Master's Thesis

Title

Biochemically-inspired, adaptive, and autonomous VNF control for service function chaining

Supervisor

Professor Morito Matsuoka

Author

Ryota Kurokawa

February 6th, 2019

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Biochemically-inspired, adaptive, and autonomous VNF control
for service function chaining

Ryota Kurokawa

Abstract

In Network Function Virtualization (NFV), various Virtual Network Functions (VNFs) are deployed on general-purpose servers. A flow receiving NFV service may have a Service Function Chaining (SFC) request that describes the order of VNFs to be applied to the flow. Therefore, to efficiently operate the NFV system, placement of VNFs on servers, resource allocation to each VNF, and flow routes are determined adaptively. Furthermore, to quickly respond to environmental fluctuations, and to maintain the scalability of the NFV services, a distributed control is more feasible than a centralized one. One way to achieve such behaviors is to exploit a biochemical mechanism with autonomous dispersibility and self organization.

Our research group has proposed a construction method of service space in virtualized network system based on biochemically-inspired tuple space model. In this method, the behaviors in the virtualized network system are described by biochemical reactions in tuple spaces. Since biochemical reaction equations are defined and executed independently in each tuple space, it is suitable for achieving autonomous and decentralized behaviors. To operate the NFV system, the above method has been extended to handle flow routes in accordance with SFC requests, and server resource limitation. The basic behaviors of the extended method have been confirmed with computer simulation. However, the evaluation assuming various situations in the NFV system has not been performed. In addition, the method has been implemented and evaluated only in a simple experimental environment, so the applicability to the actual NFV system has not been clearly shown.

In this thesis, we assess the performance of the NFV system based on the service space construction method, and show its implementation design. We first explain chemical substances and

biochemical reaction equations required for applying the method to the NFV system. We then perform computer simulation experiments to clarify that the proposed method can cope with various situations in the NFV system, such as time variation of traffic amount and a sudden network failure. We finally present the implementation design of the proposed method based on the existing NFV framework. In detail, we show the detailed implementation environment using Open Platform for NFV, one of major implementations of the NFV framework with open source softwares. We also present the implementation of SFC using Network Service Header, proposed by Internet Engineering Task Force.

Keywords

Network Function Virtualization (NFV)
Service Function Chaining (SFC)
Biochemical Mechanism
Tuple Space Model
Network Service Header (NSH)

Contents

1 Introduction **7**

2 Related Work **11**

3 VNF Control based on Tuple Space Model with Biochemical Reactions **14**

3.1 Tuple Space Model 14

3.2 Application to NFV System 16

3.2.1 Resource Allocation and Execution of VNFs 16

3.2.2 Diffusion of VNFs 19

3.2.3 Packet Forwarding 20

3.2.4 Coexistence of Multiple VNFs 21

4 Simulation Experiments **23**

4.1 τ-Leaping Method 23

4.2 Common Parameter Settings 24

4.3 Scenario 1: Placement of VNFs Considering Flow Priorities 24

4.3.1 Application Scenario 24

4.3.2 Network Topology and Parameter Settings for Simulation Experiments . 26

4.3.3 Simulation Results and Discussion 28

4.4 Scenario 2: Route changes and VNF migrations on network failures 31

4.4.1 Application Scenario 31

4.4.2 Network Topology and Parameter Settings for Simulation Experiments . 33

4.4.3 Simulation Results and Discussion 35

5 Implementation Design of the Proposed Method with the NFV Framework **39**

5.1 NFV Framework and its Integration with SDN 39

5.2 Positioning of the Proposed Method 39

5.3 Implementation Environment 41

5.4 Handle of Service Function Chaining 43

5.4.1 Utilization of Network Service Header 43

5.4.2 Stochastic Selection of Flow Routes 45

6 Conclusion and Future Work **49**

Acknowledgments **50**

Reference **51**

List of Figures

1 NFV system 8
2 Tuple space model using biochemical reaction 15
3 Application of tuple space model to NFV system 17
4 Movement of packets in accordance with gradient fields 22
5 Scenario1: Placement of VNFs considering flow priorities 25
6 Scenario1: Network topology for simulation experiments 27
7 Scenario1: Average number of executions of Reaction Equation (4) 29
8 Scenario1: Temporal change in the concentrations of $RSRC$ at node 0 and node 1 30
9 Scenario2: Route changes and VNF migrations on network failures 32
10 Scenario2: Network topology for simulation experiments 34
11 Scenario2: Average number of executions of Reaction Equation (4) 36
12 Scenario2: Temporal change in the concentrations of VNF 37
13 Scenario2: Temporal change in the concentrations of $RSRC$ at all nodes 38
14 NFV framework and its integration with SDN 40
15 System configuration of OPNFV 42
16 Network Service Header (NSH) 44
17 Implementation design of SFC using NSH in RFC 8300 44
18 Implementation design of SFC using NSH 46
19 Stochastic determination of flow route with the proposed method 48

