling technologies of physical and virtual servers have been proposed and have already been deployed in the actual environment [4–8]. However, they may not utilize server resources efficiently against the imbalanced resource demand, because the resource configuration per server or instance is generally fixed. In [5], the authors present that, in a network with many IoT devices, Mobility Management En-

tity (MME) scaling can cause unnecessary higher CPU and memory resource provisioning than actually required.

On the other hand, Server Disaggregation technologies have been proposed, allowing server resources to be isolated and individually configured [9–14]. In [9], the authors presented that better resource utilization can be achieved using disaggregated hardware in data centers. However, Server Disaggregation is an architecture for long-term server re-organization on a yearly basis, and is not a method aimed at responding to short-term load fluctuations focused in this thesis. In [14], the authors show that the delay and cost overhead associated with the process of reallocating disaggregated server resources can increase when resources are controlled on a time scale of several hours or less.

In this thesis, instead of dynamically allocating server resources according to the demand, we consider controlling the demand itself to balance server resource utilizations and increase the capacity of the mobile core network. For this purpose, a method to control state transitions of connected devices is proposed. Specifically, CPU load and memory consumption of mobile core nodes are controlled by configuring the timeout value for the device's state transition to Idle state. This is achieved by introducing a new state transition from RRC Connected Inactive state to Idle state. The potential of the proposed method to control CPU and memory resources is evaluated by mathematical analysis. In the numerical examples of the analysis, the numerical parameters from actual servers and previous experimental results are utilized to confirm the effectiveness of the proposed method.

However, the appropriate timeout value cannot be determined easily. This is because the fluctuation of the number of accommodating devices and their communication characteristics bring various changes in the server resource demand. Therefore, a control theory-based method is introduced to dynamically adjust the timeout value based on CPU load and memory consumption of mobile core nodes, so that the number of devices that can be further accommodated is maximized. The performance of the proposed method is evaluated on our self-made simulator. Specifically, the accuracy of the dynamic adjustment of the timeout value based on the consumption of server resources is presented. In addition, the effectiveness of the proposed method is confirmed by comparing with results without using the proposed method.

The rest of this thesis is organized as follows. Section 2 shows existing works related to this thesis. Section 3 explains mobile core network architecture, signaling procedures, and device's

state transition. Section 4 explains proposed method. Section 5 reveals the potential of the proposed method to control CPU and memory resources. Section 6 shows the results of the simulation and discussions. Finally, Section 7 concludes this thesis and presents some directions for future research.

## 2 Related Work

Various methods have been proposed to improve the capacity of mobile core networks.

In [15], the authors introduced an adaptive mechanism for the user plane virtualization of the LTE Packet Data Network Gateway (P-GW). The authors applied SDN and NFV concepts to their proposed mechanism so that it can be adaptive to workload changes. In [16], the authors proposed an SDN-based architecture for the Mobile Packet Core (MPC) in order to facilitate dynamic provisioning of MPC network functions. In [17], the authors presented SDN and NFV based architecture that integrates cloud and fog computing.

In [18], the authors presented two design of LTE Evolved Packet Core (EPC) architectures, one of which is based on SDN, and the other is based on NFV. They also provided the performance comparison of two EPC implementation on their LTE testbed. The results showed that the SDN approach is preferable for handling large amount of user data traffic because SDN switches are optimized for data forwarding. On the other hand, the SDN approach have its bottleneck on the communication between SDN switches and SDN controllers, thus NFV approach is adequate for handling extreme signaling processing load. However, the implementation of their EPC is not fully compliant to standards such as the utilized transport protocol for communications between radio access network and MME. Also, the fairness of comparison between SDN and NFV approach is not ensured because of the difference of implementation.

In [19, 20], the authors presented offloading algorithms based on local IP access (LIPA) and selected IP traffic offload (SIPTO) which are proposed by 3GPP. In [19], the offloading traffic volume is determined by the network utilization and user preference. In [20], the authors proposed a bearer-based offload mechanism.

In [21,22], the authors addressed the Virtual Evolved Packet Core (vEPC) placement problem. In [23], the authors presented that the reduction in network resource consumption can be achieved as a result of optimal placement of vEPC functions. However, in these evaluations, the overhead caused by synchronizing distributed vEPC was not considered. In [24], the authors evaluated the performance overhead of state synchronization for control plane and data plane operations. The result showed that synchronizing the state of distributed vEPC can incurs a prohibitive performance penalty (over 70% reduction in throughput as compared to the case of no synchronization).

In [25], the authors proposed a distributed LTE/EPC network architecture based on SDN, NFV,

and cloud computing supporting distributed P-GWs and centralized control plane in LTE/EPC networks. Also, they proposed a route optimization strategy for the internal traffics exchanged between LTE and UEs. The proposed solution was compared with the conventional LTE/EPC network' s scheme in terms of the gateway data processing volume. The simulation results showed that the proposed architecture can reduce the load in LTE/EPC core network and enhance the scalability. However, they focus only on the load caused by device's handover. Since load on control plane generated by data transmission is not ignorable in terms of accommodating M2M/IoT devices, the effect of control plane signaling must be evaluated.

# 3 Mobile Core Network

## 3.1 Network Configuration

Figure 1 depicts the configuration of a mobile core network in this thesis. Each node in this figure has the following functions [26].

- **User Equipment (UE):** User devices, including smartphones, tablets, and M2M/IoT devices.
- **evolved NodeB (eNodeB):** Radio base stations that exchange control messages and data packets with UEs through radio channels. eNodeBs also exchange data packets with the Serving Gateway / Packet Data Network Gateway (S/PGW) and control messages with the MME.
- **Mobility Management Entity (MME):** The node that performs the core of the signaling procedures such as attach, detach, and authentication. It also performs handling UE's handover in wireless networks,
- **Serving Gateway / Packet Data Network Gateway (S/PGW):** The node that exchanges IP packets with external IP networks. It also performs as an anchor point when UEs move between eNodeBs.
- **Home Subscriber Server (HSS):** The database node that manages user specific information, such as the contract information of each user, data for authentication, and the location data of each UE.

## 3.2 Signaling Procedure

When a UE connects to the network, three bearers are established before starting data transmission: a Radio Bearer between the UE and the eNodeB, an S1 Bearer between the eNodeB and the S/PGW, and an S5/S8 Bearer inside the S/PGW.

In this thesis, we assume that a UE visits one of following three states: a Connected state, an Idle state, and a Connected Inactive state. In Connected state, all bearers are established and UEs can transmit and receive user data. In Idle state, the UE is disconnected from the network and its

Radio bearer and S1 bearer are released. When the UE wants to send user data, it needs to transit to Connected state with some signaling procedures. Connected Inactive state is a new state for UEs that has been studied recently [2, 3]. In this state, the Radio Bearer is released, but S1 and S5/S8 bearers are kept established. Therefore, a part of signaling procedures related to establishing S1 and S5/S8 bearers are omitted when a UE transits from this state to Connected state. On the other hand, in Connected Inactive state, the session information of a UE is kept stored in mobile core nodes, causing an increase of memory consumption.

Figure 2 shows the signaling procedure when a UE in Idle state or in Connected Inactive state transits to Connected state. The abbreviations of Req., Res., Cmp., Cmd., Msg., and Ctxt mean request, response, complete, command, message, and context, respectively. The message exchanges and processing depicted in blue are omitted when transiting from Connected Inactive state.

### 3.3 UE's State Transition

Figure 3 shows UE's state transition diagram. A UE in Idle state transits to Connected state when a data transmission request occurs. After the data transmission, the UE starts an Inactive timer which indicates the time remaining before transiting to Connected Inactive state. When no data transmission or reception occurs until the timer expires, the UE transits to Connected Inactive state, else the UE resets the timer. A UE in Connected Inactive state transits to Connected state when a data transmission request occurs. However, when the amount of user data is enough small, the data transmission is performed without transiting to Connected state. This is achieved by including the user data in the signaling messages.

### 3.4 Problems in Accommodating Massive IoT Devices

Most of IoT devices transmit data periodically and intermittently that is quite different from conventional devices. Therefore, especially when massive IoT devices are accommodated to the mobile network, the estimation of server resource utilization cannot be conducted easily.

On the other hand, the CPU load on the mobile core node can be reduced by introducing Connected Inactive state explained in Subsections 3.2 and 3.3. However, it is expected that introducing this state brings significant increase of the memory consumption, because in Connected Inactive

state, the session information of the UE is kept stored in mobile core nodes even when the UE has no data to send. In particular, when IoT devices with extremely low frequency of data transmission is accommodated, the bearers are maintained for rare data transmission for a long time, resulting in the waste of memory. These factors would fluctuate the utilization of CPU and memory of mobile core nodes, that makes the decision of scaling out/up the server resources more difficult.

In the next section, we propose a new state transition for UEs and introduce a method of dynamically control the resource demand itself by adaptively changing the parameter for the state transition.

