Master's Thesis

Title

Construction method of service space in virtualized network system based on chemical-inspired tuple space model

Supervisor

Professor Morito Matsuoka

Author

Shun Sakurai

February 10th, 2015

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Construction method of service space in virtualized network system
based on chemical-inspired tuple space model

Shun Sakurai

Abstract

In virtualized network system for cloud computing environment, multiple services are located on virtual and physical servers for constructing service space, and these servers provide services according to user requests. For ensuring the scalability of the system to the number of servers, deployed services and user requests, and for detecting and recovering system failures in such large-scale system, we need to construct the service space in an autonomous distributed fashion. One possible way to realize this is to exploit biochemical mechanisms, that is often applied to various information network systems due to its autonomic, distributed, and self-organized behavior.

In this thesis, we propose a construction method of service space based on chemical-inspired tuple space model which realize spatial coordination of system. In the proposed method, a physical server is modelled as a tuple space, and user requests, service demands, and server resources are considered as chemical substances in the tuple space. Then, we define chemical reactions in the tuple space to represent the execution of the services, and the increase and decrease of the service demands and server resources. Furthermore, we construct the network of multiple tuple spaces to model the service space in virtualized network system, and describe the diffusion of the services and the movement of user requests.

From extensive simulation results, we confirm that the proposed method realizes distributed execution of multiple services according to user requests, autonomous sharing of server resources among services on each server, relocation of services to appropriate nodes, and load balancing among surrounding servers in various situations. Furthermore, we present the application scenario of the proposed method to placing functions and determining flow routes in Network Function Virtualization (NFV) architecture to confirm the suitability of the proposed method for actual network services.

Keywords

Virtualized network system

Chemical reaction

Tuple space model

Spatial coordination

Network Function Virtualization (NFV)

Contents

1 Introduction **5**

2 Chemical-inspired tuple space model **9**

2.1 Overview 9

2.2 Spatial coordination by chemical reactions 9

2.3 Applicability to the virtualized network system 11

3 Construction method of service space **14**

3.1 Sharing server resources 14

3.2 Diffusion of services 15

3.3 Movement of user requests 15

4 Performance evaluation **16**

4.1 τ-leaping algorithm 16

4.2 Simulation scenarios 16

4.3 Simulation results and discussions 17

5 Application to Network Function Virtualization **26**

5.1 Overview 26

5.2 Extension of proposed method for NFV 26

6 Conclusion and Future work **28**

Acknowledgement **29**

Reference **30**

List of Figures

1 Virtualized network system for cloud computing environment 6
2 Overview of chemical-inspired tuple space model 10
3 Gradient field and *REQ* movement along the gradient 12
4 SINET backbone network topology 18
5 Scenario 1: Concentration changes of *SERV*s on node 1 20
6 Scenario 1: Concentration changes of *MEDIATE*s and *CATAL* on node 1 20
7 Scenario 1: Generation rate of $toserv(SERV, REQ)$s on node 1 21
8 Scenario 1: Gradient field for services at 200 th time step (top) and 1,500 th time step (bottom) 21
9 Scenario 2: Concentration changes of *SERV* on nodes 8 (top), 3 (middle), and 1 (bottom) 22
10 Scenario 2: Gradient field at 200 th time step (top), 400 th time step (middle), and 1,500 th time step (bottom) 22
11 Scenario 2: Moving rate changes of REQ_1 on node 1 23
12 Scenario 3: Average generation rate of $toserv(SERV_1, REQ_1)$ from 500 th to 2,000 th time step 24
13 Scenario 4: Average generation rate of $toserv(SERV_1, REQ_1)$ from 500 th to 2,000 th time step 25

1 Introduction

In virtualized network system for cloud computing environment as shown Figure 1, multiple services and functions (hereinafter referred to services) are located on virtual and physical servers for constructing service space. For example, in mashup Web services, multiple service components such as databases, registration, and authentication are configured on servers in cloud computing environments at data centers [1]. Similarly, in Network Function Virtualization (NFV), virtualized network functions such as Deep Packet Inspection (DPI), firewall, network accelerator, and Network Address Translation (NAT) are installed on universal servers in the network so that packet flows go through such functions [2, 3]. Each server in the service space provides one or more services and these services share the server resources [4]. Therefore, it is important to determine server locations, service distribution to servers according to the demand for the services to construct effective service space.

For resource management in such a service space, there are two types of controls, centralized control [5] and autonomous distributed control [6]. In case of the centralized control, since a central controller needs to maintain the computing resources and services in the service space [7], the overhead would increase rapidly with the extension of the service space. Furthermore, it takes larger time and cost for detecting and recovering the system failures. Therefore, for ensuring the scalability of the system to the number of servers, deployed services and requests, and for quick detection and recovery of the system failures in such a large-scale system, we need to construct and maintain the service space in an autonomous distributed fashion.

Furthermore, today's and future's information network services are characterized by dynamism and unpredictability [8]. For such services we require autonomous distributed behavior without human intervention. One possible way to realize this is to exploit biochemical mechanisms, that is often applied to various information network systems due to its autonomic, distributed, and self-organized behaviors [9–15].

In [16], a tuple space model associated with chemistry has been proposed. In this model, a system component is modelled as a chemical tuple space, and these tuple space are connected to form a network as a whole system. In the tuple space, a tuple mimics a chemical substance, and a notion of activity value for tuples [17] represents as chemical concentration. Chemical reactions are properly installed into the tuple space, and evolve concentration of tuples over time exactly in

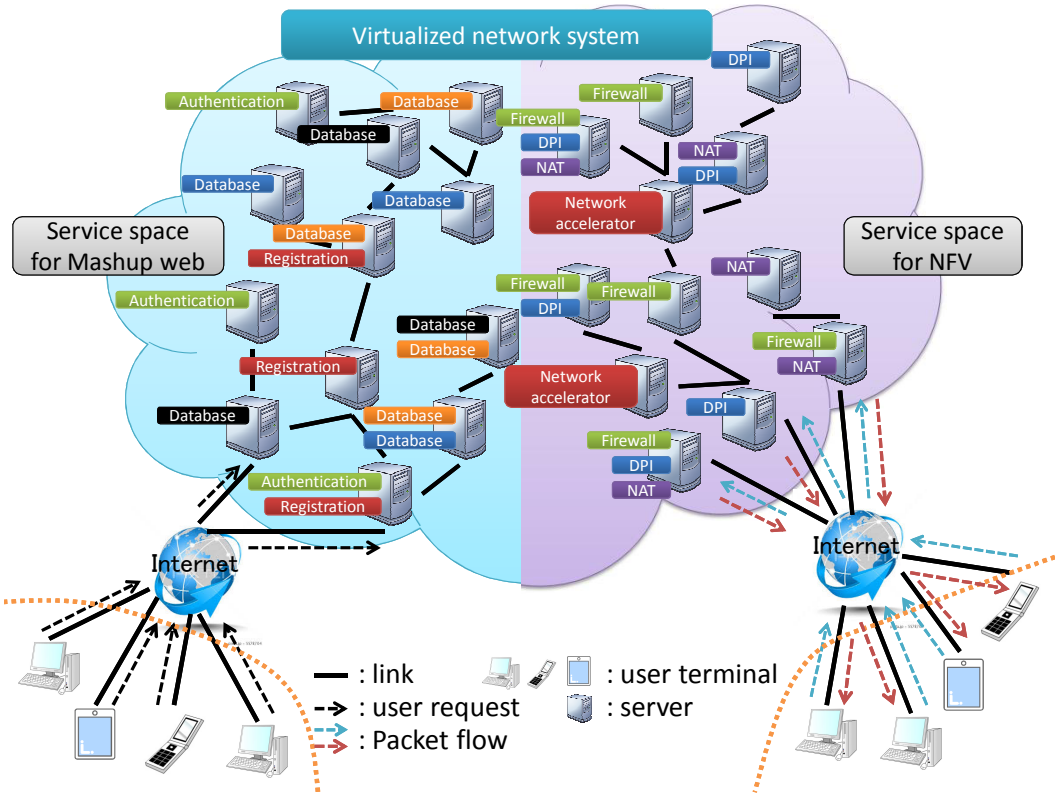

Figure 1: Virtualized network system for cloud computing environment

the same way that chemical substances would behave. Additionally, such chemical reactions are extended with a mechanism of tuple diffusion.