List of Tables

1 Comparison of VNF placement methods 13

2 Correspondence between tuple space model and NFV system 17

3 Scenario1: Temporal change in rate of flows 27

4 Scenario2: Temporal change in rate of flows 34

5 Flow entries for handling NSH 46

1 Introduction

Due to the wide and rapid spread of smartphones and tablets, and the development of Internet of Things (IoT) [1], the number of devices connected to the network is increasing. As a result, network services have become more diverse, and network traffic has also increased rapidly. In general, to launch a new network service, new dedicated hardware devices are required. It requires space and power to accommodate these devices, causing a decrease in revenue and an increase in energy consumption. In addition, due to the continuous development and expansion of network services, product life of dedicated hardware devices has been shortened, causing an increase in capital expenditures. It also results in low flexibility to deal with system failures, maintenance and operation of hardware.

Network Function Virtualization (NFV) is considered as one possible technique for resolving such problems [2]. In NFV, network functions on dedicated hardware are achieved by software, and deployed and executed on general-purpose servers. The network functions achieved by software are called Virtual Network Functions (VNFs). Typical VNFs include Firewall [3], Network Address Translation (NAT) [4], Intrusion Detection System (IDS) [5] and Evolved Packet Core (EPC) [6, 7]. Figure 1 shows an NFV system. In NFV, multiple VNFs may share the resource on a single server or one VNF may be distributed to multiple servers to provide services throughout the network [8, 9]. As a result, it is possible to suppress operational and capital expenditures by aggregating physical servers. It is also possible to flexibly respond to environmental fluctuations by reallocating server resources to VNFs, migrating VNFs, and rerouting flow packets.

A flow receiving NFV service may have a Service Function Chaining (SFC) request that describes the order of VNFs to be applied to the flow. In Figure 1, a flow arriving at the NFV system receives NFV services in accordance with the SFC request and exits the system. Therefore, to efficiently operate the NFV system, placement of VNFs to servers, resource allocation to each VNF, and flow routes are determined adaptively in accordance with the SFC requests, traffic amount of the flows, and amount of server resource. In addition, to quickly respond to environmental fluctuations such as system failures and changing demands, and to maintain the scalability of the NFV services, a distributed control is more feasible than a centralized one [10]. One way to achieve such behaviors is to exploit a biochemical mechanism with autonomous dispersibility and self organization [11, 12].