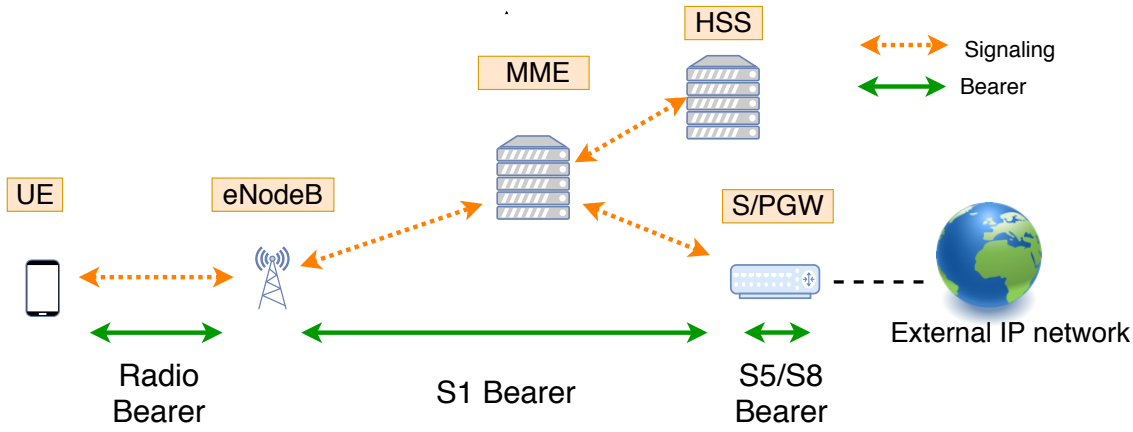


Figure 1: Mobile core network model

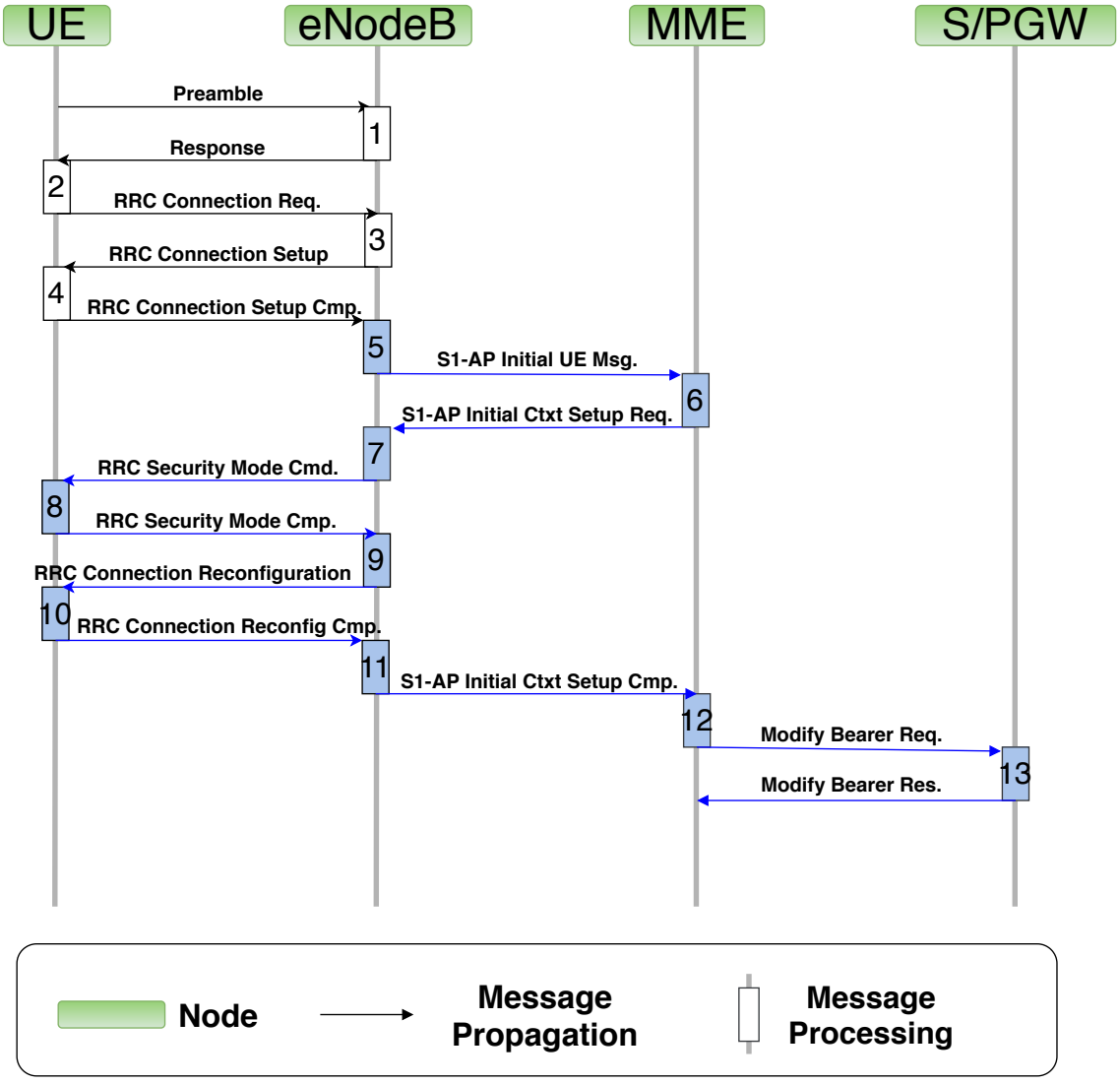


Figure 2: Signaling procedure for UE's state transition to Connected state [1, 2]

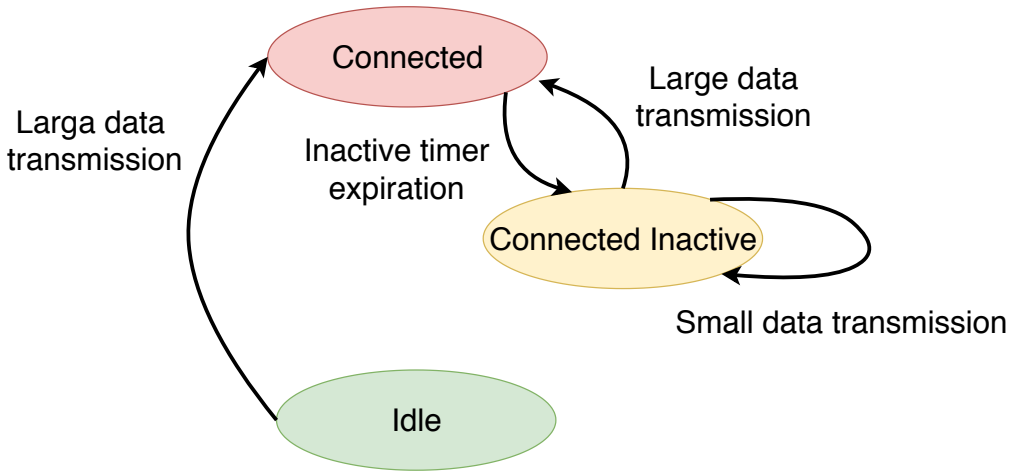


Figure 3: UE's state transition diagram

# 4 Proposed Method

## 4.1 A New State Transition

We introduce a new state transition from Connected Inactive state to Idle state. Figure 4 shows UE's state transition diagram in the proposed method. Compared to Figure 3, the state transition from Connected Inactive state to Idle state is added. We also introduce a timer, which is denoted by *Idle timer* in this thesis, to control this transition. The timer, indicating the time remaining before the transition from Connected Inactive state to Idle state, starts when the UE finishes data transmission.

The introduction of this state transition is motivated by the difference in the memory utilization of a UE in two states and the difference in CPU resource consumption when moving to Connected state. When a UE is in Connected Inactive state, the memory consumption is large, but the CPU resource is not used when moving to Connected state. On the other hand, when the UE is in Idle state, the memory consumption is small, but the transition to Connected state requires CPU resource for signaling procedure. Therefore, when the number of UEs in Connected Inactive state and that in Idle state can be controlled, we can manage CPU and memory utilization of mobile core nodes. We consider that such control can be realized by introducing a state transition from Connected Inactive state to Idle state.

## 4.2 Timer Configuration for State Transition

It is obvious that the length of Idle timer affects the number of UEs in Connected Inactive state and Idle state. When Idle timer becomes longer, the number of UEs that transits from Connected Inactive state to Idle state would decrease, meaning that the number of UEs in Connected Inactive state increases while that of Idle state decreases, and vice versa. Therefore, by carefully choosing the length of Idle timer, we can control the CPU and memory utilization of mobile core nodes. In Section 5, the potential of this method in controlling the resource utilization is assessed by the mathematical analysis.

### 4.3 Adaptive Adjusting of Idle timer

In the actual environment, an appropriate length of Idle timer cannot be determined easily. This is because the fluctuation of the number of accommodating devices and their communication characteristics brings various changes in the server resource consumption. Therefore, we introduce a method for adaptively control the length of Idle timer based on the measurement of CPU load and memory consumption of mobile core nodes, so that the number of devices that can be accommodated is maximized. The number of devices that can be accommodated is defined as the maximum number of UEs that can be further accommodated without overloading either CPU or memory resources, assuming that UEs have the same communication characteristics as currently accommodated. It is also premised that the number of currently accommodated UEs, CPU load and memory usage of mobile core nodes can be observed.

Let $N_{\mathrm{UE}}$ be the number of UEs currently accommodated. Let $C_{N_{\mathrm{UE}}}(T)$ and $M_{N_{\mathrm{UE}}}(T)$ be the consumption of CPU and memory resources, when the number of accommodating UEs is $N_{\mathrm{UE}}$ and the length of Idle timer is $T$, respectively. Let $N_{\mathrm{UE}}^{\mathrm{cap}}(T)$ be the maximum number of devices that can be accommodated when the length of Idle timer is $T$. Then $N_{\mathrm{UE}}^{\mathrm{cap}}(T)$ can be expressed as follows, where $C^{\max}$ and $M^{\max}$ are the maximum amount of CPU and memory resources that can be used.

$$N_{\mathrm{UE}}^{\mathrm{cap}}(T) = \left\lfloor N_{\mathrm{UE}} \cdot \min\left\{ \frac{C^{\max}}{C_{N_{\mathrm{UE}}}(T)}, \frac{M^{\max}}{M_{N_{\mathrm{UE}}}(T)} \right\} \right\rfloor \tag{1}$$

In the proposed method, the length of Idle timer is adaptively changed to maximize $N_{\mathrm{UE}}^{\mathrm{cap}}(T)$. Specifically, at every regular time interval (denoted as *time step* in what follows), the consumption of CPU and memory resource is observed, and the length of Idle timer is shifted so that $N_{\mathrm{UE}}^{\mathrm{cap}}(T)$ increases. Idle timer is controlled by repeating this operation continuously.

It may take a long time for Idle timer reaching an optimal length when the maximum manipulated variable in each time step is small. On the other hand, the control may become unstable with overshooting effects when the manipulated variable is too large. Therefore, in this thesis, we exploit PID control [27], which is known to be simple and highly versatile.

As qualitatively clarified in Subsections 4.1 and 4.2 and numerically confirmed in Section 5, CPU resource consumption is reduced and memory resource usage is increased as Idle timer increases. For this reason, $\frac{C^{\max}}{C_{N_{\mathrm{UE}}}(T)}$ is monotonically non-decreasing and $\frac{M^{\max}}{M_{N_{\mathrm{UE}}}(T)}$ is monotonically non-increasing as Idle timer increases. Let $\boldsymbol{T}$ be a set of $T$ which fulfill the following Equation (2)

and let $\boldsymbol{T}_{\mathrm{optimal}}$ be a set of $T$ which maximize $N_{\mathrm{UE}}^{\mathrm{cap}}(T)$. Then, $T \in \boldsymbol{T}$ is a sufficient condition for $T \in \boldsymbol{T}_{\mathrm{optimal}}$.

$$\frac{C^{\max}}{C_{N_{\mathrm{UE}}}(T)} = \frac{M^{\max}}{M_{N_{\mathrm{UE}}}(T)} \tag{2}$$

Based on Equation (2), we define measured process value $y(t)$ and desired process value $r(t)$ in PID control as follows, where $t$ is a variable that represents time step.

$$y(t) = \frac{C_{N_{\mathrm{UE}}}(T)}{C^{\max}} - \frac{M_{N_{\mathrm{UE}}}(T)}{M^{\max}} \tag{3}$$

$$r(t) = 0 \tag{4}$$

The error $e(t)$ and manipulated variable $u(t)$ in PID control are defined as follows. Then, $K_p$, $K_i$ and $K_d$ are the proportional gain, the integral gain and the derivative gain, respectively. These parameters are set based on Ziegler-Nichols' Ultimate Gain method [27].

$$e(t) = r(t) - y(t) \tag{5}$$

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt} \tag{6}$$

Idle timer ($T$) is dynamically increased and decreased based on $u(t)$ for maximizing $N_{\mathrm{UE}}^{\mathrm{cap}}(T)$.

