# Efficient Methods for Signaling Process and Information Management in Mobile Networks

Submitted to
Graduate School of Information Science and Technology
Osaka University

July 2016

Manabu ITO

# List of publication

## Journal papers

1. Manabu Ito, Satoshi Komorita, Yoshinori Kitatsuji, Hidetoshi Yokota, and Takeo Hayashi, "Cloudization of IP Multimedia Subsystem and Testbed Verification: High Reliability and Energy Saving for IMS," *IPSJ Digital Practice*, vol. 4, no. 4, pp. 323–332, October 2013 (in Japanese).

2. Manabu Ito, Kiyohide Nakauchi, Yozo Shoji, Nozomu Nishinaga, and Yoshinori Kitatsuji, "Service-Specific Network Virtualization to Reduce Signaling Processing Loads in EPC/IMS," *IEEE Access*, vol. 2, pp. 1076–1084, September 2014.

3. Manabu Ito, Nozomu Nishinaga, Yoshinori Kitatsuji, and Masayuki Murata, "Reducing state information by sharing IMSI for cellular IoT devices," *IEEE Internet of Things Journal*, 2016.

## Refereed Conference Papers

1. Manabu Ito, Satoshi Komorita, Yoshinori Kitatsuji, and Hidetoshi Yokota. "OpenFlow-based Routing Mechanism for Call Session State Migration in the IMS," in *Proceedings of the 7th WSEAS International Conference on Computer Engineering and Applications*, vol. 13, pp. 60–67, January 2013.

2. Manabu Ito, Kiyohide Nakauchi, Yozo Shoji, and Yoshinori Kitatsuji. "Mechanism of Reducing Signaling Processing Load in EPC/IMS Using Service-Specific Network Virtualization," in *Proceedings of the 6th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, March 2014.

3. Manabu Ito, Nozomu Nishinaga, and Yoshinori Kitatsuji, "Redirection-Based Rules Sharing Method for the Scalable Management of Gateways in Mobile Network Virtualization," in *Proceedings of the Second IEEE International Workshop on 5G & Beyond - Enabling Technologies and Application*, pp. 1–7, December 2015.

4. Manabu Ito, Nozomu Nishinaga, Yoshinori Kitatsuji, and Masayuki Murata, "Aggregating cellular communication lines for IoT devices by sharing IMSI," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC 2016)*, pp. 1–7, May 2016.

## Non-Refereed Technical Papers

1. Manabu Ito, Kiyohide Nakauchi, Yozo Shoji, and Yoshinori Kitatsuji. "A Method of Reducing Signaling Traffic in EPC/IMS using Network Virtualization appropriate to Services," *Technical Report of IEICE (NV2013-8)*, pp. 16–21, November 2013 (in Japanese).

2. Manabu Ito, Nozomu Nishinaga, Yoshinori Kitatsuji, and Masayuki Murata, "A Study of Aggregating Cellular Communication Lines for IoT Devices by Sharing IMSI," *Technical Report of IEICE (MoNA2015-36)*, vol. 115, no. 311, pp. 83–88, November 2015 (in Japanese).

# Preface

Mobile networks are expected to provide various services other than the current services (e.g., voice, short message service (SMS), and over-the-top (OTT) services). The emerging services include virtual reality office, autonomous driving, and the Internet of Things (IoT) services. Mobile networks need to support conflicting requirements, such as ultra-low latency, high reliability, and ultra-inexpensive network usage. A current architecture of mobile networks, which is composed of dedicated hardware and which conducts common procedures for service provision, cannot meet ultra-low latency and high reliability without tremendously increasing capital expenditure/operational expense (CAPEX/OPEX).

For mobile networks to support the conflicting requirements, mobile network operators (MNOs) need to construct dedicated mobile networks that are composed of functions specialized for each service. In each dedicated mobile network, network configuration, signaling processes, or information management are specialized for each service. Besides, the system capacity of each dedicated mobile network is changed on demand. For realizing these dedicated mobile networks without investing in extra capacity, network function virtualization (NFV) and network virtualization have attracted attention as promising technologies. NFV can flexibly assign and distribute available hardware resources to functions that constitute a mobile network, as required. Network virtualization can construct service-specific mobile networks on a single physical network infrastructure through configuring logically independent networks (hereafter called “virtual networks”) that are composed of functions specialized for particular services.

Concerning the technologies for virtualization of network resources/functions and hardware resource assignments, many ideas have been proposed and studied. In this thesis, we focus on

methodologies, through improving the benefits from virtualization, for reducing signaling load and a significant amount of information in the entire mobile network. We first investigate state information migration for flexible reconfiguration of a mobile network and use of different dedicated mobile networks depending on each service. Second, we inquire into a gateway function for realizing the dynamic use of dedicated mobile networks using network virtualization and verify that network virtualization can reduce signaling load in the entire mobile network. After that, we improve the scalability of the gateway function in consideration of practical use. Finally, we investigate an efficient information management, which is necessary but rarely argued for inexpensive IoT services.

Regarding the state information migration, we propose a method to enable it without modifying the standard procedure for an IP multimedia subsystem (IMS), which is a call session control system in a 4G mobile network. We also introduce implementations that enable a significant amount of state information to migrate by making the assumption of server failure or disaster. We verify the performance of state information migration through a testbed and discuss the feasibility of the IMS where the proposed method is applied.

To verify that network virtualization is useful for reducing signaling load, we propose a service-specific network virtualization. The proposed method creates several virtual networks that are composed of functions specialized for particular services on a single physical network infrastructure and efficiently forwards a sequence of signaling messages to the appropriate virtual networks. Using a prototype system, we verify the overheads costs of the proposal that are incurred during the inspection of application packet headers needed to appropriately forward signaling messages as well as the overheads incurred when migrating state information from one virtual network to another. We show that the proposal can reduce the signaling load by approximately 25% under certain assumptions.

In consideration of the practical use of the service-specific network virtualization, we propose a method for improving the scalability of sharing the flow rules, which are used to inspect and direct packets, between the gateway functions in the single physical network infrastructure. The proposed method enhances the centralized management mechanism, in which a controller reactively installs rules based on receiving the first packet of each flow, to reduce the load in the controller by decreasing the packets transmitted to the controller for packet inspection. Using a prototype system, we show that the enhanced method can reduce the number of packets arriving at the controller per unit time by up to 63% under our use case assumption.

We finally focus on information management in the mobile core system, called evolved packet core (EPC) in the 4G mobile network, to provide ultra-inexpensive network usage for a large number of IoT devices. We propose a method for reducing state information. The proposed method assigns the same international mobile subscriber identity (IMSI) to multiple IoT devices, which upload data with the same cycle, and manages devices uploading data at different timings. We investigate the impact of factors affecting the performance of the proposed method through several simulation-based verifications and show that the proposal can reduce the required state information to less than 0.5% compared to the current EPC method.

# Acknowledgments

This thesis could not have been accomplished without the assistance of many people, and I would like to acknowledge all of them.

First of all, I would like to express my great gratitude to my supervisor, Professor Masayuki Murata, for his generous guidance, insightful comments, and meaningful discussion. I was able to complete my thesis owing to his kind guidance, timely encouragement, and valuable advice.

I am heartily grateful to the members of my thesis committee, Professor Takashi Watanabe, Professor Toru Hasegawa, and Professor Teruo Higashino of Graduate School of Information Science and Technology, Osaka University, and Professor Morito Matsuoka of Cyber Media Center, Osaka University, for their multilateral reviews and perceptive comments.

I especially would like to express my sincere appreciation to Dr. Yoshinori Kitatsuji of Mobile Network Group, KDDI R&D Laboratories, Inc., for his continuously generous support and encouragement. This work would not have been possible without his patient encouragement, precise advice, and elaborated direction. I would like to acknowledge Dr. Hidetoshi Yokota and Mr. Satoshi Komorita of Mobile Network Group, KDDI R&D Laboratories, Inc., and Dr. Nozomu Nishinaga, Dr. Yozo Shoji, and Dr. Kiyohide Nakauchi of New Generation Network Laboratory, National Institute of Information and Communications Technology for their support and encouragement.

Furthermore, I must acknowledge Associate Professor Shin'ichi Arakawa and Assistant Professor Yuichi Ohsita of Graduate School of Information Science and Technology, Osaka University, and Associate Professor Go Hasegawa of Cyber Media Center, Osaka University, for their valuable comments and suggestions on my study.

I am also grateful to all of the members of Mobile Network Group, KDDI R&D Laboratories, Inc., and New Generation Network Laboratory, National Institute of Information and Communications Technology for their support and encouragement. Much discussion with them greatly helped and encouraged me.

Finally, I deeply thank my parents, my wife Hiromi, and my daughter Fumino for their understanding and hearty support and encouragement. This work would not have been achieved without them.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

Mobile networks providing cellular communication services (e.g., voice, short message service (SMS), and over-the-top (OTT) services) have become a lifeline and have been expected to ensure high reliability. In the future, mobile networks will be expected to deliver emerging communication services (e.g., virtual reality office [1], autonomous driving [2]) and be used for the Internet of Things (IoT) services. The requirements that need to be supported by mobile networks become not only diversified but also specialized. For example, a virtual reality office and autonomous driving require ultra-low latency, while the IoT services, which rarely send data, need an ultra-low cost for network usage.

It is hard for current mobile networks to meet these requirements. The reasons for this difficulty are as follows. First, the current mobile networks are composed of dedicated hardware, which is installed per functions and is partially aggregated at several sites. Therefore, all traffic, including traffic between nearby devices, is forced through these sites. This routing makes it difficult to provide ultra-low latency. The dedicated hardware leads to inefficient resource utilization. Hardware resources cannot flexibly be assigned to functions as required. Also, if some hardware is in failure (e.g., due to disasters), some functions halt. Then, services are temporarily disrupted until the hardware recovers or alternatives work. Second, in the current mobile networks, there are common

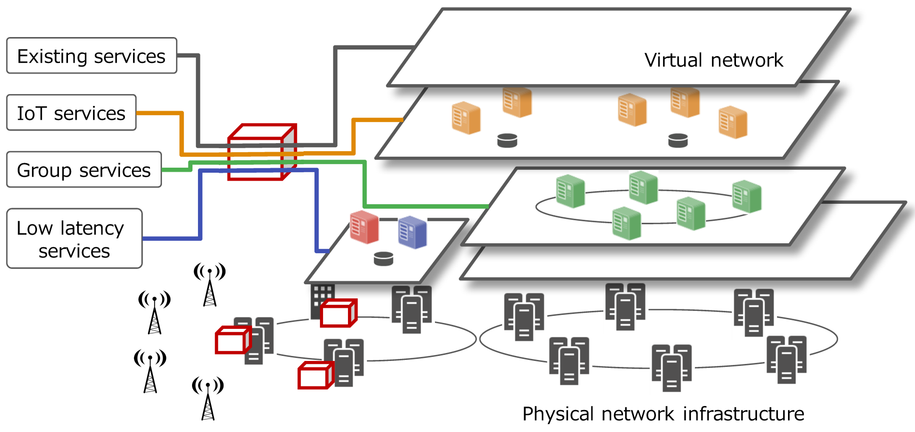


Figure 1.1: Construction of service-specific mobile networks on a single physical network infrastructure

procedures for service provision, regardless of service and the environment for service usage, as a matter of course. These procedures also are conducted for each UE and designed on the assumption that each UE individually moves. Therefore, the current mobile networks are less than effective, depending on the service or the environment for service usage. Third, a mobile core system (also referred to as evolved packet core (EPC)) that constitutes a mobile network continuously maintains connections status information for each device even if it rarely sends data. This specification requires a large number of computational resources to store and process a large amount of the information when the EPC needs to support a large number of IoT devices. Additionally, a rapid increase in demand for computational resources creates difficulties in reducing the EPC's capital expenditure (CAPEX) and operational expense (OPEX).

The descriptions above indicate that dedicated mobile networks, in which system configurations, signaling processes, and information management specialize in particular services, are desirable to satisfy diversified and specialized requirements. However, if independent and dedicated physical mobile networks are constructed to use on a case-by-case basis for satisfying different requirements, it will lead to the rise of infrastructure construction cost due to the loss of separation. Besides, such construction has difficulty dealing with the fluctuation in demand and additional network requirements immediately.

Therefore, for realizing dedicated mobile networks, network virtualization technologies [3, 4] have been promising solutions. Network virtualization technologies allow, as shown in Figure 1.1, the construction of service-specific mobile networks on a single physical network infrastructure through configuring logically independent networks (hereafter called “virtual networks”) that are composed of functions specialized for particular services. Such a construction method allows each mobile network to flexibly update its configuration or resource allocation. This ensures the provision of additional network requirements, and the total efficiency of the network.

The above-mentioned mobile network, called here “service-specific virtual mobile network,” requires the manner for configuring virtual networks and the mechanism for enabling services to be processed in an appropriate virtual network for its characteristics. In this thesis, we focus on fundamental technologies for the latter. These technologies include state information migration and gateway management. Also, we investigate service-specific signaling process and information management, which are realized by functions specialized for particular services, and we show that the service-specific virtual mobile network can achieve efficiency even when taking into account overheads caused by this virtualization. The details are described as follows.

**The mechanism for using service-specific virtual mobile network**

For devices to use virtual networks for services, we study a technology that can migrate state information, which intrinsically continues to be stored in the same function, between functions. In addition, we investigate a necessary gateway function that enables services to be assigned to appropriate virtual networks, and we tackle an improvement of the performance of the gateway function and an issue caused by the fact that many gateway functions are dispersed over a wide geographical area. The necessary routing information (which we refer to as “rules” in this study) needs to be shared promptly with few resources and a low load, even when gateways employ many rules.

**Service-specific signaling process and information management**

Specializing the signaling processes for each service can reduce the number of signaling messages, which is unnecessary or can be skipped depending on services or environments for service usage. This reduction results in reducing the load of the entire network. We have established that the reduction in load from the benefit of specializing the signaling processes is larger than the increase due to overheads of the above mechanism for using a service-specific virtual mobile network. Also, specializing information management for IoT devices, which rarely send data, can reduce the amount of state information. This reduction results in reducing a large number of computational resources to store and process a large amount of state information. We investigate a communication control method that aims to reduce the amount of state information retained in the EPC.

Note that in this thesis, virtual networks are assumed to be appropriately configured on a physical network infrastructure provided by a single mobile network operator (MNO). Virtual networks are also assumed to be not localized, that is, the deployment of functions is left out of consideration. Supposed services are provided without having packets go through multiple virtual networks, other MNO's network, and the Internet.

## 1.2 Outline of Thesis

### State Information Migration and Routing Management using Flow Control for Virtualization [5, 6]

In Chapter 2, we propose a method to enable state information migration without modifying the standard procedure in an IP multimedia subsystem (IMS) [7], which is adopted as a call control system in GSMA [8] to ensure interconnectivity between MNOs around the world. The proposed method coordinates routing control with the deployment of call/service session using OpenFlow [9], which provides an open standard protocol to program flow tables (that run at line-rate to implement firewalls, NAT, QoS, and to collect statistics) in different switches and routers [10]. We also introduce implementations that enable a large amount of state information (also referred to as call session state) to migrate by making the assumption of server failure or disaster. We verify the performance of call session state migration through a testbed and discuss the feasibility of the IMS where the proposed method is applied. Finally, we show the effectiveness of power saving.

## Reducing Signaling Load using Service-Specific Mobile Network Virtualization [11, 12]

In Chapter 3, we introduce a service-specific network virtualization to address the tremendous increase in the signaling load in the EPC and IMS of a mobile network. The proposed method creates several virtual networks that are composed of functions specialized for particular services on a single physical network infrastructure and efficiently forwards a sequence of signaling messages to the appropriate virtual networks. To prevent the forwarding delay from increasing because of the application header inspection, the mechanism effectively forwards the signaling messages with a combination of IP flow identification and application header inspection. Using a prototype system, we verify the overheads of the proposed method that are incurred during the inspection of application packet headers needed to appropriately forward signaling messages as well as the overheads incurred when replicating state information from one virtual network to another. We show that the proposed method can reduce the signaling load by approximately 25% under certain assumptions.

## Scalable Management of Gateways using Redirection-based Rules Sharing for Mobile Network Virtualization [13, 14]

In Chapter 4, we offer a method for improving the scalability of sharing the flow rules, which are used to inspect and direct packets, between gateway functions in a single physical network infrastructure. The proposed method enhances the centralized management mechanism, in which a controller reactively installs rules based on receiving the first packet of each flow, to reduce the load in the controller by decreasing the packets transmitted to the controller for packet inspection. The main feature of this method is that the gateway function redirects unknown packets to another gateway function instead of the controller, and the matched rules are installed in every gateway function on the redirection path using reverse-path forwarding. Using a prototype system, we show that the enhanced method can reduce the number of packets arriving at the controller per unit time by $\sim 63\%$ under our use case assumption.

### Reducing State Information by Time-Division Sharing IMSI for Cellular IoT Devices [15,16]

In Chapter 5, we propose a method for reducing state information maintained in the EPC. The proposed method assigns the same international mobile subscriber identity (IMSI) to multiple IoT devices, which upload data with the same cycle, and manages devices uploading data at different timings. That is, the proposed method allows a single communication line to be shared among the IoT devices using time division. The EPC's perspective sees a device that alternates between movement and communication. This is because bearers are established and released with a single IMSI by multiple IoT devices in different locations (this is not a handover, which is the process of maintaining the communication as devices move). To prevent the overlapping of communication timings, time slots at periodic intervals are introduced in the EPC. The proposed method controls, by using time slots, the timings of the registration process of IoT devices powered up (specifically powered up at random), thereby controlling the timing of uploading data after the registration process. We investigate the impact of factors affecting the performance of the proposed method through simulation-based verification and show that the proposed method can reduce the required state information to less than 0.5% compared to the current EPC method. This conclusion is based on the observation that two hundred or more IoT devices sharing a single IMSI can upload data in turns.

# Chapter 2

# Session Migration and Routing Management Using Flow Control for Virtualization

## 2.1 Introduction

To provide telecommunication services over IP, mobile network operators (MNOs) have developed an IP multimedia subsystem (IMS) [7]. The IMS is adopted as a call control system in GSMA [8] to ensure interconnectivity between MNOs around the world. The IMS enables MNOs to provide multimedia services (e.g., instant message, live streaming, and content sharing) other than legacy services (e.g., voice and short message service (SMS)). The support of these multimedia services resulted in an increase in signaling traffic in the IMS. This increase made it difficult to estimate the required system capacity. The traditional construction strategy of cellular communication systems, which deploys physical servers based on unreliable forecasts, causes excessive capacity investment in preparation to handle the increasing and unpredictable traffic. We bilieve that it is desirable to introduce the cloud concept, which is characterized by elastic and on-demand capacity, to the IMS.

Server virtualization can flexibly assign and distribute available hardware resources to functionalities called the call session control function (CSCF), which constitute the IMS, as required. This technology can adjust the number of running servers depending on the load fluctuations on call session controls. In addition to server virtualization, the number of running servers can be minimized by the use of call session state migration, which allows active call sessions to continue to be processed on another CSCF. This is because call session state migration can more sensitively adjust the number of running servers. Minimization leads to a reduction in power consumption by the system. In the case of server failure or a disaster, call session state migration can ensure reliability by continuing active call sessions on another server.

We propose a method of realizing call session state migration without extensions to the standard procedure for control of call and service sessions. The proposed method coordinates routing control with the deployment of call and service sessions using OpenFlow [9], which provides an open standard protocol to program flow tables (that run at line-rate to implement firewalls, NAT, QoS, and to collect statistics) on different switches and routers [10]. We also introduce a method that enables dynamic reconfiguration of servers constituting the IMS and an implementation for restoring a significant amount of call session state information by making the assumption of disasters. We show the feasibility of the proposed method using a testbed.

The rest of this chapter is organized as follows. Section 2.2 details the current IMS architecture and requirements for dynamic reconfiguration of the IMS. Section 2.3 describes related work. Section 2.4 explains our proposed method for routing control of service control messages using OpenFlow for call session state migration. Section 2.5 introduces implementations enabling a large amount of state information to be migrated. Section 2.6 presents verification experiments of the proposed method made by using a testbed, and discusses related issues. Section 2.7 presents our conclusions.

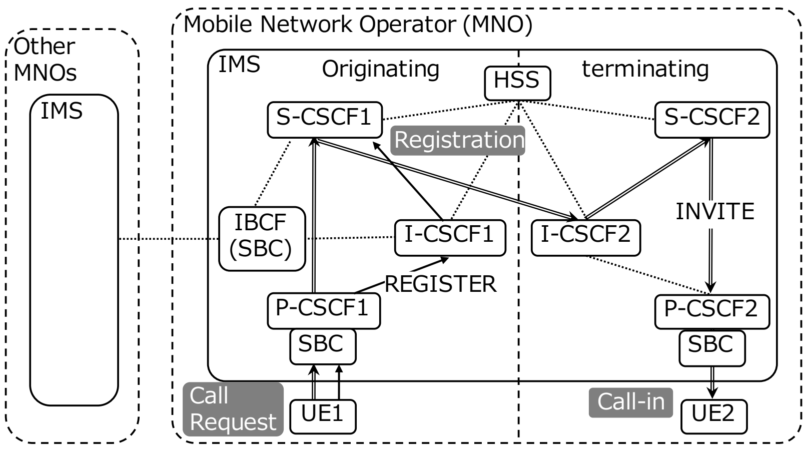


Figure 2.1: IMS architecture

## 2.2 Functionality Requirements for Cloudization of IMS

### 2.2.1 IMS Architecture and Call Session Control

Figure 2.1 shows a basic IMS architecture. The IMS is composed of CSCFs that control call/service sessions for user equipment (UE), a home subscriber server (HSS) as the database server for managing subscriber information, an interconnection border control function (IBCF) that is an entry/exit point for other IMSs, and session border controllers (SBC) that provide a variety of functions for security and connectivity (e.g., access control, topology hiding, NAT traversal, protocol interworking, and media monitoring). These functions of the SBC are often integrated into the P-CSCF (described later) and the IBCF. There are three types of CSCF: an S-CSCF (Serving CSCF) as the representative SIP server primarily dealing with call/service control and management, a P-CSCF (Proxy CSCF) that communicates with UE directly and establishes a secure connection to it, and an I-CSCF (Interrogating CSCF) that routes signaling messages to the S-CSCF managing the terminating UE.

In the IMS, SIP (session initiation protocol) [17] is used for call control between the CSCFs and UE. First, a UE registers with the IMS before obtaining IMS services. The UE registers with the S-CSCF via the P-CSCF and the I-CSCF. The S- CSCF assigned by the I-CSCF verifies the UE based on the authentication information stored on the HSS. After that, the UE uses IMS services by sending and receiving SIP messages to and from a correspondent UE via the P-CSCF, S-CSCF, and I-CSCF. For example, when the UE makes a call, the UE sends an INVITE message to the correspondent UE and establishes an active session to communicate the required information. The exchanged SIP messages take, as shown in Figure 2.1, the following path: UE1 (caller), P-CSCF1, S-CSCF1, I-CSCF1, I-CSCF2, S-CSCF2, P-CSCF2, and UE2 (callee). Note that the messages need not be forwarded through the I-CSCF (I-CSCF1) within the originating network. If the originating network operator wants to keep configuration hidden, it forwards the messages through I-CSCF1 to an entry point (I-CSCF2) in the terminating network.

This chapter defines the "Cloudization of IMS" as the means of enabling the number of running servers constituting the IMS to be adjusted on demand. This chapter refers to the status of processing services as the "call session state" and telephone services as "services" unless specifically distinguishing between telephone services and services.

### 2.2.2 Requirements for Cloudization of IMS

Once CSCFs begin to handle the active session in making the call through terminating the call of UE, it is not changed for the CSCFs to process the session because the CSCFs have the states regarding the session. However, to cope with server failure and dynamically change the number of physical servers to match the network and resource use situation, it is desirable for the call session state to migrate to another CSCF and continue to be processed (hereinafter called "call session state migration").

There are three functionality requirements for call session state information. When a call session state migrates from the original CSCS to another (conveniently called the "new CSCF"), a correspondent CSCF or UE needs to forward SIP messages to the new CSCF. The information, such as IP address in SIP headers (e.g., Route header and Via header), needs to be modified in response to session migration. However, after considering roaming UE, interconnectivity between other MNOs, compliance to standards, and performance, it is desirable to enable SIP messages to be rerouted without these modifications (Requirement 1).

To enable active call sessions in crashed CSCF to continue to be processed on a new CSCF, call sessions need to be backed up in advance and restored to the new CSCF immediately after detecting failures (Requirement 2). Call sessions are processed in one CSCF in a large amount, and the status of call sessions is frequently updated. The backup/restore function needs to ensure consistency of the status and restore call sessions without service disruption.

Besides, to minimize the number of running CSCFs with the goal of power saving, the system must dynamically determine the deployments of call sessions in CSCFs (Requirement 3). The determination process needs to calculate which CSCF should be active and which call sessions should be moved to which CSCF taking the cost of movement and permitted movement times into consideration.

Methods for satisfying the above three requirements are described in Section 2.4.

## 2.3 Related work

To realize call session state migration, methods of using the live migration of virtual machines have been studied [18, 19]. However, these approaches have two shortcomings: 1) it may cause service disruption, and 2) it may produce insufficient optimization in reducing the number of physical servers. Regarding 1), VM migration takes a long time when the VM is treating a CSCF maintaining a large number of sessions. This is because a large amount memory needs to be rewritten. Although one solution may be to have many VMs maintain a small number of sessions on a single physical server, this requires a significant overhead of computational resources and makes it difficult to minimize the number of physical servers. Also, overall power consumption increases. As a result, this leads to issue 2). There is a trade-off among these issues in the VM migration approach. Therefore, to realize power saving, the system must allow active call sessions to continue to be processed on another CSCF (hereinafter called “call session state migration”).

Studies dealing with call session state migration into the IMS have been published. In [20], an efficient mutual complementarity mechanism between P-CSCF and S-CSCF was proposed. This mechanism can support IMS reconfiguration with service continuity and cope with a single CSCF failure without backup servers. However, this coping mechanism may not work well in roaming cases where the call session state of the roaming UE is processed on P-CSCF and S-CSCF located on different MNOs. This is because the call session states need to be mutually transferred between the S-CSCF and P-CSCF over the proprietary protocol.

Y. Liu et al. [21] have proposed a mechanism to ensure reliability by restoring register and session information from a standby backup server. This mechanism can reduce the load on backup servers and produce a rapid restore by storing only the necessary information. They focus on call session state migration only between S-CSCFs.

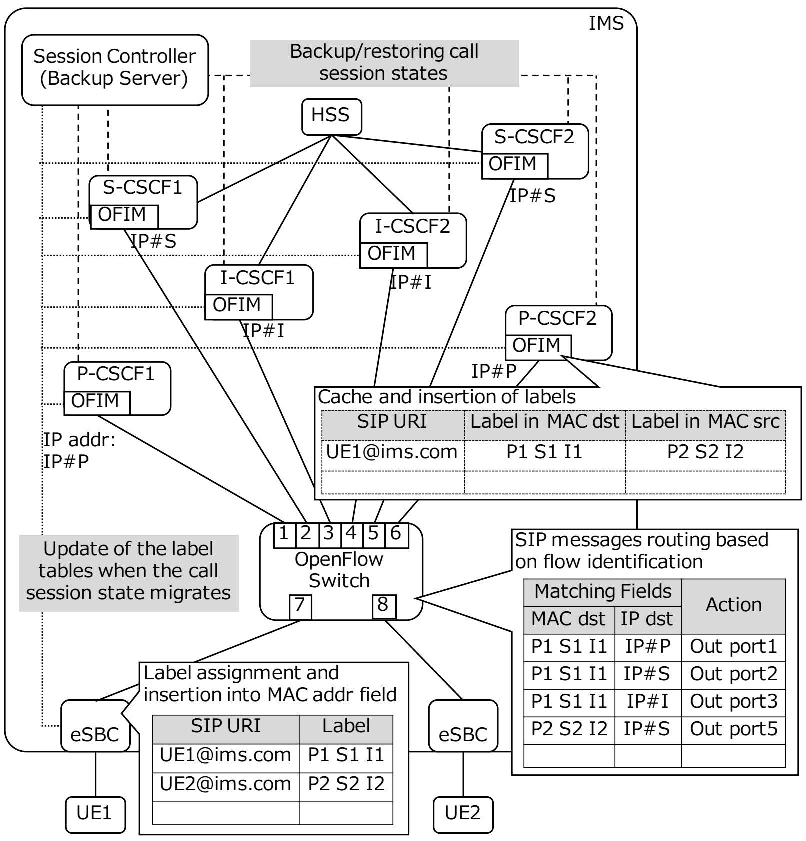
In the previous study [22], a restore mechanism for the entire IMS was proposed. This mechanism stores call session states on backup servers with the proper timing and restores the states to other CSCFs. However, to maintain IP connectivity and forward SIP messages to the new CSCF after call session state migration, it is necessary to use a specific packet forwarding mechanism that requires a proprietary tunneling technology.

## 2.4 Routing Control of SIP Messages for Call Session State Migration

To prevent the standard procedure from modifying, we assign the same IP address to the same type of CSCF and have networking equipment (e.g., switch or router) to forward SIP messages by using an identifier other than the IP address. This approach prevents SIP headers from being modified when call session states migrate to other CSCFs. We use an SIP URI (unique identifier inherent in UE) as the identifier other than the IP address.

Today's network equipment transfers data mainly using Ethernet headers or IP headers. Transmitting data by identifying an SIP URI in the payload of UDP packets drastically degrades the performance of data transfer. Therefore, we generate a label corresponding to the SIP URI and insert it in an Ethernet header or an IP header.

In recent years, the OpenFlow protocol has been used to dynamically control forwarding paths. The OpenFlow protocol separates the control plane from the data plane and allows applications to flexibly control the data plane by implementing control logic using software programming. We implement a control logic that selects which call session states should be migrated and decides the forwarding paths of SIP messages corresponding to the call session states.