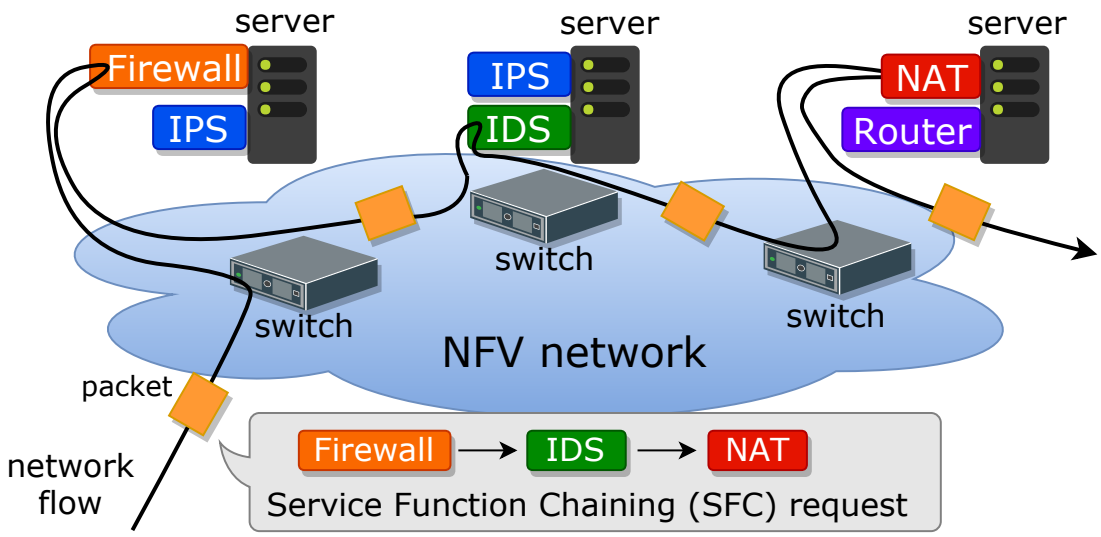


Figure 1: NFV system

Our research group has proposed a construction method of service space in virtualized network system based on biochemically-inspired tuple space model [13, 14]. In this method, a server is considered as a tuple space, and service requests, service demands and server resources are expressed as chemical substances in tuple spaces. The behaviors in the virtualized network system are then described by biochemical reaction equations in tuple spaces. Furthermore, by configuring a network by connecting multiple tuple spaces, the movement and spread of services and requests in a network system composed of multiple servers are represented. Since biochemical reaction equations are defined and executed independently in each tuple space, it is suitable for achieving autonomous and decentralized behaviors. We consider that one of possible application of the above method is an NFV system. To operate an NFV system, the above method has been extended to handle flow routes in accordance with SFC requests, and server resource limitation. By including these behaviors in the method, it is possible to get closer to the actual NFV service. The basic behaviors of the extended method have been confirmed with computer simulation. However, the evaluation assuming various situations in the NFV system has not been performed. In addition, the method has been implemented and evaluated only in a simple experimental environment, so the applicability to the actual NFV system has not been clearly shown.

In this thesis, we assess the performance of the NFV system based on the service space construction method, and show its implementation design. First, we briefly summarize how to apply the service space construction method to the NFV system, explained in [14]. In particular, we describe various behaviors in the NFV system, such as the execution of VNFs to flow packets, server resource allocation to each VNF, diffusion of VNFs, packet forwarding, and coexistence of multiple VNFs on a single server, by biochemical reaction equations in tuple spaces. Then, we perform computer simulation experiments assuming various situations in the NFV system, such as time variation of traffic amount and a sudden network failure. Through the simulation experiments, we confirm that the proposed method can cope with dynamical changes in the NFV system.

We then explain how to incorporate the proposed method to NFV framework [15] proposed by European Telecommunications Standards Institute (ETSI) Industry Specification Group (ISG). We show the detailed implementation environment using Open Platform for NFV (OPNFV) [16], one of major implementation of the NFV framework with open source softwares such as OpenStack [17], OpenDaylight [18], and Kernel-based Virtual Machine (KVM) [19]. In addition, we present an example of the implementation of SFC using Network Service Header (NSH) [20]

proposed by Internet Engineering Task Force (IETF).

The rest of this thesis is organized as follows. Section 2 summarizes the related work. Section 3 explains the tuple space model using biochemical reactions and how to apply the model to NFV system. Section 4 shows the simulation results to confirm that the proposed method can cope with dynamical network situations in the NFV system. Section 5 shows the implementation design of the proposed method with the NFV framework and a realization of the implementation of SFC using NSH. Finally, Section 6 concludes this thesis and presents some directions for future research.

2 Related Work

Various methods have been proposed for placement of VNFs in the NFV system [21]. These existing works, as well as the method proposed in our research group [13, 14], are summarized in Table 1.

The authors of [22–27] proposed various methods for dynamic placement of VNFs. In [22], the authors studied the deployment of NFV middleboxes considering the traffic changing effects by middleboxes to achieve the optimal network performance. In [23], the authors presented a dynamic placement approach of VNFs to maintain low end-to-end latency in edge and cloud computing environments. In [24], the authors presented a placement algorithm of VNFs based on the estimation of the access location of users. It considers the migration of VNF instances to higher accessed locations. In [25], the authors proposed VNF placement strategies in edge and cloud computing environments to optimize resource utilization, prevent cloud overload, and avoid the violation of QoS requirements. The authors of [26] studied the migration of flows for NFV elastic contol including scaling, load balancing, failure recovery, and upgrading of VNFs. In [27], the authors proposed a dynamic scaling algorithm of VNF instances considering the tradeoff between response time and operation cost. However, the authors of [22–27] did not consider SFC requests of accommodated flows.