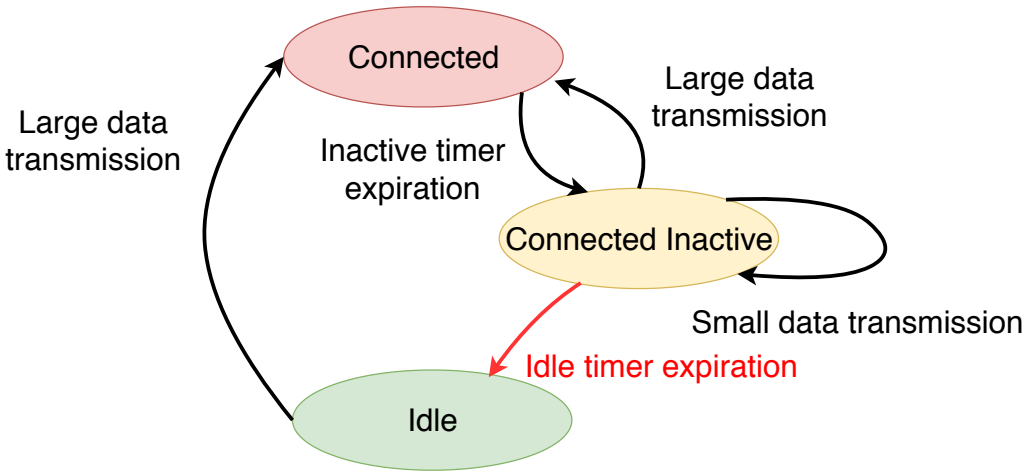


Figure 4: A new state transition in the proposed method

# 5 Analysis of Potential Performance of Proposed Method

In this section, we evaluate the potential of the proposed method based on mathematical analysis.

## 5.1 Performance Analysis

We first give the mathematical analysis for the consumption of CPU and memory resources of mobile core nodes, which can be controlled by adjusting Idle timer. Then we derive the conditions for accommodating UEs. We focus on the CPU load and memory usage of MME since MME is the most bottlenecked node in mobile core network [5]. Also, because the resources of the data plane and the control plane are generally separated in mobile core networks, we focus only on the load on the control plane. In addition, we assume that the time required for data transmission is sufficiently small compared to the communication cycle of UE, meaning that the data transmission never overlaps the control plane communication for the communication of the next cycle.

### 5.1.1 CPU Load

The CPU load is derived based on the number of signaling messages per second which is processed by MME. Let $N_{\mathrm{UE}}$ be the number of currently accommodated UEs and let $\boldsymbol{U}$ be a set of the UEs ($\boldsymbol{U} = \{u_1, u_2, \ldots, u_{N_{\mathrm{UE}}}\}$). Let $T^{\mathrm{ci}}$ and $T^{\mathrm{i}}$ be the length of Inactive timer and Idle timer respectively. Let $T_h$ be the communication interval of $u_h$ ($u_h \in \boldsymbol{U}$). Let $c_h(T^{\mathrm{i}})$ be the average number of signaling messages per second generated by $u_h$ when Idle timer is $T^{\mathrm{i}}$. Table 1 summarizes the notations for the number of signaling messages generated by UE's state transitions. Then, $c_h(T^{\mathrm{i}})$ is calculated as Equation (7), where $d_h$ is the probability that $u_h$ transits to Connected state from Connected Inactive state when transmitting data. Note that, we consider that $T^{\mathrm{i}} \geq T^{\mathrm{ci}}$ always holds.

$$c_h(T^{\mathrm{i}}) = \begin{cases} \frac{1}{T_h} \cdot s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{c}} & \text{if } T_h \leq T^{\mathrm{ci}} \\ \frac{1}{T_h} \cdot (s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{c}} + s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{ci}}) \cdot d_h & \\ \qquad + \frac{1}{T_h} \cdot s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{ci}} \cdot (1 - d_h) & \text{if } T^{\mathrm{ci}} < T_h \leq T^{\mathrm{i}} \\ \frac{1}{T_h} \cdot (s_{\mathrm{MME}}^{\mathrm{i}\to\mathrm{c}} + s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{ci}} + s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{i}}) & \text{otherwise} \end{cases} \tag{7}$$

Let $C(T^{\mathrm{i}})$ be the average CPU load of MME. $C(T^{\mathrm{i}})$ is calculated as follows.

$$C(T^{\mathrm{i}}) = \sum_{h=1}^{N_{\mathrm{UE}}} c_h(T^{\mathrm{i}}) \tag{8}$$

### 5.1.2 Memory Consumption

Let $\tau_h^{\mathrm{c}}(T^{\mathrm{i}})$, $\tau_h^{\mathrm{ci}}(T^{\mathrm{i}})$ and $\tau_h^{\mathrm{i}}(T^{\mathrm{i}})$ be the ratios of time that the UE $u_h$ is in Connected state, in Connected Inactive state and in Idle state, respectively, when Idle timer is $T^{\mathrm{i}}$. Then, $\tau_h^{\mathrm{c}}(T^{\mathrm{i}})$, $\tau_h^{\mathrm{ci}}(T^{\mathrm{i}})$ and $\tau_h^{\mathrm{i}}(T^{\mathrm{i}})$ are calculated as follows.

$$\tau_h^{\mathrm{c}}(T^{\mathrm{i}}) = \begin{cases} 1 & \text{if } T_h \leq T^{\mathrm{ci}} \\ \frac{T^{\mathrm{ci}}}{T_h} \cdot d_h + \frac{0}{T_h} \cdot (1 - d_h) & \text{if } T^{\mathrm{ci}} < T_h \leq T^{\mathrm{i}} \\ \frac{T^{\mathrm{ci}}}{T_h} & \text{otherwise} \end{cases} \tag{9}$$

$$\tau_h^{\mathrm{ci}}(T^{\mathrm{i}}) = \begin{cases} 0 & \text{if } T_h \leq T^{\mathrm{ci}} \\ \frac{T_h - T^{\mathrm{ci}}}{T_h} \cdot d_h + \frac{T_h}{T_h} \cdot (1 - d_h) & \text{if } T^{\mathrm{ci}} < T_h \leq T^{\mathrm{i}} \\ \frac{T^{\mathrm{i}} - T^{\mathrm{ci}}}{T_h} & \text{otherwise} \end{cases} \tag{10}$$

$$\tau_h^{\mathrm{i}}(T^{\mathrm{i}}) = \begin{cases} 0 & \text{if } T_h \leq T^{\mathrm{ci}} \\ 0 & \text{if } T^{\mathrm{ci}} < T_h \leq T^{\mathrm{i}} \\ \frac{T_h - T^{\mathrm{i}}}{T_h} & \text{otherwise} \end{cases} \tag{11}$$

The notations for the memory consumption generated by a UE in each state are presented as Table 2. Let $m_h(T^{\mathrm{i}})$ be the average memory consumption of MME generated by $u_h$, when Idle timer is $T^{\mathrm{i}}$. Then, $m_h(T^{\mathrm{i}})$ is calculated as follows.

$$m_h(T^{\mathrm{i}}) = m_{\mathrm{MME}}^{\mathrm{c}} \cdot \tau_h^{\mathrm{c}}(T^{\mathrm{i}}) + m_{\mathrm{MME}}^{\mathrm{ci}} \cdot \tau_h^{\mathrm{ci}}(T^{\mathrm{i}}) + m_{\mathrm{MME}}^{\mathrm{i}} \cdot \tau_h^{\mathrm{i}}(T^{\mathrm{i}}) \tag{12}$$

$M(T^{\mathrm{i}})$, which represents the average memory consumption of MME generated by all UEs, when Idle timer is $T^{\mathrm{i}}$, is calculated as follows.

$$M(T^{\mathrm{i}}) = \sum_{h=1}^{N_{\mathrm{UE}}} m_h(T^{\mathrm{i}}) \tag{13}$$

### 5.1.3 Conditions for Accommodating Devices

Let $C^{\max}$ and $M^{\max}$ be the maximum number of signaling messages that MME processes per second and the size of memory in MME, respectively. Then, all UEs can be accommodated when there exists $T^{\mathrm{i}}$ that satisfies the following both conditions at the same time.

$$\begin{aligned} C(T^{\mathrm{i}}) &\leq C^{\max} \\ M(T^{\mathrm{i}}) &\leq M^{\max} \end{aligned} \tag{14}$$

## 5.2 Numerical Evaluation Results

### 5.2.1 Parameter Setting

Table 3 shows the parameters utilized in the numerical evaluation. We set the length of Inactive timer to 10 sec [2]. The number of signaling messages associated with state transitions is determined based on [1] and [2]. The memory consumption of MME associated with UE's state is determined based on the source code of OpenAirInterface (OAI) [28], which is an open source software that implements mobile core network functions. Specifically, the memory usage is calculated by the data size stored at MME, derived from the static analysis of OAI source code. Because Connected Inactive state is not implemented in OAI, the memory consumption is estimated to be the same as Connected state based on signaling procedures in [1] and [2].

The maximum number of signaling messages that MME processes per second is configured based on the experimental results in [29]. In [29], the authors presented the relationship between the frequency of UE's attach requests to the mobile core network and the processing delay at MME by experiments using OAI. They revealed that exponentially increase in processing delay can be caused by high frequency of attach requests. In this thesis, the maximum number of signaling messages per second that MME processes is set to 1,200 messages/sec. The memory size of MME is set to 1,000 MB. In addition, we set $d_h = 1$, meaning that the size of data transmitted by UEs is enough large for transiting to Connected state.

### 5.2.2 Evaluation Scenarios

The performance of the proposed method is evaluated in two scenarios which is different in communication characteristics of UEs. In Scenario 5-1, the number of UEs is 500,000, and their communication intervals are uniformly distributed between 10 s and 6,000 s. In this scenario, we

confirm the fundamental characteristics of the proposed method in terms of the control width for server resources. In Scenario 5-2, the number of UEs is 500,000, and their communication intervals are summarized in Table 4, which is based on the typical example in [30]. In this scenario, we evaluate the proposed method in terms of the network capacity in accommodating UEs, with realistic characteristics of UE's communication.

### 5.2.3 Results and Discussions

Figure 5 shows the evaluation results for Scenario 5-1, that plots the relationship between the CPU and memory consumption when Idle timer changes. The horizontal axis represents the CPU load, and the vertical axis represents the memory usage. The vertical and horizontal dashed lines represent $C^{\max}$ and $M^{\max}$, respectively. Also, the colors of the points correspond to the length of Idle timer from 10 s to 6,000 s. From Figure 5, we can observe that the CPU and memory consumption significantly change according to the changes in Idle timer. Specifically, the CPU load decreases and the memory consumption increases with the increase of Idle timer. This is because the increase of Idle timer makes more UEs remain Connected Inactive state, causing the reduction of signaling messages and the increase of session information stored at MME. This result obviously shows that CPU load and memory consumption can be controlled by changing Idle timer. On the other hand, the CPU load exceeds $C^{\max}$ when Idle timer is smaller than 1,422, and the memory usage exceeds $M^{\max}$ when Idle timer is greater than 4,000. These results mean that we need to adjust Idle timer between 1,422 s and 4,000 s to accommodate all UEs in this scenario.

Figure 6 shows the relationship between the CPU and memory consumption in Scenario 5-2. In this scenario, Idle timer is changed from 10 s to 86,400 s. From this figure, we can observe that the CPU load step-wisely decreases with the increase of Idle timer. This is because the distribution of the communication interval of UEs is discrete. It is also observed that the memory consumption exceeds $M^{\max}$ when Idle timer is greater than 72,789 s. Therefore, we need to adjust Idle timer smaller than 72,789 s to accommodate all UEs in this scenario.