IMS
Session Controller (Backup Server)
Backup/restoring call session states
HSS
S-CSCF2
OFIM
IP#S
S-CSCF1
OFIM
IP#S
I-CSCF2
OFIM
IP#I
I-CSCF1
OFIM
IP#I
P-CSCF2
OFIM
IP#P
P-CSCF1
OFIM
IP addr: IP#P
Cache and insertion of labels
1 2 3 4 5 6
OpenFlow Switch
7
8
Update of the label tables when the call session state migrates
SIP messages routing based on flow identification
Label assignment and insertion into MAC addr field
eSBC
eSBC
UE1
UE2

Cache and insertion of labels

| SIP URI | Label in MAC dst | Label in MAC src |
|---|---|---|
| UE1@ims.com | P1 S1 I1 | P2 S2 I2 |
| | | |

SIP messages routing based on flow identification

| Matching Fields | | Action |
|---|---|---|
| MAC dst | IP dst | |
| P1 S1 I1 | IP#P | Out port1 |
| P1 S1 I1 | IP#S | Out port2 |
| P1 S1 I1 | IP#I | Out port3 |
| P2 S2 I2 | IP#S | Out port5 |
| | | |

Label assignment and insertion into MAC addr field

| SIP URI | Label |
|---|---|
| UE1@ims.com | P1 S1 I1 |
| UE2@ims.com | P2 S2 I2 |
| | |

Figure 2.2: Proposed architecture

### 2.4.1 Architecture for Migration of Call Session Information

Figure 2.2 shows the proposed architecture where new functions are introduced for routing SIP messages by using an OpenFlow switch. For the OpenFlow switch to be able to identify the SIP messages as the flows, the SBC is extended (eSBC in Figure 2.2). When the eSBC receives an SIP message from UE, it retrieves the SIP URIs of the UE included “From/To” headers and inserts the labels corresponding to the SIP URIs in specific fields recognized at the OpenFlow switch. The labels represent the combination of P/I/S-CSCFs admitting the UE. In our proposal, the labels are inserted in the MAC address fields because the original values (MAC address) in the fields are not used as identifiers of the flows at the OpenFlow switch and the field has ample length. Here, labels corresponding to the originating and terminating SIP URI are inserted in the destination and source MAC address fields. The OpenFlow switch forwards the SIP messages by identifying the flows with the label in the destination MAC address field and the destination IP address.

To maintain the labels in the MAC address field when CSCFs send or receive the SIP messages, an OpenFlow Interface module (hereinafter called "OFIM") is newly introduced in CSCFs. This module caches the labels in the destination and source MAC address fields when the CSCFs receive the SIP message. Next, it inserts the same labels into the MAC address fields when the CSCFs send the SIP messages.

A session controller has a copy of the call session states on CSCFs and restores the states to the other CSCF when that CSCF halts. When the call session state migrates, the session controller updates the label table in the eSBC and CSCFs by using the OpenFlow protocol.

The session controller also dynamically migrates call session states and changes the number of physical servers to match the network and resource use situation. To do this, the session controller estimates all power consumption required for running the IMS and executing the reconfiguration and uses a linear programming method to try to reduce the total amount while still providing stable IMS service. Such a dynamic reconfiguration might lead to an increase in total power consumption. This is because even if the number of servers and the power consumption could be reduced on a temporary basis, total power consumption might increase when the reconfiguration and call session state migration happened frequently. Thus, the execution cycle of the reconfiguration is important to avoid extra reconfiguration.

We previously proposed a model of the reconfiguration and formulations using linear programming with the goal of saving power [23]. The reconfiguration will be executed if the power consumption during a certain period (cycle) from the start of the reconfiguration is less than that during the last period. The reconfiguration will also be executed if the number of sessions exceeds the capacity of the CSCF due to fluctuations in the number of sessions. Note that the frequency of the reconfiguration depends on a value of the cycle. Thus, to improve the effectiveness of saving power, an estimation of the cycle based on the call rate in an actual environment is desirable.

The following subsections describe the label assignment solution and the detailed call flow of registration, call session establishment, and the call session state migration.

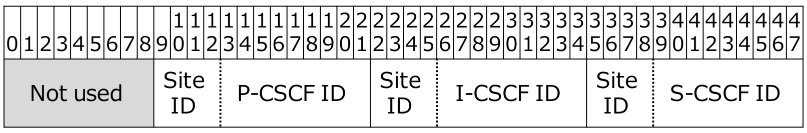


Figure 2.3: Field assignments of P/I/S-CSCF IDs in MAC address field

### 2.4.2 Label Assignment Solution

If the labels vary from one UE to another, the number of labels reaches the number of UE sets (e.g., 30 million). OpenFlow switches cannot store all of the labels. Therefore, the label represents the combination of the identifiers (IDs) of P/I/S-CSCF, where UE sets are assigned in the registration phase. The IDs also represent the combination of site identifiers (site ID) and the local identifiers of CSCFs as shown in Figure 2.3. With this label assignment solution, the number of labels stored in a OpenFlow switch become $(S_P + N - 1) \cdot (S_I + N - 1) \cdot (S_S + N - 1)$ in size. Note that $N$ denotes the number of sites and $S_P$, $S_I$, and $S_S$ denote the maximum number of P/I/S-CSCF in each site. For instance, under the assumption of 30 million UE sets, 100 individual CSCFs are required (State-of-the-art CSCF products admit 300,000 or more UE sets). In the case where 20 individual CSCFs are deployed at each site, the number of sites is five, and the number of labels stored in each OpenFlow switch is 13,824( $= (20+5-1) \cdot 3$ ). The number of labels is independent of the number of UE sets (30 million).

This reduction is responsible for the capability that OpenFlow protocol and switches deal with wildcard values. When CSCFs are distributed to several sites, OpenFlow switches are deployed at each site. In the OpenFlow switches, labels corresponding to CSCFs at other sites can be represented by labels denoting the sites.

### 2.4.3 Call Flow of Registration

Figure 2.4 shows the call flow of Registration through the OpenFlow switch. The system starts the initial settings. The session controller generates flow entries composed of matching fields and corresponding actions and labels that represent the combinations of P/I/S-CSCFs admitting the UE (step 1). The session controller adds the flow entries to the flow table in the OpenFlow Switch and

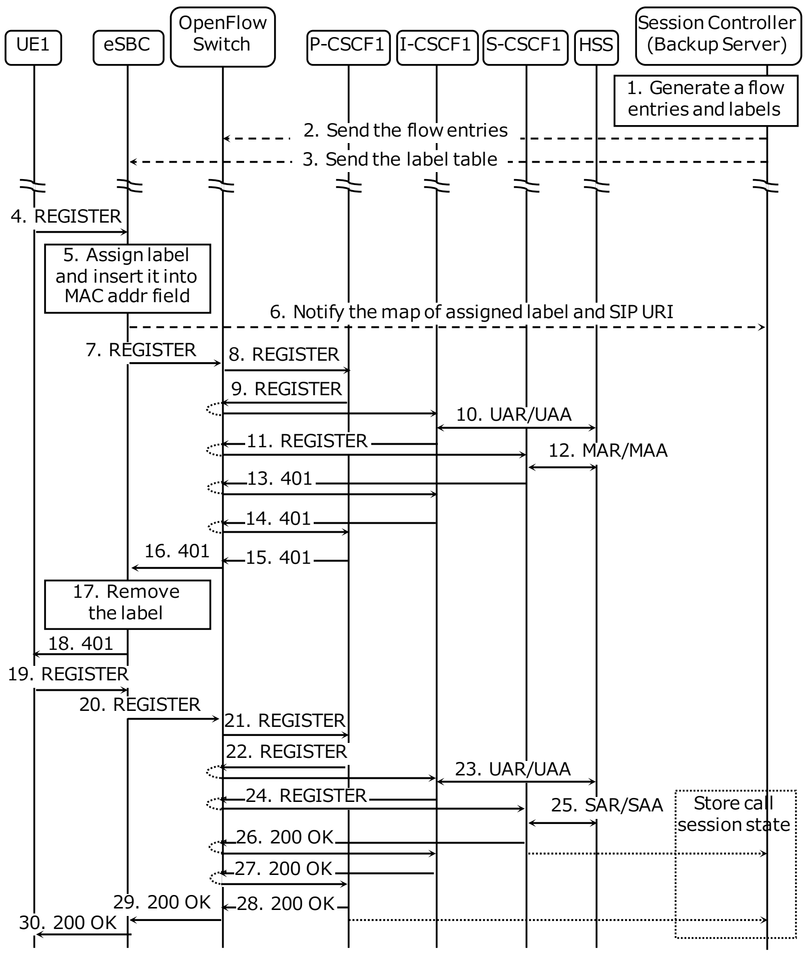


Figure 2.4: Registration flow

the labels to the label table in the eSBC by using the OpenFlow protocol (steps 2 and 3).

When a REGISTER message enters the eSBC (step 4), it first tries to retrieve the label corresponding to the SIP URI, included in the “From” header of the message. If there is none, the eSBC assigns a label that represents the combinations of P/I/S-CSCFs (step 5). The eSBC notifies the mapping information to the session controller (step 6). This is done only if the label is newly assigned. Here, the session controller prepares to store the new call session state coming later (at steps 26 and 28). The eSBC inserts the label into the destination MAC address field and sends the REGISTER message (step 7). The OpenFlow switch forwards the REGISTER message to P-CSCF1 admitting UE1 by retrieving the destination IP address and the label in the destination MAC address field (step 8). The destination IP address specifies the type of CSCF server (P-CSCF), and the label specifies the CSCF ID (P-CSCF1) to which the message forwards. When P-CSCF1 receives the REGISTER message, it first caches the label to be able to supply the same label to other CSCFs. Next, when P-CSCF1 sends the REGISTER message to the I-CSCFs (step 9), it inserts the same label into the destination MAC address fields. The IP header has P-CSCF and I-CSCF as the source and destination IP addresses, respectively. In this way, the REGISTER message and the response message (401 Unauthorized) are forwarded to CSCFs through the OpenFlow switch. When the 401 Unauthorized message returns to UE1, the eSBC removes the label from the message (step 17) and forwards it toward UE1 through IP routing (step 18). UE1 sends the REGISTER message again with the authentication information (step 19), which is included in the 401 Unauthorized message, and the eSBC inserts the same label as at step 7 (step 20). The second REGISTER message and the response message (200 OK) are forwarded to CSCFs through the OpenFlow switch in the same way (steps 21-29). Finally, the eSBC removes the label from the 200 OK and forwards it towards UE1 (step 30).

### 2.4.4 Call Flow of Call Session Establishment

After registration, the UE establishes a call session as shown in Figure 2.5. In this figure, UE1 makes a call to UE2. First, UE1 sends an INVITE message to UE2 (step 1). The SBC inserts two labels corresponding to the originating (UE1) and terminating (UE2) SIP URIs, which are retrieved from the “From” and “To” headers, to the destination and source MAC address fields, respectively (step 2). Note that the label represents a combination of the P/I/S-CSCF admitting the UE components. When the INVITE message is processed by the originating CSCFs (steps 3-6), the label corresponding to the caller’s SIP URI should be referred by the OpenFlow switch. On the other hand, when the INVITE message is processed by the callee-side CSCF servers (steps 8-13), the label corresponding to the callee’s SIP URI is referred instead. For this purpose, the I-CSCF1 in Figure 2.5 reverses the labels in the destination/source MAC address field (step 7). In the response message (steps 16-27), I-CSCF2 reverses the labels in the same manner as the request messages

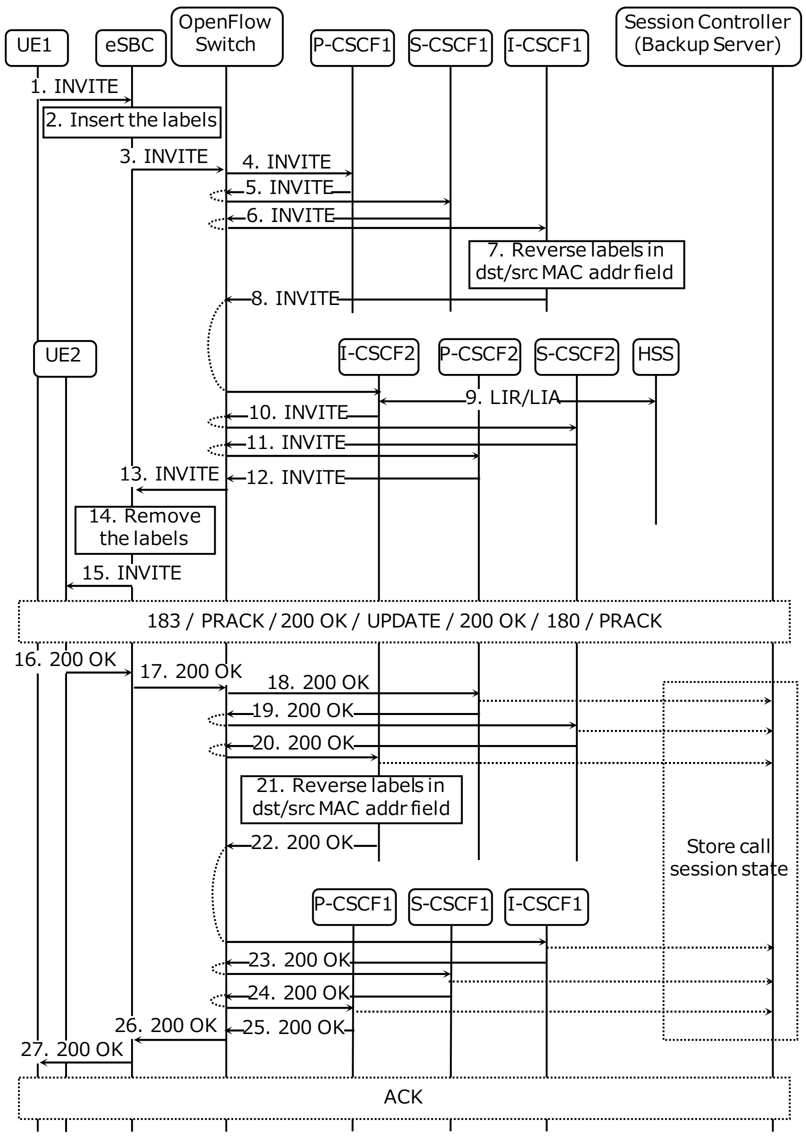


Figure 2.5: Call session establishment flow

when the messages are forwarded from the terminating-side to the originating-side (step 21). The call session state is stored in the response with proper timing [22]. After sending ACK, they can communicate with each other like a VoIP.

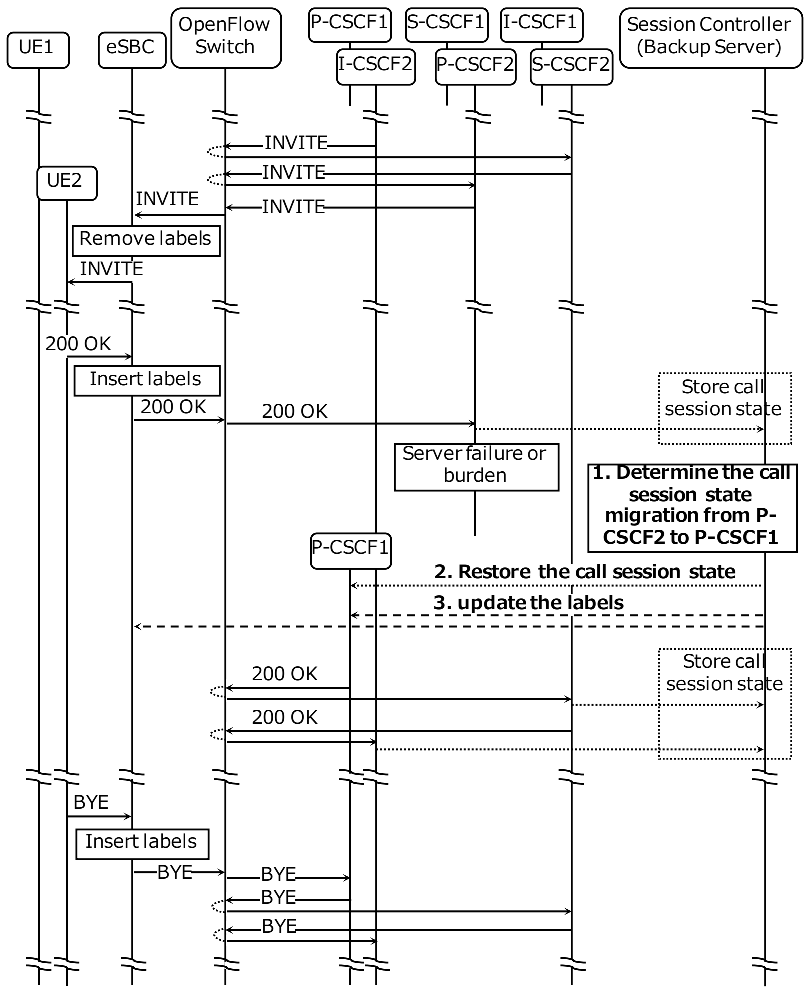


Figure 2.6: Call session state migration from P-CSCF2 to P-CSCF1

### 2.4.5 Procedure of Call Session State Migration

Call session state migration is triggered when availability monitoring detects a CSCF failure or burden. Figure 2.6 shows the procedures by which the call session states on P-CSCF2 migrate to P-CSCF1. The migration takes the following steps:

1. The session controller chooses the available CSCF (e.g., P-CSCF1) to which the call session states are restored.

2. P-CSCF1 restores the call session states from the provided data.

3. The session controller notifies the new labels corresponding to the restored session's SIP URIs to P-CSCF1 and the eSBC.

After the migration, the message (e.g., 200 OK) related to the call session state is forwarded by the insertion of the new labels on P-CSCF1. The message (e.g., BYE) from UE is on the eSBC.

There are various monitoring methods to monitor the availability of CSCFs, including periodic workload querying, network availability checks, and monitoring error and/or warning logs. The workload and/or number of admitted sessions of each CSCF can be used in choosing the available CSCF. This depends on what is monitored at the session controller according to network operator policy.

## 2.5 Implementation of Call Session State Migration

This section discusses implementations for performance improvement to deal with a large-scale server failure or disaster.

### 2.5.1 Backup Method of Call Session State

For services to be continued in the case of a large-scale CSCF failure, call session states must be restored just before the failure. To realize this, CSCFs need to store call session states on backup servers at every update of the states (states are updated for every SIP message). Besides, to cope with the failure of sites, call session states have to be stored on backup servers at other sites. However, if such a backup is frequently carried out, the delay of completing the backup accumulates. This degrades the processing performance of CSCFs. In the case of a mobile phone service, call session states are updated 60 times during call session establishment. (Details are omitted.)

We previously proposed a method to reduce the number of backup tasks from 60 to 23 [22]. We implement this method in this chapter. CSCFs retransmit SIP request messages when they do not receive the response messages corresponding to the request messages. By utilizing the retransmission mechanism, the number of backup tasks can be reduced. CSCFs back up the call session states updated by response messages, while they do not back up the call session states updated by request messages. The call session states updated by request messages can be restored when the retransmissions of the request messages are received. A restoration takes the following steps:

1. When a CSCF (old CSCF) has failed, the UE starts retransmitting request messages due to not receiving the response message corresponding to the request message. In general, UE retransmits the request message several times.

2. The call session states that maintained the old CSCF are restored to another CSCF (new CSCF). At the same time, the messages, which reach the old CSCF, are directed to the new CSCF.

3. The new CSCF receives the request messages retransmitted by the UE and restores the call session state just before the old CSCF failed.

### 2.5.2 Implementation method for high-speed processing of the eSBC

The eSBC retrieves the SIP URIs of the UE included in the “From/To” headers and inserts the labels corresponding to the SIP URIs in the MAC address fields. There are two bottlenecks for processing SIP messages: (1) the process for retrieving SIP URIs from the payload of packets, (2) the processes for entry and search in the session management table that stores correspondences between SIP URIs and labels.

To realize high-speed processing of the eSBC, we built the eSBC on the Cavium Octeon Plus CN5650 network processor. CN5650 has 12 MIPS cores running at 750 MHz. There are two programming modes provided by Cavium SDK: Linux mode (with a Linux OS) and Simple Executive mode (without OS support). We program codes in the Simple Executive mode for SIP message processing including the above bottlenecks and Linux mode for the label management function.

Figure 2.7 shows a block diagram of the eSBC. SIP message processing is composed of a packet analysis component that analyzes received packets and identifies SIP packets, session management components that retrieve SIP URIs included in the “From/To” headers and the assign labels corresponding to the SIP URIs, and a header conversion component that inserts the labels in the MAC address fields. Each component is assigned to each CPU.

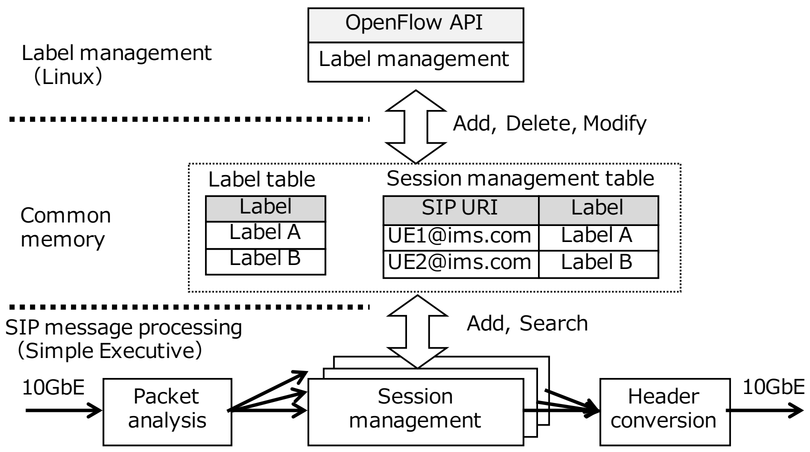


Figure 2.7: Block diagram of eSBC

To improve bottlenecks, session management was implemented in the following way. Regarding the process for retrieving SIP URIs, the Boyer-Moore algorithm [24] was used for pattern matching. The processing speed achieved 10,000 packets per second (pps). Regarding the process for entry and search in the session management table, the processes are distributed by one packet and each process accesses a common session management table. The session management table can contain one million entries and have the performance of about 200,000 accesses per second.

## 2.6 Testbed-based Evaluation

We measured the performance of call session state migration using an OpenFlow testbed, which is the constructed GICTF [25]. Figure 2.8 shows our experimental network configuration for verifying the behavior of the proposal on a real system. There are three sites: Sendai, Tokyo, and Hakata in Japan. Each site has one OpenFlow switch and several physical servers. On each server, several virtual machines are deployed. Each CSCF runs individually as software on a virtual machine. CSCFs are connected to OpenFlow switches and L3 switches. Several HSS are installed at the Sendai site, and a session controller and backup servers are installed at the Hakata site.

The CSCFs and HSS were built based on the Open IMS core [26], which is an open-source SIP server. We implemented the required module (OFIM) of the CSCFs. We simulated a large UE by

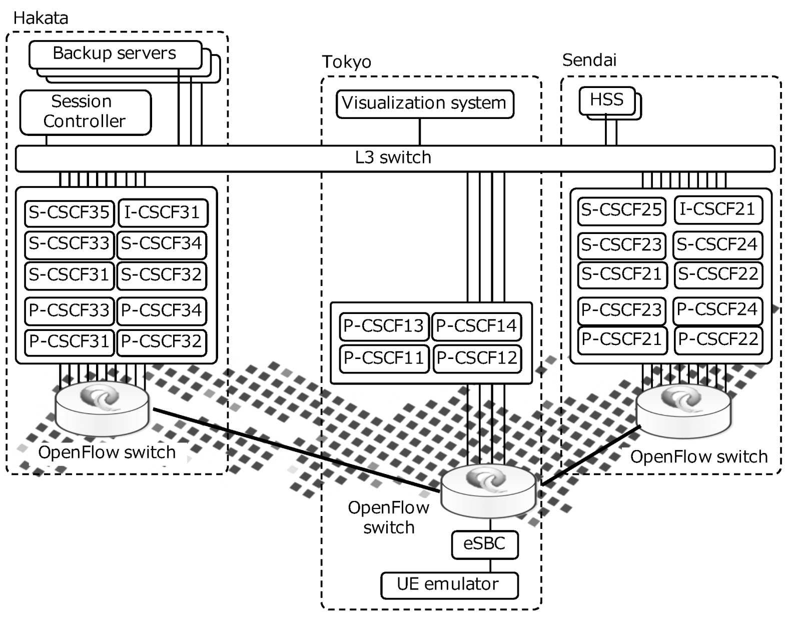


Figure 2.8: Experimental network configuration

using IMS Bench SIPp [27], which is also open-source UE emulator.

We verified the impact on CSCF performance from the backup of the call session state (Subsection 2.6.1), the restoration performance of call session state (Subsection 2.6.2), and the effectiveness of power saving (Subsection 2.6.3).

## 2.6.1 Impact of Call Session State Backup

When the CSCFs store call session states on the backup server, they keep call process waiting. Figure 2.9 shows the performance decrease of call processing due to the backup. This figure shows the call termination rate at a P-CSCF while increasing the call initiation rate. Note that the P-CSCF runs on the virtual machine deployed on the physical server. For this P-CSCF to become a bottleneck, several other CSCFs run using multiple physical servers.

The figure shows that call processing does not degrade unless the call initiation rate exceeds 100 calls per second (cps). Considering that one UE makes 1.3 phone calls per day on average [28], the IMS admitting one hundred thousand UE sets needs to handle 1,500 calls per second. If the call initiation rate reaches 20 times due to the concentration of calls in a rush hour, 300 VMs only have

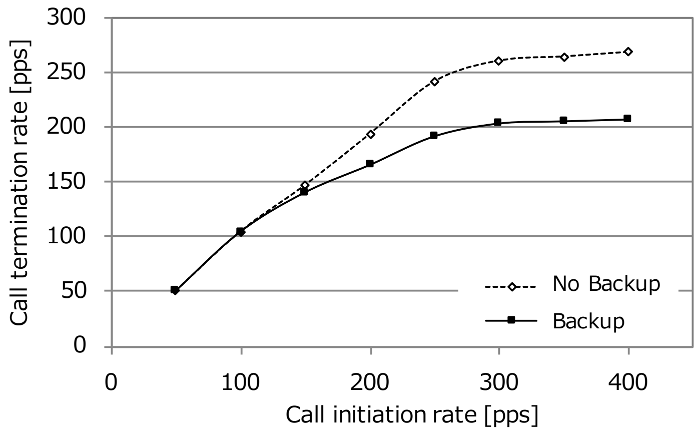


Figure 2.9: Performance of call processing at P-CSCF

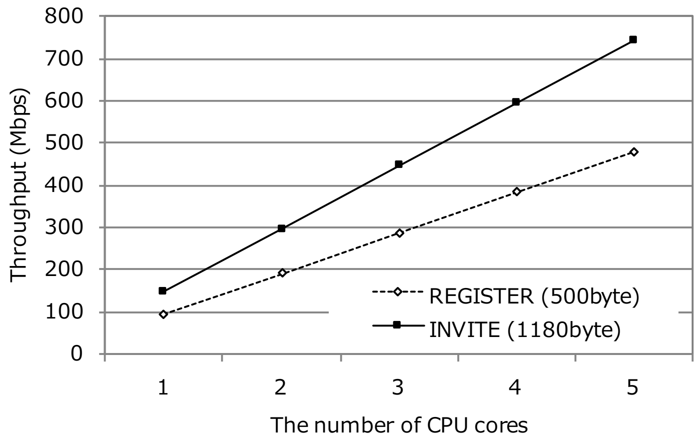


Figure 2.10: Performance of throughput at eSBC

to be prepared for preventing the performance from decreasing.

### 2.6.2 Restoration Performance of Call Session State

We verified the performance improvement of eSBC by distributing the session management tasks to CPU cores. Figures 2.10 and 2.11 show the performance of processing SIP messages at eSBC while increasing the number of CPU cores. The performance improved by increasing the number of CPU cores allocated to the session management tasks. The eSBC can handle the REGISTER messages at 120,000 pps (480 Mbps) and the INVITE messages at 78,000 pps (750 Mbps) in the case of five CPU cores. In this implementation, the increase to ten CPU cores was a meaningful

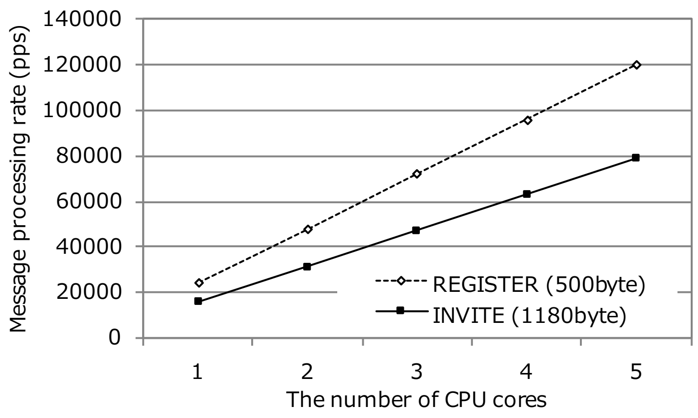


Figure 2.11: Performance of processing messages at eSBC

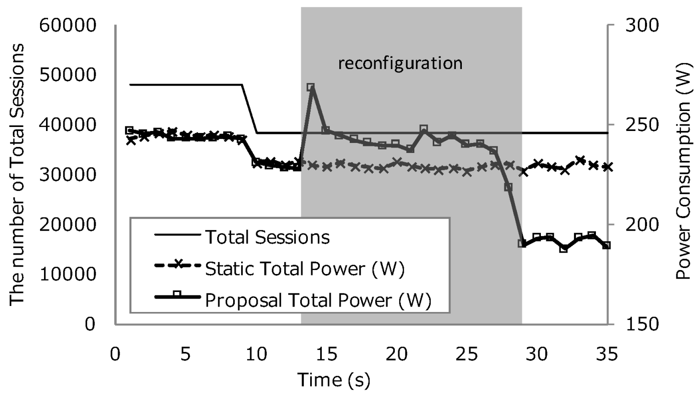


Figure 2.12: Power comsunption when the number of sessions is reduced

way for performance improvement because the session management table has the performance of about 200,000 accesses per second.