The authors of [28–31] proposed VNF placement methods considering SFC requests. In [28], the authors proposed a dynamic programming algorithm for VNF placement that maximizes acceptance rate of SFC requests, resource utilization, and provider's revenue. In [29], the authors studied SFC orchestration mechanism across multiple data centers to minimize overall costs including the deployment cost of VNFs and bandwidth cost between data centers. It is possible to scale-in and scale-out VNF instances depending on the number of jobs in the system. However, [28, 29] did not consider the relocation of VNFs against changes in network environment over time.

In [30], the authors presented a model of the dynamic and adaptive placement of VNFs considering the relocation of VNFs. In [31], the authors proposed a dynamic SFC deployment approach to reduce hop violations of SFC requests in data centers. The hop violations are reduced by migrating VNF instances to the other servers. However, the authors of [30, 31] did not consider the dynamic scaling of VNF instances and the reconfiguration of flow paths.

In [32], the authors proposed a VNF placement algorithm to meet latency requirements of SFC requests, as well as minimizing energy consumption. The proposed algorithm considers the migration of VNFs, the integration of multiple VNFs with low demand into one, and the replication of highly-demanded VNFs. The paths of SFC requests are then reconfigured. The authors of [33] studied a dynamic placement algorithm of VNFs to maximize network throughput, that considers the migration of VNF instances and instantiation of new VNF instances.

In our proposed method, we consider the SFC requests of accommodated flows in the NFV system. In addition, the proposed method enables the dynamic and adaptive relocation of VNFs, and scaling of VNF instances in accordance with changes in network environments. Furthermore, the proposed method controls the NFV system in a distributed fashion, while most of existing methods assume centralized control.

Table 1: Comparison of VNF placement methods

	[22, 25]	[23, 24]	[26]	[27]	[28, 29]	[30, 31]	[32, 33]	Proposed
Dynamic placement of VNFs	✓	✓	✓	✓	✓	✓	✓	✓
Consideration of SFC requests					✓	✓	✓	✓
Migration of VNFs		✓				✓	✓	✓
Scaling of VNF instances			✓	✓			✓	✓
Reconfiguration of flow paths			✓				✓	✓
Distributed system control								✓

3 VNF Control based on Tuple Space Model with Biochemical Reactions

In this section, we summarize the tuple space model using biochemical reactions and how to apply the model to NFV system, described in [14].

3.1 Tuple Space Model

A tuple space model in [14] is one of the models that describes a distributed system. Figure 2 depicts the tuple space model in this thesis. In the figure, a component of the distributed system is modeled as a tuple space. In a tuple space, biochemical reactions occur. Then, tuples in the tuple space correspond to chemical substances, and the amount of tuples corresponds to the concentrations of chemical substances. The concentrations of tuples can be increased and decreased by defining and executing biochemical reactions in tuple spaces.

A reaction rate of a biochemical reaction is determined by the product of the concentration of each reactant and the rate coefficient defined in the biochemical reaction equation. For example, we consider that the following reaction equation is defined, which defines X and Y as reactants, Z as a product, and a as a reaction rate coefficient.

$$X \mid Y \xrightarrow{a} Z$$

If the concentrations of reactants X and Y are respectively x and y, the reaction rate is axy. Due to this property, the reaction rates in biochemical reactions are controlled by the concentrations of reactants and the rate coefficients defined in biochemical reaction equations.

In addition, a network can be configured by connecting multiple tuple spaces. It is possible to achieve the interaction among multiple tuple spaces by defining biochemical reactions that describe the diffusion and movement of tuples among tuple spaces. Since biochemical reactions in each tuple space occur independently, autonomous and decentralized behaviors in networked system can be described.