By comparing Scenarios 5-1 and 5-2, we confirmed that the CPU load and memory consumption are different even when the number of UEs is identical, because they significantly depend on the communication characteristics of accommodated UEs. Also, it can be confirmed that the conditions on Idle timer for accommodating all UEs are different in both scenarios. Therefore, an

adaptive control of Idle timer is essential to maximize the number of UEs that can be accommodated, especially when their communication characteristics are unknown, such as for IoT devices.

We next evaluate the effect of Idle timer on the capacity of mobile core network, using numerical examples with Scenario 5-2. Figure 7 shows the evaluation results when the number of UEs is 469,200, 925,700, and 1,178,100. Focusing on the case of 469,200 UEs, we can observe that when Idle timer is enough small, the CPU and memory consumption are both lower than the system capacity ($C^{\max}$, $M^{\max}$). However, when Idle timer is greater than 80,000 s, memory usage reaches $M^{\max}$. This means that when the length of Idle timer is longer than 80,000 s, the maximum number of UEs that can be accommodated is 469,200. The plots with 925,700 UEs shows that, when Idle timer is smaller than 1,800 s, the CPU load reaches $C^{\max}$. This means that when the length of Idle timer is smaller than 1,800 s, the maximum number of UEs that can be accommodated is 925,700. Finally, when focusing on the results with 1,178,100 UEs, it is observed that when Idle timer is greater than 1,800 s and smaller than 3,281 s, the CPU load and memory consumption are on $C^{\max}$, $M^{\max}$, respectively. This means that when the length of Idle timer is adjusted between 1,800 s and 3,281 s, the maximum number of UEs that can be accommodated is 1,178,100.

From the above evaluation, it is confirmed that the number of UEs that can be accommodated changes greatly depending on Idle timer. Specifically, in Scenario 5-2, by setting Idle timer appropriately, the number of UEs that can be accommodated is improved by 151 % and 27 %, compared the case when Idle timer is set greater than 80,000 s and that when the timer is set smaller than 1,800 s, respectively.

When the proposed method is not used, there is no state transition from Connected Inactive state to Idle state. This is equivalent to the case when Idle timer is set to enough large value. Based on the above discussion, with Scenario 5-2, we can conclude that by setting Idle timer appropriately based on the proposed method, the number of UEs that can be accommodated is improved by 151% compared to the case without the proposed method.

Table 1: Notations for the number of signaling messages associated with UE's state transitions

| State Transition | | The number of |
|---|---|---|
| Source | Destination | signaling messages |
| Connected | Connected | $s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{c}}$ |
| Connected Inactive | Connected Inactive | $s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{ci}}$ |
| Connected | Connected Inactive | $s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{ci}}$ |
| Connected Inactive | Connected | $s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{c}}$ |
| Connected Inactive | Idle | $s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{i}}$ |
| Idle | Connected | $s_{\mathrm{MME}}^{\mathrm{i}\to\mathrm{c}}$ |

Table 2: Notations for the memory consumptions associated with UE's states

| State | Memory consumption |
|---|---|
| Connected | $m_{\mathrm{MME}}^{\mathrm{c}}$ |
| Connected Inactive | $m_{\mathrm{MME}}^{\mathrm{ci}}$ |
| Idle | $m_{\mathrm{MME}}^{\mathrm{i}}$ |

Table 3: Parameter settings

| Parameter | Numerical setting |
|---|---|
| $T^{\mathrm{ci}}$ | 10 s |
| $s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{c}}$ | 0 messages |
| $s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{ci}}$ | 0 messages |
| $s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{ci}}$ | 0 messages |
| $s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{c}}$ | 0 messages |
| $s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{i}}$ | 5 messages |
| $s_{\mathrm{MME}}^{\mathrm{i}\to\mathrm{c}}$ | 5 messages |
| $m_{\mathrm{MME}}^{\mathrm{c}}$ | 17878 bits |
| $m_{\mathrm{MME}}^{\mathrm{ci}}$ | 17878 bits |
| $m_{\mathrm{MME}}^{\mathrm{i}}$ | 408 bits |
| $C^{\max}$ | 1200 messages/s |
| $M^{\max}$ | 1,000 MB |
| $d_h$ | 1 |

Table 4: The distribution for communication interval for Scenario 5-2

| | 1 day | 2 hours | 1 hour | 30 minutes |
|---|---|---|---|---|
| The ratios of UEs | 40% | 40% | 15% | 5% |