The authors in [18] proposed a chemical-inspired tuple space model as an enhanced version of the above model for supporting a system of pervasive services. In this model, the case study based on the pervasive and self-adaptive display infrastructure for the airport scenario is considered. In this case, a node, where a displayed installed, is modelled as a tuple space. Requests from users, services for users, and local information in the system component are represented as chemical substances. The concentration of the chemical substances means the amount of service demands and requests from users, and so on. In each tuple space, chemical reactions are defined to determine the behavior of the node, the display, and displayed services, such as competition among displayed services, the increase and decrease of the demand of the services, and their matching to user's preferences. Furthermore, multiple tuple space are connected to form a network of system components to represent the spatial coordination of the system, e.g. diffusion and movement of services and requests in the system. We believe that the above model can also be applied to constructing service space needs to determine services to be provided according to user requests with a spatial coordination.

In this thesis, we propose a construction method of service space based on chemical-inspired tuple space model to realize distributed execution of multiple services according to user requests and autonomous sharing server resources among these services. In the proposed method, a physical server in the virtualized network system is modelled as a tuple space. User requests, demands, and server resources are considered as chemical substances in the tuple space. Then, we define chemical reactions in the tuple space to represent various behaviors such as the execution of the services and the increase and decrease of the service demands and server resources according to the amount of user requests. Moreover, we construct a network of multiple tuple spaces to model the service space in virtualized network system, and represent the diffusion of the services and the movement of user requests.

We confirm the behavior of the proposed method through simulation experiments, in terms of distributed execution of services, autonomous sharing of server resource according to the amount of user requests, service relocation to appropriate servers in the network. Furthermore, we present an application scenario of the proposed method to Network Function Virtualization (NFV) architecture. For that purpose we extend the proposed model to accommodate the Network Function

Chaining (NFC), that is one of important services in NFV.

The rest of this thesis is organized as follows. In Section 2, we explain the chemical-inspired tuple space model and its applicability to the virtualized network system. In Section 3, we propose the construction method of service space. In Section 4, we show extensive simulation results to confirm the behavior of the proposed method. In Section 5, we explain the application of the proposed method to NFV architecture. Finally, in Section 6, we present a conclusion and some directions for future research.

2 Chemical-inspired tuple space model

2.1 Overview

In [18], a chemical-inspired tuple space model has been proposed on the idea of embodying the distributed state of a pervasive services in the network of system component. Figure 2 depicts the overview of the model. A system component is modelled as a tuple space, and multiple tuple spaces are connected to form a network as a whole system. In each tuple space, local information is described by chemical substances (corresponding to tuples in the original model [16]), and the amount of the information is represented by the concentration of corresponding chemical substance. Chemical reactions are set into the tuple space to describe behaviors of the system component. By conducting chemical reactions, we obtain the evolution of chemical substances in the tuple space over time.

In the next subsection, we present an application example of the model in the context of pervasive display infrastructure in airport scenario introduced in Section 1.

2.2 Spatial coordination by chemical reactions

Each display in the airport is modelled as a tuple space. At the tuple space, the following reactions are defined to describe the execution of a service for a request and the evolution and decay of the service in the tuple space.

$$SERV \mid REQ \xrightarrow{u} SERV \mid SERV \mid toserv(SERV, REQ) \quad (1)$$

$$SERV \xrightarrow{d} 0 \quad (2)$$

Substance $SERV$ represents the service identifier provided by the display. Substance REQ means a request for the display service from people nearby the display. Substance $toserv(SERV, REQ)$ is the information on actual display service for the request. Reaction (1) includes two behaviors in the system component. One is that a request for the service is processed by the display. Another is that the concentration of $SERV$ increases as a result of the request processing to represent the increase of service demand. On the other hand, Reaction (2) represents decay of the service.

The rate at which a reaction occurs in a tuple space is determined in proportion to the product of the concentrations of chemical substances in reactants of the reaction and the rate parameter (u

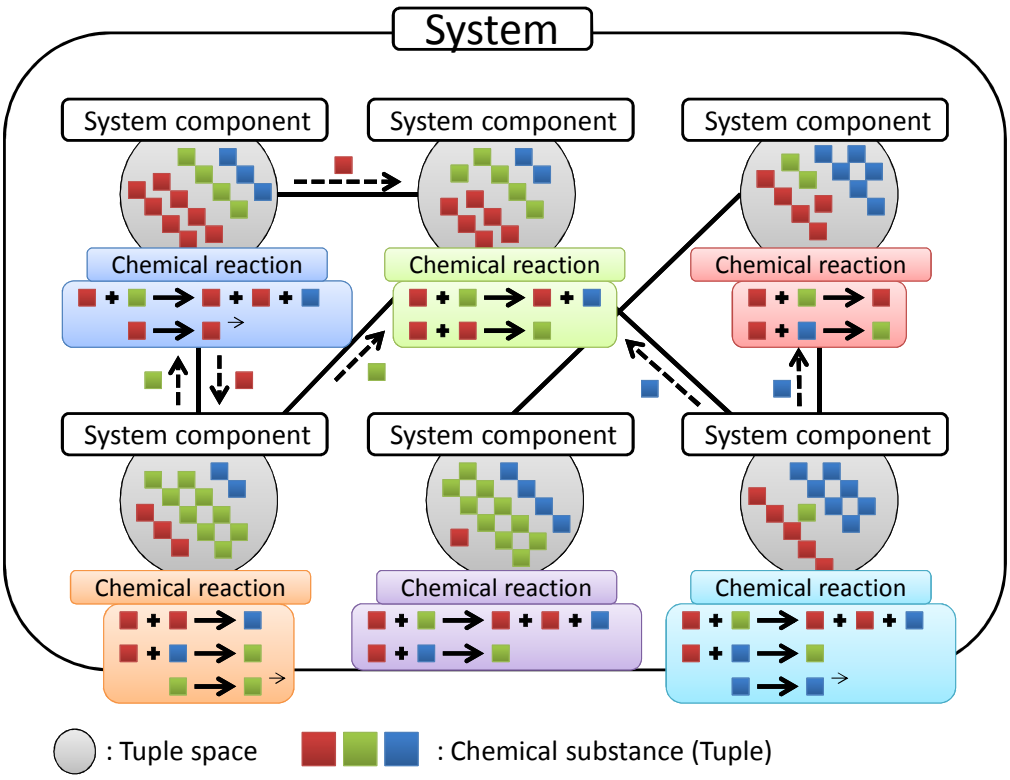

Figure 2: Overview of chemical-inspired tuple space model

and d in Reactions (1) and (2), respectively). Therefore, by these two reactions we can represent the evolution and decay of the service according to the amount of demand for the service.

Reaction (3) represents the diffusion of a service to other interconnected tuple spaces, where m means the rate parameter.

$$SERV \xrightarrow{m} SERV^{\rightsquigarrow} \tag{3}$$

By this reaction, substance *SERV* is diffused to surrounding tuple spaces at a rate proportional to its concentration. Combined with Reaction (1) and (2), we can describe the spatial evolution of the service according to the service demand.