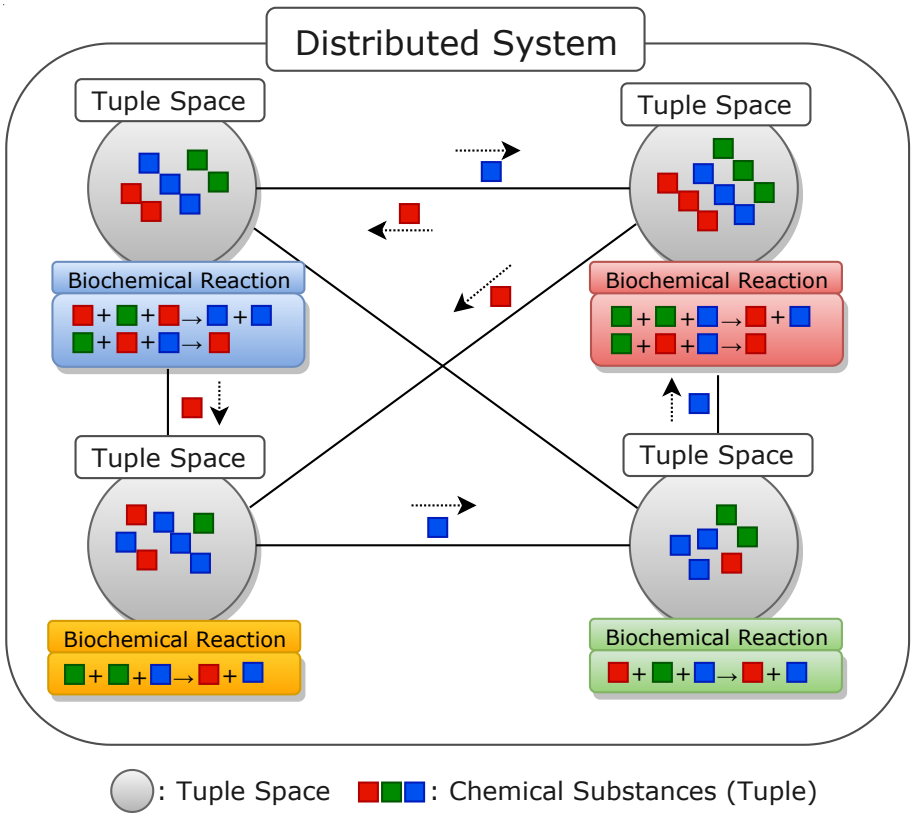


Figure 2: Tuple space model using biochemical reaction

3.2 Application to NFV System

Figure 3 depicts the application of the model described in SubSection 3.1 to NFV system. To apply the tuple space model to NFV system, a tuple space is associated with a server that deploys and executes VNFs. Tuples in the tuple spaces correspond to demands of VNFs, flow packets, server resources, and so on. The behaviors in the NFV system are described by biochemical reaction equations in tuple spaces. Biochemical reaction equations are defined to adaptively and autonomously determine placement of VNFs on the servers, the resource allocation to each VNF, and flow routes in accordance with SFC requests, traffic amount of the flows, and the amount of server resources. Table 2 shows the correspondence between the tuple space model and the NFV system.

An SFC request for a flow, represented by a series of VNFs, $f_1, f_2, f_3, ..., f_{end}$ is described as follows.

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

When VNF f_1 is executed to the flow with an SFC request c, c changes as follows.

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

A VNF that is executed at first in c is represented by $f^1(c)$. In this thesis, the subscript f, c and t of chemical substances represent a VNF, an SFC request, and a server, respectively. In what follows, we present biochemical reaction equations that achieve various behaviors for the NFV system.

3.2.1 Resource Allocation and Execution of VNFs

It is desirable that placement of VNFs on servers and resource allocation to each VNF are determined in accordance with demands of VNFs. It is required that VNFs in low demand have low priority in the server and those in high demand have high priority to be executed. When a packet of a flow with an SFC request c arrives at a server, VNF $f^1(c)$ is applied to the packet. Then, when c is composed of multiple VNFs, the SFC request c changes so that the executed VNF is deleted from c. On the other hand, when c is composed of one VNF, the packet disappears. The above behaviors are described by Reaction Equations (1) and (2).