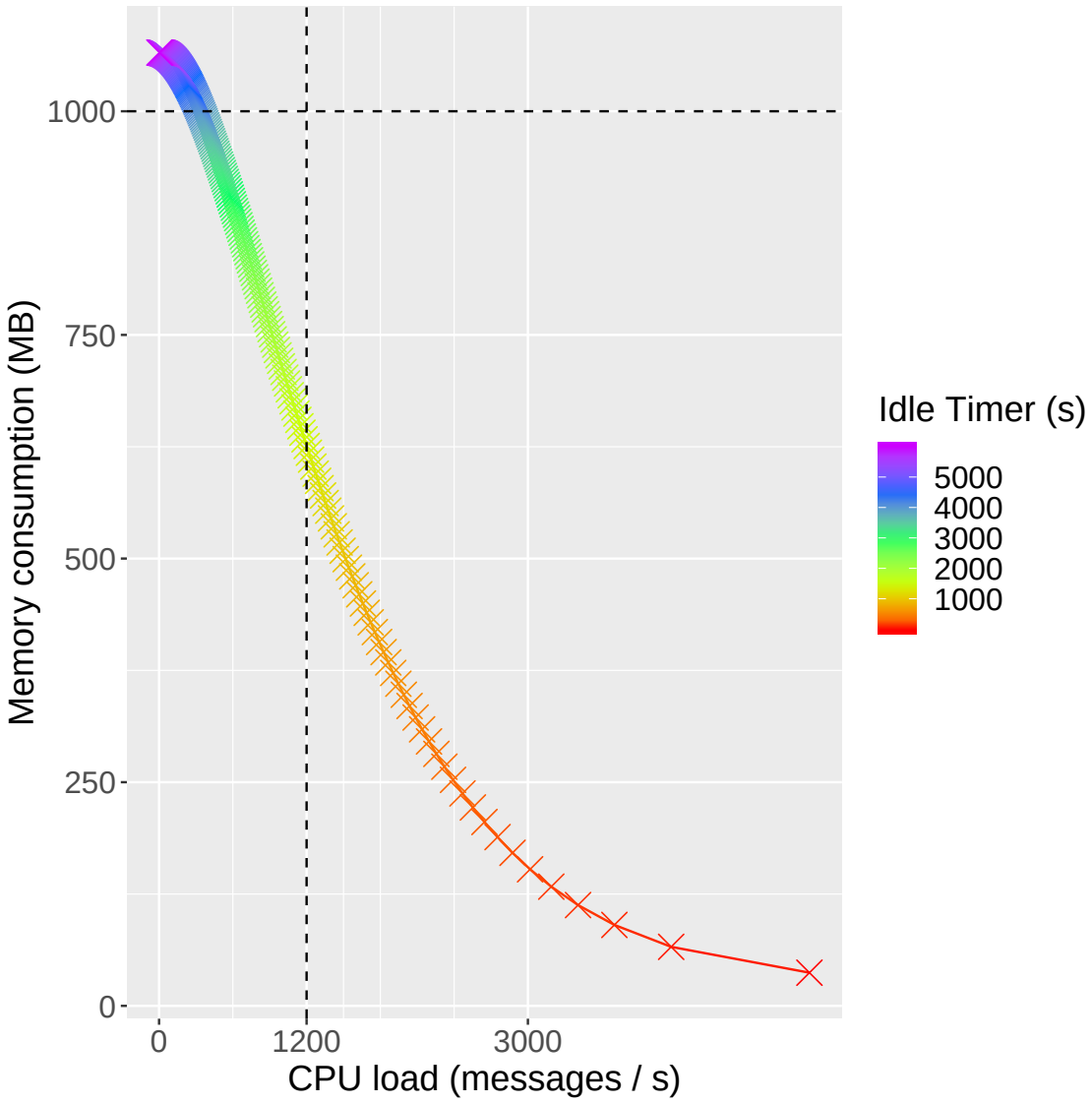


Figure 5: Relationship between the CPU and memory consumption in Scenario 5-1

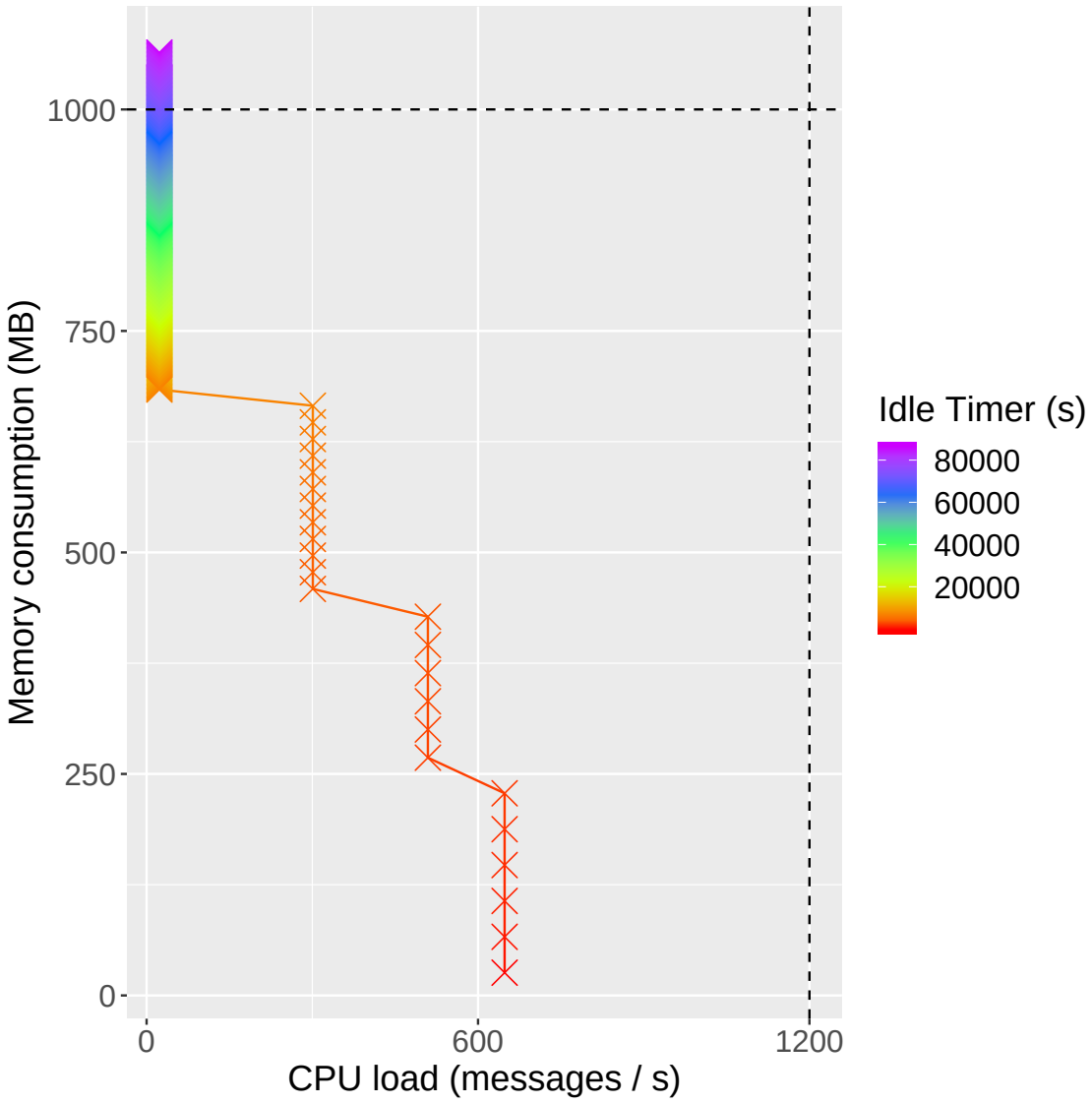


Figure 6: Relationship between the CPU and memory consumption in Scenario 5-2

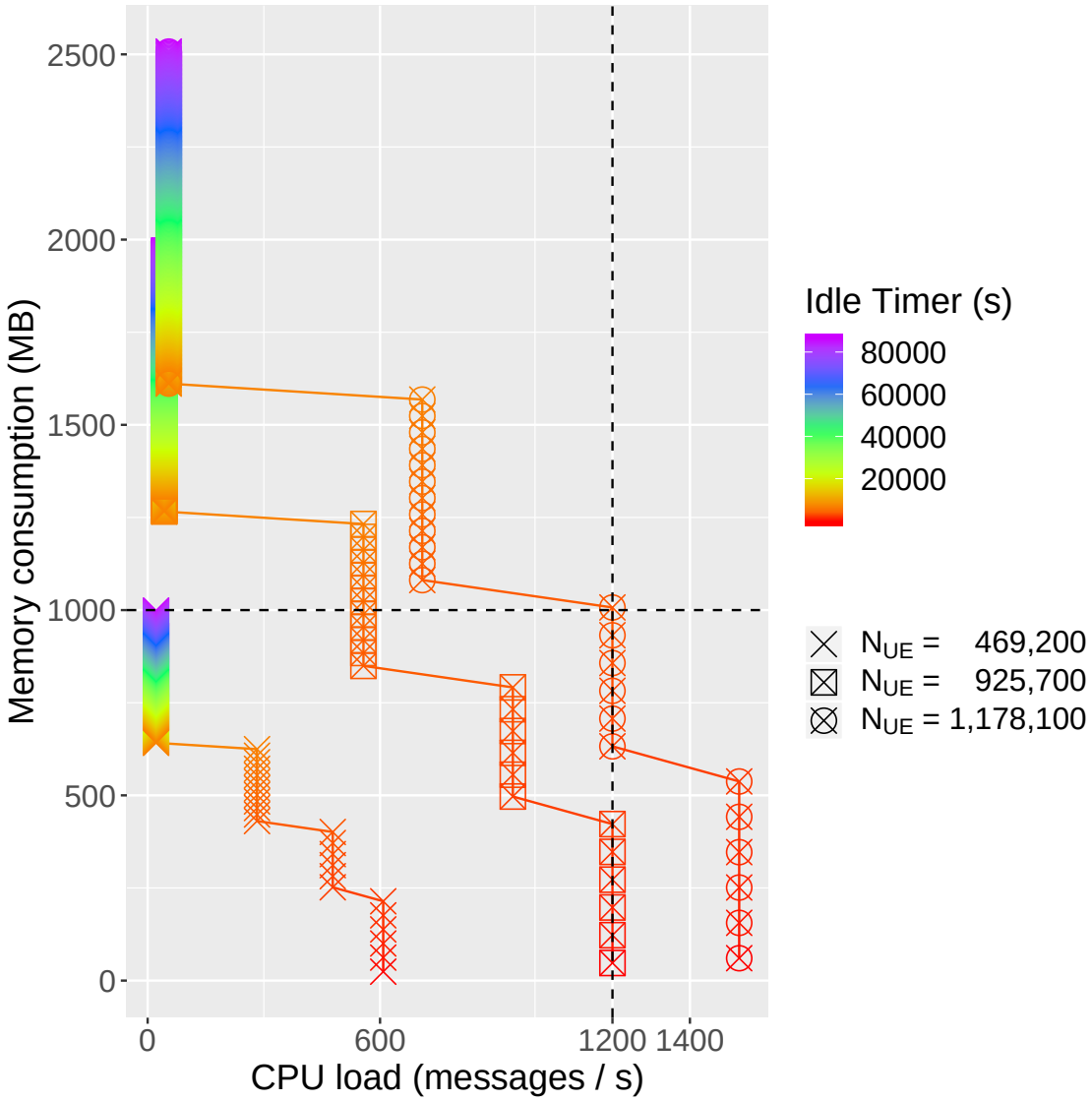


Figure 7: Relationship between the CPU and memory consumption in Scenario 5-2 with 469,200, 925,700, and 1,178,100 UEs

# 6 Simulation Studies on Adaptive Control of Idle timer

In this section, we evaluate the performance of adaptive control of Idle timer based on the proposed method by simulation experiments.

## 6.1 Simulator

We constructed a simulator to simulate the behavior of UEs in a mobile core network, and resource consumption of MME. At every time step, the simulator updates the state of UEs and the CPU and memory consumption of MME, and Idle timer is updated based on the proposed method for the next time step.

### 6.1.1 UE's State Transition

The behavior of UEs is simulated by conducting state transitions as explained in Subsection 4.1 with Figure 4. The conditions for state transitions are evaluated every time step. Each UE has an individual communication cycles and communication timings. Idle timer and Inactive timer of each UE are both set when the UE performs data transmission.

### 6.1.2 CPU Load

The CPU load of MME is derived based on the number of signaling messages per second which is processed by MME. Let $C(t)$ be the CPU load of MME at time $t$. Table 1 summarizes the notations for the number of signaling messages generated by UE's state transitions. Also, Table 5 summarizes the notations for the number of UEs that make state transitions at time $t$. Then $C(t)$ is calculated as Equation (15).

$$\begin{aligned} C(t) &= s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{c}} \cdot n^{\mathrm{c}\to\mathrm{c}}(t) + s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{ci}} \cdot n^{\mathrm{ci}\to\mathrm{ci}}(t) + s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{ci}} \cdot n^{\mathrm{c}\to\mathrm{ci}}(t) \\ &+ s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{c}} \cdot n^{\mathrm{ci}\to\mathrm{c}}(t) + s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{i}} \cdot n^{\mathrm{ci}\to\mathrm{i}}(t) + s_{\mathrm{MME}}^{\mathrm{i}\to\mathrm{c}} \cdot n^{\mathrm{i}\to\mathrm{c}}(t) \end{aligned} \tag{15}$$

### 6.1.3 Memory Consumption

Let $M(t)$ be the memory consumption of MME by accommodated all UEs at time $t$. Table 2 presents the notations for the memory consumption generated by a UE in each state. Also, Table 6

presents the notations for the number of UEs in each state at time $t$. Then $M(t)$ is calculated as Equation (16).

$$M(t) = m^{\mathrm{c}}_{\mathrm{MME}} \cdot n^{\mathrm{c}}(t) + m^{\mathrm{ci}}_{\mathrm{MME}} \cdot n^{\mathrm{ci}}(t) + m^{\mathrm{i}}_{\mathrm{MME}} \cdot n^{\mathrm{i}}(t) \tag{16}$$

#### 6.1.4 Update of Idle timer

The Idle timer is adaptively controlled by the proposed method in Section 4. Specifically, the Idle timer is dynamically updated by PID control based on the calculated results of the CPU and memory consumptions of MME, by Equations (15) and (16). The Idle timer is applied to a UE when it performs data transmission.

### 6.2 Evaluation Results

#### 6.2.1 Parameter Setting

We utilize the same parameters in Table 5 for Section 5. The length of time step of the simulation is set to one second. Table 7 shows the constants in PID control that are configured by the Ziegler-Nichols limit sensitivity method. To determine these constants, we conducted the preliminary experiments where the number of UEs is 648,000 and their communication intervals are summarized in Table 4. Then, we determined the constants based on the results when Idle timer is converged from the initial value (600 s) to the optimal value (3,280 s).

By comparing the results of control types in PID control, we revealed that the P control is the most stable and converges Idle timer fastest among P, PI, and PID controls. Therefore, in this section, we compare the proposed method with P control and the method without the proposed method, meaning that Idle timer remains unchanged from 2,017 s that maximizes the number of UEs that can be accommodated in the initial state of each scenario. There are mainly two reasons why P control is the best among the three PID controls. The first is that the integral control did not work effectively because no steady-state error occurred in the proposed method. The second is that the discrete load fluctuations in simulation experiments make differential control unstable.

#### 6.2.2 Evaluation Scenarios

The performance of the proposed method is evaluated in two scenarios. In both scenarios, at the start of the simulation, the number of UEs is 240,000, and their communication intervals are

summarized in Table 8. Then, UEs with a specific communication intervals are newly connected to the network.

In Scenario 6-1, 320,000 UEs connects to the network at random timings between 60,000s and 70,000s. Their communication intervals are fixed to 100 s. After that, they are disconnected from the network at random timings between 140,000 s and 150,000 s.

In Scenario 6-2, 320,000 UEs connects to the network at random timings between 60,000s and 72,000s. In addition, 120,000 UEs connects to the network at random timings between 100,000s and 112,000s. Their communication intervals are fixed to 6,000 s. After that, they are disconnected from the network at random timings between 140,000 s and 152,000 s.

### 6.2.3 Results and Discussions

Figures 8 and 9 show the evaluation results for Scenario 6-1, that plots the temporal changes in the average CPU load and memory consumption every 10 s. Figure 8 is a result without the proposed method where Idle timer is fixed, while Figure 9 is a result with the proposed method. Temporal changes in Idle timer by the proposed method is shown in Figure 10. The dashed lines in these figures represent the $C^{\max}$ and $M^{\max}$.

From Figure 8, we can observe that the CPU load is temporarily increasing with increasing the number of UEs. There are two reasons for that. First, the newly connected UEs cause signaling procedures with their transition to Connected state. Second, once they transit to Connected state, they never transit to Idle state and cause signaling procedures because their communication intervals are shorter than Idle timer. Also, it is observed that memory consumption increases to 1,044 MB with increasing the number of UEs. This is because the UEs with short communication intervals remain Connected state or Connected Inactive state, causing session information remain stored in MME.

On the other hand, with the proposed method shown in Figure 9, Idle timer is controlled so that the CPU load increases and the memory consumption decreases. Specifically, Idle timer is decreased to reduce the number of UEs in Connected state or in Connected Inactive state. With this control, the memory consumption reduces, while CPU load increases because the number of UEs in Idle state increases. As a result, it is confirmed that while the CPU load increases to 1,133 messages/s, the memory usage decreases to 835 MB. From Figure 10, we can confirm the decrease of Idle timer when the number of UEs increases.

Without proposed method, it is difficult to accommodate further UEs because the memory consumption exceeds $M^{\max}$. On the other hand, with the proposed method, more UEs can be accommodated because both CPU load and memory usage have not reached $C^{\max}$ and $M^{\max}$, respectively. This clearly explains the effectiveness of the proposed method that adaptively adjusts the CPU and memory resource demand by currently accommodated UEs to increase the number of UEs that can be further accommodated.