The remaining four reactions are utilized to move substance *REQ* to tuple spaces where the corresponding service is provided.

$$PUMP \xrightarrow{p} PUMP \mid GRAD \tag{4}$$

$$GRAD \xrightarrow{d} 0 \tag{5}$$

$$GRAD \xrightarrow{m} GRAD^{\rightsquigarrow}(GRAD^{-}) \tag{6}$$

$$REQ \xrightarrow{m'} REQ^{\rightsquigarrow}(GRAD^{+}) \tag{7}$$

p, d, m, and m' are rate parameters for Reactions (4), (2), (3), and (7), respectively. Substance *PUMP* is inserted into a tuple space that provides the service. *GRAD* is a substance for establishing a gradient field in the system shown in Figure 3. Reaction (6) represents the diffusion of *GRAD* to surrounding tuple spaces that have less concentrations of *GRAD*. Therefore, the gradient field is constructed so that the tuple space providing the service becomes a summit with the largest concentrations of *GRAD*, and surrounding tuple spaces have smaller concentrations of *GRAD* according to the distance (e.g. hop count) from the summit. Reaction (7) describes the movement of *REQ* to tuple spaces with larger concentrations of *GRAD*, depicted in Figure 3.

2.3 Applicability to the virtualized network system

We consider that the above chemical-inspired coordination model can be applied to constructing a service space in the virtualized network system by the following analogies. System components in Figure 2 are replaced into servers that provide services. Substances *SERV* and *REQ* correspond to a service and a user request for the service, respectively. The network of the system components in

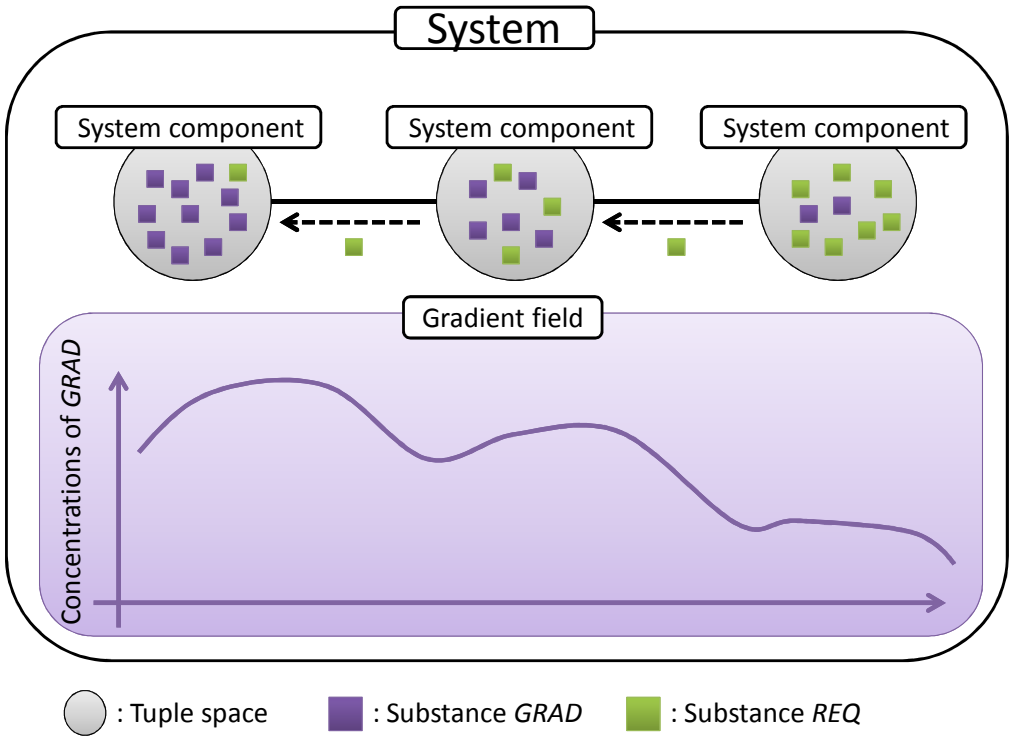

Figure 3: Gradient field and *REQ* movement along the gradient

Figure 2 can be constructed according to the network of the servers. By conducting chemical reactions in each server, we can realize distributed service execution, sharing server resource among multiple services, relocation of services to appropriate servers, and the movement of user request to appropriate servers. In the next section, we describe how the model is applied to constructing a service space in the virtualized network system in more detail.

3 Construction method of service space

We propose a construction method of service space in the virtualized network system by extending the chemical-inspired space coordination model in Section 2. In what follows we explain the chemical reactions for realizing various behaviors in each server of the service space in the virtualized network system.

3.1 Sharing server resources

We introduce the following reactions for service i in tuple space t to represent the execution of multiple services according to user requests.

$$SERV_i \mid REQ_i \mid CATAL_t \overset{u_{i,t}}{\rightarrow} MEDIATE_{i,t} \tag{8}$$

$$MEDIATE_{i,t} \overset{v_{i,t}}{\rightarrow} SERV_i \mid REQ_i \mid CATAL_t \tag{9}$$

$$MEDIATE_{i,t} \overset{w_{i,t}}{\rightarrow} SERV_i \mid SERV_i \mid CATAL_t \mid toserv(SERV_i, REQ_i) \tag{10}$$

$$SERV_i \overset{d_{i,t}}{\rightarrow} 0 \tag{11}$$

$u_{i,t}$, $v_{i,t}$, $w_{i,t}$, and $d_{i,t}$ means the rates parameters for Reactions (8), (9), (10), and (11), respectively. Substance $SERV_i$ represents an identifier of service i. The existence of this substance in a tuple space means that the corresponding server can execute service i. Substance REQ_i means a request for service i from users. Substance $toserv(SERV_i, REQ_i)$ expresses the information on actual execution of service i for the request.

As described in the previous section, the reaction rate is determined in proportion to the product of the concentrations of reactants in the reaction. It means that in Reaction (1), when the concentrations of $SERV$ and REQ increase, the reaction rate increases without any limitation. This is not suitable for service spaces in the virtualized network system since a server has limited amount of resources such as processing performance and memory size. Therefore, we exploit Michaelis-Menten mechanism [19], that explains enzyme-catalyzed reactions [20] in the context of chemical kinetics. In enzyme-catalyzed reaction, the reaction rate is controlled by the concentrations of catalyst while the catalyst itself does not affect the reaction [21]. The authors in [19] proposed a model for determining the rate of the enzyme-catalyzed reaction by introducing an enzyme-substrate complex into the reaction.

Based on the mechanism in [19], we extend Reaction (1) into Reactions (8), (9), and (10). Substance $CATAL_t$ means the catalyst in tuple space t, that corresponds to available server resources in a server. Substance $MEDIATE_{i,t}$ is the complex, whose concentration means the amount of resources in the tuple space t utilized for service i. Therefore, we configure the initial concentration of $CATAL_t$ in tuple space t according to the resource amount in the corresponding server.

3.2 Diffusion of services

We add the following reaction to realize the diffusion of services to other interconnected servers.

$$SERV_i \overset{m_{i,t}}{\rightarrow} SERV_i^{\rightsquigarrow} \tag{12}$$

$m_{i,t}$ means the rate parameter. By this reaction, $SERV_i$ is diffused from tuple space t to surrounding tuple spaces. When the destination tuple space of the diffusion has requests for the service, they are processed according to Reactions (8), (9), and (10), resulting the evolution of the service at the tuple space. On the other hand, when the destination tuple space has no corresponding request, the service would decay according to Reaction (11).

3.3 Movement of user requests

When a request in a tuple space can not find the corresponding service, it should move to another tuple space (e.g. server) to meet the service. Furthermore, the moving direction should be determined based on the remaining server resources and the service demand in each tuple space.