### 2.6.3 Effectiveness of Power Saving

Figure 2.12 shows the behavior of the reconfiguration when the number of sessions was reduced. “Static Total Power” in the figure is the power consumption in the static IMS configuration, as in traditional operation. In our proposal, all sessions of a P-CSCF on a node were migrated to other P-CSCFs, and the node was turned off. As shown the figure, the power consumption was reduced by only 8% in the case of the static IMS configuration, which the power consumption was reduced by 20% by changing the IMS configuration (turning off the physical server). On a temporary basis,

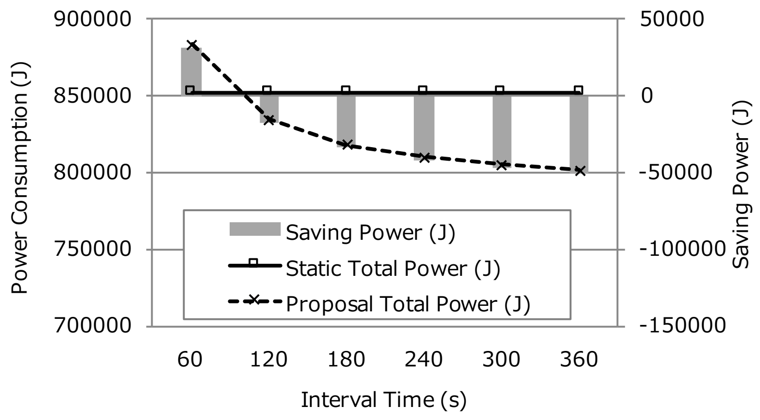


Figure 2.13: Effectiveness of power saving while increasing the reconfiguration cycle

power consumption increased for the optimization calculation (11–13 seconds), call session state migration (14–21 seconds), and the turning off of the physical server (22–28 seconds). However, it became smaller than the static power consumption after the reconfiguration.

Figure 2.13 shows the effectiveness of saving power while increasing the reconfiguration cycle. Note that the duration of each call was 120 seconds and the call initiation rate repeated between 400 cps and 320 cps. When the cycle of reconfiguration was more than double for the duration of calls, the power consumption was reduced by more than 5%. Since the call initiation rate for a day fluctuates widely (3–4 times), the IMS reconfiguration can significantly reduce the power consumption with selecting the appropriate value of the cycle.

## 2.7 Conclusion

To realize the mobile network with energy efficiency and high-reliability, we introduced call session state migration, the strategy of conducting the IMS reconfiguration, and implementations for improving the backup and restoration of call session state. We also discussed the feasibility of the proposed method using the testbed system. Our method features the migration of call session state per session and has the advantage of network operations over VM migration. Since the memory size of call session states is sufficiently smaller than that of VMs, service disruption does not occur during the migration. This advantage enables the reconfiguration reactively and on a flexible basis, resulting in the high effectiveness of saving power. Further, since the proposed method can migrate all of the call session states in a CSCF to another CSCF, operators can upgrade server software on demand.

# Chapter 3

# Reducing Signaling Load using Service-Specific Mobile Network Virtualization

## 3.1 Introduction

Mobile network operators (MNOs) have provided multimedia services (e.g., instant message, live video streaming, and content sharing) other than legacy telecommunication services (e.g., voice and short message services (SMS)) by developing all-IP based mobile networks. In recent years, they are engaged in delivering emerging communication services and controlling a large number of sophisticated network policies by introducing fifth-generation (5G) mobile networks [29]. In 5G mobile networks, mobile networks must manage a huge number of objects (e.g., base stations and mobile terminals) and perform complex controls. These requirements lead to a significant increase in the signaling processing load that may cause communication failures. As one solution to this increased load, there is a standard [30] and a concept [31] in which the functions, constituting an evolved packet core (EPC) and an IP multimedia subsystem (IMS) in the mobile network, are implemented as software on a virtualization infrastructure composed of Intel Architecture (IA) servers.

This technology (hereafter called "network function virtualization (NFV)") can immediately allocate computational resources to necessary functions if the signaling processing load increases.

The amount of signaling will increase tremendously for 5G (at the compound annual growth rate of 140% [32]), however, MNOs will need to continue to not only scale IA servers but also add a large number of them to increase the amount of computation resources in the infrastructure. Today, it is becoming harder to obtain locations and power for new rack housings [30]. Hence, it is becoming important to reduce the signaling processing load.

We presume that the signaling processing load of an entire network can be reduced by constructing several dedicated mobile communication networks, specialized for particular services, on a virtualization infrastructure. These specializations include conducting multiple common signaling processes at a time to omit redundant processes and skipping the processes that are unnecessary for certain services or circumstances. We can reduce the load by enabling services to be individually processed in logical networks (hereafter called "virtual networks" [3,4]) that are composed of these specialized functions on a physical network.

For the services to be processed in an appropriate virtual network for its characteristics, it is necessary to distribute a sequence of signaling messages processed over multiple functions to the appropriate virtual networks. Note that service is defined as the service to which is applied some policies (e.g., QoS) that are provisioned by exchanging signaling messages of application-layer control protocols (such as SIP and HTTP). The signaling messages are transmitted and received with the same port number, regardless of the service. To distribute signaling messages depending on service, the signaling messages need to be identified at the service level granularity, which is finer than the IP-flow level by indexed a 5-tuple (IP address and port number for both source and destination as well as transport type). In previous virtualization infrastructures for mobile communication networks [31, 33–36], the signaling messages are, however, identified only by the IP-flows passing through the EPC bearers (as described later). In other words, there has been no study on enabling the signaling messages such as SIP to be distributed to the appropriate virtual networks.

With the goal of reducing the signaling processing load in the EPC/IMS, we propose a mechanism that enables services to be processed in virtualized networks in which the signaling processes are specialized for particular services. The proposed mechanism forwards a sequence of signaling messages to the appropriate virtual networks and replicates state information from one virtual network to another. To prevent the forwarding delay from increasing because of the application header inspection, the mechanism effectively forwards the signaling messages with a combination of IP flow identification and application header inspection.

The rest of this chapter is organized as follows. Section 3.2 describes the scheme for providing services in the EPC/IMS and current techniques to address the significant increase in the signaling processing load. Section 3.3 presents related work. Section 3.4 discusses our approach to reducing the load using service-specific virtual networks in which the signaling processes are specialized for particular services. This section also explains our proposed mechanism that enables services to be processed in appropriate virtual networks. Section 3.5 classifies the types of service-specialized signaling processes and explains a method for reducing the number of signaling messages related to a multi-device service as an example of a specialized signaling process in the EPC/IMS. Section 3.6 evaluates the overhead and feasibility of the proposed mechanism using a prototype system and discusses challenges for its practical use. Finally, Section 3.7 concludes this chapter.

## 3.2 Existing Mobile Telecommunication Network

### 3.2.1 Scheme for Providing Services in EPC/IMS

Figure 3.1 illustrates the mobile communication network architecture assumed in this study, composed of an EPC and an IMS. The EPC is an All-IP-based packet switching network that can coordinate several wireless access systems besides Long Term Evolution (LTE). The EPC is composed of a mobility management entity (MME) that performs mobility management of user equipment (UE), a serving gateway (SGW) that accommodates base stations (called "eNBs") of the LTE, a packet data network gateway (PGW) that is a point of connection to external networks (e.g., IMS or the Internet), and a policy and charging rules function (PCRF) that controls QoS and charging policies for a communication path (called a "bearer") in the EPC. The IMS is composed of call session control functions (CSCFs) that control call/service sessions for the UE, a home subscriber

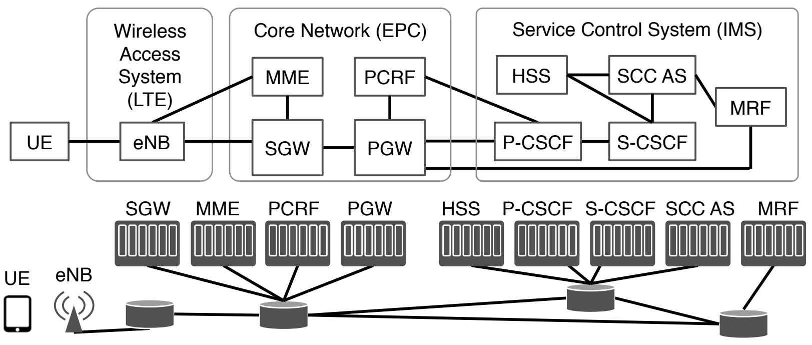


Figure 3.1: Mobile communication network architecture.

server (HSS) that is a database server for managing subscriber information, and application servers (ASs) that control various supplementary services. There are three types of CSCFs: a serving CSCF (S-CSCF) that is the representative SIP server that primarily handle call/service control and management, a proxy CSCF (P-CSCF) that establishes a secure connection and communicates with the UE directly, and an interrogating CSCF (I-CSCF) that routes signaling messages to the S-CSCF managing the terminating UE (I-CSCF is omitted in Figure 3.1). Examples of ASs include the telephony AS (TAS) that provides telephony supplementary services, the service centralization and continuity AS (SCC AS) that provides a seamless handover between heterogeneous wireless access systems or UEs, and the media resource function (MRF) that duplicates or combines media.

When the power is turned on, the UE registers itself with the EPC (this is the “Attach” procedure). In this procedure, the EPC performs location registration of the UE, assigns it an IP address, and establishes a default bearer for exchanging SIP messages between the UE and IMS. After completing Attach, the UE exchanges SIP messages (REGISTER and its response) with the IMS and registers itself. When using IMS services (e.g., a telephone service), the UE then sends/receives SIP messages to/from the correspondent UE via the P-CSCF, S-CSCF, and I-CSCF. It then establishes an active communication media for the required information. In this procedure, the PGW exchanges messages using the authentication protocol “Diameter” with the P-CSCF via the PCRF. It then establishes a dedicated bearer with a QoS for the communication media.

### 3.2.2 Rapid Increase in Signaling Processing Load and Current Actions

The signaling processing load in mobile communication networks is predicted to continue to increase [37] because of the emergence of new communication services with the introduction of All-IP networks and the shift of the environment for IMS service usage. Note that in this chapter, signaling is defined as sequences of messages exchanged to establish LTE and EPC bearers and SIP and Diameter messages exchanged for service control in IMS. Multi-device services [38] and machine-type communication (MTC) services [39] are examples of services that increase the signaling processing load. These services request to interconnect many UEs closely, causing a concentration of signaling messages triggered by concurrent requests.

Although MNOs have added hardware to their infrastructures to avoid communication failures caused by the rapid increase of the signaling processing load, they have not been able to add this hardware immediately because of the large lead time required to source proprietary hardware. MNOs also incur substantial operating costs for their infrastructures because of the increasing complexity caused by adding new proprietary hardware for new services. To solve these problems, network function virtualization technologies, which can run network functions as software on IA servers, are presently being studied. However, MNOs need to continue to add hardware, even if it is not proprietary, to address the continuous increase in the load. They also need to prepare a large number of IA servers to compensate for the low performance caused by processing packets at software switches (hereafter called "virtual switches"). Today, it is becoming more difficult to obtain a location and power for new rack housings. To curb the increase in the number of servers, it is important to reduce the signaling processing load.

## 3.3 Related Work

There are some concepts and implementations where mobile communication networks are constructed on virtualization infrastructures. Arati et al. [31] have investigated a concept that the virtualization of functions, constituting the EPC, can configure private networks, separate hardware and network resources, and optimize communication paths by appropriately deploying virtualized functions. They also have described requirements and use cases for the concept. Hampel et al. [33]

have propounded a concept that the U-plane is separated from the C-plane and is replaced with a general packet switch for simplifying and virtualizing a mobile communication network. Li et al. [34] have discussed the challenges of scalability for realizing such a software-defined network (SDN)-based mobile communication network.

Kempf et al. [35] have designed the enhancement of OpenFlow protocol to identify GTP packets (bearers in the EPC) as flows. Pentikousis et al. [36] have implemented a prototype of the SDN-based mobile communication network by using OpenFlow and enhancing a controller.

The above works are geared to the identification of the granularity of devices (the bearers are established per devices). There has been no study for the granularity of services and the overheads caused by the use of virtual networks for services.

## 3.4 Service-Specific Network Virtualization to Reduce Signaling Processing Load

### 3.4.1 Service-Specific Network Virtualization in EPC/IMS

We presume that network function virtualization technologies can reduce the signaling processing load. Specializing the signaling processes in the EPC/IMS with respect to services can reduce the number of signaling messages related to service provision. This will reduce the signaling processing load. In the EPC/IMS, there are common procedures for service provision, regardless of service and the environment for service usage, as a matter of course. These procedures are conducted with respect to each UE. In addition, these procedures are designed on the assumption that each UE individually moves. Therefore, certain procedures can be conducted at a time or skipped, depending on the service or the environment for service usage. For example, in a multi-device service, the number of signaling messages can be decreased by collectively dealing with the requests for allocating resources to the dedicated bearers (normally, these requests are generated with respect to each UE and performed simultaneously). In an MTC service, the number of signaling messages can be decreased by omitting mobility management procedures for stationary devices. Other solutions are presented in Subsection 3.5.1. The MNO configures virtual networks that are composed of functions specialized with respect to the service, as presented above, on a physical network. This configuration (hereinafter called “service-specific network virtualization”) would be able to reduce the number of signaling messages in the entire mobile communication network.

A procedure that is specialized depending on the service could be achieved by adding new functions into the EPC/IMS and extending existing functions. However, it is not easy to promptly update all related functions and eliminate mismatches between these functions with every new procedure that is specialized for services. The determination of appropriate procedures at each function per signaling message may also degrade the performance of signaling processing. Therefore, we consider a mechanism that creates several virtual networks, which are composed of functions specialized for particular services, and enables these services to be processed in the appropriate virtual networks.

For services to be processed in an appropriate virtual network, it is necessary to identify the signaling messages in the EPC and SIP messages, which are sent from the UE, with service level granularity at the eNB or the node in front of the SGW. This requires inspections of the UE identifications in the EPC Attach messages and service identifications in the SIP header for the IMS. The signaling messages provided by the S1AP protocol [40] and encapsulated in the stream control transmission protocol (SCTP) [41] are sent to the same MME, regardless of the type of UE, identified with the international mobile subscriber identity (IMSI), in the case of the EPC. Furthermore, SIP messages from a UE are sent to the same P-CSCF, regardless of the service, i.e., the services provided by the SIP messages cannot be identified using the 5-tuple. However, this inspection that covers all messages causes not only long delays in packet forwarding processing but also increases the load throughout the entire network.

In the real world, to make the service-specific network virtualization usable, it is necessary to manage resource allocation by considering the resource usage situation. There have been several studies on this subject [42, 43]. In this chapter, we focus on the mechanism to efficiently allocate services for processing in the appropriate virtual networks.

### 3.4.2 Mechanism for Services Allocation in Appropriate Virtual Networks

Figure 3.2 gives an overview of the proposed mechanism for processing services in the appropriate virtual networks using the service-specific network virtualization in the case of IMS. The key of this proposal is that a new function, added between the eNB and SGW, inspects SIP headers only when it is essential by using the information held in the SIP server. This node identifies SIP messages sent from particular UEs and inspects the SIP headers of only these messages. Therefore, all other SIP messages avoid inspection. In addition, processes performed in Virtual Network 1 are taken over by Virtual Network 2. Note that the standard EPC/IMS runs in Virtual Network 1, but Virtual Network 2 is where the EPC/IMS is specialized for a particular service. In addition, Virtual Network 2 has been already created and allocated the appropriate amount of network/server resources.

A particular service is processed in Virtual Network 2 according to the following procedures. A service request message (SIP message) sent from a UE goes through a flow identification function (FIF) and reaches a P-CSCF in Virtual Network 1 (step 1). The P-CSCF identifies a service from information in the SIP header during SIP message processing (step 2). The P-CSCF notifies a virtual network management function (VNMF) of the service information (step 3). If the service needs to be processed in Virtual Network 2, the VNMF replicates the state information on the nodes in Virtual Network 1 (in Figure 3.2, SGW, PGW, and P-CSCF are shown while the others are omitted) to the correspondent nodes in Virtual Network 2. At the same time, the VNMF inserts a flow table into the FIF for a header inspection/translation function (HIF) to inspect the SIP headers of only the SIP messages sent from the particular UE (step 4). Note that there are several methods for replicating state information: server redundancy backup methods, standardized relocation methods, and methods that use signaling messages for the handover. The P-CSCF in Virtual Network 2 takes over the process of the SIP message and sends the message to the correspondent node (step 5). The response message to this reaches the service binding function via each node in Virtual Network 2 (step 6). If each node in Virtual Network 2 is configured with a different IP address realm from Virtual Network 1, the HIF translates the parameters (e.g., the source IP address) of the SIP, IP, and GTP headers (GTP is a tunneling protocol for carrying data, i.e., SIP messages, through bearers) into appropriate parameters. The HIF then forwards the message to the UE (step 7). When receiving

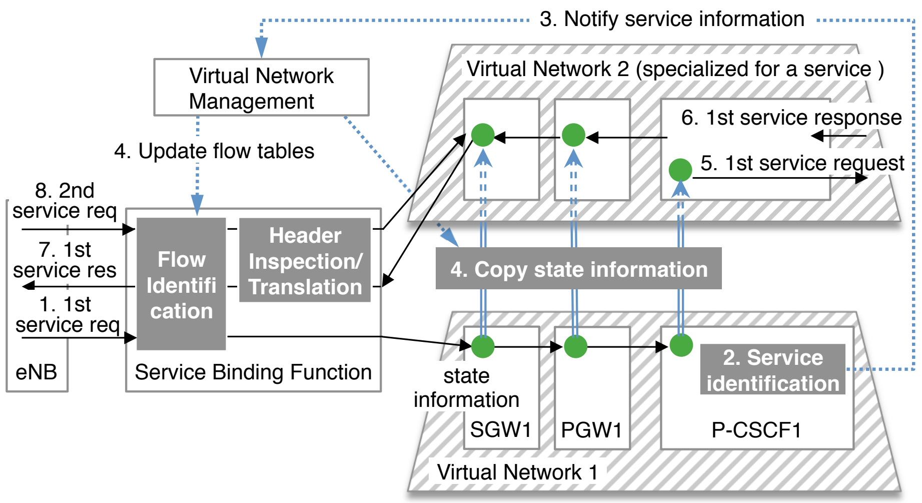


Figure 3.2: Overview of service allocation to virtual networks.

the messages from the UE, the FIF determines whether the message is the SIP message sent from the particular UE. The HIF then inspects this message's SIP header and determines whether it is related to the message in step 1. If it is, the HIF translates the parameters of the headers as necessary in the same way as in step 7 and forwards the message to Virtual Network 2 (step 8).

## 3.5 Example Solutions for Reducing the Number of Signaling Messages

### 3.5.1 Types of Specialized Controls in Virtual Networks

Solutions for reducing the number of signaling messages can be classified into the following approaches:

1. Certain procedures are conducted at a time. In the services where many UEs are closely interconnected, the resource allocation requests to initiate communications can be treated collectively (as described in Subsection IV. B). In addition to this example, the signaling messages common to a group of devices, especially messages related to mobility management (e.g., Tracking Area Update (TAU) requests), can also be aggregated for bulk handling in the

network [44]. For any function in the EPC, the signaling messages are held back for a predefined timeout or until a number of signaling messages arrive (or a combination of the two) before starting a bulk procedure. In [45], by using multicast to transmit paging information from the MME, the number of signaling messages related to paging can be reduced. This method is beneficial in making the tracking area size much larger. When the tracking area size is 90, the number of messages transmitted from the MME is reduced to one-third.

2. Certain procedures are skipped. For non-moving devices, mobility management procedures can be omitted [44]. For sporadic and small data (e.g., sensor data) generated by a huge number of terminals, the procedures for the setup and release of bearers can be omitted by alternatively transmitting the data in isolation from signaling messages in the control plane [46]. This method can reduce the number of signaling messages depending on the proportion of sporadic and small data traffic.

3. The frequency of communication attempts is reduced by setting appropriated parameters depending on services and circumstances. The number of signaling messages can be reduced by dynamically changing parameters such as the SIP retransmission timer and the paging area size. Dynamically controlling the SIP retransmission timer according to the server loads can mitigate an increase in the number of SIP messages [47]. Dynamic control of the paging area according to the movement characteristics of terminals can reduce the paging area update messages by about 64% [48].

### 3.5.2 Method to Reduce the Number of Signaling Messages Related to Multi-Device Services

We focus on session replication [39] as a representative example of multi-device services, where a UE (called the "controller UE") participating in a real-time communication service replicates the received media to several UEs (called the "controlee UEs"). Such a multi-device service could be used for not only single users but also for multiple users. Therefore, it is assumed that the number of controlee UEs is ten or more. A controller UE sends a SIP message, including the SIP URIs of the controlee UEs, to the IMS, and the media is replicated. During this replication, a significant signaling message occurs intensively because the EPC/IMS calls each controlee UE and allocates resources (prepares dedicated bearers) for these UE simultaneously. This section discusses a method for reducing the number of these signaling messages. In this method, the EPC/IMS allocates resources for UEs at a time instead of individually and, on calling each controlee UE, indicates that the resource for the UE is already ensured.

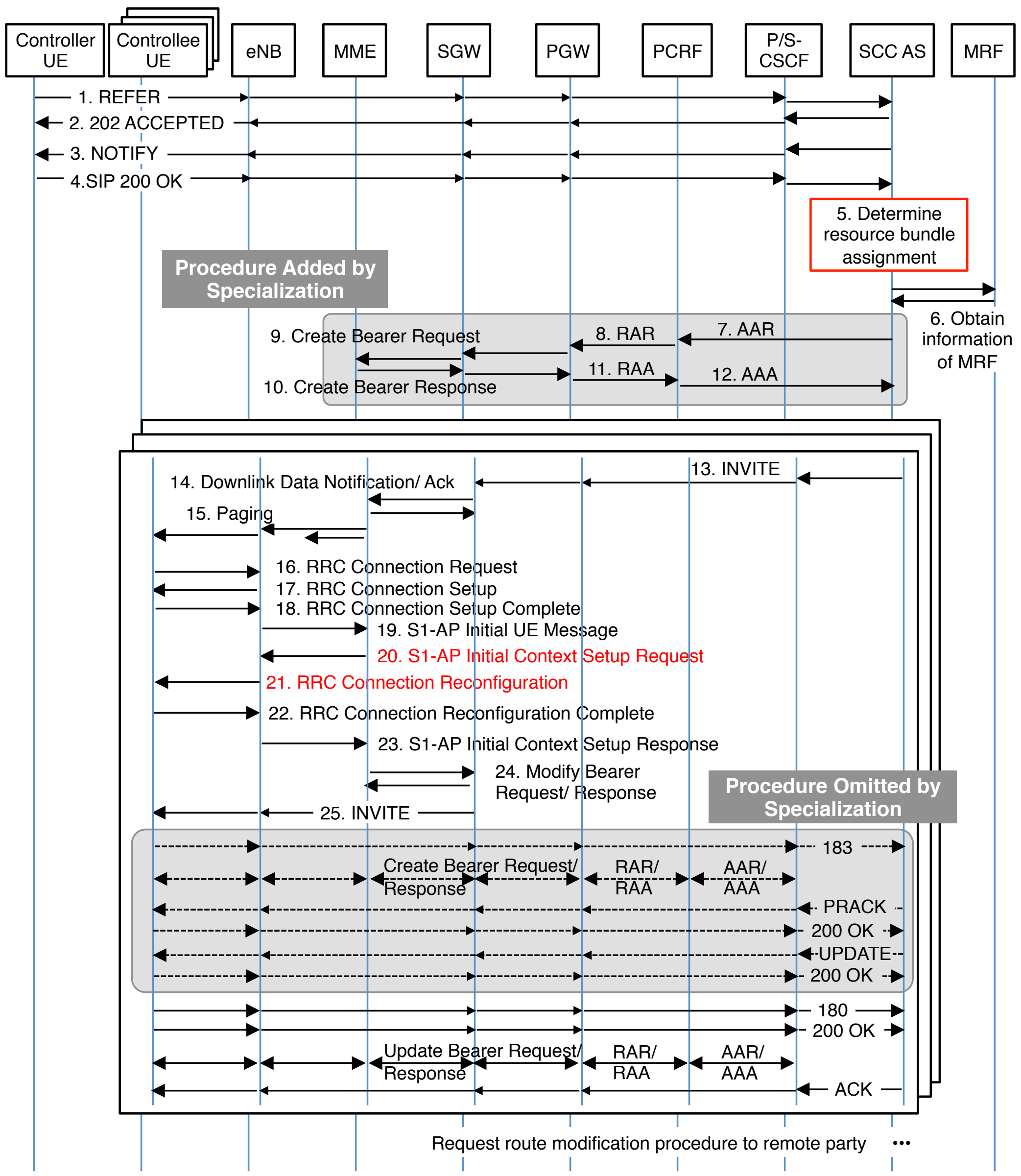

Controller UE
Controllee UE
eNB
MME
SGW
PGW
PCRF
P/S-CSCF
SCC AS
MRF
1. REFER
2. 202 ACCEPTED
3. NOTIFY
4.SIP 200 OK
5. Determine resource bundle assignment
Procedure Added by Specialization
6. Obtain information of MRF
7. AAR
8. RAR
9. Create Bearer Request
10. Create Bearer Response
11. RAA
12. AAA
13. INVITE
14. Downlink Data Notification/ Ack
15. Paging
16. RRC Connection Request
17. RRC Connection Setup
18. RRC Connection Setup Complete
19. S1-AP Initial UE Message
20. S1-AP Initial Context Setup Request
21. RRC Connection Reconfiguration
22. RRC Connection Reconfiguration Complete
23. S1-AP Initial Context Setup Response
24. Modify Bearer Request/ Response
25. INVITE
Procedure Omitted by Specialization
183
Create Bearer Request/ Response
RAR/ RAA
AAR/ AAA
PRACK
200 OK
UPDATE
200 OK
180
200 OK
Update Bearer Request/ Response
RAR/ RAA
AAR/ AAA
ACK
Request route modification procedure to remote party

Figure 3.3: Call flow in session replication for reducing the number of signaling messages.

Figure 3.3 shows the call flow in the session replication to reduce the number of signaling messages. A controller UE exchanges service request and response messages (SIP messages) with an SCC AS (steps 1-4). The SCC AS retrieves SIP URIs from this request message and obtains the corresponding terminal information by referring to the HSS. Using this information, the SCC AS determines that the resources for dedicated bearers of each controlee UE can be allocated at a time (step 5). The SCC AS obtains the information about an MRF (step 6) and sends a request message for resource allocation to a PCRF (step 7). This message contains the information (e.g., src/dst IP address, src/dst port, bandwidth and codec) about media traveling in the dedicated bearers of each controlee UE. The PCRF sends the policy information (including QoS parameters) for allocating resources to a PGW (step 8). The PGW installs the policies of each controlee UE for each dedicated bearer and sends the dedicated bearer information to an SGW. The SGW installs the policies in the same manner and forwards the information to an MME (step 9). After allocating resources at a time, the SCC AS sends INVITE messages to each controlee UE for session establishments (step 13). These messages have parameters that indicate that the resource for the UE is already ensured. When receiving these INVITE messages, the SGW buffers and sends downlink data notifications to the MME (step 14). The MME sends paging messages to each controlee UE and receives the service requests (steps 15-19). After conducting procedures such as authentication, the MME sends the request messages, which contain the information about the default and dedicated bearers retained in step 9, to the eNB (step 20). This information about dedicated bearers is transmitted to each controlee UE (step 21). Note that this transmission can be achieved without changing the standardized I/F between the eNB and the UE. After establishing wireless access bearers, the eNB sends responses to the MME (steps 22-23). The MME establishes communication paths between the eNB and SGW by sending the modify bearer requests (step 24). After that, the INVITE messages buffered at step 13 are sent to each controlee UE (step 25). The session establishment procedures between the SCC AS and each UE are conducted on the condition that the resources are already ensured. In this result, the procedures for resource allocation (the SIP messages from 183 session progress to the second 200 OK and the messages for creating the dedicated bearers in Figure 3.3) are omitted. The session between the SCC AS and each UE are established by exchanging the SIP messages from 180 ringing to ACK, and the dedicated bearers are updated to enable the transfer of

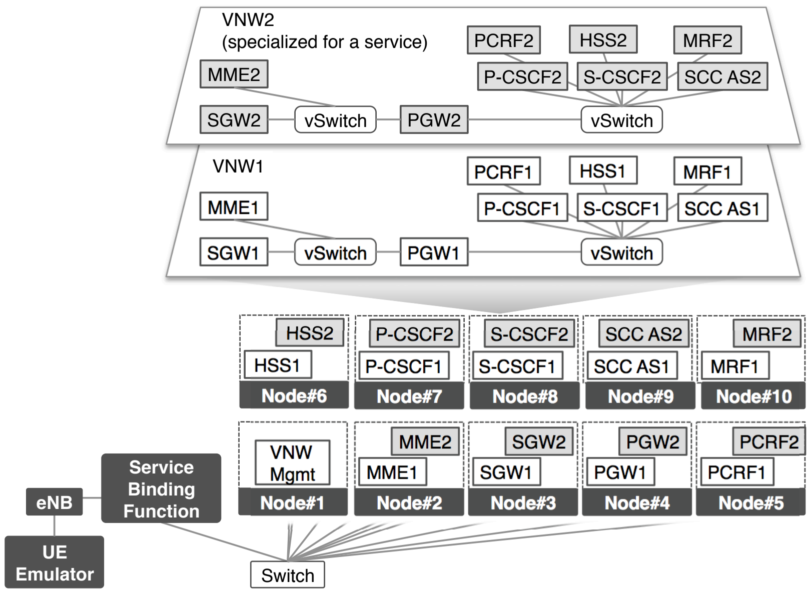


Figure 3.4: Experimental network configuration.

media flows.

## 3.6 Experimental Evaluation

This section evaluates the overheads and feasibility of the proposed mechanism using a prototype implementation. There are overheads for inspecting the SIP header at the service binding function, which is newly added in this proposal, as well as for replicating state information from one virtual network to another.