In addition, we can observe that Idle timer has resumed to the level before the increase of UEs when the UEs disconnect from the network.

Figures 11 and 12 show the evaluation results for Scenario 6-2, that plots the temporal changes in the average CPU load and memory consumption every 10 s. Figure 11 is a result without the proposed method where Idle timer is fixed, while Figure 12 is a result with the proposed method. Temporal changes in Idle timer by the proposed method is shown in Figure 13. The dashed lines in these figures represent the $C^{\max}$ and $M^{\max}$.

From Figures 11 and 12, we can observe that the CPU load and memory consumption increase with increasing the number of UEs. The increase of CPU load is caused by increasing the number of UEs transiting to Idle state causing signaling procedures because the communication intervals of the newly connected UE are longer than Idle timer. The memory usage increases because the number of UEs in Connected state or in Connected Inactive state increases with increasing the number of UEs. From Figure 11, it is observed that the CPU load increases to 1,041 messages/s and the memory consumption increases to 602 MB with increasing 320,000 UEs. In addition, we can observe that the CPU load increases to 1,253 messages/s and the memory usage increases to 692 MB with increasing further 120,000 UEs.

On the other hand, with the proposed method shown in Figure 12, contrary to scenario 6-1, Idle timer is controlled so that the CPU load decreases and the memory consumption increases. As a result, when 320,000 UEs connected, the CPU load and memory consumption increase to 965 messages/s and 719 MB, respectively, and when further 120,000 UEs connected, they increase to 1,166 messages/s and 858 MB, respectively. From Figure 13, we can confirm the increase of Idle timer when the number of UEs increases.

Without proposed method, it is difficult to accommodate further UEs because the CPU load exceeds $C^{\max}$ when 440,000 UEs added. On the other hand, with the proposed method, more UEs can be accommodated because both CPU load and memory usage have not reached $C^{\max}$

and $M^{\max}$, respectively. This results also explains the effectiveness of the proposed method.

Table 5: Notations for the number of UEs that make state transitions at time $t$

| State Transition | | The number of |
|---|---|---|
| Source | Destination | UEs |
| Connected | Connected | $n^{\mathrm{c}\to\mathrm{c}}(t)$ |
| Connected Inactive | Connected Inactive | $n^{\mathrm{ci}\to\mathrm{ci}}(t)$ |
| Connected | Connected Inactive | $n^{\mathrm{c}\to\mathrm{ci}}(t)$ |
| Connected Inactive | Connected | $n^{\mathrm{ci}\to\mathrm{c}}(t)$ |
| Connected Inactive | Idle | $n^{\mathrm{ci}\to\mathrm{i}}(t)$ |
| Idle | Connected | $n^{\mathrm{i}\to\mathrm{c}}(t)$ |

Table 6: Notations for the number of UEs in each state at time $t$

| State | The number of UEs |
|---|---|
| Connected | $n^{\mathrm{c}}(t)$ |
| Connected Inactive | $n^{\mathrm{ci}}(t)$ |
| Idle | $n^{\mathrm{i}}(t)$ |

Table 7: Constants setting in PID control

| Constant | Numerical setting |
|---|---|
| $K_p$ | 1.5 |
| $K_i$ | 0 |
| $K_d$ | 0 |

Table 8: The distribution for communication interval for Scenarios 6-1 and 6-2

| | 10 s | 20 s | 30 s | $\cdots$ | 6,000 s | 合計 |
|---|---|---|---|---|---|---|
| The number of UEs | 400 | 400 | 400 | $\cdots$ | 400 | 240,000 |

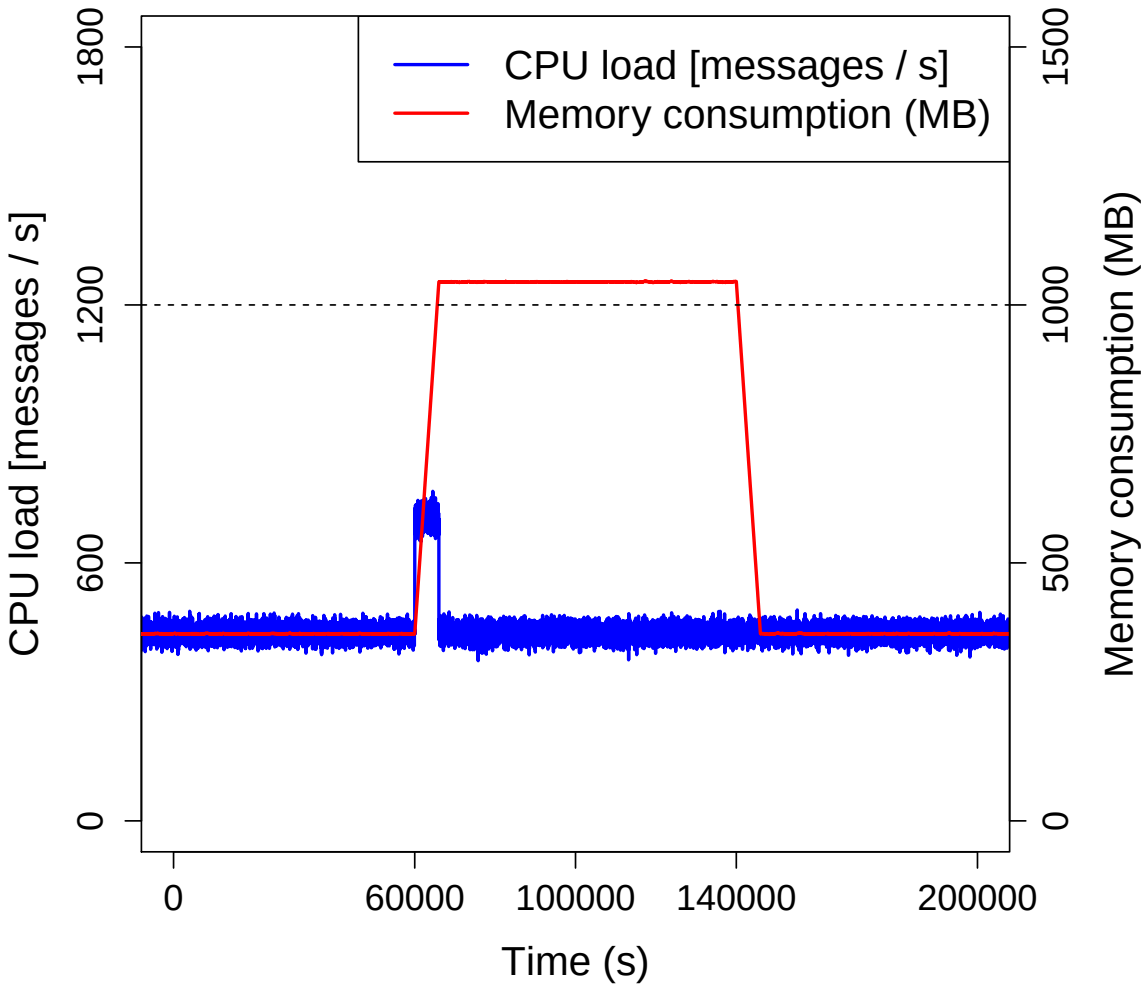


Figure 8: Temporal changes in CPU load and memory consumption without the proposed method in Scenario 6-1

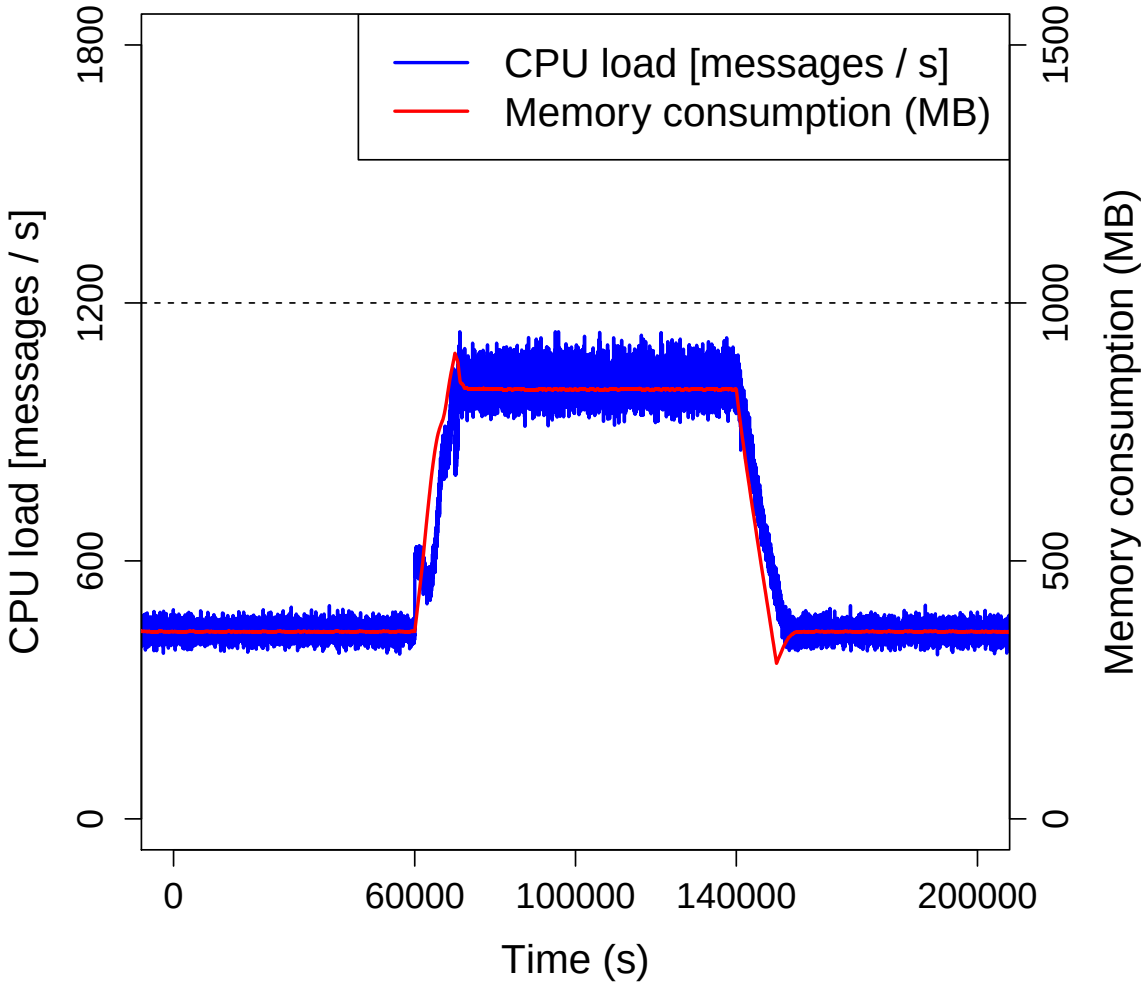


Figure 9: Temporal changes in CPU load and memory consumption with the proposed method in Scenario 6-1