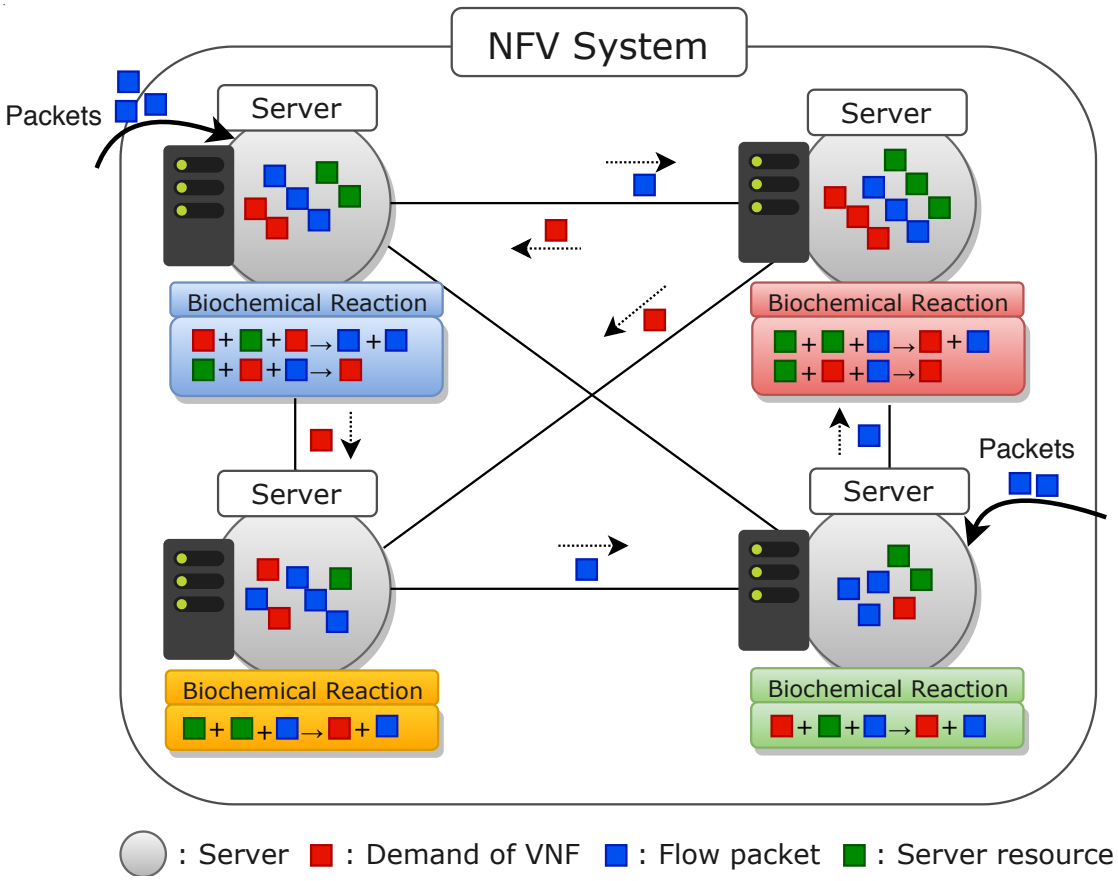


Figure 3: Application of tuple space model to NFV system

Table 2: Correspondence between tuple space model and NFV system

Tuple Space Model	NFV System
Tuple Spaces	General-purpose Servers
Chemical Substances	Demand of VNFs, Flow Packets, Server Resources, Gradient Fields for VNFs
Biochemical Reactions	Apply VNFs to Packets, Demand Increase of VNFs, Decay of VNFs Server Resource Allocation to VNFs, Diffusion of VNFs, Packet Forwarding

$$VNF_{f^1(c)} \mid PKT_c \xrightarrow{r_{us}} \begin{cases} VNF_{f^1(c)} \mid VNF_{f^1(c)} \mid PKT_{c\backslash\{f^1(c)\}} \\ \mid toserve(VNF_{f^1(c)}, PKT_c) \;\; (c\backslash\{f^1(c)\} \neq \emptyset) \\ \\ VNF_{f^1(c)} \mid VNF_{f^1(c)} \\ \mid toserve(VNF_{f^1(c)}, PKT_c) \;\; (c\backslash\{f^1(c)\} = \emptyset) \end{cases} \quad (1)$$

$$VNF_f \xrightarrow{r_{ds}} 0 \quad (2)$$

In the above Equations, substance $VNF_{f^1(c)}$ indicates the VNF to be applied for a flow. A VNF with a large concentration value means that its execution is highly demanded. Substance PKT_c represents a packet constituting a flow with c. Substance $toserve(VNF_{f^1(c)}, PKT_c)$ indicates result of applying the VNF to a packet of a flow with c. r_{us} and r_{ds} are the rate coefficients of Reaction Equations (1) and (2), respectively, to determine the rate of reactions. Reaction (1) indicates that a VNF is executed to packets of a flow on a server, and the concentration of VNF increases to represent the demand increase for the corresponding VNF. Reaction (2) indicates that VNF decays at a rate proportional to its concentration.