We introduce the following chemical reactions to describe these behaviors.

$$SERV_i \mid CATAL_t \overset{g_{i,t}}{\rightarrow} SERV_i \mid CATAL_t \mid GRAD_i \tag{13}$$

$$GRAD_i \overset{h_{i,t}}{\rightarrow} 0 \tag{14}$$

$$GRAD_i \overset{k_{i,t}}{\rightarrow} GRAD_i^{\rightsquigarrow}(GRAD_i^{-}) \tag{15}$$

$$REQ_i \overset{n_{i,t}}{\rightarrow} REQ_i^{\rightsquigarrow}(GRAD_i^{+}) \tag{16}$$

$g_{i,t}$, $h_{i,t}$, $k_{i,t}$, and $n_{i,t}$ are the rate parameters for Reactions (13), (14), (15), and (16), respectively. We extend Reaction (4) into Reaction (13) so that we construct the gradient field for service i according to the remaining resource of the server and the service demand, instead of just using the existence of the service.

4 Performance evaluation

In this section, we confirm the behavior of the proposed method through simulation experiments.

4.1 τ-leaping algorithm

We exploit τ-leaping algorithm [22] to simulate the behavior of the proposed method. It is one of stochastic simulation algorithms [23], which can capture the inherent stochasticity in many biochemical systems such as the chemically reacting system and the cell system. We briefly explain the procedures of τ-leaping algorithm for a tuple space in the proposed method as follows.

- Step 1: Set τ for the time step of the simulation
- Step 2: Calculate the reaction rates of chemical reactions in the tuple space
- Step 3: Generate a Poisson random variable for each chemical reaction whose mean is the product of corresponding reaction rate and time τ
- Step 4: Execute chemical reactions at the number of times in Step 3
- Step 5: Progress simulation time by τ
- Step 6: Turning back to Step 2

In [24], the authors describe that simulation with τ-leaping algorithm can be carried out more faster than other stochastic simulation algorithms such as Gillespie's algorithm [25], without deteriorating the quality of the simulation result. We also believe that τ-leaping algorithms is suitable for parallel simulation of multiple tuple spaces with interactions among them, that is required for the evaluation of the proposed method.

4.2 Simulation scenarios

We utilize the topology of the backbone network of Science Information Network (SINET) [26] in Japan. SINET provides the academic backbone network that is constructed and operated by National Institute of Informatics (NII) for academic information infrastructure [27] of Japanese universities and research institutions. We depict the network topology for the performance evaluation in Figure 4, where nodes are located on major cities in Japan. We assume that there is one

server at each node with limited resource amount to provide service(s). User requests for the services are generated on the nodes, assuming that the requests come from the network of universities and institutes connecting to the node.

We evaluate the performance of the proposed method by using the following four scenarios. Note that all node has 1,000 of the initial concentration of $CATAL$, that corresponds to the capacity of serving roughly 50 requests per unit time.

- Scenario 1: We locate service 1 and service 2 to node 1 by initially setting concentrations of $SERV_1$ and $SERV_2$ to 2,000. User requests for services 1 and 2 are generated by injecting 10 and 30 of the concentrations of REQ_1 and REQ_2 to node 1 per unit time, respectively. By this scenario we show the server resource sharing among services according to the amount of requests for them.

- Scenario 2: We locate service 1 to node 8 by initially setting concentrations of $SERV_1$ to 3,000. User requests for service 1 are generated by injecting 40 of the concentrations of REQ_1 to node 1 per unit time. We exhibit the relocation of the service to an appropriate node.

- Scenario 3: Initial service location is identical to Scenario 3. However, the request rate for service 1 at node 1 is increased from 40 to 80. Since 80 requests per unit time can not be served by a single server, the load sharing among surrounding nodes would occur in this scenario.

- Scenario 4: We locate service 1 to node 6 by initially setting concentrations of $SERV_1$ to 1,000. User requests for service 1 are generated by injecting 80 of the concentrations of REQ_1 to node 3 per unit time. Furthermore, we limit the diffusion area of substance $SERV_1$ to nodes 4, 5, and 6. By this scenario we present the load balancing among three servers to serve the requests.

4.3 Simulation results and discussions

In Figure 5, we plot the concentration changes of $SERV$ for each service on node 1 in Scenario 1. We can see from this figure that the concentrations of $SERV$ for each service converge at constant values according to the injection rate of REQ for both services. This result means that

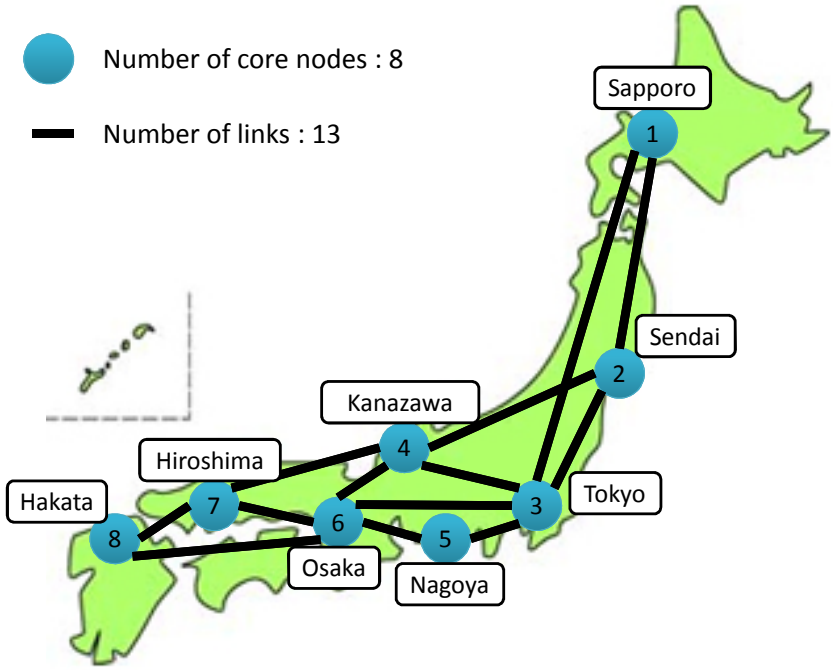

Figure 4: SINET backbone network topology

service demand on a node for each service is determined according to user requests injected to the node. Figure 6 presents the changes in the concentrations of $MEDIATE$ for each service and $CATAL$ on node 1 in Scenario 1. We can confirm from this figure that the concentrations of $MEDIATE$ for each service also converge at constant values according to the injection rate of REQ for both services. Furthermore, we can see that the concentration of $CATAL$ decrease according to increasing the concentrations of $MEDIATE$s. This result means that limited resources are shared among two services. In Figure 7, we plot the generation rate of $toserv(SERV, REQ)$ for each service on node 1. This result means that the requests for both services are processed at almost identical rate to their injection rates. We also confirmed that almost no REQ_1 and REQ_2 moves to other nodes from node 1 because node 1 has both much service demand and enough resources to process all injected requests and it becomes a summit with the largest concentrations of $GRAD$ as shown in Figure 8, that exhibits the gradient fields of $SERV_1$ and $SERV_2$.

In Figure 9, we plot the concentration changes of $SERV_1$ on nodes 8, 3, and 1 in Scenario 2. These results clearly show that $SERV_1$ moves from node 8 to node 1, where user requests for the service are injected. Figure 10 plots the gradient field for the service at 200 th time step, 400 th time step, and 1,500 th time step. We can see from these figure that the gradient field is generated according to the movement of $SERV_1$. We also plot the moving rate changes of REQ_1 on node 1 in Figure 11. We can explain in this figure that user requests firstly can not be processed in node 1 so that these requests move to other nodes which can process them. However, since $SERV_1$ moves to node 1 over time, all user requests injected in node 1 are processed in node 1.