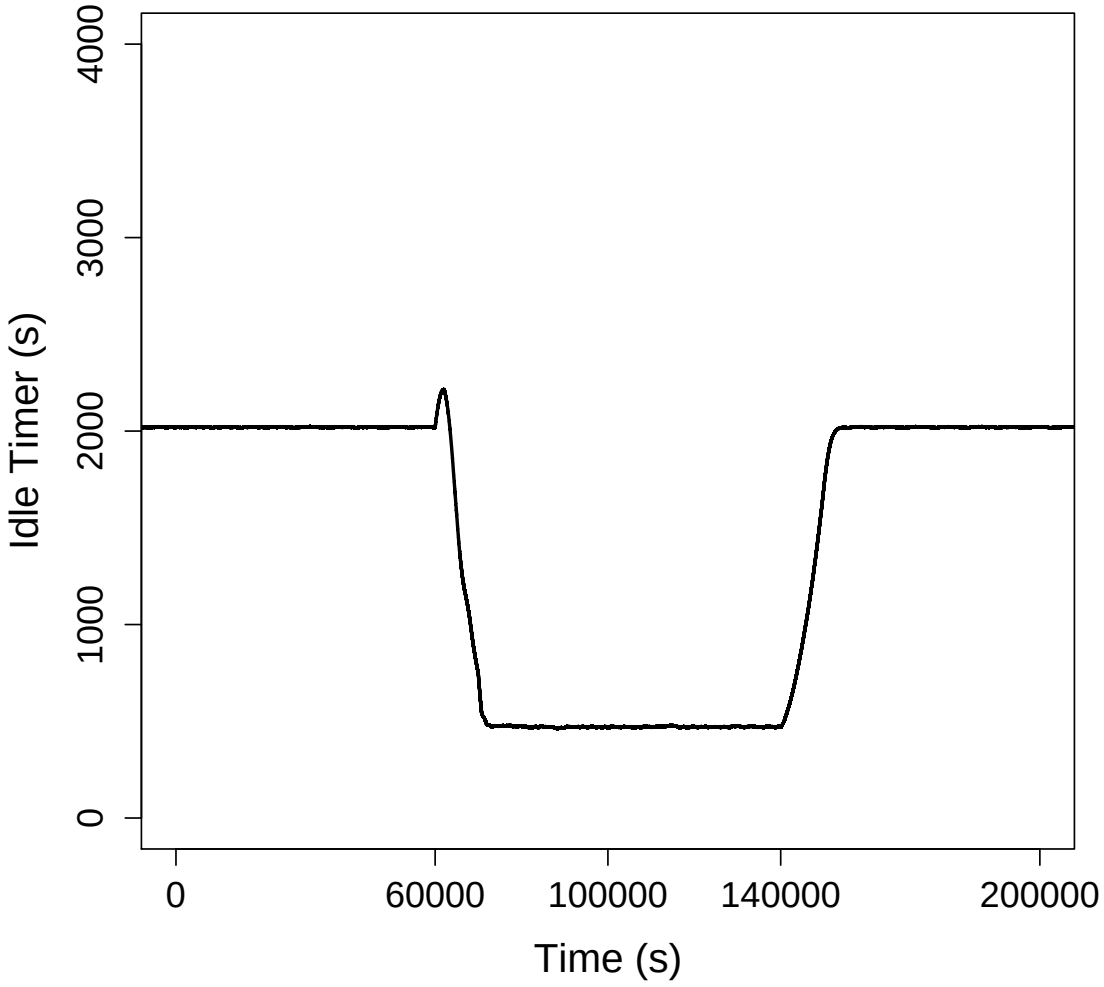


Figure 10: Temporal changes in Idle timer with the proposed method in Scenario 6-1

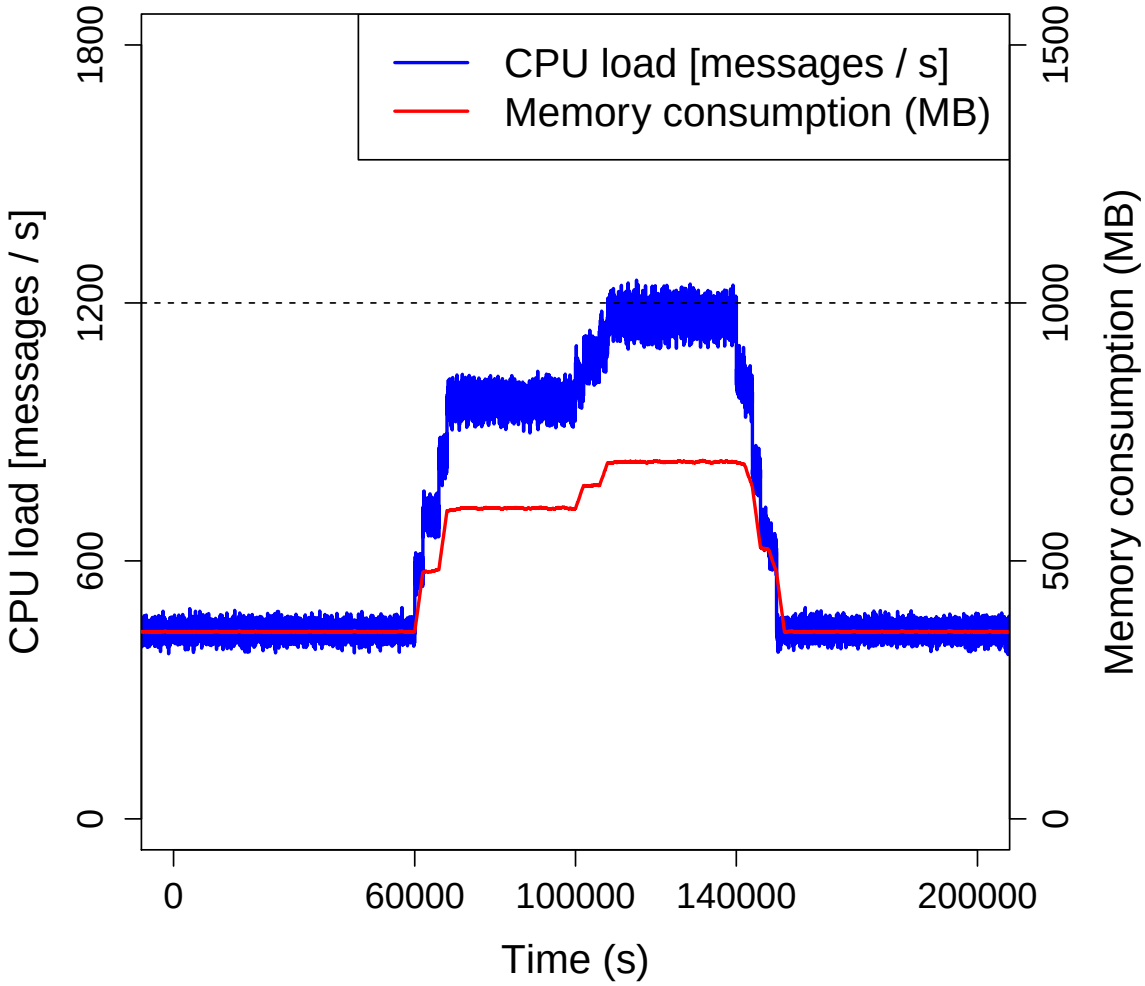


Figure 11: Temporal changes in CPU load and memory consumption without the proposed method in Scenario 6-2

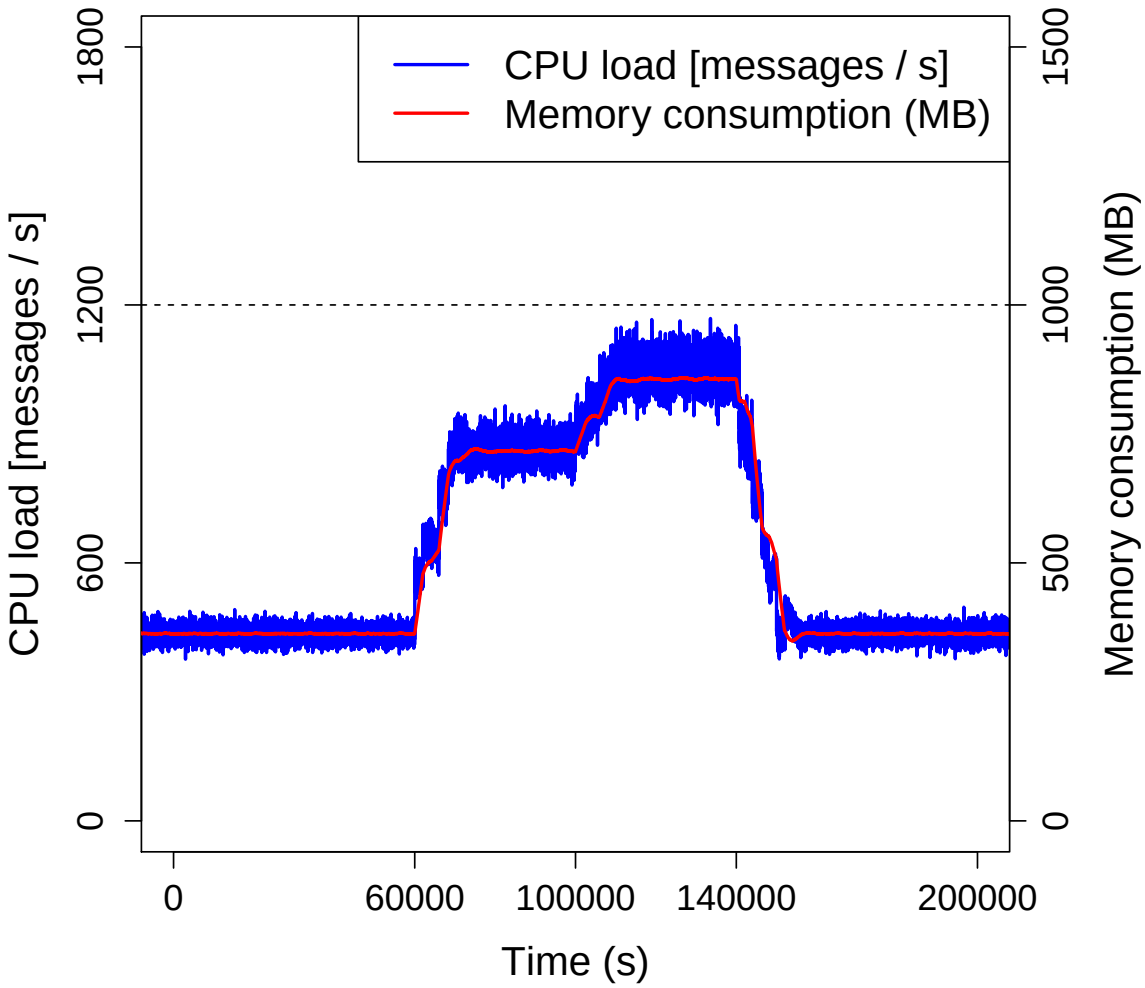


Figure 12: Temporal changes in CPU load and memory consumption with the proposed method in Scenario 6-2