### 3.6.1 Experimental Configuration and Measurement Method

Figure 3.4 shows our experimental network configuration. Ten IA servers are connected via a switch. We constructed two virtual networks using nine IA servers (Node#2-10 in Figure 3.4). Each virtual network is configured such that each function, which constitutes the EPC/IMS, runs as software on a virtual machine individually, and each virtual machine is deployed in an IA server. We used an overlay technique, Virtual eXtensible Local Area Network (VXLAN), to construct virtual networks. In the first virtual network (VNW1 in Figure 3.4), each standard function is

Table 3.1: Experimental network components

| | Spec | | |
|---|---|---|---|
| Node | CPU | Memory | OS (VM OS) |
| IA Server (Node#1-10) | Intel Core i7-3612QE 2.1GHz | 8 GBytes | Ubuntu 12.04.4 LTS |
| Service Binding Function | | | |
| eNB | Intel Core i5-2520M 2.5GHz | 8 GBytes | |
| UE Emulator | Intel Core i7-3770T 2.5GHz | 16 GBytes | |

running. In the second virtual network (VNW2 in Figure 3.4), some functions are specialized for the multi-device service. A virtual network management function (VNW Mgmt in Figure 3.4) runs on the virtual machine deployed in the other IA server (Node#1 in Figure 3.4). A service binding function, which runs on a physical node, is added behind an eNB. A UE emulator connects to the EPC/IMS through the eNB and service binding function. Table 3.1 shows the experimental network components.

The EPC/IMS were built based on OpenEPC [49], which provides a reference implementation of 3GPP's EPC and IMS, developed by the Fraunhofer Institute FOKUS. We implemented the required module in the functions that constitutes the EPC/IMS for our proposed mechanism and the required software for the service binding function, virtual network management, SCC AS, and MRF. We also modified the UTC IMS Client [50], an open-source IMS client, for the multi-device service. We emulated a large number of UEs by using IMS bench SIPp [27], an open-source load testing software for the SIP.

We verified the overheads for inspecting the SIP header at the service binding function and replicating the state information. In the first experiment, we sent a given number of INVITE requests per second, which follows a Poisson distribution, to the service binding function from the UE emulator. We increased the request rate and measured the CPU usage and the processing time at the service binding function with SIP header inspection of either all or part (10%) of the INVITE messages. In the second experiment, we increased the number of state information replications per second (the state information is replicated from a certain node in the VNW1) and measured the CPU usage of the relevant nodes. We performed this experiment ten times and calculated the average. We also changed the number of simultaneous state information replication. We also verified the effectiveness of applying service-specific network virtualization to a mobile communication network using the multi-device service as an example. In this experiment, we set up two types of network configurations. One was configured such that the telephone and multi-device services are processed in the same virtual network (VNW1 in Figure 3.4). The other was configured such that the multi-device service is processed in a virtual network specialized for its service (VNW2 in Figure 3.4) as opposed to the virtual network for the telephone service (VNW1 in Figure 3.4) by the proposed mechanism. We fixed the arrival rate of the telephone service at 20 cps, where the utilization of the P-CSCF (which was higher than the other nodes of the utilization) was approximately 50%. During execution of the telephone service, we performed the multi-device service, where a real-time media was replicated to ten controlee UEs. We then measured the resulting CPU usage of all IA servers. We performed this experiment ten times and calculated the average.

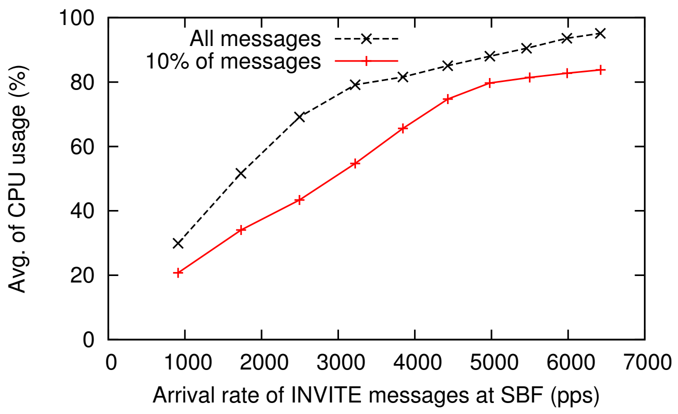


Figure 3.5: CPU usage at service binding function.

### 3.6.2 Experimental Results

Figure 3.5 shows the CPU usage of the service binding function when the mean number of INVITE messages per second is increased (in the first experiment). The solid line represents the case where

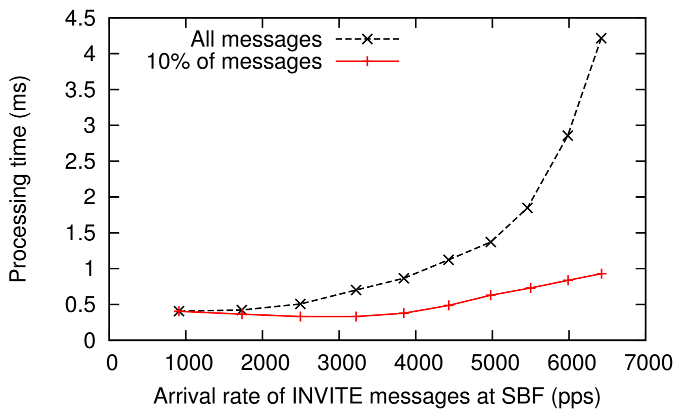


Figure 3.6: Processing time at service binding function.

10% of all arrival messages were inspected, while the dotted line represents the case where all messages were inspected. This indicates that the SIP header inspection of only some selected messages by the flow identification function can reduce the CPU usage more that the inspection of all messages. In this experiment, the CPU usage is reduced by up to 40% (at the arrival rate of 2500 pps in Figure 3.5). Figure 3.6 shows the processing time of the SIP header inspection at the service binding function. This indicates that the selection of inspection messages also can prevent the processing delay from increasing. This is because an increase in CPU usage is prevented. These results indicate that introducing the service binding function has an insignificant impact on call session establishment when the CPU usage is below 80% of maximum capacity.

Figure 3.7 shows the sum of CPU usage of the virtual machines involved in replicating state information when increasing the rate of replication (in the second experiment). Note that the CPU usage was normalized as values from 0 to 100%. When the rate of state information replication is the same, the CPU usage of VNW Mgmt function, which triggers the replication of state information and inserts flow entries to the service binding function, remained about the same, regardless of the number of simultaneous state information replications (blue boxes in Figure 3.7). On the other hand, as the number of simultaneous state information replications increased, the usage of the CPUs for the virtual machines where the destination EPC/IMS functions ran (i.e., on VNW2) increased. This is because our implementation replicates state information by sending messages per state information. In order to reduce the overhead for replications, it is necessary to reduce the number of messages by replicating in bulk. On the other hand, when a large amount of state information needs to be replicated, it is necessary to adjust the transmission rate of the state information so as not to occupy CPU time.

Figure 3.8 compares the total CPU usage of IA servers when applying and not applying service-specific network virtualization, labeled as SSNV and non-SSNV, respectively[1]. Note that the CPU usage was normalized to values from 0 to 100%. The CPU usage for processing the multi-device service was reduced by approximately 29% by enabling its service to be processed in the specialized virtual network. On the other hand, VNW1, VNW2, VNW Mgmt, and the service binding function incurred CPU usage overheads. This resulted in an approximately 25% reduced CPU usage under our assumption, even when taking the overhead into consideration.

### 3.6.3 Discussion

In the above experiments, we verified the overhead of the proposed mechanism. The processing delay overhead caused by inspecting application headers in signaling messages could be prevented from increasing unless the portion of signaling messages to be inspected became large. The CPU usage overhead for state information replication from one virtual network to another changed according to the rate of state information replication. This CPU usage became high as the rate increased. We also showed the effectiveness of service-specific network virtualization using a multidevice service as an example. Service-specific network virtualization can reduce the total CPU usage for processing signaling messages, as long as the effect of reducing the number of signaling messages is greater than the impact of processing overheads from service-specific network virtualization. This effect becomes greater as the number of terminals interacting closely increases (as in the example of the multi-device service), and this impact remains small when the requests of the corresponding service are not intensive.

In 5G, there will be the needs of rich services that enhance users' experiences (such as multi-device services). These services will burden the mobile communication network, although the busy

[1]This chapter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes a video that shows the effectiveness of service-specific network virtualization by taking the multi-device service as an example. This material is 63.4 MB in size.

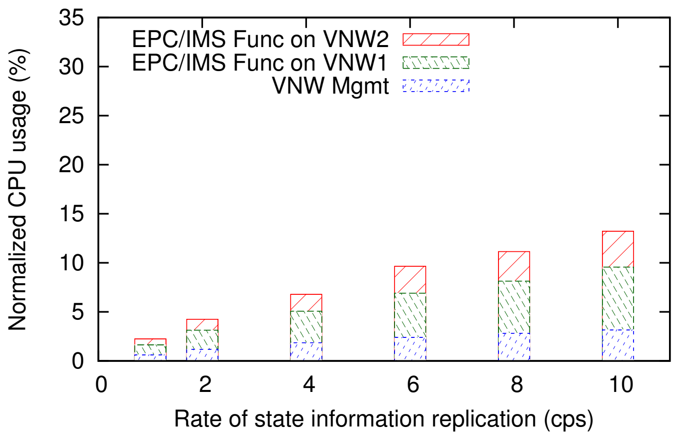
hour call attempt (BHCA) of its services will be comparatively small. Under these circumstances, MNOs will benefit from service-specific network virtualization.

To maximize the benefits of using service-specific network virtualization, it is necessary to establish a model for the management of virtual networks. There will be several service-specific virtual networks applied the efficiency solutions (as shown in Subsection 3.5.1). It is important to determine the amount of resources allocated to each service-specific virtual network at different times. It will also be important to determine the timing for forwarding a sequence of signaling messages to the appropriate virtual network by considering the situation of the mobile communication network.

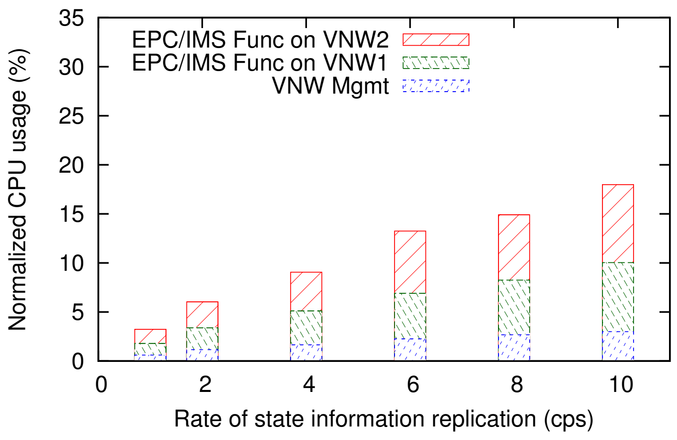
In the real world, the service binding function is distributed geographically. In addition, to prevent signaling message processing overheads from increasing, a certain number of service binding functions are deployed. In the case of multiple service binding function deployment, some service binding function are required to share the necessary information to distribute a sequence of signaling messages, which are sent and received by mobile terminals, to the appropriate virtual network. The challenge is to determine a reasonable sharing mechanism that can promptly share the necessary information with few resources and a low load, even when the information referenced by service binding functions becomes huge in volume.

## 3.7 Conclusion

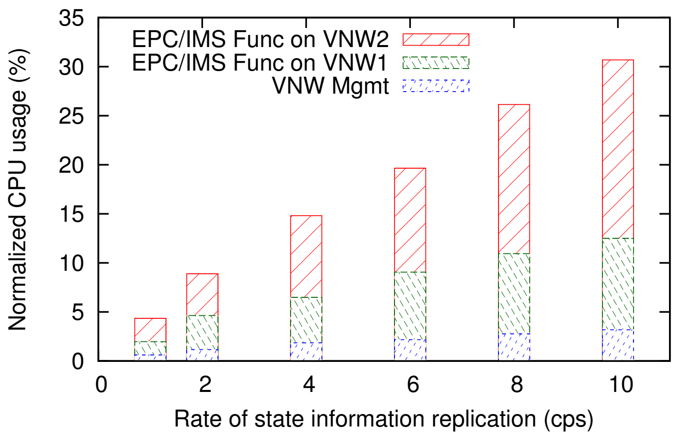
To reduce the load of signaling message processing in a mobile communication network, we proposed an approach that creates several virtual networks that are composed of network functions specialized for particular services. We also proposed a mechanism that enables services to be processed in the appropriate virtual networks by forwarding a sequence of signaling messages to these networks and replicating state information from one virtual network to another. The mechanism can effectively forward the signaling messages using a combination of IP flow identification and application header inspection. Using a prototype implementation and actual measurement, we showed that the processing delay overhead caused by forwarding the signaling messages does not have significant impact on service delay. We also showed that the CPU usage overhead for replicating state information can be low, depending on the frequency of the replication. In addition, using a virtual network specialized for a multi-device service as an example, we showed that the proposed mechanism could reduce the load by approximately 25% under our assumptions. We also discussed the challenges for the proposed mechanism with respect to maximizing load reduction and implementing it in practice.



(a) Number of state information replicated at time: 1



(b) Number of state information replicated at time: 5



(c) Number of state information replicated at time: 10

Figure 3.7: CPU Usage for state information replication. (a) Number of state information replicated at time: 1. (b) Number of state information replicated at time: 5. (c) Number of state information replicated at time: 10.

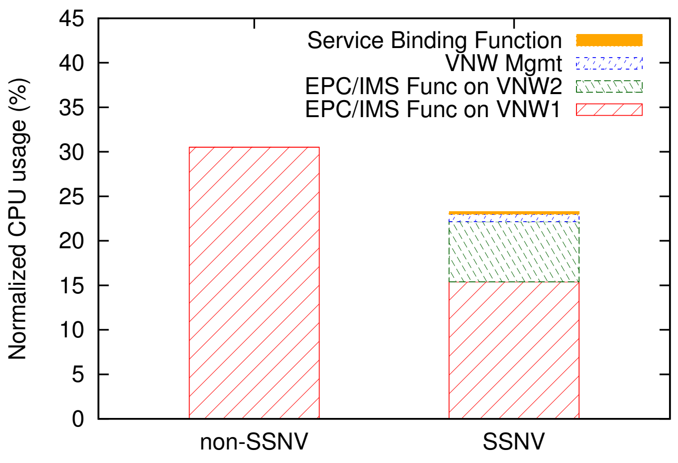


Figure 3.8: Comparison of total CPU usage between applying and not applying service-specific network virtualization.

# Chapter 4

# Scalable Management of Gateways using Redirection-based Rules Sharing for Mobile Network Virtualization

## 4.1 Introduction

Fifth-generation (5G) mobile networks [29] need to deliver newly emerging communication services and control a large number of sophisticated network policies for these services. A key of the 5G mobile networks is the agile service provisioning by meeting each service requirements with specific customizations through all the segments (the wireless, backhaul, core network, and service system (service servers) segments). For example, device-to-device (D2D) communication requires the dynamic network-controlled allocation of resources to the D2D link [51], while a virtual reality office requires the dynamic placement of data files or functions close to group members as well as the efficient setup of group communications [1]. In terms of the agility, the incorporation of new technologies in exiting equipment disrupts prompt responses to new changes, and the common procedure employed by one model of network deployment may be inefficient for some services.

The use of virtualization technologies, such as network and network function virtualization, is a promising solution for promptly introducing new technologies and efficiently meeting the requirements of all services. Previous studies [11,31,43,52] have shown that complex controls and network configurations can be optimized for individual services by constructing a number of dedicated mobile networks, customized for particular services, on a virtualization infrastructure (hereafter the dedicated mobile network is referred to as "virtual mobile network").

To provide services using an appropriate virtual mobile network that has been customized to their characteristics, it is necessary to introduce a function at base stations (called eNBs in long-term evolution (LTE)) or nodes in front of the virtualization infrastructure [11, 52]. This function identifies the communications among services and/or user terminals using packet inspection, thereby directing a sequence of packets related to the service to the appropriate virtual mobile networks. Previously, we demonstrated the feasibility of this type of function (which we refer to as a service binding function (SBF)) by proposing a method that can effectively direct packets using a combination of IP flow identification and application header inspection [11]. In the real world, multiple SBFs are dispersed over a wide geographical area and the challenge of addressing the user terminal's mobility demands the sharing of rules that facilitate flow control (which we simply refer to as "rules" in this study), which are installed in each SBF and used to inspect and direct packets. The necessary rules need to be shared promptly with few resources and a low load, even when many rules are employed by SBFs.

Utilizing centralized management [53], adopted in software defined network (SDN) technologies, is a promising solution to allow the reasonable sharing of rules. A centralized controller has two main procedures: one that installs flow rules preliminarily into switches directing the flows, the other that returns the flow rules to switches, which receive the first packet belonging to unknown flows (the controller did not specified to direct flows), in order to install the queried flow rules in the switch. In the latter case, the switches send the first packet to the controller.

Using this scheme, each SBF (as a role of SDN switches) can obtain the necessary rules by transmitting the first packet ("unknown packets") during each transaction to the controller that has all the rules. However, the centralized controller becomes a bottleneck due to the high rate of unknown packet arrivals, which triggers a packet inspection process that requires a certain amount of time because of a large number of rules. This arrival rate will increase with the SBFs in response to the increase in capacity and the number of eNB. Distributing the load over replicated controllers may appear to be a natural solution to this bottleneck, but it is not easy to coordinate numerous other replicas while maintaining consistency due to the large number of rules and the frequent changes in the rules. Therefore, it is very important to reduce the number of unknown packets transmitted to the controller for benefiting from load distribution solutions by preventing the controller replicas from increasing.

In order to improve the scalability of the rules sharing for centralized management, we propose a method for redirection-based rules sharing that can reduce the load on the controller and result in preventing numerous replicas over multiple controllers. In the proposed method, the SBF redirects the unknown packets to another SBF, which is likely to have the necessary rules, instead of the controller. During redirection, the target SBF inspects the packets with stored rules and forwards them to appropriate virtual networks without invoking the controller if it has the necessary rules. In addition, the matched rules are installed in other SBFs along the redirection path. This results in increasing the number of rules cached in SBFs and reduces the transmission of unknown packets to the controller. To detect the target SBF, the proposed method centrally manages the locations where the user terminals have requested their services previously. Since this location management stores only the mapping between a user terminal and the last location of it, this centralized management has an insignificant load. This chapter shows that the proposed method can reduce the arrival rate of unknown packets to the controller by up to 63%. Preliminary simulation results of this work have been published in [14]. This chapter shows the effectiveness of the proposed method through the implementation-based verifications.

The rest of this chapter is organized as follows. In Section 4.2, we introduce service-specific mobile network virtualization and the flow control scheme in this technology. This section also includes a description of the scalability issues when sharing rules. In Section 4.3, we describe related research. In Section 4.4, we explain our proposed method, which can reduce the need to invoke the controller to inspect packets. In Section 4.5, we present performance evaluations of the proposed method and the centralized-based method using a prototype implementation, and we discuss related issues. In Section 4.6, we give our conclusions.

## 4.2 Technologies for Service-Specific Mobile Network Virtualization

### 4.2.1 Current Mobile Networks

Figure 4.1 shows the LTE and evolved packet core (EPC) architecture, which is a representative mobile network that is deployed at present. Packets from user terminals (called user equipment (UE)) are routed using two GPRS tunneling protocol (GTP) tunnels, which comprise a data path (a "bearer"), before they are sent to external networks (e.g., IP multimedia subsystem (IMS) or the Internet). There are base stations ("eNBs") for the LTE, a serving gateway (SGW), and a packet data network gateway (PGW) on the data path, where their main function is packet routing/forwarding and policy enforcement. The SGW serves as a mobility anchor to enable seamless communication when the UE moves from one eNB to another. The PGW performs traffic monitoring, billing, and access control, and it acts as a gateway to the external networks. The GTP tunnels are created, removed, and updated by eNB, SGW, and PGW in a coordinated manner using a mobility management entity (MME) and the policy and charging rules function (PCRF). The MME is a control plane function, which is responsible for mobility management and user authentication via a home subscriber server (HSS). The MME interacts with the eNB and SGW to facilitate bearer session management, through which the GTP tunnels are created, removed, and updated. The PCRF provides the PGW with the QoS authorization (QoS class identifier and bit rates) based on the user's subscription profile and/or the media information (e.g., the TCP/IP 5-tuple and media codecs) received from external networks.

Suppose that the UE wants to use a communication service over the IMS or Internet. First, the UE sends a bearer session establishment request, which includes an access point name (APN) that specifies the type of external network where the desired service is provided, to the MME via the eNB. The MME detects a SGW that can be contacted to create a GTP tunnel to a PGW, which is the point of connection to the external network identified by the requested APN. The contacted SGW creates the GTP tunnels with the PGW and the eNB. Next, the UE exchanges the application-layer control protocol messages (such as SIP or HTTP messages) with service functions in the external networks over the GTP tunnels. During this process, the service functions send media information to the PCRF, and the PCRF sends the QoS authorization and media information to the PGW. This

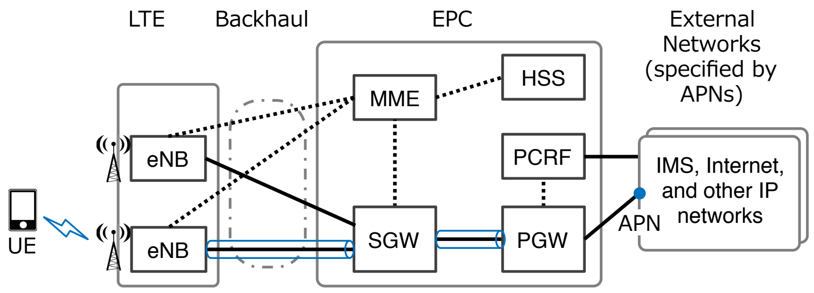

LTE
Backhaul
EPC
External Networks (specified by APNs)
MME
HSS
PCRF
eNB
eNB
SGW
PGW
UE
APN
IMS, Internet, and other IP networks

Figure 4.1: Standard mobile network architecture.

information is then forwarded to the MME via the SGW. The MME then requests the eNB to allocate radio resources and to establish a connection with the UE. The PGW and SGW create new GTP tunnels (a "dedicated bearer") or update existing GTP tunnels for the media, as necessary.

Current mobile networks have several limitations. In particular, they force all traffic, including traffic between UEs on the same operator's network, through the PGW or service functions in the external networks, thus they make it difficult to provide very low latency. In addition, mobile networks are inefficient because of the common procedures and parameters used for session management. Regardless of the services and the environment employed for service, the data paths are handled using common procedures and parameters. These procedures and parameters are also used for each UE and they are designed based on an assumption that each individual UE moves separately. It is not realistic to specialize the procedures and parameters depending on the services by extending existing functions. This is because it is not easy to promptly update all related functions and to eliminate mismatches between these functions with every new procedure and parameter that are specialized for services.

### 4.2.2 Service-Specific Mobile Network Virtualization and Flow Control Schemes

Previously, we assumed that these concerns can be resolved by configuring several dedicated mobile networks, specialized for particular services, on a virtualization infrastructure [11]. Figure 4.2 shows an example of this configuration (referred to as a "service-specific mobile network virtualization" in our earlier study). There are one common virtual mobile network (labeled as COMN-vMNW in Figure 4.2), which is composed of standardized functions and is regularly used, and service-specific virtual mobile networks (labeled as SS-vMNW in Figure 4.2), which are composed of the necessary functions specialized for services. A SS-vMNW has only data plane functions (SGW, PGW, service functions in external networks, etc), another has control plane functions (MME, PCRF, etc) as well as data plane functions. For example, the former case requires running data plane functions on physical nodes close to UEs for very low-latency communication services to multiple UEs (e.g., virtual reality office or online gaming). The latter case requires modifying procedures at each function from the standardized for efficiently handling multiple data paths in a group communication service. Each virtual mobile network is configured such that each function runs as software on a virtual machine individually, and virtual machines are deployed in Intel Architecture (IA) servers. Packets exchanged among functions are encapsulated by a virtual switch in the IA server using overlay protocols, such as Virtual eXtensible Local Area Network (VXLAN). In order to allow packets that sent from the UE to be processed in an appropriate virtual network, these packets are required to be encapsulated at the UE or the node between the UE and IA server. In the case of the UE approach, it is necessary for the UE to be equipped with fast packet I/O mechanisms and to handle layer-2 frames. Thus, the UE approach has the limitation that only modified UEs obtain benefits from service-specific mobile network virtualization.

We have demonstrated the prototype of service-specific mobile network virtualization where a new function (called SBF in our earlier work), added behind the eNB, inspects service request packets for identifying services and directs them to the appropriate virtual mobile network using encapsulation. The reason why packet inspection is needed is that the 5-tuple cannot identify the bearer session management messages in the EPC and the service session control messages (such as SIP and HTTP messages) that are sent from the UE with service level granularity. The signaling messages that are provided by the S1AP protocol [54] and encapsulated in the stream control transmission protocol (SCTP) [41] are sent to the same MME, regardless of the types of UE and APN in the case of the EPC. Furthermore, SIP messages and HTTP messages from a UE are sent to the same SIP server and webRTC server, respectively, regardless of the service. This requires inspections of the UE and APN identifications in the bearer session management messages and service identifications in the service session control messages.

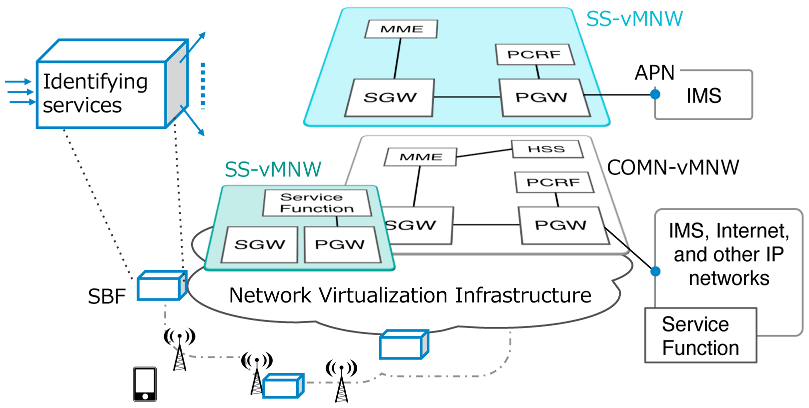


Figure 4.2: Service-specific mobile network virtualization.

### 4.2.3 Requirements for Sharing Rules and Scalability Issues

In the real world, the SBFs are dispersed geographically. In addition, multiple SBFs need to be deployed for distributing the load. In terms of end-to-end delay, it is desirable that the SBFs are located near UE, i.e., at the eNB. This is because gateway and service functions, which act as anchor points, can be moved close to UE. However, given the number of eNBs increases with decreasing cell size for providing large capacity, it is not realistic in terms of CAPEX and OPEX that all eNBs are equipped with the SBF. Alternatively, the SBF can be installed in a backhaul network and part of the macro-eNB. In the case of multiple SBF deployment, some SBFs must share the necessary rules in order to inspect packets and direct a sequence of packets to the appropriate virtual network even if the UE moves. Installing all of the rules in advance is not attractive due to the large number of rules employed and dynamic changes in the rules. However, by utilizing centralized management technology [53] instead, adopted in SDN technologies, the rules can be installed reactively in response to the mobility of user terminal. Thus, the SBF sends the unknown packet during each transaction to a centralized controller that has all the rules. The controller inspects the packet and installs the matched rules in the SBF, which sent the unknown packet.

Identifying services by inspecting packets at the controller requires a specific amount of time because the controller has many rules [55]. Therefore, the large number of unknown packets that arrive at the controller per unit time due to the deployment of many SBFs creates a bottleneck that prevents the transmission of packets to the virtual network. Distributing the load over replicated controllers may appear to be a natural way of scaling the system, but this approach requires that each controller maintains of all the rules and that it coordinates with the other replica controllers to maintain consistency when the rules change [56]. It is not easy to distribute the load over many replicas. To prevent the number of replicas from increasing, it is important to reduce the number of unknown packets transmitted to the controller for packet inspection.

## 4.3 Related Work

To reduce the load on the controller, some approaches can decrease the number of interactions between the controller and the flow-based switches that it manages. DIFANE [57] employs a flow management architecture that maintains all of the traffic in the data plane by selectively redirecting packets through intermediate switches, which store the necessary rules in advance. In this architecture, the controller recomputes the appropriate partitioning of the rules and updates new partition rules in all of the switches if the rules change, or due to the mobility of the user terminal. This process takes longer as the number of switches increases.

DevoFlow [58] enhances the openflow protocol where switches only invoke the controller after detecting long-lived or high-throughput flows. This approach targets re-routing or load balancing for the flows, which are characterized as significant based on statistical analyses. However, this approach does not work well if the flows cannot be identified based on statistics alone (our method needs to inspect application headers in packets).

## 4.4 Redirection-based Rules Sharing

Our proposed method, redirection-based rules sharing, improves the scalability of the rules sharing approach using the centralized management method. The key feature of this method is that instead of the controller, the SBF obtains the necessary rule from another SBF, which accommodates the eNBs where the UE camped previously, by redirecting the unknown packet to this SBF. The target SBF inspects the packet payload with stored rules and returns a rule that identifies a specific service to the reverse path. This rule is cached and forwarded hop-by-hop at every SBF on this path. Thus, we add capabilities for redirection management to the controller and cache management to the SBFs. In this section, we present an overview of the proposed method. We then show how the packets travel to the virtual networks.

### 4.4.1 Overview of the Architecture

Figure 4.3 gives an overview of the proposed method. In the SBF controller (SBFC), a redirect manager function is added. When the SBFC receives a partial packet containing a UE identifier from a SBF (this method is describes below), the redirect manager function returns the redirection destination corresponding to the UE if this is present in the function; otherwise, the redirect manager function returns a request to have the SBF send the whole packet. When the SBFC receives the whole packet, the service identification manager function inspects the packet payload with all the rules for identifying a service and installs the matched rule in the SBF that sends the whole packet.