For Scenario 3 we plot the average generation rate of $toserv(SERV_1, REQ_1)$ between 500 th to 2,000 th time step. This result exhibits that the injected requests are served at nodes 1, 2, and 3 and the service rate of node 1 is highest since the requests are injected to node 1.

Finally, we show the evaluation results of Scenario 4. Figure 13 show the average generation rate of $toserv(SERV_1, REQ_1)$ from 500th to 2,000th time step. We can observe from this figure that the requests injected to node 3 are distributed to nodes 4, 5, and 6 for load balancing. However, The service rate of node 6 is slightly larger than that of nodes 4 and 5. This is because of the network topology among nodes 4, 5, and 6. Since node 6 has two links to nodes 4 and 5, while nodes 4 and 5 only has a link to 6, the concentration of $SERV_1$ at node 6 becomes higher that nodes 4 and 5 as a result of the diffusion of $SERV_1$. Note that we have confirmed that when nodes 4 and 5 have an additional interconnected link, the service rates of nodes 4, 5, and 6 become identical.

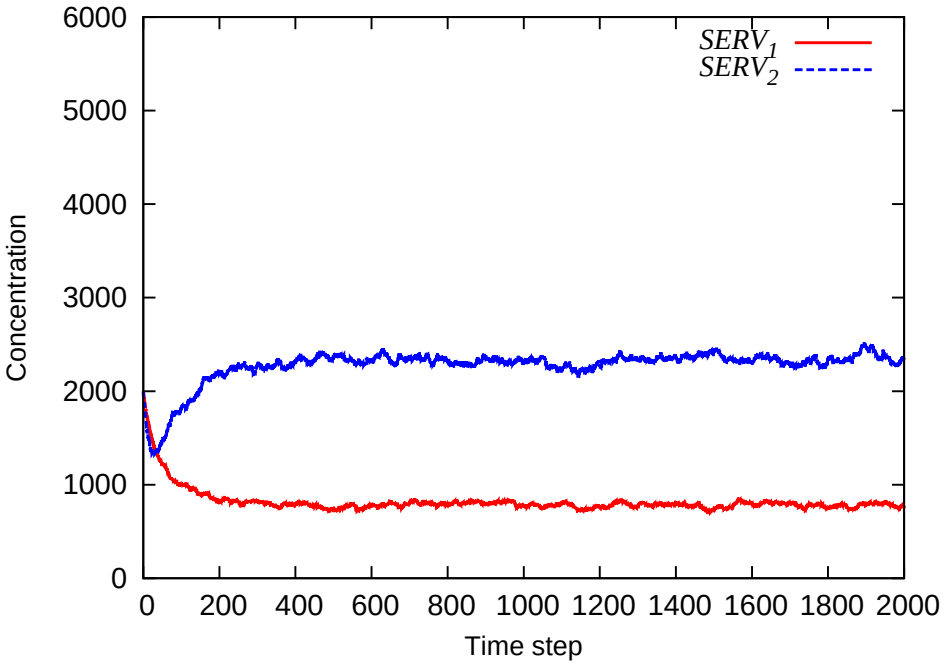

Figure 5: Scenario 1: Concentration changes of *SERV*s on node 1

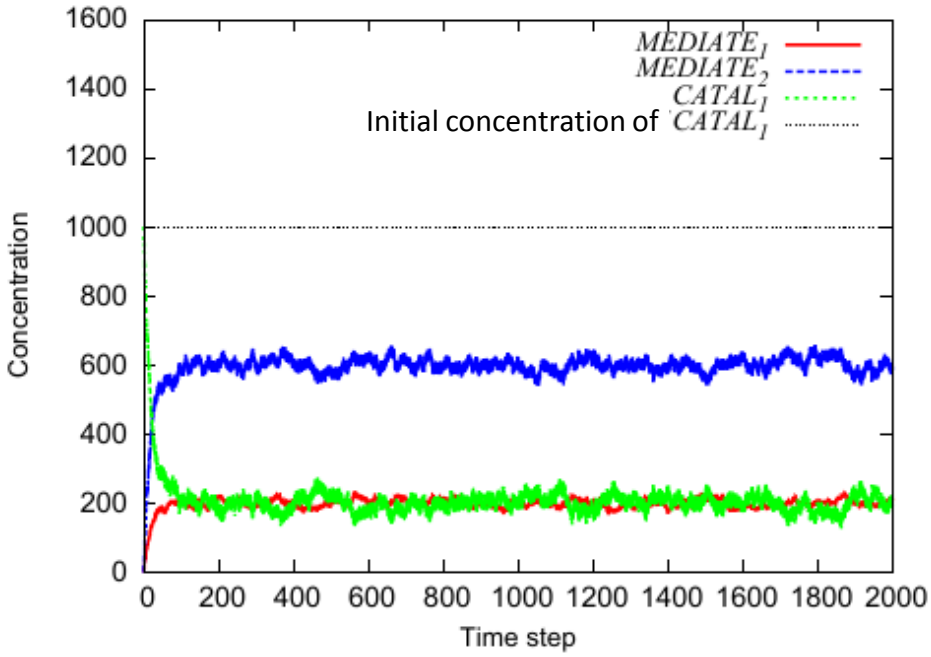

Figure 6: Scenario 1: Concentration changes of *MEDIATE*s and *CATAL* on node 1

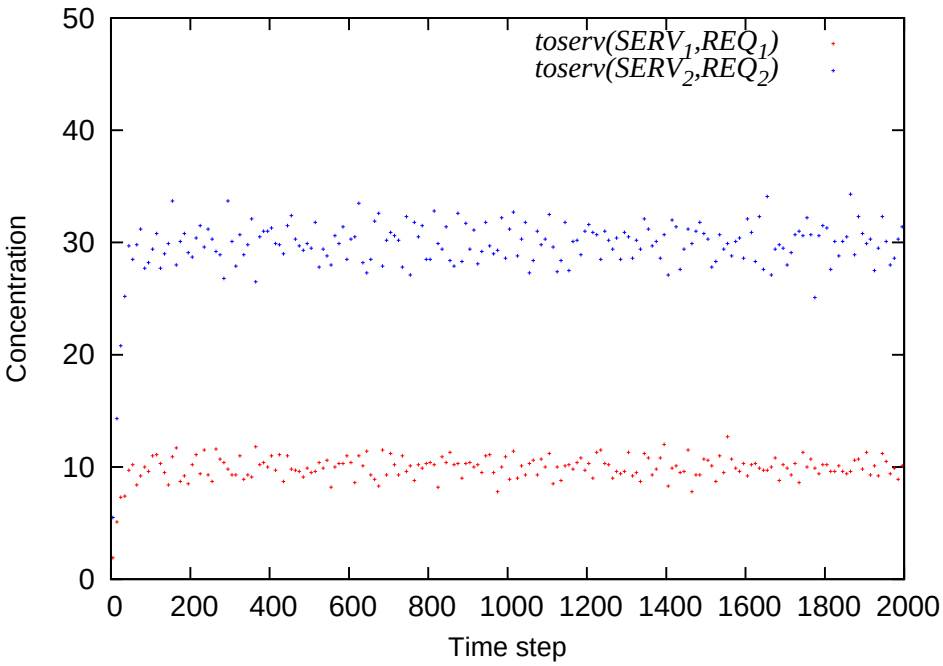

Figure 7: Scenario 1: Generation rate of $toserv(SERV, REQ)$s on node 1

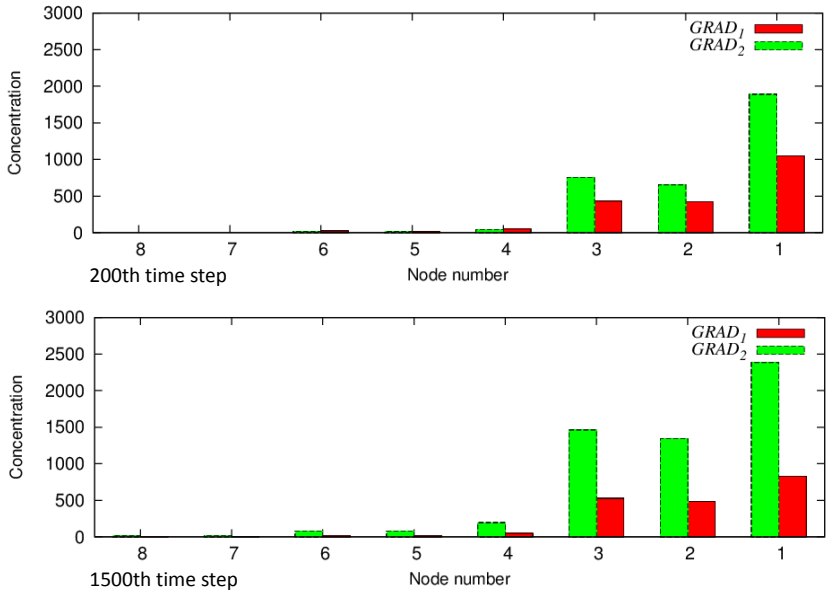

Figure 8: Scenario 1: Gradient field for services at 200 th time step (top) and 1,500 th time step (bottom)

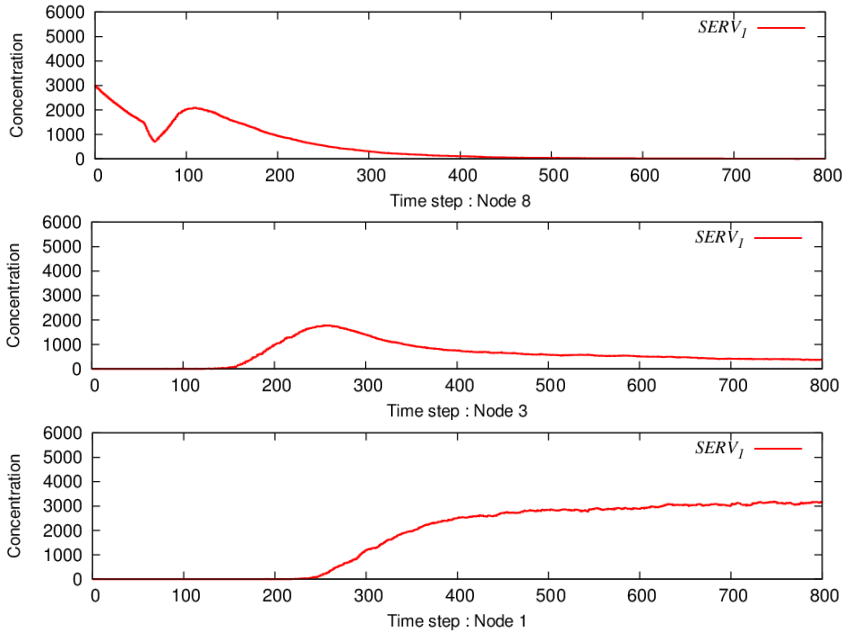

Figure 9: Scenario 2: Concentration changes of $SERV$ on nodes 8 (top), 3 (middle), and 1 (bottom)

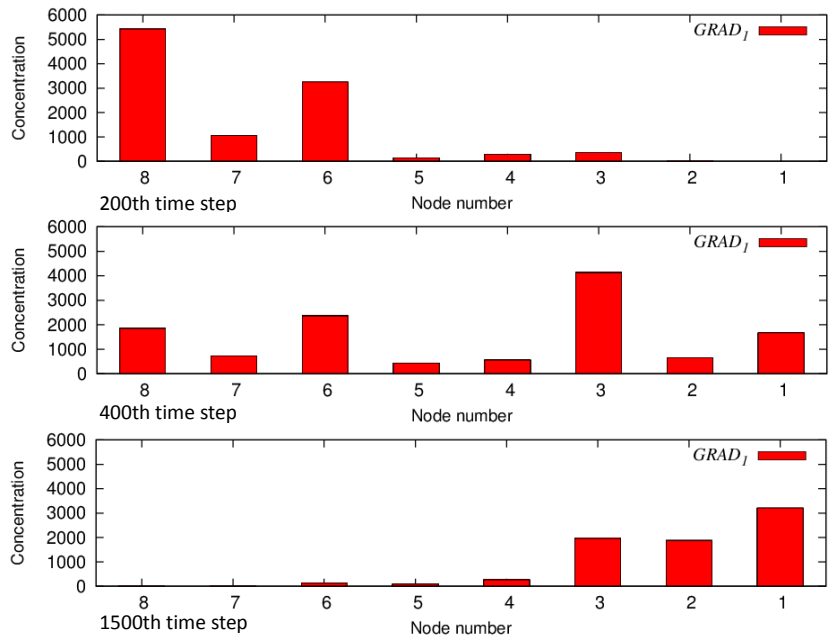

Figure 10: Scenario 2: Gradient field at 200 th time step (top), 400 th time step (middle), and 1,500 th time step (bottom)