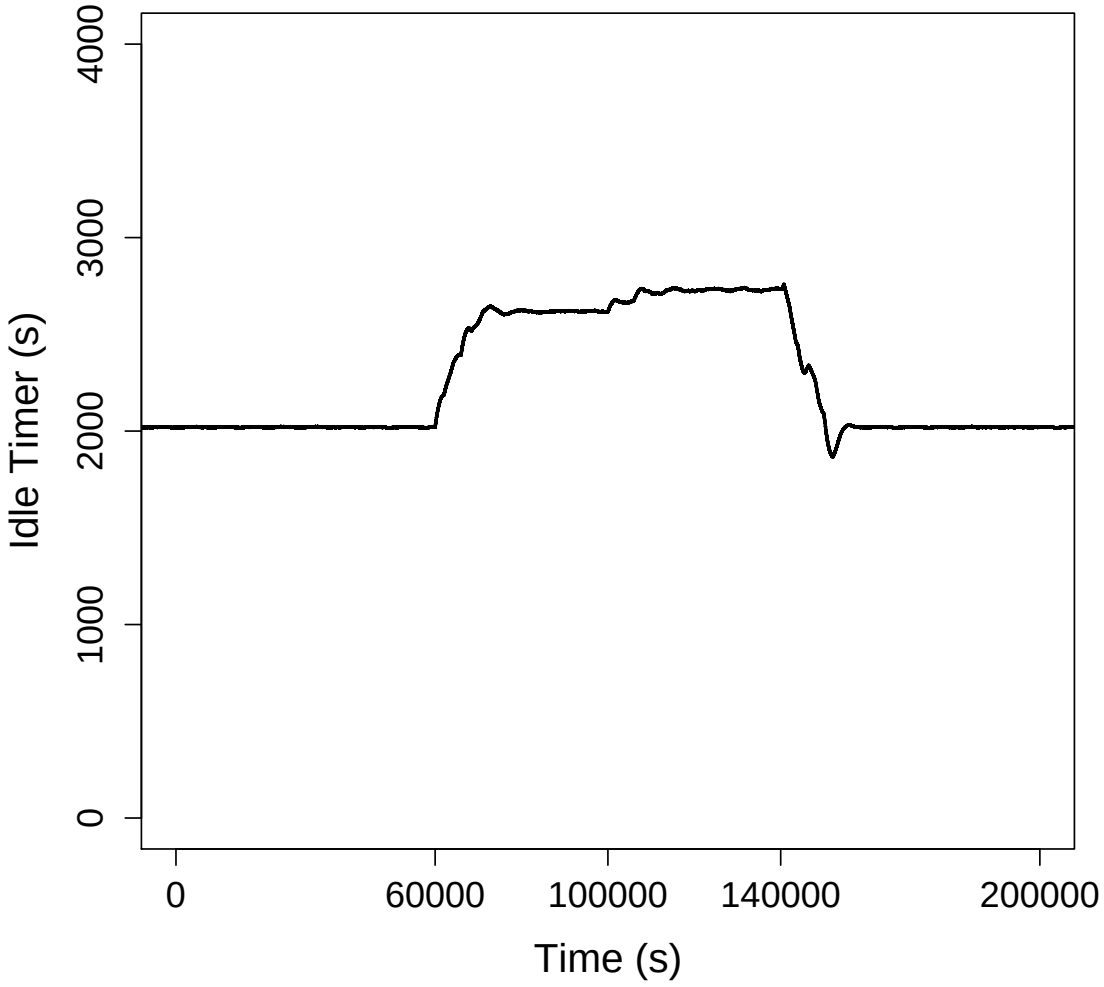


Figure 13: Temporal changes in Idle timer with the proposed method in Scenario 6-2

# 7 Conclusion

In this thesis, we propose a method that dynamically controls server resource demand of mobile core networks. Specifically, the CPU load and the memory resource demand of mobile core nodes are adaptively adjusted by controlling the accommodated device's state transitions.

The potential of the proposed method is evaluated by mathematical analysis. We confirm that the resource utilization of the mobile core network can be improved by balancing CPU load and memory consumption. Also, the proposed method can increase the capacity of the mobile core network up to around 150 % without increasing server resources. We then evaluate the effectiveness of proposed method that adaptively adjusts the control parameter of UE's state transitions by simulation experiments. We confirm that the proposed method can adaptively control server resource demand and avoid them from becoming short when the number of accommodating device changes.

For future work, we plan to extend the proposed method by combining the resource control architecture such as Server Disaggregation and scale-out/scale-in to respond to short-term and long-term load fluctuations. It is also important to consider a more efficient control theory for controlling device's states.

## Acknowledgment

There are so many people to thank for helping me during my master thesis degree studies at Osaka University. I would like to show my great appreciation to my supervisor, Professor Morito Matsuoka of Osaka University. He always gave me warm encouragements and elaborated guidance. Also, I am deeply grateful to Professor Masayuki Murata of Osaka University. His comments and suggestions were inestimable value of my study. And I would like to express the deepest appreciation to Professor Go Hasegawa of Tohoku University. He provided carefully considered feedback and meticulous comments. Finally, I gratefully appreciate for the great encouragement and support to the members of Matsuoka Laboratory.

## References

[1] 3GPP, "Study on architecture enhancements for Cellular Internet of Things (CIoT)," TR 23.720 Version 13.0.0, 3rd Generation Partnership Project (3GPP), Mar. 2016.

[2] I. L. Da Silva, G. Mildh, M. Sily, and S. Hailu, "A Novel State Model for 5G Radio Access Networks," in *Proceedings of 2016 IEEE International Conference on Communications Workshops (ICC)*, pp. 632–637, May 2016.

[3] S. Hailu, M. Saily, and O. Tirkkonen, "RRC State Handling for 5G," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 106–113, Jan. 2019.

[4] M. Shimizu, H. Nakazato, and H. Seshake, "Scale-Out Architecture for Service Order Processing Systems," in *Proceedings of 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 880–883, May 2013.

[5] P. C. Amogh, G. Veeramachaneni, A. K. Rangisetti, B. R. Tamma, and A. A. Franklin, "A Cloud Native Solution for Dynamic Auto Scaling of MME in LTE," in *Proceedings of 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, Oct. 2017.

[6] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin, and D. Darche, "On the Scalability of 5G Core Network: The AMF Case," in *Proceedings of 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 1–6, Jan. 2018.

[7] Y. Ren, T. Phung-Duc, J. Chen, and Z. Yu, "Dynamic Auto Scaling Algorithm (DASA) for 5G Mobile Networks," in *Proceedings of 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Dec. 2016.

[8] C. H. T. Arteaga, F. Rissoi, and O. M. C. Rendon, "An Adaptive Scaling Mechanism for Managing Performance Variations in Network Functions Virtualization: A Case Study in an NFV-Based EPC," in *Proceedings of 2017 13th International Conference on Network and Service Management (CNSM)*, pp. 1–7, Nov. 2017.

[9] M. Mahloo, J. M. Soares, and A. Roozbeh, "Techno-Economic Framework for Cloud Infrastructure: A Cost Study of Resource Disaggregation," in *Proceedings of 2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 733–742, Sept. 2017.

[10] "Intel' s Disaggregated Server Rack," technical report (tr), Moor Insights Strategy, Aug. 2013.

[11] C. Devaki and L. Rainer, "Enhanced Back-off Timer Solution for GTP-C Overload Control," Patent 20160057652, Feb. 2016.

[12] B. Abali, R. J. Eickemeyer, H. Franke, C. Li, and M. Taubenblatt, "Disaggregated and Optically Interconnected Memory: When will it be cost effective?," *CoRR*, vol. abs/1503.01416, Aug. 2015.

[13] "Disaggregated Servers Drive Data Center Efficiency and Innovation," technical report (tr), Intel Corporation, June 2017.

[14] S. Legtchenko, H. Williams, K. Razavi, A. Donnelly, R. Black, A. Douglas, N. Cheriere, D. Fryer, K. Mast, A. D. Brown, A. Klimovic, A. Slowey, and A. Rowstron, "Understanding Rack-Scale Disaggregated Storage," in *Proceedings of 9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17)*, p. 2, July 2017.

[15] L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, "An Adaptive Mechanism for LTE P-GW Virtualization Using SDN and NFV," in *Proceedings of 2017 13th International Conference on Network and Service Management (CNSM)*, pp. 1–9, Nov. 2017.

[16] M. Karimzadeh, L. Valtulina, H. v. d. Berg, A. Pras, M. Liebsch, and T. Taleb, "Software Defined Networking to Support IP Address Mobility in Future LTE Network," in *Proceedings of 2017 Wireless Days*, pp. 46–53, Mar. 2017.

[17] R. Chaudhary, N. Kumar, and S. Zeadally, "Network Service Chaining in Fog and Cloud Computing for the 5G Environment: Data Management and Security Challenges," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 114–122, Nov. 2017.

[18] A. Jain, Sadagopan N S, S. K. Lohani, M. Vutukuru, "A Comparison of SDN and NFV for Re-designing the LTE Packet Core," in *Proceedings of 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 74–80, Nov. 2016.

[19] S. Lee, H. Cheon, S. Kang, and J. Kim, "Novel LIPA/SIPTO Offloading Algorithm According to the Network Utilization and Offloading Preference," in *Proceedings of 2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 314–318, Oct. 2014.

[20] L. Ma and W. Li, "Traffic Offload Mechanism in EPC Based on Bearer Type," in *Proceedings of 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, Sept. 2011.

[21] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile Core Network Virtualization: A Model for Combined Virtual Core Network Function Placement and Topology Optimization," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–9, Apr. 2015.

[22] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, "Network Function Placement on Virtualized Cellular Cores," in *Proceedings of 2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, pp. 259–266, Jan. 2017.

[23] A. Gupta, M. Tomatore, B. Jaumard, and B. Mukherjee, "Virtual-Mobile-Core Placement for Metro Network," in *Proceedings of 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pp. 308–312, June 2018.

[24] P. Satapathy, J. Dave, P. Naik, and M. Vutukuru, "Performance Comparison of State Synchronization Techniques in a Distributed LTE EPC," in *Proceedings of 2017 IEEE Conference on*

*Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–7, Nov. 2017.

[25] Y.-h. Kim, H.-k. Lim, K.-h. Kim, and Y.-H. Han, "A SDN-Based Distributed Mobility Management in LTE/EPC Network," *The Journal of Supercomputing*, vol. 73, pp. 2919–2933, July 2017.

[26] M. Ueno, "Experimental Evaluation of Accommodation Methods of M2M/IoT Terminals in Mobile Core Networks," Master's thesis, Graduate School of Information Science and Technology, Osaka University, Feb. 2019.

[27] M. Johnson and M. Moradi, *PID Control: New Identification and Design Methods*. Probability and its applications, Springer London, 2006.

[28] "OpenAirInterface." available at `http://www.openairinterface.org/`.

[29] M. Ueno, G. Hasegawa, and M. Murata, "Experimental Evaluation of Mobile Core Networks on Simultaneous Access from M2M/IoT Terminals," in *Proceedings of 2019 International Conference on Information Networking (ICOIN)*, pp. 13–18, Jan. 2019.

[30] 3GPP, "Cellular System Support for Ultra-low Complexity and Low Throughput Internet of Things (CIoT)," Technical Report (TR) 45.820 Version 13.1.0, 3rd Generation Partnership Project (3GPP), Dec. 2015.