In the SBF, piecemeal inquiry and packet redirection behaviors are added to the service identification function, and a cache manager function is also included. The piecemeal inquiry capability performs appropriate actions according to the type of message received and/or the destinations. When an unknown packet with a service that is not identified by the installed rules arrives from the UEs, the SBF sends the partial packet containing the UE identifier to the SBFC (as described above) and buffers the whole packet. After receiving the packet that indicates the requirement to send the whole packet, the SBF sends the buffered packet to the SBFC. After receiving the information related to the redirection destination (target SBF) from the SBFC, the SBF redirects the buffered packet to the target SBF. After receiving the redirected packet, the target SBF inspects the packet payload with the installed rules to identify the service. If the service can be identified, the SBF encapsulates and forwards the packet to the appropriate virtual network; otherwise, the target SBF sends the whole packet to the SBFC.

The cache manager function caches and forwards the rule, which identifies the service related to the redirected packet, hop-by-hop. In this behavior, the cache manager function records the arrival interface of the redirected packet when the redirected packet arrives, before forwarding the packet

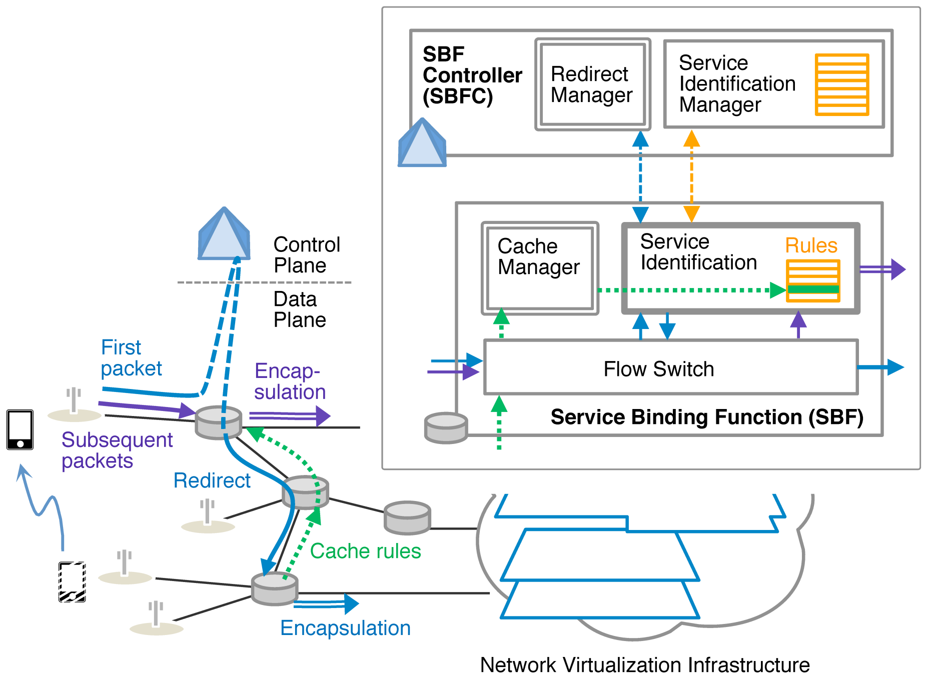


Figure 4.3: Overview of redirection-based rule sharing.

based on the forwarding information base. After receiving the rule, it is stored by the cache manager function and forwarded toward the recorded interface.

### 4.4.2 Inquiry of Redirection Destinations

The proposed method centrally manages the redirection destinations. The SBFs contact the SBFC to obtain the redirection destination. To avoid wasting the bandwidth between the SBFC and the SBFs, the SBFs send a partial packet when requesting the redirection destination. At this time, the SBFs buffer the whole packet. If the buffer is full, the SBFs send the whole packet to the SBFC without the inquiry of redirection destinations. That is, the redirection of unknown packets is not performed. The buffer also has a timer for discarding buffered packets. If the buffer size and the timer are large, the unknown packets will spend more time waiting inside. This causes a large number of retransmission of unknown packets (the applications, which send the service requests, normally has a retransmission mechanism). If the buffer size is too small, the number of redirection decreases. Therefore, the appropriate values need to be configured.

### 4.4.3 Redirection of Unknown Packets and Reverse Path Forwarding of Rules

The unknown packets are redirected to the target SBF in consequence of the inquiry to the SBFC. The SBFC stores the last SBF where the UE camped and requested a service by invoking the SBFC to inspect the packets. Therefore, the target SBF does not necessarily have the required rules (e.g., the rules are removed due to a timeout, or the service requested at this time is not the same at other times). In this case, the target SBF invokes the controller to inspect the packets. To reduce the need to invoke the SBFC, the proposed method shares the rules between SBFs that utilize the redirection. The rules that identify services at the target SBF are cached and forwarded hop-by-hop to every SBF on the redirection path. This sharing method makes redirection useful even if the target SBF does not have the desirable rules. The redirection process helps to increase the cache rule in SBFs and it reduces the need to invoke the controller.

### 4.4.4 Processes for Transferring Service Requests

Figure 4.4 shows the sequence of operations that takes place to transfer service requests to the appropriate virtual network through the SBFs and SBFC. Note that a UE moves from point A to point C. When a new service request packet arrives at SBF#3 (step 1), SBF#3 inspects the packet payload using its installed rules. Because SBF#3 does not identify a service, SBF#3 sends the SBFC a message containing the UE identification (e.g., IP address) to retrieve the redirection destination of the service request packet (step 2). If the SBFC has the SBF that corresponds to the UE, the SBFC replies with its address; otherwise, the SBFC replies to the request to send the whole packet (step 3). If the latter is received, SBF#3 sends the SBFC the whole packet to allow entrusted service identification (step 4'). The SBFC identifies the service and installs the rule, which detects the service, at SBF#3 (step 5'). SBF#3 encapsulates the packet and forwards it to the appropriate virtual network (step 6'). If the address of target SBF (SBF#1) is received, SBF#3 encapsulates the packet and forwards it to SBF#1 (step 4 and 5). SBF#1 de-encapsulates the redirected packet, inspects it with the installed rules, and forwards it to the appropriate virtual network (step 6). If SBF#1 does not identify the service (due to the removal of rules after a timeout or a request for different services at another time), SBF#1 sends the SBFC the whole packet (this sequence is omitted from Figure

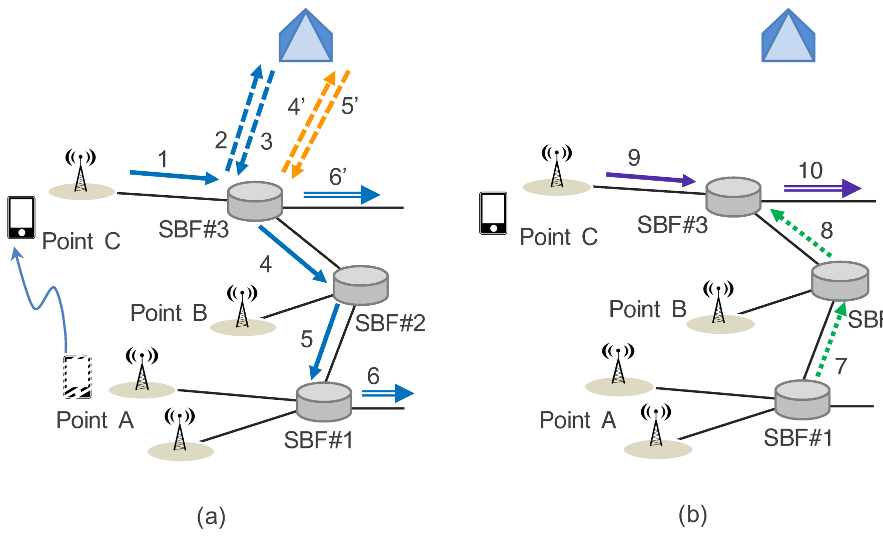


Figure 4.4: Steps during the transfer of service request packets: (a) step 1 - 6, (b) step 7 - 10.

4.4). SBF#1 forwards the rules that detect the service to SBF#2 for reverse-path forwarding (step 7). On the path, SBF#2 caches the rule and forwards it to SBF#3 (step 8). When subsequent packets arrive at SBF#3 (step 9), SBF#3 encapsulates and forwards them directly to the virtual network (step 10). In addition, when UE moves from point A to point B and sends the service request, SBF#2 can identify the service without invoking the controller and it forwards the service request to the appropriate virtual network. This is because SBF#2 caches the corresponding rule at step 8.

### 4.4.5 Handling Rules Change

When the rules, which are cached in SBFs, change at the SBFC, handling depends on whether the rules inconsistency is critical. In the case that the rules consistency is required, the SBFC updates the cache rules in SBFs where the SBFC directly installed the rules in past times. The SBFs forward the updated rules toward the recorded interface corresponding to the old rules. To enable updating the old rules in all the SBFs on the previous redirection paths as well as the most recent redirection path (the multiple paths may be caused by the movement of UE), the SBFs record all the interfaces where the rules are forwarded in past times. In the case of the rules whose consistency is not indispensable, the SBFs remove the cache rules after a timeout time expires.

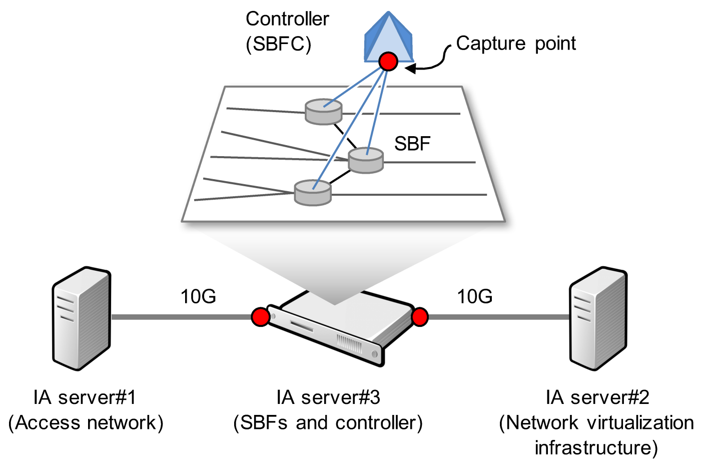

Controller (SBFC)
Capture point
SBF
10G
10G
IA server#1 (Access network)
IA server#3 (SBFs and controller)
IA server#2 (Network virtualization infrastructure)

Figure 4.5: Network configuration used in the experimental evaluations.

Table 4.1: Experimental network components

| | Spec | | |
|---|---|---|---|
| Node | CPU | Memory | OS |
| IA Server#1 (Access network) | Intel Core i7-2600 3.4GHz | 8 GBytes | Ubuntu 12.04.5 |
| IA Server#2 (Network virtualization infrastructure) | | | |
| IA Server#3 (SBFs and controller) | Intel Xeon E5-2630 2.6GHz (24 processor) | 208 GBytes | |

## 4.5 Experimental Evaluation

This section evaluated, using a prototype implementation, the performance of the redirection-based (proposed) method against the centralized-based methods in terms of three key metrics: (1) unknown packet arrival rate at the controller; (2) ratio of the traveling path pattern; and (3) one-way delay. The first metric represents the effectiveness of the proposed method, the second metric breaks down the causes of this effectiveness, and the last metric represents the overhead of the proposed method.

### 4.5.1 Experimental Configration

Figure 4.5 shows our experimental network configuration. The SBFs were deployed in front of the network virtualization infrastructure, where virtual mobile networks are constructed. Three IA servers were connected using 10 GB Ethernet. These servers emulated an access network (UEs and eNBs), a network virtualization infrastructure, and SBFs and a controller (SBFC in the proposed method), respectively. The first server (IA server#1 in Figure 4.5) sent the prepared packets, which emulated packets passing the eNBs (details are given below). The second server (IA server#2 in Figure 4.5) only responded received packets for simplifying. In the last server (IA server#3 in Figure 4.5), the SBFs and controller were constructed using the Linux Container (LXC) [59]. Table 4.1 shows the experimental network components.

The SBFs formed a transit-stub network topology, which was generated using the GT-ITM topology generator [60]. The topology comprised transit domains at the top level with several SBFs in each. Each transit SBF had a specific number of stub domains attached, where each stub had several SBFs. Each SBF possessed a number of eNBs, which was determined by the capacity of the eNB and the interface on the SBF. The overall number of eNBs and the capacity of the SBF's interface were assumed to be constant, and the required number of SBFs changed depending on the capacity of the eNB. An SBFC was connected to each SBF.

The eNBs were assumed to be tightly arranged in 50,000 $\times$ 58,890 m area, and the coverage area of each eNB was assumed to be a hexagonal cell. Each SBF domain comprised $n$ rings with cells of the same size. The value of $n$ was varied according to the number of SBF. Figure 4.6 shows the SBF domains and the connections between the SBFs in the case of 27 SBFs for instance.

The packets passing through the eNB were prepared in advance. The packets included service requests (INVITE messages), which followed a Poisson arrival process with mean arrival rate 150 (packets per second). The service requests were sent ten times by each UE (fifty hundred UEs). The duration of the packets was slightly shorter than one hour. The service requests comprised several types, which depended whether the service was the same as last time based on predetermined probabilities ($P_{same}$). The outer source IP addresses (eNBs' IP addresses) of the packets were assigned according to the location of UEs in the 50,000 $\times$ 58,890 m area.

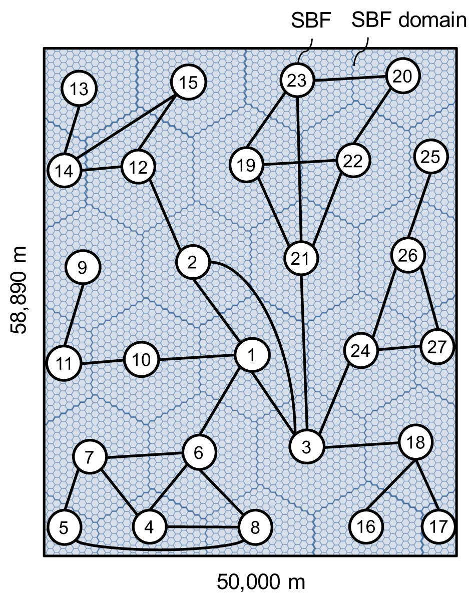


Figure 4.6: SBF domains and connections between the SBFs in the case of 27 SBFs.

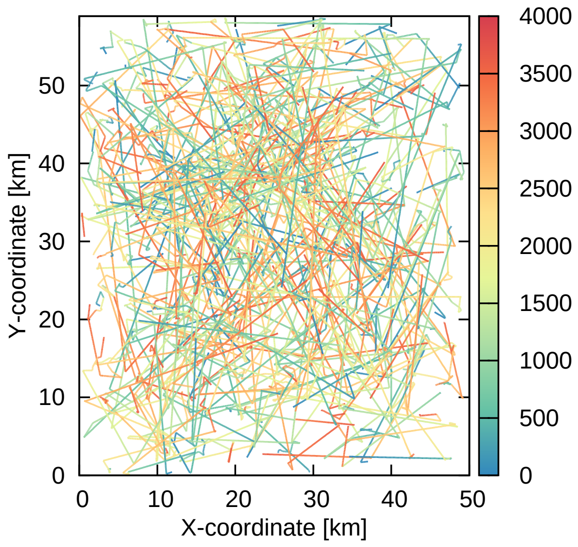


Figure 4.7: Movement traces of two hundred UEs.

The UEs' movement traces for one hour were generated using SMOOTH [61, 62], which can generate synthetic traces that match the statistical features of real human movement. SMOOTH initially places each UE near a cluster. Some clusters (Total number of clusters is given by <clusters> parameter) are distributed evenly about the simulation area (<sim_x>, <sim_y>), and far

Table 4.2: Parameter Settings

| Parameters | Values |
|---|---|
| Number of UEs | 50,000 |
| Number of eNBs | 4,727 |
| Radius of a cell $[m]$ | 500 |
| Area $[m^2]$ | 50,000 $\times$ 58,890 |
| Number of rings in SBF domain $(n)$ | 8 5 3 |
| Number of SBFs $(N_{SBF})$ | 27 64 144 |
| Sum of the service request rate $[/s]$ | 150 |
| Number of INVITE messages per UE | 10 |
| Probability of the same service as last time $(P_{same})$ | 0.5 / 0.8 |
| Number of rules at SBF Controller | 17,500,00 |
| Limit on the number of rules cached at SBF | 10,000 / 5,000 |
| Link delay time $[ms]$ | 1 |

enough apart so that no cluster is in the transmission range of another (<range>). When a UE changes location, the UE moves to either a new location, or a previously visited location, based on predetermined probabilities. These movements follow power-law distribution, where <alpha> is the flight distribution, <f_min> is the minimum value for the flight distribution, and <f_max> is the maximum value for the flight distribution. The time that a UE pauses at a location also follows power-law distribution (<beta>, <p_min>, <p_max>). In the experiments, we choose parameters as follow, clusters = 254, sim_x = 50000, sim_y = 58890, range = 100, alpha = 1.4, f_min = 100, f_max = 40000, beta = 1.5, p_min = 10, and p_max = 1800. We generated fifty hundred UEs' traces in two hundred fifty batches due to the inability to generate at one time. The movement traces of two hundred UEs were shown in Figure 4.7, where the colors indicate the time (seconds).

The prepared packets were transmitted from UE emulator using Tcpreplay [63]. Packets through the system were captured at the controller and the interface of access network-side and network virtualization infrastructure-side in the IA server emulating SBFs. Each SBF logs the information (the type of messages received and sent at each interface) to identify the routing of the service requests later. The parameter settings used in our experiments are listed in Table 4.2.

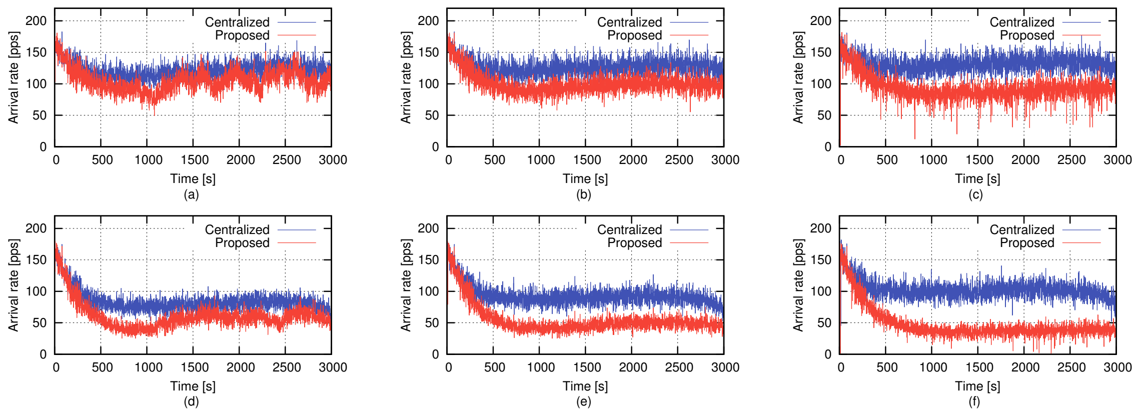


Figure 4.8: Number of packets arriving at the controller per unit time (Size of cache table: 10,000). (a) $N_{SBF} = 27, P_{same} = 0.5$. (b) $N_{SBF} = 64, P_{same} = 0.5$. (c) $N_{SBF} = 144, P_{same} = 0.5$. (d) $N_{SBF} = 27, P_{same} = 0.8$. (e) $N_{SBF} = 64, P_{same} = 0.8$. (f) $N_{SBF} = 144, P_{same} = 0.8$.

### 4.5.2 Experimental Results

Figure 4.8 and 4.9 show the number of packets that arrived at the controller for inspection per unit time ("packet arrival rate") in the cases where the size of cache table is 10,000 and 5,000, respectively. In these figures, (a)-(f) refer to experimental conditions with $N_{SBF}$ = 27, 64, 144 and $P_{same}$ = 0.5, 0.8. $P_{same}$ is equals to the probability that the destination SBF has the desired rules. At any experimental conditions, the proposed method (red lines in the figures) reduced the concentration of requests on the controller as compared with the centralized-based method (blue lines in the figures). This is because services were identified without invoking the controller. This effect differed depending on the size of cache table, the number of SBFs, and the values of $P_{same}$. A large size of cache table (Figure 4.8) made the reduction of the packet arrival rate larger than a small size of cache table (Figure 4.9). In addition, the increase of the number of SBFs increased the reduction of the packet arrival rate (in the centralized-based method, the opposite trend was observed). Under our assumption, the proposed method reduced the packet arrival rate at the controller by approximately 14-63 %.

Figure 4.10 and 4.11 show the ratio of the transmission path pattern for the first service requests. These results break down the causes of the reduction, as shown in Figure 4.8 and 4.9, respectively. There are three transmission path pattern: the first (orange parts in the Figure 4.10 and 4.11) is

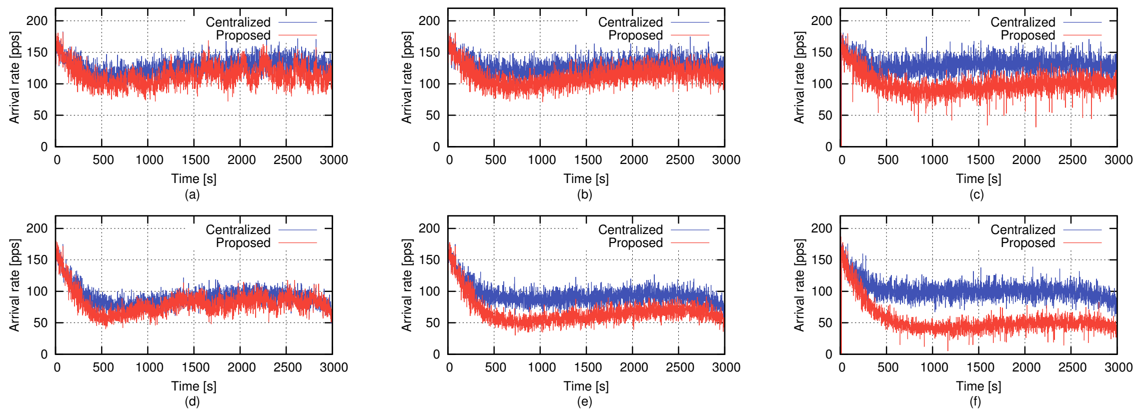


Figure 4.9: Number of packets arriving at the controller per unit time (Size of cache table: 5,000). (a) $N_{SBF} = 27, P_{same} = 0.5$. (b) $N_{SBF} = 64, P_{same} = 0.5$. (c) $N_{SBF} = 144, P_{same} = 0.5$. (d) $N_{SBF} = 27, P_{same} = 0.8$. (e) $N_{SBF} = 64, P_{same} = 0.8$. (f) $N_{SBF} = 144, P_{same} = 0.8$.

through only one SBF (the SBF received the requests from eNBs identified services without redirecting the requests and invoking SBFC); The second (green parts in the figures) is through some SBFs (the requests were redirected to the target SBFs and inspected at the target SBFs); and the last (blue parts in the figures) is invoking SBFC for inspection. The proposed method decreased the ratio of invoking SBFC (the blue parts in Figure 4.10 and 4.11) as the number of SBFs, while the centralized-based method increased (but only slightly in this experimental environment). In the cases where the size of cache table and the number of SBFs were small (with the exception of $N_{SBF} = 144$ in Figure 4.10 (c)), the proposed method made the ratio of packets traveling through only on SBF (the orange parts) less than the centralized-based method. That is, these experimental conditions did not benefit from the caching mechanism in the proposed method. This was caused by the eviction of cached rules due to the limitation of the size of cache table. To benefit not only from the redirection mechanism but also form the caching mechanism, it is necessary to reduce the eviction of cached rules. An increase in the size of cache table may grow the orange parts, resulting in reducing the ratio of invoking SBFC.

We also evaluated the overheads of the redirection scheme. Figure 4.12 and 4.13 show the one-way delay time of first request packets in the case where the size of cache table was 10,000 and 5,000, respectively. These results indicate the overheads of piecemeal inquiries to obtain the redirection destination and the overheads of redirecting packets. The one-way delay in the proposed

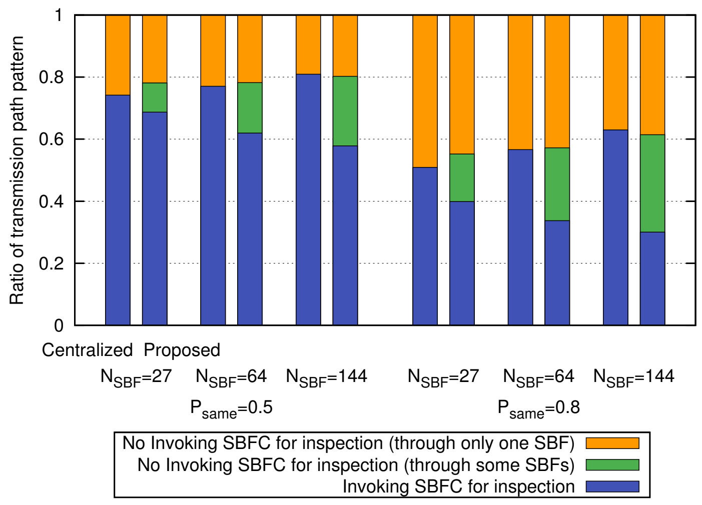


Figure 4.10: Ratio of the transmission path pattern of the first service request (Size of cache table: 10,000).

method was larger than that in the centralized-based method due to the overhead. The difference value between the proposed and centralized-based method increased when assuming that the number of SBFs was not large (the load on the SBFC was low), whereas the difference value was prevented from increasing by assuming a large number of SBFs. This is because the proposed method avoided increasing the load on the SBFC. Fig. 4.14 shows the hop counts of redirection path between SBFs where the size of the cache table is 10,000 and $P_{same}$ is 0.5 (Note that under other condition, the behavior was similar). This figure breaks down the communication costs of redirecting packets. The hop counts increased as the number of SBFs increased. This increase caused that the one-way delay time did not shorten, even if the proposed method can reduce the ratio of invoking SBFC as the number of SBFs (as described above). At any hand, these results demonstrate that redirecting packets has an insignificant impact on packet transfers to virtual networks.

### 4.5.3 Discussion

The proposed method can reduce the need to invoke the controller to inspect packets by redirecting unknown packets to the other SBFs. However, if the target SBF does not have the required rules, the redirected packet is sent to the controller. To reduce this inefficiency, it would be more effective

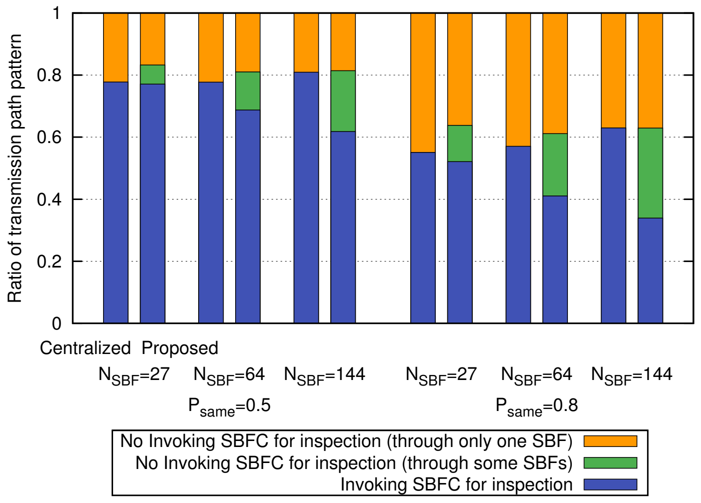


Figure 4.11: Ratio of the transmission path pattern of the first service request (Size of cache table: 5,000).

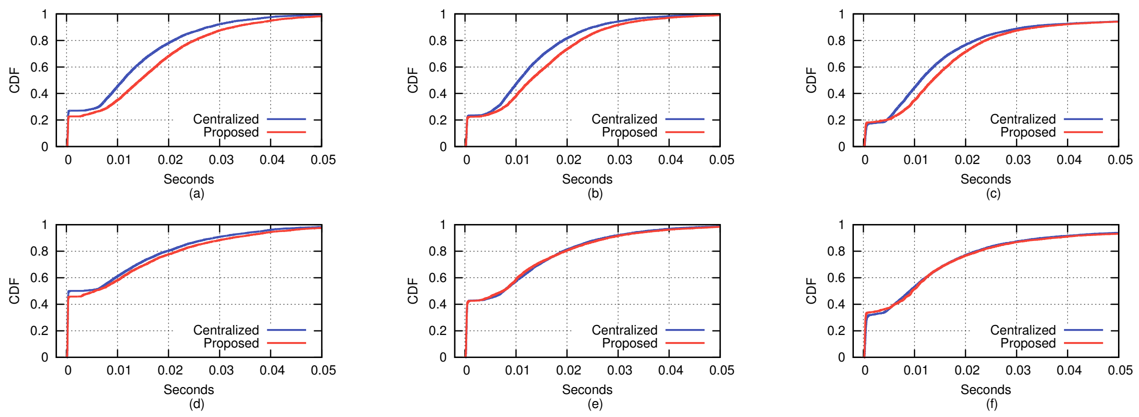


Figure 4.12: One-way delay time (Size of cache table: 10,000). (a) $N_{SBF} = 27, P_{same} = 0.5$. (b) $N_{SBF} = 64, P_{same} = 0.5$. (c) $N_{SBF} = 144, P_{same} = 0.5$. (d) $N_{SBF} = 27, P_{same} = 0.8$. (e) $N_{SBF} = 64, P_{same} = 0.8$. (f) $N_{SBF} = 144, P_{same} = 0.8$.

if some of the intermediate SBFs on the redirection path inspect the packets (in the present study, all of the intermediate SBFs did not inspect the redirected packets to prevent the one-way delay from increasing). Thus, it will be necessary to select appropriate SBFs and to enhance the redirection method by considering the tradeoff between the reduced invocations of the controller and the delay overheads.