The execution rate of a biochemical reaction is determined in proportion to the product of the concentration of each reactant of the reaction. Therefore, in Reaction Equation (1), as the concentrations of VNF and PKT increase, the reaction rate increases without limitation. However, servers have their performance constraints determined by server resources such as CPU capacity and memory size. Therefore, using only Reaction (1) is not suitable for describing the behaviors in the NFV system. To describe the above constraints, enzyme-catalyzed reactions mechanism in biochemical reactions are exploited [34]. In enzyme-catalyzed reactions, the reaction rate can be controlled by the concentration of the catalyst which does not affect the reaction itself. The basic equation of the enzyme-catalyzed reaction is shown in the following Reaction Equation, which defines E as an enzyme, S as a substrate, ES as an enzyme-substrate complex, and P as a product.

$$E \mid S \leftrightarrows ES \rightarrow E \mid P$$

The execution rate of the enzyme-catalyzed reaction can be determined by introducing an enzyme-substrate complex into the reaction [35]. To describe the constraints of server resources, Reaction

Equation (1) is extended into the following Reaction Equations (3) and (4) by applying enzyme-catalyzed reactions mechanism.

$$RSRC_t \mid VNF_f \underset{r_{u_1}}{\overset{r_{v_1}}{\leftrightarrows}} RS\_VNF_f \tag{3}$$

$$RS\_VNF_{f^1(c)} \mid PKT_c \underset{r_{u_2}}{\overset{r_{v_2}}{\leftrightarrows}} MEDIATE_c \xrightarrow{r_w} \begin{cases} VNF_{f^1(c)} \mid VNF_{f^1(c)} \mid PKT_{c\backslash\{f^1(c)\}} \mid RSRC_t \\ \mid toserve(VNF_{f^1(c)}, PKT_c) \ \ (c\backslash\{f^1(c)\} \neq \emptyset) \\ \\ VNF_{f^1(c)} \mid VNF_{f^1(c)} \mid RSRC_t \\ \mid toserve(VNF_{f^1(c)}, PKT_c) \ \ (c\backslash\{f^1(c)\} = \emptyset) \end{cases} \tag{4}$$

The concentrations of substances $RSRC_t$, RS_VNF_f, and $MEDIATE_c$ respectively represent the amount of available resources of a server t, the amount of server resources allocated to VNF f, and the amount of server resources allocated to the flow packets with SFC request c. r_{v_1} and r_{u_1} are the rate coefficients for Reaction Equation (3), and r_{v_2}, r_{u_2} and r_w are the rate coefficients for Reaction Equation (4). Reaction Equation (3) indicates that server resources are allocated in accordance with the demand of each VNF, and that the allocation is controlled by the concentration of $RSRC$. Reaction Equation (4) indicates that VNF f is executed on the basis of the amount of allocated resources.

3.2.2 Diffusion of VNFs

To describe the diffusion of highly-demanded VNFs to other servers, Reaction Equation (5) is described.

$$VNF_f \xrightarrow{r_{mf}} VNF_f{}^{\rightsquigarrow} \tag{5}$$

r_{ms} is the rate coefficient for Reaction Equation (5). This Reaction Equation indicates that a highly-demanded VNF in a server diffuses to the surrounding connected servers at a rate proportional to its concentration. This diffusion destination of VNFs is stochastically determined in accordance with the concentrations of VNF at connected tuple spaces. As a result, highly-demanded VNFs are distibuted to multiple servers.

3.2.3 Packet Forwarding

When packets remain unprocessed in a server due to a lack of server resources for corresponding VNF, it is required that the packets move to another server that can process the corresponding VNF. Furthermore, the forwarding direction of packets should be determined so that the packets would approach a server executing the corresponding VNFs with enough server resources. To achieve these behaviors, a gradient field is exploited to determine the moving directions of packets. A gradient field for each VNF is constructed based on the demand of VNFs and the available resources on each server. The moving direction of packets is then determined in accordance with the gradient field. For that purpose, Reaction Equations (6)-(10) are introduced.

$$VNF_f \mid RSRC_t \xrightarrow{r_{rg}} VNF_f \mid RSRC_t \mid GRAD_f \tag{6}$$

$$VNF_f \mid RS\_VNF_f \xrightarrow{r_{rg}} VNF_f \mid RS\_VNF_f \mid GRAD_f \tag{7}$$