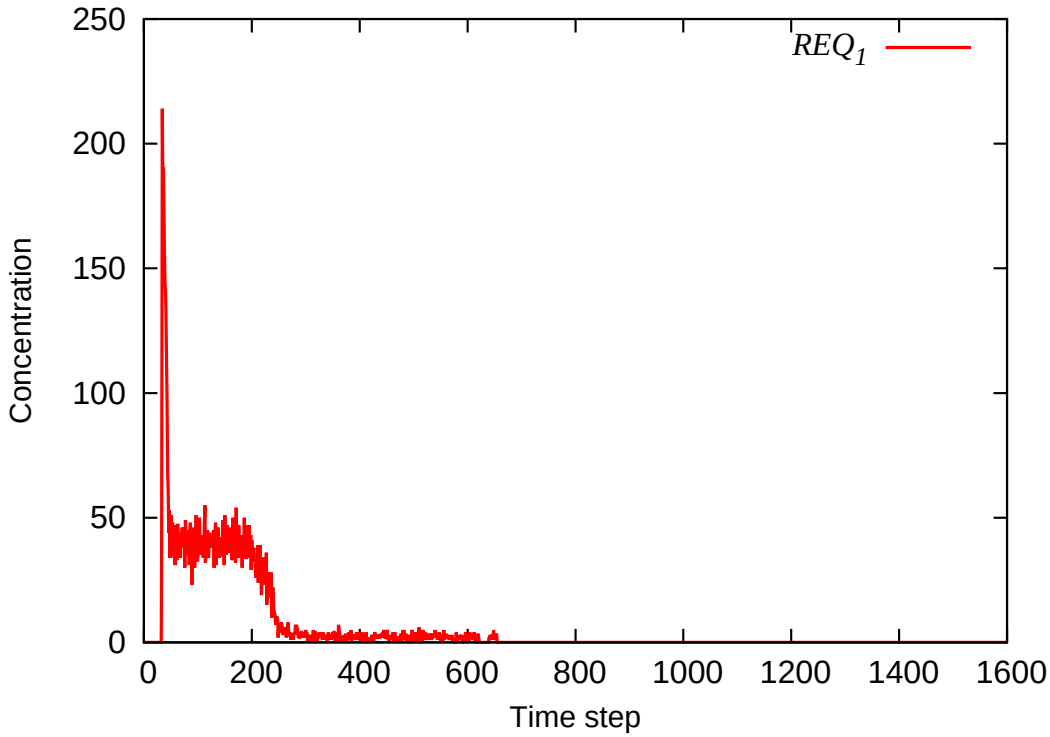

Figure 11: Scenario 2: Moving rate changes of REQ_1 on node 1

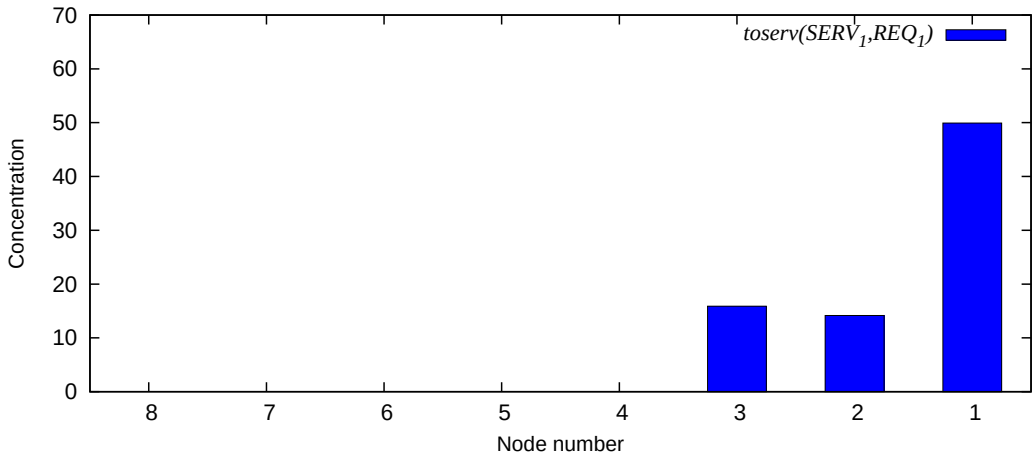

Figure 12: Scenario 3: Average generation rate of $toserv(SERV_1, REQ_1)$ from 500 th to 2,000 th time step

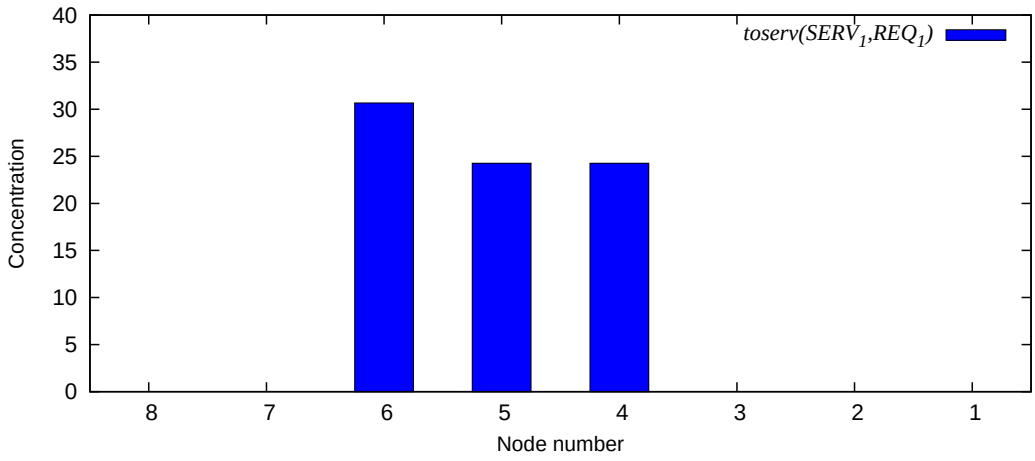

Figure 13: Scenario 4: Average generation rate of $toserv(SERV_1, REQ_1)$ from 500 th to 2,000 th time step

5 Application to Network Function Virtualization

In this section, we present the applicability of the proposed method to Network Function Virtualization (NFV) and describe the extension of the proposed method for accommodating NFV scenario.

5.1 Overview

We consider that proposed method is suitable for NFV, by which network functions are virtualized and they are executed on servers so as to simplify to share and relocate network functions. Typically, network flows go through several network functions. These flows traverse these network functions in specific order so that the required functions are applied to the flows. The notion is known as Network Function Chaining (NFC) or Network Service Chaining (NSC) [28,29]. Therefore, it is important for NFV that we decide at which server each network function is located and on which route flows go through.

In [30], the authors proposed the method which determines the location of network functions for NFV according to chaining requests of all network flows. However, the method requires that chaining requests of all network flows must be known in advance and it takes large calculation time to obtain the placement of all required functions in the network. Accordingly, we can apply the proposed method in Section 4 to placing function in NFV architecture so that we can determine where to apply each network function in chaining request to network flow and location of network functions in an autonomous distributed fashion.

5.2 Extension of proposed method for NFV

We define network functions applied to a certain flow as $f_1, f_2, ...$ We also define the first and last functions to be processed on the flow as f_{start} and f_{end}, respectively. To represent the chaining request, we introduce the following notation that describes the order of the functions to be applied to a flow.

$$c = \{f_{start}, f_1, f_2, f_3, ..., f_{end}\}$$

When a flow with a chaining request of $c = \{f_1, f_2, f_3, ..., f_{end}\}$ is processed by a network function f_1, the chaining request changes as follows.

$$c \leftarrow c \backslash \{f^1\} = \{f_2, f_3, .., f_{end}\}$$

Moreover, we denote the network function by which a chining request c should be nextly processed as $f^1(c)$ (e.g. in case $c_k = \{f_1, f_2, f_3\}$, $f^1(c) = f_1$). We then introduce the following reactions so that a server provides NFV functions, where $F = \{f_1, f_2, ...\}$, and C is a set of all chaining requests in the network.

$$\begin{aligned}
\forall c \in C, \forall f \in F \ \mathit{SERV}_f \mid \mathit{FLOW}_c \ &\overset{r_{us}}{\rightarrow} \ \mathit{SERV}_f \mid \mathit{SERV}_f \\
&\quad \mid \mathit{FLOW}_{c \backslash \{f^1(c)\}} \qquad (17) \\
&\quad \mid \mathit{toserv}(\mathit{SERV}_f, \mathit{FLOW}_c) \\
\forall f \in F \ \mathit{SERV}_f \ &\overset{r_{ds}}{\rightarrow} \ 0 \qquad (18) \\
\forall f \in F \ \mathit{SERV}_f \ &\overset{r_{df}}{\rightarrow} \ \mathit{SERV}_f{}^{\rightsquigarrow} \qquad (19) \\
\forall c \in C \ \mathit{FLOW}_c \ &\overset{r_{as}}{\rightarrow} \ \mathit{FLOW}_c{}^{\rightsquigarrow}(\mathit{GRAD}_{f^1(c)}{}^{+}) \qquad (20) \\
\forall f \in F \ \mathit{SERV}_f \ &\overset{r_{rg}}{\rightarrow} \ \mathit{SERV}_f \mid \mathit{GRAD}_f \qquad (21) \\
\forall f \in F \ \mathit{GRAD}_f \ &\overset{r_{dg}}{\rightarrow} \ 0 \qquad (22) \\
\forall f \in F \ \mathit{GRAD}_f \ &\overset{r_{mg}}{\rightarrow} \ \mathit{GRAD}_f{}^{\rightsquigarrow}(\mathit{GRAD}_f{}^{-}) \qquad (23)
\end{aligned}$$

Substance SERV_f represents a function f and substance FLOW_c means a flow which has a chaining request c. Substance $\mathit{toserv}(\mathit{SERV}_f, \mathit{FLOW}_c)$ expresses that a flow has chaining request c is processed. In Reaction (17), FLOW_c is replaced by $\mathit{FLOW}_{c \backslash \{f^1(c)\}}$ for remaining functions in the chaining request to be processed, that is different from Reaction (1) where REQ is removed after being processed. Therefore, when a flow which has only f_{end} in its chaining request is processed, FLOW_c is removed. Reactions (18), (19), (20), (21), (22), and (23) are corresponding to Reaction (11), (12), (16), (13) ,(14), and (15) in proposed method, respectively. These reactions are required for routing flows to appropriate servers for functions to be processed.