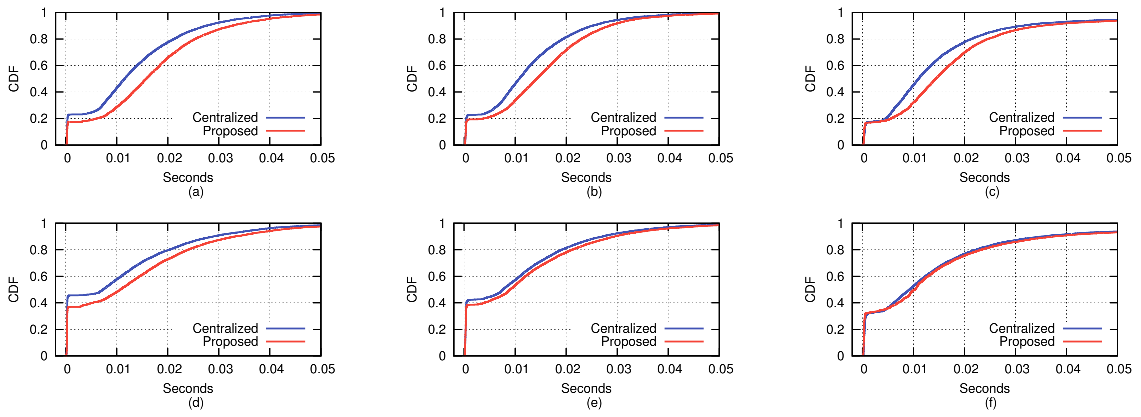


Figure 4.13: One-way delay time (Size of cache table: 5,000). (a) $N_{SBF} = 27, P_{same} = 0.5$. (b) $N_{SBF} = 64, P_{same} = 0.5$. (c) $N_{SBF} = 144, P_{same} = 0.5$. (d) $N_{SBF} = 27, P_{same} = 0.8$. (e) $N_{SBF} = 64, P_{same} = 0.8$. (f) $N_{SBF} = 144, P_{same} = 0.8$.

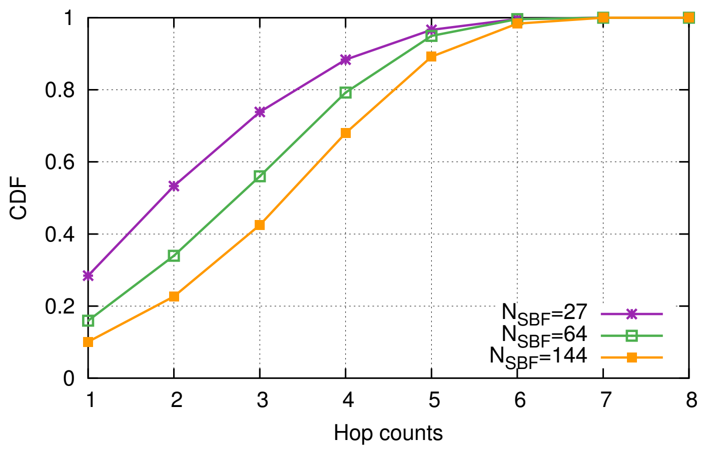


Figure 4.14: Hop counts of redirection path between SBFs (Size of cache table: 10,000, $P_{same} = 0.5$).

In the evaluations, a small number of SBFs had the frequent eviction of rules due to the limitation of the number of cached rules. Figure 4.15 shows the number of incidence of caching rules in the case where the number of SBFs is 144 and $P_{same}$ is 0.5. The difference is mainly caused by the topology of SBFs. The SBFs, which have a lot of incidence of rule cache, are nodes constituting transit domains in the transit-stub network topology. To further reduce the need to invoke the controller for packet inspection, it will be necessary to consider the topologies of SBFs.

To evaluate the proposed method at different topologies of SBFs, we can use a multi-cache approximation (MCA) algorithm [64] for general-topology cache networks. The MCA approximates the behavior of multi-cache networks by leveraging an existing approximation algorithm for isolated caches (a single-cache approximation (SCA) algorithm [65]). Let $r_{i,v}$ be the combined incoming rate of requests for rule $i$ at SBF $v$, and let $m_{i,v}$ be the miss rate of rule $i$ at SBF $v$. Then the rate of requests at each SBF can be expressed as

$$r_{i,v} = \lambda_{i,v} + \sum_{h:i \in R(h,v)} m_{i,h} \tag{4.1}$$

Note that while $\lambda_{i,v}$ is a Poisson stream of exogenous requests for rule $i$. $R(h, v)$ is the set of all rules $i$ for which $v$ is on the next hop from $h$ along the shortest path to destination SBFs. The MCA algorithm solves Eq. (4.1) and the following equations iteratively and determines the miss rate for each rule at each SBF.

$$\begin{aligned} p_{i,v} &= \frac{r_{i,v}}{\sum_{j=1}^{N} r_{j,v}} && (4.2) \\ \overrightarrow{q_v} &= contents(\overrightarrow{p_v}, |v|) && (4.3) \\ m_{i,v} &= r_{i,v} \cdot (1 - q_{i,v}) && (4.4) \end{aligned}$$

where $\overrightarrow{p_v} = (p_{1,v}, p_{2,v}, \cdots, p_{N,v})$ is the steady-state incoming request distribution for rules at a certain cache $v$. The SCA algorithm can be thought of as a function $contents(\overrightarrow{p_v}, |v|) = \overrightarrow{q_v}$, where $\overrightarrow{q_v} = (q_{1,v}, q_{2,v}, \cdots, q_{N,v})$ is the vector consisting of the probability for each rule to be present in the cache at a random point in time. In this paper, we confirmed the feasibility of the proposed method through the experimental evaluations. To evaluate the impact on large network topologies, we will conduct a comprehensive computational analyses using the above model in future work.

In this evaluation, we set no timeout for the buffer with the assumption of off-peak hours. To configure this value appropriately for practical usage, it will be necessary to evaluate the impact of the values on the arrival rate of unknown packet at the SBFC.

We added three functionalities to the SBF: step-by-step invocation the controller, redirection of packets to the other SBF, and hop-by-hop caching between SBFs on the reverse-redirection path. In

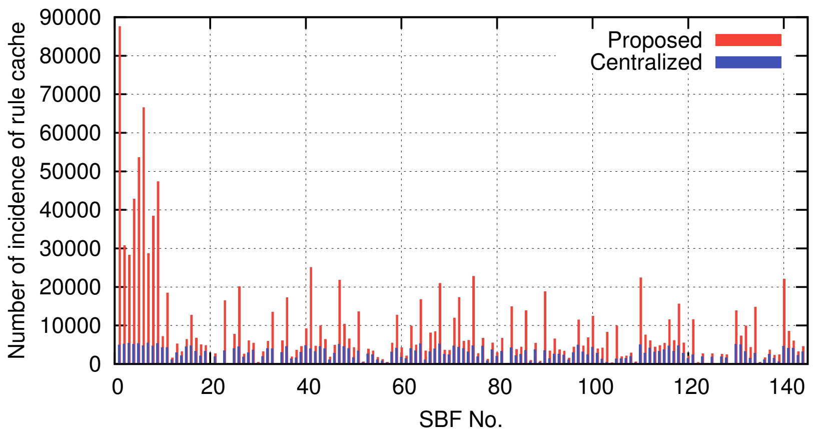


Figure 4.15: Number of incidence of caching rules in the case where $N_{SBF}$ = 144 and $P_{same}$ = 0.5.

this prototype system, we implemented the function from scratch. Therefore, it will be necessary to confirm that these functionalities can be implemented easily on current flow-based switches.

## 4.6 Conclusion

Centralized management that reactively installs rules for flow control into SBFs (gateway functions in mobile network virtualization infrastructure) by transmitting the unknown packet to the controller is a promising solution for sharing rules between SBFs. To improve the scalability of sharing rules by reducing the concentration of unknown packets on the controller, we proposed a method that redirects unknown packets to another SBF instead of the controller and shares the matched rules between SBFs on the redirection path. This can reduce the number of unknown packets transmitted to the controller for inspection. Our simulations showed that the proposed method reduced the number of unknown packets that arrived at the controller per unit time by approximately 14-63%. This can result in preventing the number of replica controllers from increasing. Although the proposed method increased the one-way delay due to the overhead of redirecting packets, the proposed method brought benefits that more than compensate for the overhead when assuming a large number of SBFs. In future research, we will study the enhanced inspection mechanism for the redirected requests to reduce the inefficiency of the redirection mechanism and evaluate the topologies between SBFs to improve the effectiveness of the caching mechanism.

# Chapter 5

# Reducing State Information by Time-Division Sharing IMSI for Cellular IoT Devices

## 5.1 Introduction

Mobile network technologies for the Internet of Things (IoT) are expected to provide wide area coverage and low power communications at a low cost (for devices and usage) [66]. IoT services, which must meet these requirements, include metering of gas and water utilities, and tracking logistics containers, cars, high-value bicycles, and pets. The IoT devices used in these services have no power source, are dispersed geographically, and some of them move around frequently.

Widely deployed cellular networks have made possible low power consumption for IoT devices and cost reductions for inexpensive IoT devices and services. To reduce the power consumption of devices, a sleep mode called power saving mode (PSM) [67], where the device does not respond to calls from the network, has been introduced to devices. The PSM enables devices to operate for more than ten years on two long life AA batteries [68]. To reduce the cost of devices, simplified radio frequency (RF) hardware has been developed [69]; one that uses a single antenna and half-duplex communication to ensure the minimum bandwidth and data rate requirements for IoT services. Several methods for reducing the costs associated with cellular network usage have been studied [70], including the effective utilization of hardware resources using virtualization techniques and optimization of communication controls. However, these methods are still not sufficient for the realization of inexpensive IoT services.

The evolved packet core (EPC), which is a representative mobile core system that constitutes the cellular network at present, controls data paths (called "bearers") for each accommodated user terminal such as smartphones and IoT devices. The EPC continuously maintains the connection status (also referred to as state information) of bearers for each user terminal even if it rarely sends data. This requires a large number of computational resources to store and process a large amount of state information when the EPC needs to support a large number of IoT devices. Additionally, a rapid increase in demand for computational resources creates difficulties in reducing the EPC's capital expenditure/operational expense (CAPEX/OPEX). The best way to address this problem would be reducing the amount of state information as much as possible.

The method in [71] reduces the number of IoT devices that establish a bearer in the EPC by having them communicate with the EPC via a gateway. However, in the case of providing IoT services over a widely distributed area, this method may impose a burden on the device side. Appropriate devices need to be selected as gateways in order to efficiently reduce state information. In addition, this method may not reduce state information to any significant degree unless mobile devices and device density are taken into account.

This chapter proposes a communication control method that aims to reduce the amount of state information retained in the EPC without adding additional sophistication to the connections among IoT devices. The proposed method is an approach based on the enhancement of the EPC (in contrast to the previous method [71] that is based on a user-side approach). The proposed method assigns the same international mobile subscriber identity (IMSI) to multiple IoT devices, which upload data with the same cycle, and manages devices uploading at different times. From the EPC's perspective, it sees a device that alternates between movement and communication. This is because bearers are established and released with a single IMSI by multiple IoT devices in different locations (this is not a handover, which is the process of maintaining the communication as devices move). To prevent communication timings from overlapping, time slots at periodic intervals are introduced in

Table 5.1: Comparison between Related Works and Proposed Method

| | Features | | | |
|---|---|---|---|---|
| Methods | Solution type | Amount of reduction of state information | Lead time to market | Prolong battery life |
| Enhancement of standardized functions [72, 73] | EPC-side | Low (Only U-plane) | Short | Favorable |
| C/U-planes separation [74, 75] | EPC-side | Low (Only U-plane) | Long | Favorable |
| New architecture [76] | EPC-side | Medium $\sim$ Large (C/U-plane) | Long | Favorable |
| Proposed | EPC-side | Significant (C/U-plane) | Short | Favorable |
| Gateway-based [71] | Device-side | Large (C/U-plane) | Short | - |

the EPC. The proposed method controls, by using time slots, the timings of the registration process of IoT devices powered up (specifically powered up at random), thereby controlling the timing of uploading data after the registration process. This chapter shows that the proposal can reduce the required state information to less than 0.5% compared to the current EPC method. This conclusion is based on the observation that two hundred or more IoT devices sharing a single IMSI can upload data in turns. Preliminary results of this work have been published in [16]. This chapter investigates the impact of important factors provisionally configured in our previous work, carefully configures them, and, as a result, shows the improved performance of the proposed method.

The rest of this chapter is organized as follows. Section 5.2 describes related work aimed at reducing state information in the EPC. Section 5.3 introduces the current procedure that IoT devices follow for uploading data through the EPC. Section 5.4 explains our proposed method for the communication control of IoT devices that share a single IMSI. Section 5.5 presents performance evaluations of the proposed method made by using simulation experiments, and discusses related issues. Section 5.6 presents our conclusions.

## 5.2 Related Work

Methods that can reduce state information in the EPC are categorized into two types: first, methods that modify the EPC; and second, methods that utilize gateway devices in access networks. The former is referred to as EPC-side methods, and the latter is referred to as device-side methods.

The EPC-side methods include the enhancement of standardized functions [72, 73], the use of implementation by separating the C-plane from the U-plane [74,75], and a new architecture [76]. In the enhancement of standardized functions, when devices do not transmit data, the communication tunnels that constitute bearers in the EPC are released (in general, the communication tunnels are maintained based on the number of devices irrespective of communication status of the devices). The methods can reduce the number of the communication tunnels, which are constantly established in the EPC. In C/U-planes separation, data paths are controlled without using communication tunnels. The enhancement of standardized functions [72, 73] and C/U-planes separation [74, 75] have limited effectiveness in reducing state information. That is, although state information maintained in U-plane can be reduced, that maintained in C-plane cannot be reduced. In contrast, the new architecture [76] can reduce state information in C-plane. In this architecture, the mobile core network comprises general switches and middle boxes, and the architecture has adopted a routing control model that is based on a Software Defined Network (SDN) that controls data paths without using communication tunnels. In addition, this architecture reduces the amount of information (rules for routing control) maintained in a whole mobile core network by aggregating rules for routing control. However, this architecture cannot reduce information such as users' profiles and locations of devices. Considering market trends, cellular networks need to support IoT services before migrating from the current EPC to the new architectures based on an SDN (e.g., 5G networks). It takes some time to migrate from the current EPC to a new architecture. Consequently, it is favorable to use methods based on the current standardized architecture.

In the device-side method [71], devices communicate with the EPC through gateway devices appropriately chosen among the devices by using device-to-device (D2D) communication. This method can reduce the number of devices that establish bearers in the EPC, thereby reducing the

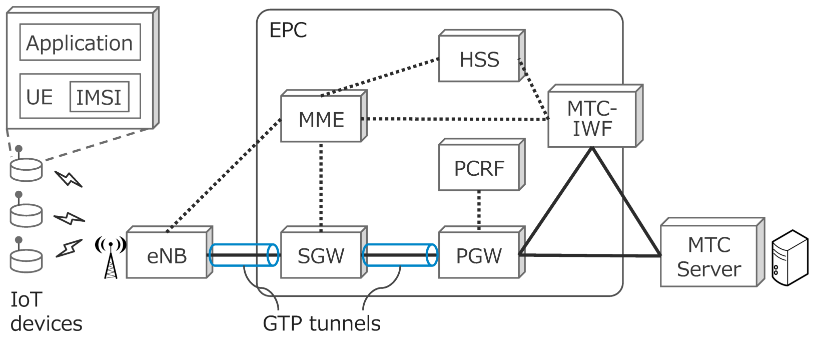


Figure 5.1: Standard cellular network architecture.

amount of state information in the EPC. However, to provide IoT services over a wide area, it is necessary that multiple gateway devices be optimally selected from devices to improve the reduction of state information. This is because there are limitations to the number of devices that can be managed by a single gateway device and the communication distance between the gateway device and other devices. Although the combined use of multi-hop communication technologies may improve the flexibility of selection of gateway devices, some calculation processes, such as routing control and congestion control in the device, are required, thereby reducing the battery life of the device. In addition, moving devices cannot always establish network connectivity via the gateway devices, and a region of low device density requires a certain number of gateways due to the limitation of communication distance between devices. These prevent state information from being effectively reduced. In contrast, the proposed method does not require any assistance from devices due to the independence of the moving devices and device density. Even so, the proposed method can reduce the amount of state information to the same degree or better than the device-side method (details in Section 5.5.5). Table 5.1 shows summary of comparison between the proposed method and related works.

Table 5.2: Responsibilities of Function Blocks

| Block/Module | Responsibilities |
|---|---|
| UE | The UE has an ability to access the EPC. |
| Application | The application records sensor's data and transmits the data. |
| IMSI | The IMSI is used as a unique identification number in the EPC. |
| eNB | The eNB is a base station for a LTE access network. |
| MME | The MME is responsible for mobility management and user authentication via a HSS. |
| SGW | The SGW serves as a mobility anchor that enables devices to have seamless communication when the device moves from one eNB to another. |
| PGW | The PGW performs traffic monitoring, billing, and access control, and acts as a gateway to external networks. |
| PCRF | The PCRF provides the PGW with QoS authorization. |
| HSS | The HSS has subscriber profiles and the corresponding information between an EID and IMSI. |
| EID | The EID is newly defined for identifying and calling IoT devices. |
| MTC-IWF | The MTC-IWF serves as an intermediary between the EPC and an MTC server. |
| MTC Server | The MTC server identifies IoT devices by EIDs. |

## 5.3 IoT Communication using Cellular Network

### 5.3.1 EPC and Communication Procedures for IoT Devices

Figure 5.1 shows the evolved packet core (EPC) architecture with a newly standardized function for IoT devices (which we refer to as "devices", unless specifically distinguishing between "IoT devices" and "devices"). The device is equipped with user equipment (UE) that has the ability to access the EPC, and an application for recording sensor's data and transmitting the data. The UE has an IMSI, which is used as a unique identification number in the EPC. For machine type communication (MTC) such as IoT, an MTC inter-working function (MTC-IWF) is added [67]. This function serves as an intermediary between the EPC and an MTC server where the applications for IoT services run. The MTC server is liberated from the responsibility of managing the IMSIs and IP addresses of devices. The MTC server uses external identifiers (EIDs), which are newly defined, for identifying and calling the devices (called "device triggering").

The packets that are transmitted and received between devices and MTC server are routed using two GPRS tunneling protocol (GTP) tunnels, which constitute a bearer. There are base stations (called "eNBs") for a long term evolution (LTE) access network, a serving gateway (SGW), and a packet data network gateway (PGW) on the data path, wherein the primary function is packet routing/forwarding and policy enforcement. The SGW serves as a mobility anchor that enables devices to have seamless communication when the device moves from one eNB to another. The PGW performs traffic monitoring, billing, and access control, and acts as a gateway to external networks. The GTP tunnels are created, removed, and updated by the eNB, SGW, and PGW in a coordinated manner using a mobility management entity (MME) and a policy and charging rules function (PCRF). The MME is a C-plane function, which is responsible for mobility management and user authentication via a home subscriber server (HSS), which has subscriber profiles and the corresponding information between the EID and IMSI. The PCRF provides the PGW with QoS authorization based on user's profile and/or the media information (e.g., the TCP/IP 5-tuple and media codecs) received from external networks. Table 5.2 summarizes function of blocks in Figure 5.1.

When the power is turned on, the device registers itself with the EPC—also known as the "Attach" procedure. In this procedure, the EPC registers the location of the device, assigns it an IP address, and establishes a bearer. After the completion of the Attach procedure, state information (IP address, location, bearer IDs, external network information, QoS, etc.), which is indexed by the IMSI, is maintained in the functions (eNB, MME, SGW, and PGW) that constitute the EPC. When there is no data on the data path after the Attach procedure or data transmission, only the GPRS tunnel between the eNB and SGW is released, whereas the tunnel between the SGW and PGW is allowed to remain. The device then enters idle mode. IoT devices often use PSM, where they do not respond to calls from the network (details in the next subsection), to save battery. After returning from PSM, a device executes a tracking area update (TAU) to update its location. If the device receives paging messages from the EPC shortly after performing the TAU, the device re-establishes

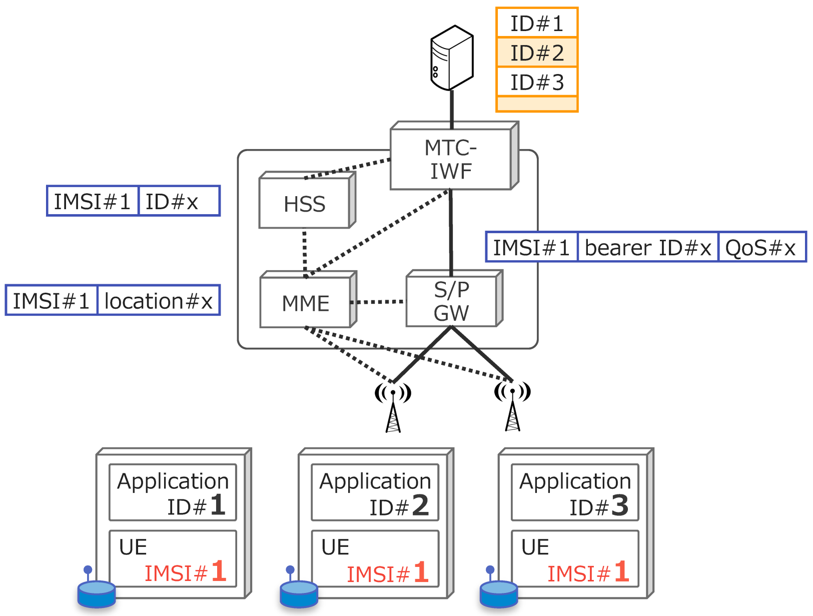


Figure 5.2: Overview of proposed method.

the GPRS tunnel between the eNB and SGW and the bearer returns to being available.

### 5.3.2 Power Saving Mode

In PSM, the devices switch off RF modules, thereby entering sleep mode, during which the devices do not respond to paging messages. The device activates PSM using two timers, which are obtained in the Attach or TAU procedure. The first timer (called T3324) is the time during which the device remains in idle mode following the Attach or TAU procedure. The second timer (called T3412) is the periodic TAU timer. Subsequent to the EPC releasing radio resource and eNB-to-SGW tunnel, T3324 and T3412 are initiated. Until T3324 expires, the device can remain in idle mode and respond to the paging messages or any other signaling messages from the EPC. Once T3324 expires, the device will then enter the PSM for the duration of T3412. The device can cancel PSM by conducting a TAU or by sending a service request that is triggered by the application layer to re-establish its bearer.

## 5.4 Proposed Methods

To aggregate cellular communication lines (bearers for IoT devices) in the EPC, this chapter proposes a method that allows a single communication line to be shared among the IoT devices using time division. Note that the IoT devices communicate with a MTC server at a constant frequency. Figure 5.2 gives an overview of the proposed method. Its key feature is that multiple devices have the same IMSI (lower layer ID) and transmit data to the MTC server in turns (at different times). The application ID (upper layer ID) is responsible for identifying IoT devices. From the perspective of the EPC, a single device alternatives between movement and communication (uploading data). Therefore, the proposed method can reduce the amount of state information managed in the EPC.

Note that it is assumed that devices sharing a single IMSI are used in the same IoT service. In other words, the devices accommodated in the same IoT service have the same communication policy (upload cycle, traffic, etc.). After completing the Attach procedure following power-on, devices upload their data to the MTC server periodically. Note that time taken to upload their data does not change largely. Data retransmissions do not occur frequently due to the assumption of a LTE access network in which high reliability is ensured.

This section presents a control method designed to prevent communication timings from overlapping (Section 5.4.1) and a method for providing device triggering in a limited case where the devices support the PSM (Section 5.4.2). Section 5.4.3 describes additional functions and procedures for realizing the proposed methods on standard cellular network architecture.

### 5.4.1 Method for Controlling Communication Timings

Wireless access networks and the EPC are subject to load fluctuations. The load fluctuations vary the time it takes devices to communicate with the EPC and MTC server (the devices establish a bearer, upload data, and release the bearer). Therefore, to prevent periods of communication from overlapping, a certain time interval between the periods is required. To ensure this, time slots at periodic intervals are introduced in specific IMSIs stored by HSS in the EPC. Note that a time slot is defined as the time period in which a specific IMSI is active. The EPC is enabled to accept Attach requests caused by the random switching-on of devices with a specific IMSI while only in time

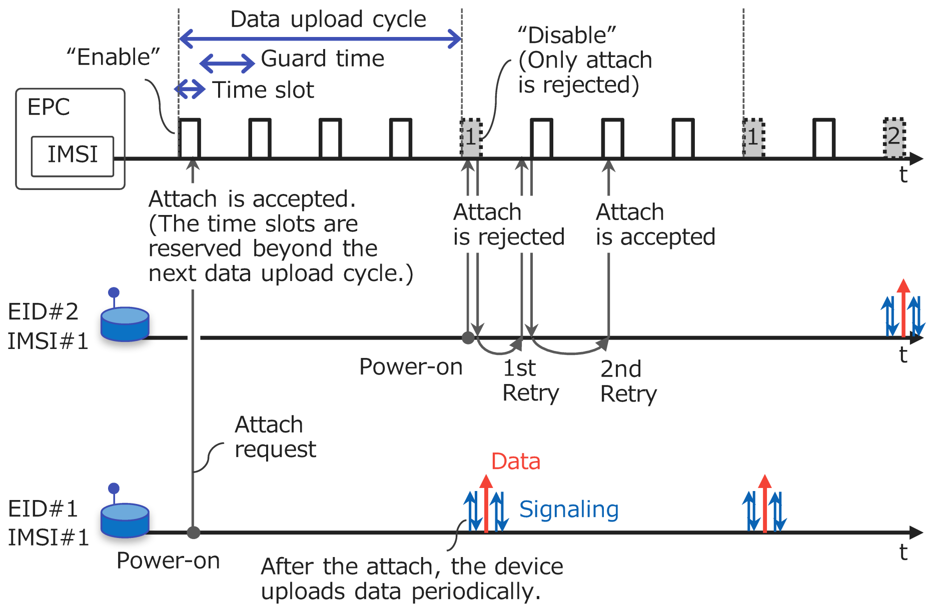


Figure 5.3: Overview of redirection-based rule sharing.

slots, thereby controlling the timings of uploading data following the Attach procedure.

Figure 5.3 provides an overview of the control method. Note that two devices have the same IMSI (IMSI#1), and EID#1 and EID#2 are assigned to them, respectively (the way of assigning EIDs is described in Section 5.4.3). The data upload cycle is the period during which the devices upload data. The number of time slots in this cycle is equal to the maximum number of devices sharing a single IMSI. A guard time is inserted between the time slots. The guard time blocks the Attach procedure triggered by a request that arrives late in a time slot from overlapping the next time slot. The duration of each data upload cycle, time slot, and guard time, which depends on IoT services, can be managed by the EPC itself (it is up to the implementation method to choose which nodes manage the parameters) or the MTC server (described in Section 5.4.3). The time slots have two statuses, "Enable" (solid line boxes in Figure 5.3)—where the Attach request is accepted, and "Disable" (dotted line in boxes in Figure 5.3)—where the Attach request is rejected. After accepting the Attach request from the device (EID#1), the HSS changes the status of the time slot to "Disable" from "Enable." The "Disable" state continues beyond the next data upload cycle. This reservation process for a time slot is inspired by packet reservation multiple access (PRMA) [77].

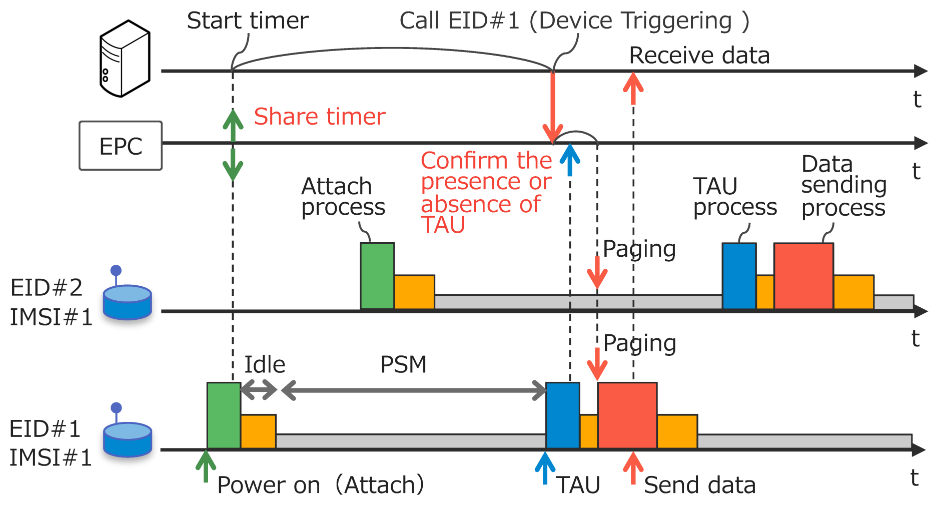


Figure 5.4: Overview of device triggering using PSM.

After the attach procedure is completed, the device (EID#1) uploads data at intervals of the data upload cycle. In the example of Figure 5.3, a device (EID#2) powered up and sends an Attach request at the same timing that the device (EID#1) uploads data. In this case, the EPC rejects the Attach request due to the “Disable” state, while the EPC accepts a bearer establishment request for uploading data. When the EPC rejects the Attach request, the EPC replies with Attach reject messages that contain a back-off timer to the device (EID#2).

The back-off timer value can be determined using various algorithms. The value can be random for the sake of simplicity, or it can be calculated to determine the time elapsed before the next “Enable” (vacant) time slot (although the calculation makes management cumbersome and complicated). After the back-off timer expires, the device will retry the Attach procedure. Attach reject messages containing the back-off timer are also answered when Attach request messages arrive at the EPC while in the guard times. The devices repeat this retransmission process until a Attach request is accepted. In this way, the EPC controls the timing of the Attach by fitting it into the “Enable” time slot, thereby preventing communication timings following the Attach from overlapping.

### 5.4.2 Device Triggering using PSM

This subsection presents a method for providing device triggering. To change the behavior of the application running on a device or improve the reliability of the application (recover the application suspended due to some sort of failure), the MTC-server-initiated requests need to be delivered to the devices.

In general, assigning the same IMSI to multiple devices makes it difficult to enable device triggering from the MTC server. This is because the devices, which are assigned the same IMSI, respond to paging messages when they reside in the same paging area (the paging message is broadcast within the paging area composed of several eNBs). Provided that all the devices support PSM and their communication timings (non-PSM time) do not overlap, device triggering can invoke a specific device. This is because devices in PSM do not respond to paging messages.

To synchronize device triggering with the timing of the non-PSM, the EPC need to notify the timer (T3324 in Section 5.3.2) to the MTC server. In the standardized procedure, this timer is notified only to a device during the Attach procedure. In addition, to eliminate an inconvenience caused by the time lag between when the device and MTC server send messages after the timer expires, a function of buffering the messages need to be introduced to the EPC. Figure 5.4 shows an overview of device triggering using the PSM. Note that two devices have the same IMSI (IMSI#1), and EID#1 and EID#2 are assigned to them, respectively (the way of assigning EIDs is described in Section 5.4.3). After the timer in the MTC server expires, the MTC server sends a data transmission request to the device by specifying its EID (EID#1). The EPC receives this request and subsequently identifies IMSI#1 corresponding to EID#1 and determines if a TAU corresponding to IMSI#1 has been performed. If no TAU exists no later than a short time (for fewer than one seconds), the request is buffered. Otherwise, paging messages are sent to the devices for the request to be transmitted. Although the paging messages reach the devices sharing IMSI#1, only the device (EID#1) that returns from PSM responds to the paging messages, thereby establishing the data path in the EPC. Subsequently, the device receives the data transmission request and sends data to the MTC server.

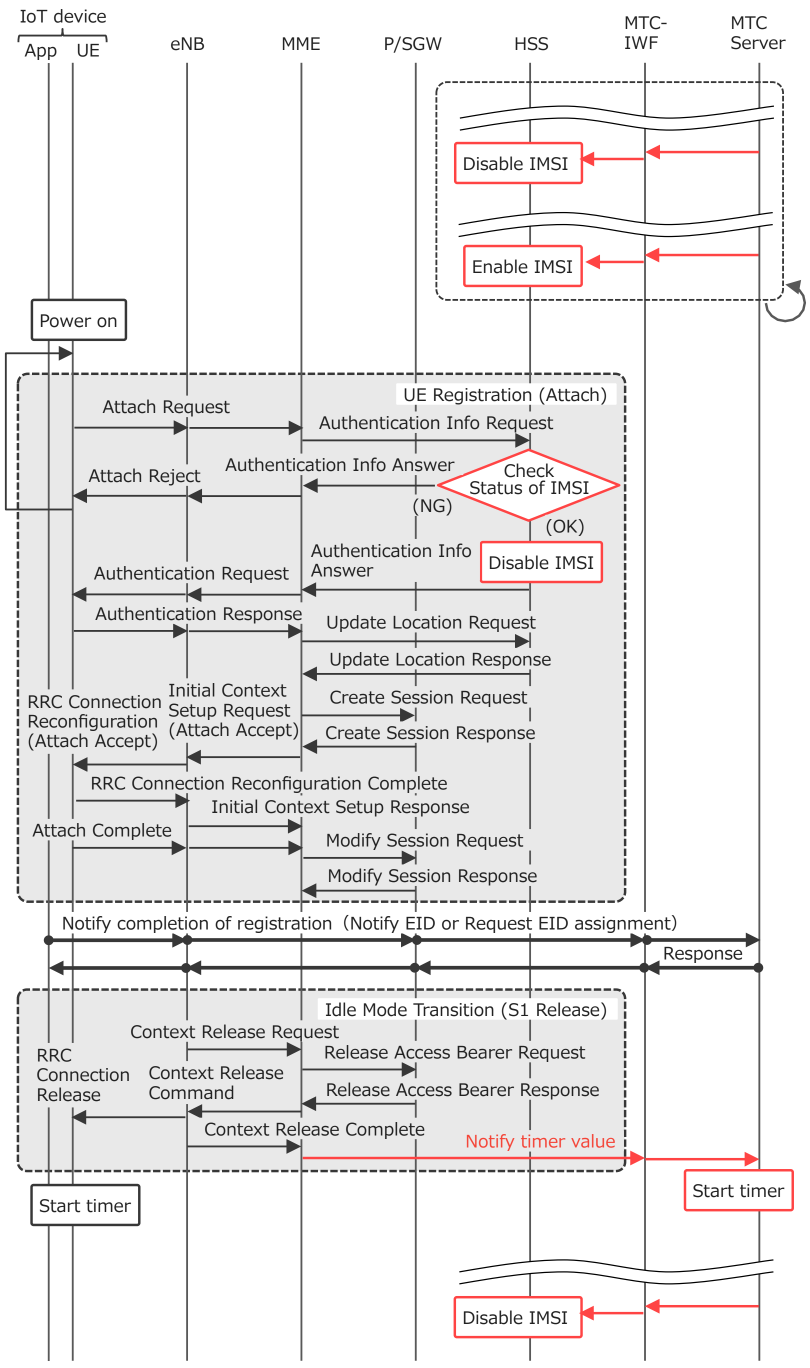


Figure 5.5: Call flow in Attach procedure.

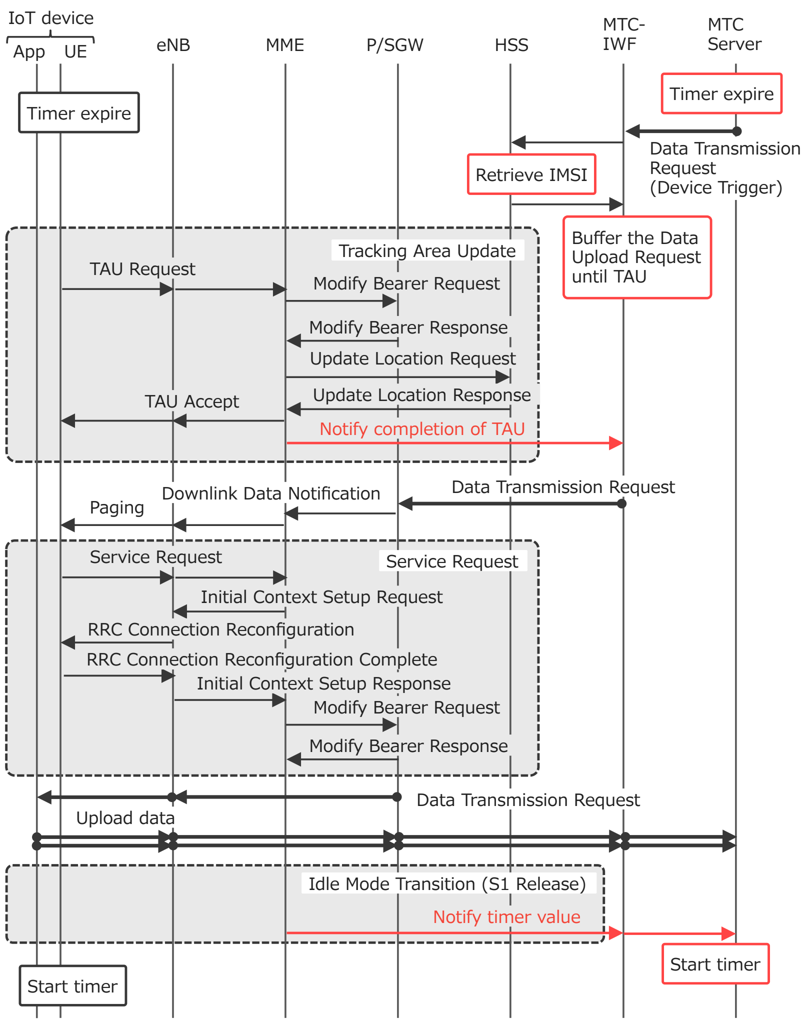


Figure 5.6: Call flow in data upload procedure using device triggering.

### 5.4.3 Applying the Proposed Methods to Standard Architecture

This subsection presents the procedure for applying the proposed method to the standard architecture shown in Section 5.3.1. In the EPC, HSS, MME, and MTC-IWF need to be enhanced. The MTC server also needs to include new operations. To simplify the management of time slots in the EPC, the MTC server is mainly responsible for the management of time slots. The MTC server activates each EID in turn for a particular period and changes the status of the IMSI, stored in the HSS, in synchronization with this activation. Once an EID is assigned to a device, the MTC server stops changing the status of the IMSI when the EID's turn comes. This eliminates the need for the HSS to manage time slots and each EID, thereby preventing the amount of information in the HSS from increasing. Only the status (Enable/Disable) of the IMSI and the back-off timer value are added in the HSS.

Note that the back-off timer value is determined using a random number algorithm for the sake of simplicity. There are several well-known back-off algorithms using random number algorithm. Uniform back-off algorithm (hereinafter referred to as the "Uniform") selects back-off timers in a constant range. Binary exponential and Adaptive back-off algorithms (hereinafter referred to as the "Binary exponential" and "Adaptive," respectively) select back-off timers in a variable range. In Binary exponential, the range is determined at devices and is incremented in a binary exponential manner according the the number of retransmissions. By contrast, in Adaptive, the range is determined at the EPC and varies depending on the rate of the Attach requests. These back-off algorithms are detailed in Section 5.5.1.

Figure 5.5 shows the call flow of an Attach procedure (modifications to the standard procedure are indicated in red). An MTC server periodically updates the status of an IMSI, which is stored in an HSS, via an MTC-IWF. When a device is powered on, an Attach request arrives at an EPC. In the EPC, the HSS checks the status of the IMSI. If the status is "Disable," an Attach reject containing a back-off timer is returned to the device. The device resends the Attach request after the back-off timer expires. If the status is "Enable," the HSS changes the status to "Disable" so that the HSS can reject the Attach requests from other devices. After completing the Attach, an application in the device notifies the MTC server of the completion of registration. At this time, if the device has a static EID, the device notifies the MTC server about it; otherwise, the device requests the assignment of an EID. The MTC server updates the binding between the time slot and the EID. The MTC server also stops updating the status of the IMSI in the HSS during the time slot to which the EID is assigned. When the bearer is released in the EPC ("S1 Release" in Figure 5.5), the MME notifies the MTC server of the timer value (T3324) which the device in the Attach is notified of. Note that the eNB starts the procedure of S1 Release upon detecting user inactivity using a timer (RRC inactivity timer). Although the period of the timer is usually a few seconds to a few tens of seconds in commercial networks [78–80], this period needs to be shortened to increase the number of IoT devices sharing a single IMSI. The MTC server triggers the timer. When this timer expires, the MTC server calls the device using the EID.

Figure 5.6 shows the call flow of a data upload procedure using device triggering (modifications to the standard procedure are indicated in red). The MTC server manages each timer of the EID. When a timer expires, the MTC server performs device triggering on the device of the EID corresponding to the timer. The MTC server sends a data transmission request to the device. The request arrives at the MTC-IWF. The MTC-IWF retrieves the IMSI corresponding to the EID, contained in the request, from the HSS. Note that the EID is composed of a group id and device id, and the HSS stores only the correspondence information between the group id and the IMSI. This eliminates the need for every EID that is related to the IMSI to be stored, thereby reducing the amount of information retained in the HSS. After retrieving the IMSI, the MTC-IWF determines the presence of the TAU corresponding to the IMSI. If no TAU exists no later than a short time, the MTC-IWF buffers the data transmission request. When the device returns from the PSM and performs the TAU, the MME notifies the MTC-IWF of the completion of the TAU. The MTC-IWF then transfers the buffered request to the device. The request goes through a PGW and reaches an SGW. To re-establish the eNB-to-SGW tunnel for transferring the request to the device, paging messages are broadcast. The device receiving the paging messages re-establishes the data path, receives the data transmission request, and uploads data to the MTC server. After uploading, the device releases the eNB-to-SGW tunnel. This behavior is repeated periodically.

## 5.5 Simulation-based Evaluation

This section evaluates the effectiveness of sharing a single IMSI using packet level simulations. In the proposed method, an appropriate value for the guard time needs to be configured to prevent communication timings from overlapping. In addition, the back-off algorithms and the time slot values have an effect on the number of re-transmitted Attach requests, resulting in variations in the amount of time spent on Attach completion and load operations on the EPC.

First, we determined the value for the guard time taking the background traffic into consideration (Section 5.5.2). Then, we compared the back-off algorithms (Section 5.5.3). Finally, we

Table 5.3: Parameter Settings

| Parameters | Values | |
|---|---|---|
| Number of devices | 100 - 240 | |
| RRC inactivity timer | 2 s | |
| Processing time | *C-plane* | *U-plane* |
| Device | 0.004 s | 0.004 s |
| eNB | 0.004 s | 0.0015 s |
| MME | 0.004 s | - |
| SGW, PGW | 0.004 s | 0.001 s |
| HSS | 0.004 s | - |
| MTC-IWF | 0.004 s | 0.001 s |
| MTC server | - | 0.004 s |
| Link delay time (wireless) | 0.001 s | |
| Link delay time (eNB-SGW) | 0.0075 s | |
| Link delay time (in the EPC) | 0.001 s | |
| Link delay time (EPC-MTC server) | 0.001 s | |

evaluated the number of devices sharing the single IMSI at different time slot values (Section 5.5.4).

### 5.5.1 Simulation Setup and Parameters

We implemented a discrete-event driven simulation of the packet delivery process using the queuing network system in C++ (OMNeT++) [81]. The call flows were simulated as shown in Figures 5 and 6. Each node in the EPC was modeled as a queuing server. The queuing servers identifies received messages and transmit them to the next queuing servers. Thus, messages go through the queuing servers according to the call flows shown in Figures 5 and 6. Messages contained only the parameters required for this evaluation. Realistic processes at each node were substituted for processing delays. To simplify the network configuration of the simulation, all devices were connected to a single eNB. The communication pattern of the device was assumed such that the device recorded the sensor's data (100 byte) in intervals of one second and uploaded the data (called 'sensing data' hereafter) to a MTC server at half-hour intervals. This communication pattern is categorized as a high-frequency IoT service [66]. Although there were several communication patterns, we focused only on the high-frequency IoT service to evaluate the performance limitations of the proposed method. Note that the proposed method is designed on the assumption that a single IMSI is shared among the same communication patterns (as described in Section 5.4). That is, IoT devices in a different communication pattern are assigned different IMSIs. Consequently, we assumed that there is only one communication pattern in order to evaluate the number of IoT devices sharing a single IMSI. The data transmission speed was assumed to be 1 Mbps and took about 1.44 seconds to upload the sensing data.

The measurements were carried out with varying the number of devices powered-on (the number of devices sharing a single IMSI). The power-on timings assumed the cases where the devices were uniformly powered on for 60 seconds or 1200 seconds (except for the evaluation of the guard time in Subsection 5.5.2).

The simulation was conducted 1000 times with different random seeds (in general, the precision error becomes a few percent) for individual power-on patterns and varying the number of devices that were powered on. The parameter settings used in our simulation are listed in Table 5.3. The processing times and link delay times were determined with reference to [82].

We evaluated the following back-off algorithms.

- *Uniform*: All back-off timers are chosen in the range $[0, M]$, where $M$ is the maximum back-off range. In the evaluations, we choose $M = 60$ and $M = 120$.

- *Binary exponential*: The back-off timer is uniformly distributed in the range $[0, 2^{i-1}w]$, where w is the initial back-off range and $i$ is the number of Attach reject received by the device. This means that the back-off range is incremented in a binary exponential manner according to the number of rejections. In the evaluations, we chose $w = 30$ and put a cap on the maximum back-off range ($i \leq 4$).

- *Adaptive*: The back-off time is uniformly distributed in the range $[0, r(T_S + T_G)]$, where $T_S$ is time slot value, $T_G$ is guard time value, and $r$ is the number of Attach requests received by the EPC during $T_S + T_G$. This means that the back-off range changes depending on the rate of Attach requests. In the evaluations, the rate was calculated per $T_S + T_G$ seconds or per the

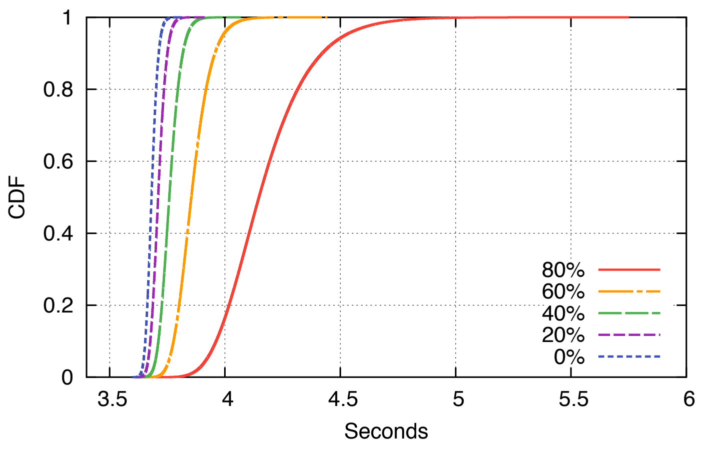


Figure 5.7: Occupation time of IMSI in uploading data. Percentages refer to the utilization of each node in the EPC.

timing when the EPC received ten Attach requests if the number of requests was less than ten during $T_S + T_G$.

We evaluated performance using the following two metrics: (1) maximum time spent on Attach completion, and (2) Attach reject messages per unit time.

### 5.5.2 Evaluation of Guard Time

The guard time needs to be longer than the time required for the devices to upload data. This time includes not only the time spent uploading data but also the time taken to establish communication tunnels. The latter time is influenced by the load of the EPC.

In this simulation, background packets were transmitted to each node (eNB, MME, S/PGW) in the EPC. The sending rates were changed, with the result that the utilization of each node was from 0 to 80%. One hundred devices uploaded data in turn at ten-second intervals. The time that elapsed from "TAU Request" to "Notify timer value" (see Figure 5.6) was measured.

Figure 5.7 shows the occupation time of the IMSI in uploading data. In our assumed IoT service, a six-second guard-time is enough. This is because the EPC restricts requests when the utilization of the node becomes 60 80% in actual operation [83–85]. In the following simulations, the guard time was configured to six seconds.

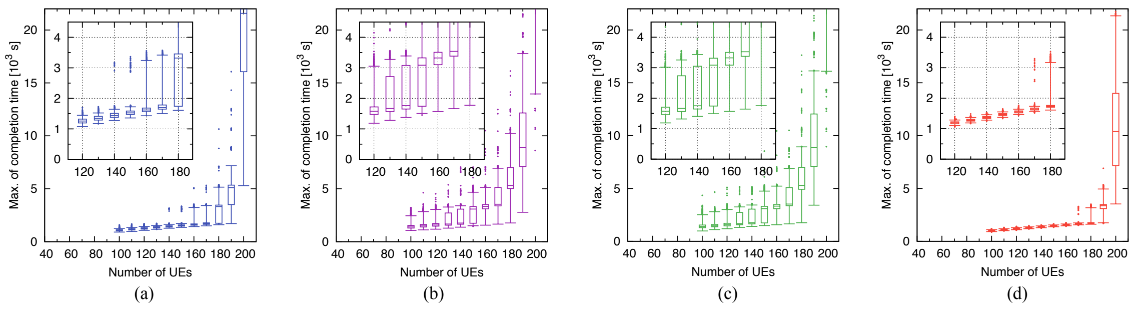


Figure 5.8: Maximum time of Attach completion in the case where the devices were powered on during a short power-on period (60 seconds) in the back-off algorithms: (a) Uniform [0s, 60s], (b) Uniform [0s, 120s], (c) Binary Exponential, (d) Adaptive.

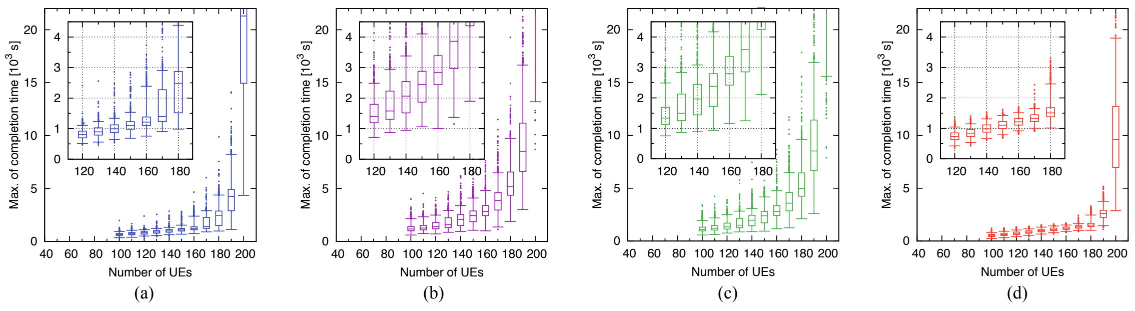


Figure 5.9: Maximum time of Attach completion in the case where the devices were powered on during a long power-on period (1200 seconds) in the back-off algorithms: (a) Uniform [0s, 60s], (b) Uniform [0s, 120s], (c) Binary Exponential, (d) Adaptive.

### 5.5.3 Comparison of Back-off Algorithms

In the comparison of back-off algorithms, the time slot was set to three seconds. This configuration ensured that a single IMSI is occupied by each device for nine seconds (equal to the sum of the time slot and guard time) in a half-hour period (1800 seconds), that is, a single IMSI can be shared among 200 devices using time division.

Figures 8 and 9 show the maximum time of Attach completion in each back-off algorithm. The data are displayed as a boxplot, showing the median value (line), interquartile range (boxes), and 5% to 95% percentile (whiskers), and outliers (dots). The maximum time increased exponentially beyond the ratio of the number of devices sharing the IMSI to the maximum number (200 devices), although the ratio differed depending on the back-off algorithms and the power-on patterns. The

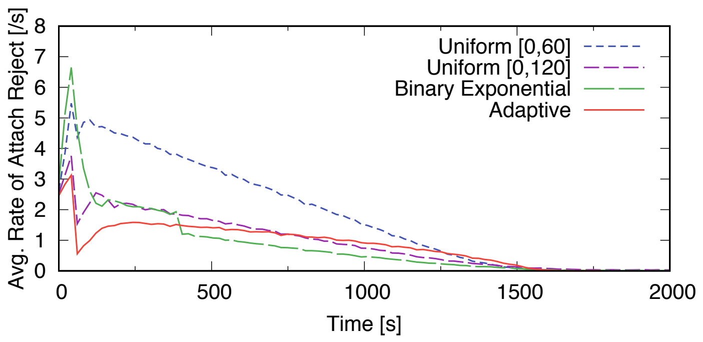


Figure 5.10: Attach Reject rate per unit time in the case where 160 devices were powered on during a short power-on period (60 seconds).

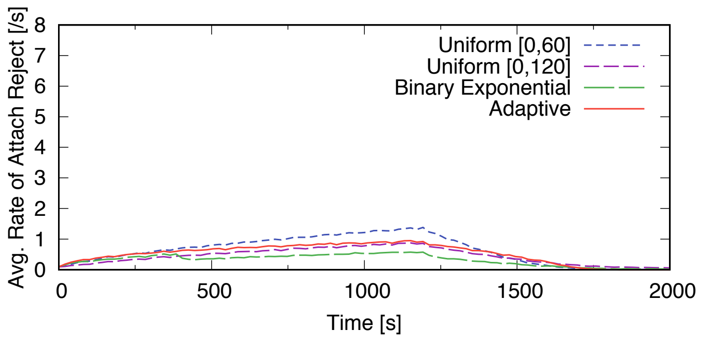


Figure 5.11: Attach Reject rate per unit time in the case where 160 devices were powered on during a long power-on period (1200 seconds).

difference among back-off algorithms showed a similar tendency in the case where the devices were powered on during a short power-on period (Figure 5.8) and also where there was a long power-on period (Figure 5.9). Adaptive back-off algorithms made the maximum time of Attach completion smaller than other algorithms (Figure 5.8 (d) and Figure 5.9 (d)). Although the Uniform back-off algorithm, where the range of random values of the back-off timer is small (Figure 5.8 (a) and Figure 5.9 (a)), reduced the maximum time to a certain extent, the small back-off timer caused the load on the EPC to increase (this is shown in the next paragraph). The Binary exponential back-off algorithms was ineffective in reducing the maximum time. The maximum time of Attach completion was a function of the range of the back-off timer values.

Figures 10 and 11 show the number of Attach Reject per unit time in each back-off algorithm when 160 devices share a single IMSI. Under the condition that the devices were powered on during a long period (see Figure 5.11), the back-off algorithms had little effect on the Attach Reject rate. In contrast, under the condition that the devices were powered on during a short period (see Figure 5.10), the back-off algorithms influenced the Attach Reject rate. The Uniform back-off algorithm, where the back-off range is small, caused a high Attach Reject rate while the Adaptive back-off algorithm decreased the Attach Reject rate.

In order to analytically compare the maximum time of Attach completion in each back-off algorithm, we express the probability that the request from the last device arrives within an "Enable" (vacant) time slot. Let $T_{max}$ be the maximum back-off range, $l_s$ be the duration of a time slot, $l_g$ be the duration of a guard time, $N_s$ be the number of time slots per data upload cycle, and $N_d$ be the number of "Disable" time slots per data upload cycle. Thus, the probability of request occurrence is $2/T_{max}$, the probability that requests arrives within a time slot is $l_s/(l_s + l_g)$, and the probability of an "Enable" time slot is $(N_s - N_d)/N_s$. The probability that requests arrive within an "Enable" time slot, $p$, can be expressed as

$$p = \frac{2}{T_{max}} \cdot \frac{l_s}{l_s + l_g} \cdot \frac{N_s - N_d}{N_s}. \tag{5.1}$$

The probability of not completing the Attach procedure after $n$ retransmission, $q$, can be given by

$$q = (1 - p)^n. \tag{5.2}$$

We evaluate $n$ when $q$ is lower than threshold $\alpha$. The number of retransmissions ($n$) to satisfy $q \leq \alpha$ is expressed as

$$n \geq \frac{\log \alpha}{\log(1 - p)}. \tag{5.3}$$

Obviously, $n$ increases as $T_{max}$ is incremented. The values of $T_{max}$ depend on the back-off algorithms.To evaluate the optimal number of devices sharing a single IMSI, we obtain lower limits of the number of retransmissions ($n$) for the number of "Disable" time slots ($N_d$) from (5.3) under the condition $q \leq 0.05$. Figure 5.12 shows the relation of lower limits of $n$ to $N_d$. The lower limits

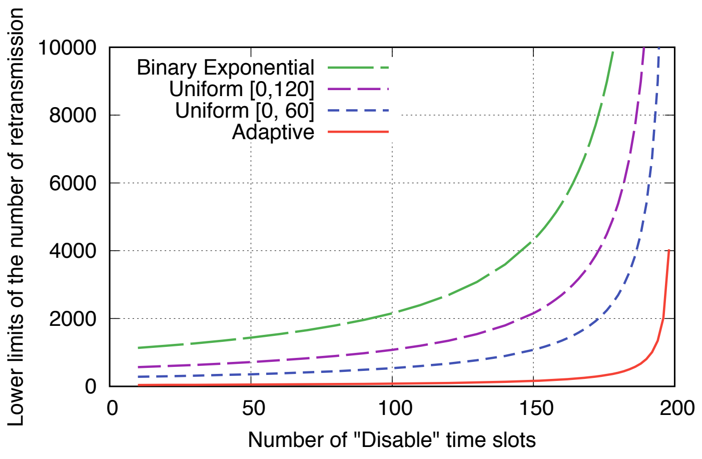


Figure 5.12: Lower limits of the number of retransmissions to the number of “Disable” time slots.

Table 5.4: Analytical Comparison of Back-off Algorithms

| Back-off Algorithms | Maximum of completion time | Attach reject rate | Other characteristics |
|---|---|---|---|
| Uniform | Depends on maximum back-off range ($T_{max}$) | Becomes high as making the maximum of completion time short and conversely | Simple to implement |
| Binary Exponential | Becomes long due to the fact that $T_{max}$ is increased by retransmissions | Becomes least due to the fact that $T_{max}$ is increased by retransmission | Need to count Attach reject at IoT devices |
| Adaptive | Becomes short due to the fact that $T_{max}$ is decreased when the Attach request rate is low | Adjusted by the Attach request rate | Need to figure out the Attach request rate for each IMSIs at EPC |

of $n$ increases exponentially beyond a certain value of $N_d$. Thus, it is desirable to share a single IMSI among the number of devices corresponding to this certain value.

Table 5.4 shows analytical comparisons of the back-off algorithms, including a summary of above evaluations.

The *Adaptive* back-off algorithm can be of some benefit in preventing the number of Attach

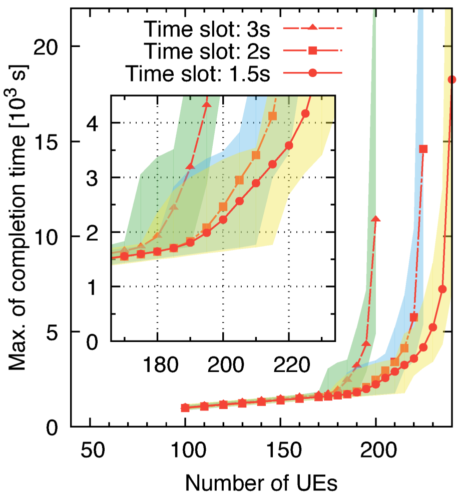


Figure 5.13: Maximum time of attach completion in the case where the devices were powered on during a short power-on period (60 seconds).

Rejects from increasing in some circumstances. The Attach Reject causes an increase in the number of signaling messages (from “Attach Request” to “Attach Reject” as shown in Figure 5.5). The condition wherein the devices were powered on during a short period will increase the load on the EPC. To reduce the impact on the EPC as much as possible, it will be necessary to use the *Adaptive* back-off algorithm instead of the *Uniform* or *Binary Exponential* back-off algorithm.

### 5.5.4 Impact of time slot on the number of devices sharing IMSI

The number of devices sharing a single IMSI was evaluated in the case of where the Adaptive back-off algorithm was used. The guard time was six seconds and the time slot value was varied.