6 Conclusion and Future work

In this thesis, we proposed a construction method of service space in virtualized network system based on the chemical-inspired tuple space model. the proposed method construct a service space in virtualized network system by autonomous distributed spatial coordinate behavior between each server in virtualized network system, where the server resource sharing among services provided by a server, the load balancing among surrounding servers on various situations, service relocation, and request forwarding to appropriate servers can be realized. We have confirmed the behavior of the proposed method by presenting extensive simulation results. Furthermore, we also explained the application of the proposed method to NFV architecture and present that the proposed method can be applied to actual application by extension of the method.

For future work, we plan to simulate the proposed method on various scenarios which include in system failure and explain the effectiveness of the proposed method. We also need to conduct the mathematical analysis of proposed method to reveal the fundamental characteristics of the proposed method. Furthermore, we should confirm the behavior of the extended model for NFV architecture shown in Section 5 by simulation experiments.

Acknowledgement

I would like to express my greatest gratitude to Professor Morito Matsuoka of Osaka University. His helpful comments and suggestions were invaluable during the course of my study.

Special thanks also go to Associate Professor Go Hasegawa of Osaka University. ALL works of this thesis also would not have been possible without his appropriate guidance and empathic advice. I am most grateful to him that he always helped me.

I am deeply grateful to Professor Masayuki Murata of Osaka University, for his excellent guidance and continuous support through my studies of this thesis.

Finally, I want to give thanks to my friends and colleagues in the Department of Information Networking of the Graduate School of Information Science and Technology of Osaka University for their support. Our conversations and work together have greatly influenced this thesis.

References

[1] J. Yang and M. P. Papazoglou, "Web component: A substrate for web service reuse and composition," *Lecture Notes in Computer Science*, Vol. 2348, pp. 21–36, May 2002.

[2] J. Carapinha and J. Jimenez, "Network virtualization: View from the bottom," in *Proceedings of VISA 2009*, pp. 73–80, June 2009.

[3] D. King and C. Ford, "A Critical Survey of Network Functions Virtualization," in *Proceedings of the Accounting and Finance 2013*, pp. 1–21, July 2013.

[4] R. lyer, R. Illikkal, O. Tickoo, L. Zhao, P. Apparao, and D. Newell, "VM: Measuing, modeling and managing VM shared resources," *Computer Networks*, Vol. 53, pp. 2873–2887, Apr. 2009.

[5] V. Nollet, T. Marescaux, P. Avasare, and D. Verkest, "Centralized run-time resource management in a network-on-chip containing reconfigurable hardware tiles," in *Proceedings of Automation and Test in Europe 2005*, pp. 234–239, Mar. 2005.

[6] C. Kesselman, C. Lee, and B. Lindell, "A distributed resource management architecture that supports advance reservations and co-allocation," in *Proceedings of IWQos 1999*, pp. 27–36, June 1999.

[7] A. G. Tsikalakis and N. D. Hatziargyrou, "Centralized control for optimizing microgrids operation," in *Proceedings of the Power and Energy Society General Meeting 2011*, pp. 24–29, July 2011.

[8] M. Viroli, M. Casadei, and Matteo, "Chemical-inspired self-composition of competing services," in *Proceedings of the ACM Symposium on Applied Computing 2010*, pp. 2029–2036, July 2006.

[9] S. Balasubramaniam, K. Leibnitz, P. Lio, D. Botvich, and M. Murata, "Biological principles for future Internet architecture design," *IEEE Communications Magazine*, Vol. 49, pp. 44–52, July 2011.

[10] M. Meisel, V. Pappas, and L. Ahang, "A taxonomy of biologically inspired research in computer networking," *Computer Networks*, Vol. 54, pp. 901–916, Apr. 2010.

[11] T. Wilhelm, "The smallest chemical reaction system with bistability," *BCM System Biology*, Vol. 3, pp. 1–9, Sept. 2009.

[12] F. Dressler and O. B. Akan, "Bio-inspired networking: From theory to practice," *IEEE Communications Magazine*, Vol. 48, pp. 176–183, Nov. 2010.

[13] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki, "Rules for biologically inspired adaptive network design," *Science*, Vol. 327, pp. 439–442, Jan. 2010.

[14] W. Ma, A. Trusina, H. El-samad, W. A. Lim, and C. Tang, "Defining network topologies that can achieve biochemical adaptation," *ScienceDirect*, Vol. 138, pp. 760–773, Aug. 2009.

[15] N. Barkai and S. Leibler, "Robustness in simple biochemical networks," *Letters to Nature*, Vol. 387, pp. 913–917, Apr. 1997.

[16] M. Viroli and M. Casadei, "Biochemical tuple spaces for self-organising coordination," Vol. 5521, pp. 143–162, June 2009.

[17] M. Bravetti, R. Gorrieri, R. Lucchi, and G. Zavattaro, "Quantitative information in the tuple space coordination model," *Theoretical Computer Science*, Vol. 346, pp. 28–57, Nov. 2005.

[18] M. Viroli, M. Casadei, and S. Montagna, "Spatial coordination of pervasive services through chemical-inspired tuple spaces," *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 6, pp. 14:1–14:24, June 2011.

[19] M. L., M. ML., J. KA., and G. RS., "The original Michaelis constant: Translation of the 1913 Michaelis-Menten paper," *Biochemistry*, Vol. 50, pp. 8264–8269, Sept. 2011.

[20] R. N. Goldberg, Y. B. Tewari, and T. N. Bhat, "Thermodynamics of enzyme-catalyzed reactions," *Science and Mathematics, Bioinformatics*, Vol. 20, pp. 2874–2877, May 2004.

[21] W. W. Cleland, "The kinetics of enzyme-catalyzed reactions with two or more substrates or products," *ScienceDirect*, Vol. 67, pp. 173–187, May 1962.

[22] D. T. Gillespie, "Stochastic simulation of chemical kinetics," *Annual Review of Physical Chemistry*, Vol. 58, pp. 35–55, Oct. 2006.

[23] D. T. Gillespie, “Stochastic simulation of chemical kinetics,” in *Proceedings of Review in Advance 2006*, pp. 35–55, Oct. 2006.

[24] H. Li, Y. Cao, and D. T. Gillespie, “Algorithms and software for stochastic simulation of biochemical reacting systems,” *Biotechnology Progress*, Vol. 24, pp. 56–61, Feb. 2008.

[25] C. V. Rao and A. P. Arkin, “Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the Gillespie algorithm,” *Chemical Physics*, Vol. 118, pp. 1–13, Oct. 2003.

[26] H. Takaba, K. Wakamatsu, and M. Yoshida, “Development of the gigabit real-time VLBI using the super-SINET,” in *Proceedings of Petrologist and Economic Geologists 2004*, pp. 6–12, Dec. 2006.

[27] B. Galen and L. Robert, “Advancing the academic information infrastructure,” *Research on Compuring in Education*, Vol. 25, pp. 464–472, Feb. 2014.

[28] F. Schneider, T. Egawa, S. Schaller, S. Hayano, M. Scholler, and F. Zdarsky, “Standardizations of SDN and ITs practical implementation,” *NEC Technical Journal*, Vol. 8, pp. 16–20, Apr. 2014.

[29] P. Quinn and T. Nadeau, “Service function chaining problem statement.” Active Internet-Draft, IFTF Secretariat, Aug. 2014.

[30] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network Functions,” *Networking and Internet Architecture*, Vol. 1058, pp. 1–7, June 2014.