Figures 13 and 14 show the maximum time of attach completion for various time slots. The dotted lines are the average values and the fill areas are the range of 1th-99th percentile. The short time slot reduced the maximum time where the devices were powered on during a short power-on period and also during a long power-on period. This is a result of increasing the maximum number of devices sharing the IMSI. The impacts of reducing the time slot on the EPC are shown in the Figures 15 and 16. The difference between curves in Figures 15 and 16 is induced by changes in the back-off range in the Adaptive back-off algorithm. In the case where the devices were powered

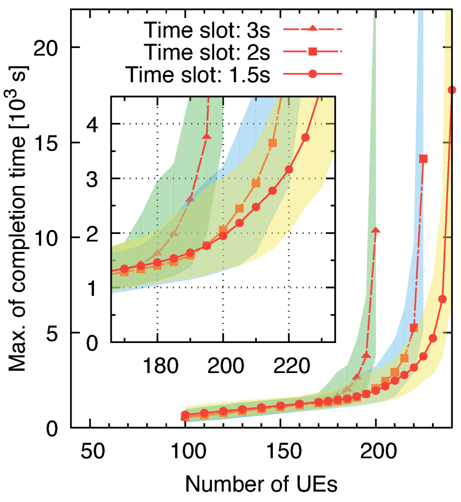


Figure 5.14: Maximum time of attach completion in the case where the devices were powered on during a long power-on period (1200 seconds).

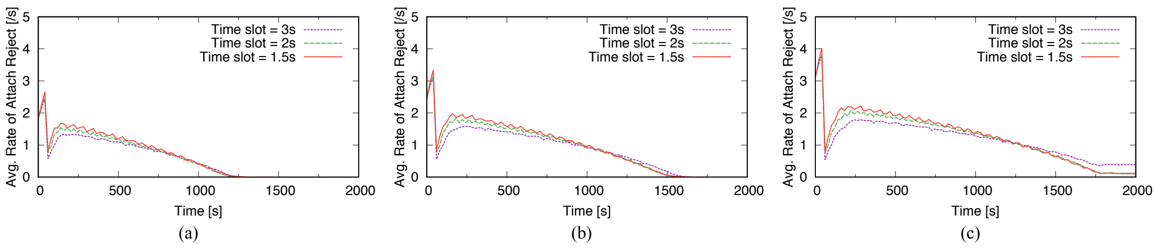


Figure 5.15: Attach Reject rate per unit time in the case where the devices were powered on during a short power-on period (60 seconds). (a) UE=120. (b) UE=160. (c) UE=200.

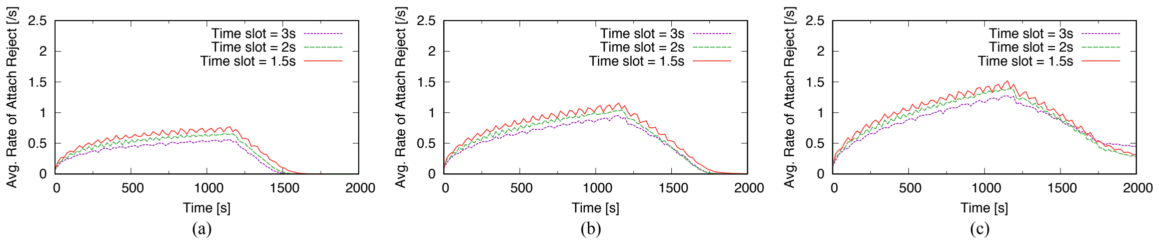


Figure 5.16: Attach Reject rate per unit time in the case where the devices were powered on during a long power-on period (1200 seconds). (a) UE=120. (b) UE=160. (c) UE=200.

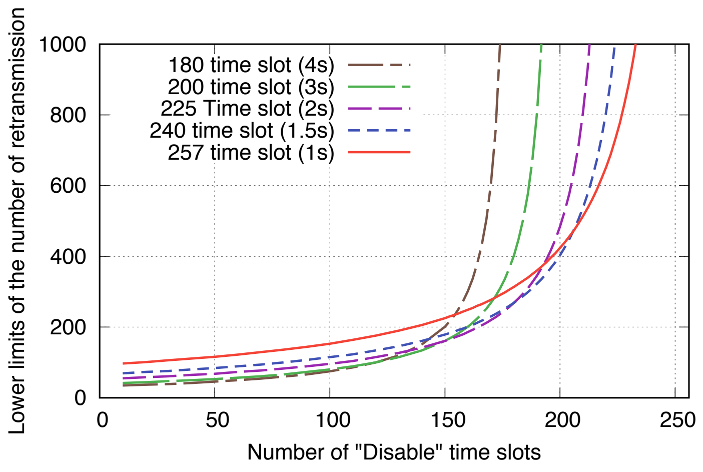


Figure 5.17: Lower limits of the number of retransmissions to the number of “Disable” time slots while varying the duration of a time slot.

on for a short power-on period (60 seconds), Attach requests are initially sharply concentrated. This results in a long back-off range, inducing a low rate of Attach requests. The back-off range, in turn, becomes short, resulting in a high rate of Attach requests. In the case where the power-on period is long (1200 seconds), there is no concentration of Attach requests. This does not cause large fluctuations in the back-off range. In this situation, the rate declines after the power-on period as shown in Figure 5.16. Varying the time slot had little effect on the Attach Reject rate regardless of the scenario (power-on period and the number of devices powered-on). This is because although the duration of time slots becomes short, the number of time slots increases.

We evaluate the tradeoff between the number of devices sharing a single IMSI and the duration of a time slot using (3) in previous subsection (Section 5.5.3). Figure 5.17 shows the relation of lower limits of $n$ to $N_d$) under the condition $q \leq 0.05$. In the case where the number of IoT devices sharing a single IMSI is not large, it is desirable to make a time slot longer.

Considering the battery life of the devices, it is preferable for the devices to complete the Attach and activate the PSM as soon as possible after powering-on. For example, assuming that the expected time of Attach completion is 3600 seconds with a probability of 99%, the sharing of the IMSI among approximately 200 devices meets the expectation for any power-on pattern when the time slot is less than or equal to two seconds, as can be seen from Figures 13 and 14. This result

Table 5.5: Analytical Comparison between GW-based and Proposed Method

| | Handling for improvement of aggregation | | | |
|---|---|---|---|---|
| Methods | Moving devices | Device density | Load | Limitations |
| GW-based | Mobility management is required on the device side. | The limitation of distance and gateway capacity need to be considered. | The load on the gateway needs to be considered. | Difficult to prolong battery life. |
| Proposed | Mobility management is inherently supported in the EPC. | Relies on area design of base stations. | MTC server needs to determine time slot and guard time. | Load due to retransmission. Only devices under the same IoT service. |

shows that the proposed method can reduce the amount of state information in the EPC by two orders of magnitude.

### 5.5.5 Discussion

Table 5.5 shows analytical comparisons between our proposed method and the previous method by which devices communicate with the EPC through the devices selected as gateway (hereinafter referred to as the "GW-based method"). In the GW-based method, devices require strategies appropriate to the device-side situation or the number of aggregating communication lines decreases. In the case of the IoT service targeting moving devices, the GW-based method needs to select appropriate devices as gateways to increase the opportunities for moving devices to obtain the connectivity using gateway devices. The GW-based method also needs to determine the number of gateways according to device density. In an area that is not densely packed with IoT devices, the number of gateway devices is determined by taking the limitation of communication distances in to account. On the other hand, in an area that is densely packed with IoT devices, the limitation of the number of devices that can be managed by one gateway device is an important factor in determining the number of gateways. In addition, the loads on the gateway devices need to be considered.

In the proposed method, the network (the EPC and the MTC server) is responsible for management. For moving devices, the proposed method does not need additional functions since the EPC inherently supports mobility management. The strategy for dealing with device density relies on area designs (capacity and coverage of eNB). For load fluctuations in wireless access networks or the EPC, the MTC server plans and determines the durations of time slots and guard times, which affect the number of aggregating communication lines. The presumption of large load fluctuations leads to a longer time slot and guard time, thereby reducing the number of devices sharing a single IMSI. When load fluctuations are restrained by constructing a dedicated network using virtualization technologies, the number of devices sharing a single IMSI can be increased. In this chapter, the time slot and guard time were set based on the assumption that the EPC had a certain load. To evaluate the performance of the proposed method in detail, it will be necessary to evaluate the load tolerance by varying the durations of the time slot and guard time.

Considering the impact on devices or the EPC, both methods have certain limitations. The GW-based method makes it difficult to prolong battery life of devices using PSM and requires engineering for selecting appropriate gateway devices. On the other hand, the proposed method increases the load on the EPC to some extent. This is because the proposed method may result in the retransmission of the Attach request, which is sent from the powered-on devices. In addition, the proposed method is intended only for devices under the same IoT services.

Table 5.6 shows a numerical comparison of the amount of state information in the EPC for a IoT service used to track bicycles. The number of bicycles in a dense zone (Tokyo, Japan: 13,500 km2) and in a non-dense zone (Hokkaido, Japan: 83,500 km2) is 9,000,000 (666.7 per square kilometer) and 2,800,000 (33.5 per square kilometer), respectively [87]. The communication pattern for uploading location information of bicycles is the same as the pattern used in the simulation (as shown in Section 5.5.1). For numerical calculations with the GW-based method, completely optimal situations were assumed. That is, it was assumed that each device selected as a gateway was placed in an area of 1.4 km mesh (note that the maximum distance between the gateway and devices is 1 km) so that all devices communicated with the EPC via the gateway devices. In the dense zone, multiple gateway devices were placed in one area due to the limitation on the number of devices that can be managed by one gateway device (150 devices [86]). In the proposed method, the number

Table 5.6: Numerical Comparison in The Amount Of State Information in Bicycle Tracking Service

| Methods | Amount of state information in the EPC | | Remarks |
|---|---|---|---|
| | Donse zone | Non-dense zone | |
| Curent EPC | 9000 K [context] | 2800 K [context] | - |
| GW-based | 60 K [context] | 42.5 K [context] | No.ofdevicesperGW:150 [86] Communication distance: 1 km |
| Proposed | 45 K [context] | 14 K [context] | No. of device sharing an IMSI: 200 (from the simulation results) |

of devices sharing a single IMSI was assumed to be 200 devices, which was obtained from the simulation results. This comparison shows that the proposed method can reduce the amount of state information as well as or better than the GW-based method in some situations.

## 5.6 Conclusion

To reduce the amount of state information in the EPC by aggregating communication lines in the EPC without gateway devices, this chapter proposed a method that assigns the same IMSI to multiple IoT devices and prevents the communication timings from overlapping. Our simulations showed that the proposed method provided cellular communication lines for two hundred or more IoT devices using only a single IMSI, thereby reducing the amount of state information in the EPC to a significant degree.

The proposed method can reduce the amount of state information significantly without device-supported planning and engineering for aggregating communication lines. Therefore, the proposed method can help mobile network operators provide a large number of IoT devices with cellular connectivity on limited resources. This may bring about benefits especially for mobile virtual network operators (MVNOs) that put a high priority on reducing CAPEX/OPEX to achieve low-cost communications.

In future research, we will evaluate the feasibility and performance of our method using a prototype implementation. In addition, we will study the reduction of state information by using our method in a realistic IoT service.

# Chapter 6

# Conclusion

For satisfying diversified and specialized requirements, it is essential to construct dedicated mobile networks, in which system configuration, signaling processes, and information management are specialized for particular services, without investing in extra capacity. Network function virtualization and network virtualization are promising technologies to realize this goal. However, it is not sufficient that these virtualization technologies are applied to mobile networks. Therefore, we studied methodologies for improving the benefits from virtualization and focused on the reduction of signaling load and a significant amount of information in mobile networks.

In Chapter 2, we investigated the state information migration for improving the benefits from virtualization and proposed state information migration without modifying the standard procedures. This method coordinates routing control with the deployment of state information using OpenFlow. We also investigated the strategy of conducting the IMS reconfiguration using state information migration, and the implementations for improving the performance of migration. We discussed the feasibility of the proposed methods using the testbed system and showed that the proposed methods cause the IMS to have greater reliability and flexibility. Under the experiments using the testbed system, we assumed only a telephone service. Therefore, there is room for further investigation under the existence of multiple services to clarify performance limitations.

In Chapter 3, we investigated a service-specific network virtualization that creates several virtual networks that are composed of network functions specialized for particular services. We proposed a method that enables services to be processed in the appropriate virtual networks by forwarding a sequence of signaling messages to these networks and migrating state information from one virtual network to another. The method can effectively forward the signaling messages using a combination of IP flow identification and application header inspection. Our prototype implementation showed that the service-specific network virtualization could reduce the load of the entire mobile network compared to the current mobile network, even if there are overheads of application header inspection and state information migration. In our prototype, we limited the condition as follows: two different services, two virtual networks without localization, and static configuration of each virtual network. Therefore, there is room for a quantitative analysis under no or few limitations to clarify drawbacks and further research to overcome the disadvantages. In the case of taking into account localized virtual networks, the gateways need to identify users and devices as well as services. Besides, the controller, which manages the gateways, must manage location information of devices and create routing information coupled with the location information.

In Chapter 4, we investigated a scalable management of SBFs (gateway functions in mobile network virtualization infrastructure) for the service-specific network virtualization in consideration of the practical use of it. We proposed a method to enhance a centralized management that reactively installs rules for flow control into SBFs by transmitting the unknown packet to the controller. The proposed method redirects unknown packets to another SBF instead of the controller and shares the matched rules between SBFs on the redirection path. This approach can reduce the number of unknown packets transmitted to the controller for inspection. Our simulations showed that the proposed method reduced the number of unknown packets that arrived at the controller per unit time by approximately 14-63%. This can result in preventing the number of replica controllers from increasing. Although the proposed method increased the one-way delay due to the overhead of redirecting packets, it brought benefits that more than compensate for the overhead when assuming a large number of SBFs.

In Chapter 5, we proposed a method that assigns the same IMSI to multiple IoT devices and prevents the communication timings from overlapping to reduce the amount of state information in the EPC by aggregating communication lines. Our simulations showed that the proposed method provided cellular communication lines for two hundred or more IoT devices using only a single

IMSI, thereby reducing the amount of state information in the EPC by a significant degree. The proposed method can help mobile network operators provide a significant number of IoT devices with cellular connectivity on limited resources. This may bring about benefits, especially for mobile virtual network operators (MVNOs) that put a high priority on reducing CAPEX/OPEX to achieve low-cost communications.

In the future, it will become increasingly necessary to streamline signaling processes and information management for each service. A lot of service-specific mobile networks will be configured in a single physical network infrastructure with the progress of virtualization technologies. The remaining work involves verification of the feasibility of the service-specific network virtualization under the condition that virtual networks are localized and that services are provided using multiple virtual networks. Also, it is necessary to consider the dynamic configuration of virtual networks and performance improvement under the existence of many service-specific virtual networks. We believe that the work in this thesis has fruitful implications for future research. The construction of service-specific mobile networks poses new challenges for network operation and management. It is very hard for MNOs to dynamically adjust hardware resource assignment and to optimally tune parameters for each network. Therefore, a future mobile network may need to be capable of self-adjustment and self-sustained growth.

# Bibliography

[1] M. Fiorani, P. Monti, B. Skubic, J. Mårtensson, L. Valcarenghi, P. Castoldi, and L. Wosinska, "Challenges for 5G Transport Networks," in *Proceedings of the 2014 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, Dec 2014.

[2] "Understanding 5G: Perspectives on Future Technological Advancements in Mobile," GSMA Intelligence, December 2014.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet Impasse through Virtualization," *IEEE Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[4] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network Virtualization Architecture: Proposal and Initial Prototype," in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '09, (New York, NY, USA), pp. 63–72, ACM, 2009.

[5] M. Ito, S. Komorita, Y. Kitatsuji, H. Yokota, and T. Hayashi, "Cloudization of IP Multimedia Subsystem and Testbed Verification: High Reliability and Energy Saving for IMS," *IPSJ Digital Practice*, vol. 4, pp. 323–332, October 2013.

[6] M. Ito, S. Komorita, Y. Kitatsuji, and H. Yokota, "Openflow-based Routing Mechanism for Call Session State Migration in the IMS," in *Proceedings of the 7th International Conference on Recent Advances in Computer Engineering Series (WSEAS)*, 2013.

[7] "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3," 3GPP TS 24.229 V9.13.0, 2012.

[8] *Global System for Mobile communications Association*, [Online]. Available: http://www.gsma.com/.

[9] *The Stanford OpenFlow Team: OpenFlow Switch Specification Version1.1.0 Implemented*, [Online]. Available: http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69–74, Mar. 2008.

[11] M. Ito, K. Nakauchi, Y. Shoji, N. Nishinaga, and Y. Kitatsuji, "Service-Specific Network Virtualization to Reduce Signaling Processing Loads in EPC/IMS," *IEEE Access*, vol. 2, pp. 1076–1084, 2014.

[12] M. Ito, K. Nakauchi, Y. Shoji, and Y. Kitatsuji, "Mechanism of Reducing Signaling Processing Load in EPC/IMS Using Service-Specific Network Virtualization," in *Proceedings of the 2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, March 2014.

[13] M. Ito, N. Nishinaga, and Y. Kitatsuji, "Enhanced Centralized Management of Gateways using Redirection-based Rules Sharing for Mobile Network Virtualization," *Springer Telecommunication Systems*, 2016.

[14] M. Ito, N. Nishinaga, and Y. Kitatsuji, "Redirection-Based Rules Sharing Method for the Scalable Management of Gateways in Mobile Network Virtualization," in *Proceedings of the 2015 IEEE Globecom Workshops*, pp. 1–7, Dec 2015.

[15] M. Ito, N. Nishinaga, Y. Kitatsuji, and M. Murata, "Reducing State Information by Sharing IMSI for Cellular IoT Devices," *IEEE Internet of Things Journal*, June 2016.

[16] M. Ito, N. Nishinaga, Y. Kitatsuji, and M. Murata, "Aggregating Cellular Communication Lines for IoT Devices by Sharing IMSI," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC 2016)*, pp. 1–7, May 2016.

[17] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," Internet Engineering Task Force, RFC 3261, 2012.

[18] M. Fakhfakh, O. Cherkaoui, I. L. Bedhiaf, and M. Frikha, "High Availability in IMS Virtualized Network," in *Proceedings of the First International Conference on Communications and Networking*, pp. 1–6, Nov 2009.

[19] A. B. Letaifa, A. Haji, M. Jebalia, and S. Tabbane, "State of the Art and Research Challenges of New Services Architecture Technologies: Virtualization, SOA and Cloud Computing," *International Journal of Grid and Distributed Computing*, vol. 3, no. 4, pp. 69–88, 2010.

[20] S. Komorita, H. Yokota, A. Dutta, C. Makaya, S. Das, D. Chee, F. J. Lin, and H. Schulzrinne, "User-Transparent Reconfiguration Method for Self-Organizing IP Multimedia Subsystem," in *Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1137–1144, June 2011.

[21] Y. Liu, Y. Liu, and X. Wang, "Reliability Mechanism for the Core Control Network-Element S-CSCF in IMS," in *Proceedings of the 2011 International Conference on Network Computing and Information Security (NCIS)*, vol. 1, pp. 57–61, May 2011.

[22] T. Usui, Y. Kitatsuji, H. Yokota, and N. Nishinaga, "Restoring CSCF by Leveraging Feature of Retransmission Mechanism in Session Initiation Protocol," in *Proceedings of the Third International Conference on Emerging Network Intelligence*, pp. 50–56, 2011.

[23] S. Komorita, M. Ito, Y. Kitatsuji, and H. Yokota, "Dynamic IMS Reconfiguration using Session Migration for Power Saving," in *Proceedings of the Ninth Advanced International Conference on Telecommunications*, pp. 191–196, Citeseer, 2013.

[24] R. S. Boyer and J. S. Moore, “A Fast String Searching Algorithm,” *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[25] *Global Inter-Cloud Technology Forum*, [Online]. Available: http://www.gictf.jp/.

[26] *The Open Source IMS Core Project*, [Online]. Available: http://www.openimscore.org/.

[27] *IMS Bench SIPp*, [Online]. Available: http://sipp.sourceforge.net/index.html.

[28] *Japan’s Communication Usage Status based on Traffic*, [Online]. Available: http://www.soumu.go.jp/main_content/000194119.pdf.

[29] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, “Design Considerations for a 5G Network Architecture,” *IEEE Communications Magazine*, vol. 52, pp. 65–75, Nov 2014.

[30] “Network Functions Virtualisation–Introductory White Paper,” ETSI, October 2012.

[31] A. Baliga, X. Chen, B. Coskun, G. de los Reyes, S. Lee, S. Mathur, and J. E. Van der Merwe, “VPMN: Virtual Private Mobile Network Towards Mobility-as-a-service,” in *Proceedings of the Second International Workshop on Mobile Cloud Computing and Services*, MCS ’11, pp. 7–12, 2011.

[32] “LTE Diameter Signaling Index,” whitepaper, Oracle Company, Redwood Shores, CA, USA, 2013.

[33] G. Hampel, M. Steiner, and T. Bu, “Applying Software-Defined Networking to the Telecom Domain,” in *Proceedings of the 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 133–138, April 2013.

[34] L. E. Li, Z. M. Mao, and J. Rexford, “Toward Software-Defined Cellular Networks,” in *Proceedings of 2012 European Workshop on Software Defined Networking*, pp. 7–12, Oct 2012.

[35] J. Kempf, B. Johansson, S. Pettersson, H. Lüning, and T. Nilsson, “Moving the Mobile Evolved Packet Core to the Cloud,” in *Proceedings of the 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 784–791, Oct 2012.

[36] K. Pentikousis, Y. Wang, and W. Hu, “Mobileflow: Toward Software-Defined Mobile Networks,” *IEEE Communications Magazine*, vol. 51, pp. 44–53, July 2013.

[37] G. Brown, “The Evolution of the Signaling Challenge in 3G & 4G Networks,” white paper, Heavy Reading, New York, NY, USA, 2012.

[38] “IP Multimedia Subsystem (IMS) Service Continuity; Inter-UE Transfer Enhancements; Stage 2,” 3GPP TR 23.831 v10.0.0, 2010.

[39] “Service Requirements for Machine-Type Communications (MTC); Stage 1,” 3GPP TS 22.368 v12.0.0, 2012.

[40] “S1 Application Protocol (S1AP) Release 11,” 3GPP TS 36.413 v11.5.0, 2013.

[41] R. Stewart, “Stream Control Transmission Protocol,” RFC 4960, IETF, 2007.

[42] A. Belbekkouche, M. M. Hasan, and A. Karmouch, “Resource Discovery and Allocation in Network Virtualization,” *IEEE Communications Surveys Tutorials*, vol. 14, pp. 1114–1128, Fourth 2012.

[43] M. Hoffmann and M. Staufer, “Network Virtualization for Future Mobile Networks: General Architecture and Applications,” in *Proceedings of the 2011 IEEE International Conference on Communications Workshops (IEEE ICC 2011)*, pp. 1–5, June 2011.

[44] T. Taleb and A. Kunz, “Machine Type Communications in 3GPP Networks: Potential, Challenges, and Solutions,” *IEEE Communications Magazine*, vol. 50, pp. 178–184, March 2012.

[45] I. Widjaja, P. Bosch, and H. L. Roche, “Comparison of MME Signaling Loads for Long-Term-Evolution Architectures,” in *Proceedings of the 2009 IEEE 70th Vehicular Technology Conference Fall (VTC 2009-Fall)*, pp. 1–5, Sept 2009.

[46] I. Sato, A. Bouabdallah, and X. Lagrange, "Improving LTE/EPC Signaling for Sporadic Data with a Control-Plane based Transmission Procedure," in *Proceedings of the 2011 14th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 1–5, Oct 2011.

[47] Y. Hong, C. Huang, and J. Yan, "Mitigating SIP Overload Using a Control-Theoretic Approach," in *Proceedings of the 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)*, pp. 1–5, Dec 2010.

[48] H. Tang, "An Adaptive Paging Area Selection Scheme," in *Proceedings of the 2008 Third International Conference on Communications and Networking in China*, pp. 1106–1110, Aug 2008.

[49] *OpenEPC Rel. 4*, [Online]. Available: http://www.openepc.net/_docs/OpenEPC-Whitepaper_nov2012.pdf.

[50] BerliOS, *UCT IMS Client*, [Online]. Available: http://sourceforge.net/projects/uctimsclient.berlios/, 2006.

[51] V. Yazc, U. C. Kozat, and M. O. Sunay, "A New Control Plane for 5G Network Architecture with a Case Study on Unified Handoff, Mobility, and Routing Management," *IEEE Communications Magazine*, vol. 52, pp. 76–85, Nov 2014.

[52] A. Banerjee, X. Chen, J. Erman, V. Gopalakrishnan, S. Lee, and J. Van Der Merwe, "MOCA: A Lightweight Mobile Cloud Offloading Architecture," in *Proceedings of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '13, (New York, NY, USA), pp. 11–16, ACM, 2013.

[53] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker, "Rethinking Enterprise Network Control," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1270–1283, Aug 2009.

[54] "Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 Application Protocol (S1AP)," 3GPP TS 36.413 v11.5.0, ETSI.

[55] A. Tongaonkar, S. Vasudevan, and R. Sekar, "Fast Packet Classification for Snort by Native Compilation of Rules," in *Proceedings of the 22nd Conference on Large Installation System Administration Conference*, LISA'08, (Berkeley, CA, USA), pp. 159–165, USENIX Association, 2008.

[56] R. Ahmed and R. Boutaba, "Design Considerations for Managing Wide Area Software Defined Networks," *IEEE Communications Magazine*, vol. 52, pp. 116–123, July 2014.

[57] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable Flow-based Networking with DIFANE," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 351–362, 2010.

[58] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-performance Networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, (New York, NY, USA), pp. 254–265, ACM, 2011.

[59] *Linux Containers*, [Online]. Available: https://linuxcontainers.org.

[60] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," in *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies Conference on The Conference on Computer Communications*, vol. 2, pp. 594–602 vol.2, Mar 1996.

[61] A. Munjal, T. Camp, and W. C. Navidi, "SMOOTH: A Simple Way to Model Human Mobility," in *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '11, (New York, NY, USA), pp. 351–360, ACM, 2011.

[62] *SMOOTH: A Simple Way to Model Human Mobility*, [Online]. Available: http://toilers.mines.edu/Public/Code/smooth.html.

[63] *Tcpreplay*, [Online]. Available: http://tcpreplay.appneta.com.

[64] E. J. Rosensweig, J. Kurose, and D. Towsley, “Approximate Models for General Cache Networks,” in *Proceedings of the 2010 IEEE INFOCOM*, pp. 1–9, March 2010.

[65] A. Dan and D. Towsley, “An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 18, no. 1, pp. 143–152, 1990.

[66] T. Rebbeck, M. Mackenzie, and N. Afonso, “Low-Powered Wireless Solutions have the Potential to Increase the M2M Market by over 3 Billion Connections,” Analysys Mason Limited, London UK, 5177472, 2014.

[67] “Architecture Enhancements to Facilitate Communications with Packet Data Networks and Applications,” 3GPP TS 23.682 V12.2.0, 2014.

[68] “Nokia LTE M2M Optimizing LTE for the Internet of Things,” Nokia, Helsinki Finland, C401-01085-WP-201409-1-EN, 2014.

[69] “LTE Release 13,” Ericsson, Stockholm Sweden, Uen 284 23-8267, 2015.

[70] V.-G. Nguyen, T.-X. Do, and Y. Kim, “SDN and Virtualization-based LTE Mobile Network Architectures: A Comprehensive Survey,” *Wireless Personal Communications*, vol. 86, no. 3, pp. 1401–1438, 2016.

[71] K. Samdanis, A. Kunz, M. I. Hossain, and T. Taleb, “Virtual Bearer Management for Efficient MTC Radio and Backhaul Sharing in LTE Networks,” in *Proceedings of the 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 2780–2785, September 2013.

[72] S. Sakurai, G. Hasegawa, N. Wakamiya, and T. Iwai, “Performance Evaluation of a Tunnel Sharing Method for Accommodating M2M Communication to Mobile Cellular Networks,” in *Proceedings of the 2013 IEEE Globecom Workshops*, pp. 157–162, Dec 2013.

[73] Y. Kitatsuji and H. Yokota, “On-Demand Session Establishment for All IP-based Mobile Networks,” in *Proceedings of the 6th EURO-NF Conference on Next Generation Internet*, pp. 1–8, June 2010.

[74] J. Kaippallimalil and H. A. Chan, “Network Virtualization and Direct Ethernet Transport for Packet Data Network Connections in 5G Wireless,” in *Proceedings of the 2014 IEEE Global Communications Conference (GLOBECOM 2014)*, pp. 1836–1841, Dec. 2014.

[75] S. Matsushima and R. Wakikawa, “Stateless User-Plane Architecture for Virtualized EPC (vEPC),” Work in Progress, draft-matsushima-stateless-uplane-vepc-04, 2015.

[76] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, “Softcell: Scalable and Flexible Cellular Core Network Architecture,” in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, pp. 163–174, ACM, Dec. 2013.

[77] D. J. Goodman, R. A. Valenzuela, K. Gayliard, and B. Ramamurthi, “Packet Reservation Multiple Access for Local Wireless Communications,” *IEEE Transactions on Communications*, vol. 37, no. 8, pp. 885–890, 1989.

[78] “LTE RAN Enhancements for Diverse Data Applications,” 3GPP TR 36.822 V11.0.0, September 2012.

[79] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, “A Close Examination of Performance and Power Characteristics of 4G LTE Networks,” in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, pp. 225–238, 2012.

[80] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto, “Streaming over 3G and LTE: How to Save Smartphone Energy in Radio Access Network-Friendly Way,” in *Proceedings of the 5th Workshop on Mobile Video*, pp. 13–18, 2013.

[81] *OMNeT++*, [Online]. Available: http://www.omnetpp.org.

[82] “Feasibility study for Further Advancements for E-UTRA (LTE-Advanced),” 3GPP TR 36.912 V12.0.0, September 2014.

[83] “NEC Virtualized Evolved Packet Core – vEPC,” NEC Corporation, Tokyo Japan, White Paper TE-524262, 2014.

[84] “Nokia Networks Telco Cloud is on the Brink of Live Deployment,” Nokia Solutions and Networks, Helsinki Finland, C401-01096-B-201410-1-EN, 2013.

[85] “Implement Overload Protection for Gateways and Neighboring Network Elements on the ASR5x00 Series,” Cisco, CA US, 119196, 2015.

[86] “Study on LTE Device to Device Proximity Services (ProSe) – Radio Aspects,” 3GPP TR 36.843 V1.0.0, November 2013.

[87] “Bicycle Holdings Trend Survey,” Japan Bicycle Promotion Institute, Tech. Rep., Mar. 2008. In Japanese.