$$GRAD_f \xrightarrow{r_{dg}} 0 \tag{8}$$

$$GRAD_f \xrightarrow{r_{mg}} GRAD_f^{\rightsquigarrow}(GRAD_f^{-}) \tag{9}$$

$$PKT_c \xrightarrow{r_{mp}} PKT_c^{\rightsquigarrow}(GRAD_f^{+}) \tag{10}$$

Substance $GRAD_f$ establishes a gradient field for VNF f. r_{rg} is the rate coefficient for Reaction Equation (6) and (7), and r_{dg}, r_{mg} and r_{mp} are the rate coefficients for Reaction Equation (8), (9) and (10), respectively. Reaction Equation (6) and (7) indicate that $GRAD$ is generated at a rate proportional to the concentrations of VNF, $RSRC$, and RS_VNF. Reaction Equation (8) indicates that $GRAD$ decays at a rate proportional to its concentration. Reaction Equation (9) indicates that $GRAD$ spreads to the surrounding servers with smaller concentration of $GRAD$. Therefore, the gradient field is constructed so that the server providing VNFs with enough resources becomes a summit with the largest concentration of $GRAD$, and the surrounding servers have smaller concentration of $GRAD$ in accordance with the distance from the summit. Reaction Equation (10) describes the movement of PKT to the surrounding servers with large concentration of $GRAD$. The forwarding direction of packets are stochastically determined at a proportional to the concentrations of $GRAD$ at connected tuple spaces. Figure 4 depicts the movement of packets with the SFC request $c = \{f_0, f_1, f_2\}$. The gradient fields are respectively generated for each VNF. First, packets move in the direction of the summit of the gradient field for f_0. Then, after applying f_0 to the packets, they move in the direction of the summit of the gradient field for f_1.

Finally, after applying f_1 to the packets, they move in the direction of the summit of the gradient field for f_2.

3.2.4 Coexistence of Multiple VNFs

When multiple VNFs coexist on a single server, it is required to share server resources by allocating them in accordance with the demand of each VNF. Therefore, the above-mentioned biochemical reaction equations are defined for each VNF.

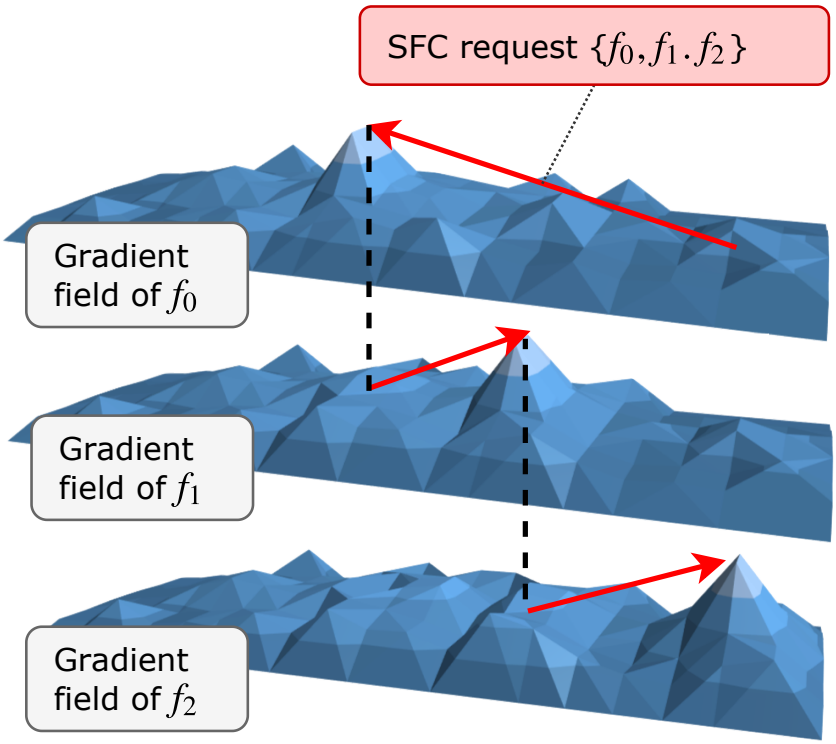


Figure 4: Movement of packets in accordance with gradient fields

4 Simulation Experiments

In this section, we assess the performance of the NFV system based on the method described in Section 3. The basic behaviors of the proposed method, such as placement of VNFs on servers, resource allocation to each VNF, and flow routing in accordance with SFC requests, have been confirmed in [14]. We then confirm that the proposed method can cope with dynamical changes in the NFV system.

4.1 τ-Leaping Method

In order to simulate the model with biochemical reactions, we exploit τ-leaping method [36], which is one of stochastic simulation algorithms that can capture the inherent stochasticity in many biochemical systems. The basic idea of τ-leaping method is to obtain temporal change in concentrations of chemical substances by executing reactions simultaneously during preselected time τ. The procedures of τ-leaping algorithm are briefly explained as follows.

Step 1 Set τ for the time step of the simulation

Step 2 Calculate reaction rates of biochemical reactions by the product of concentrations of reactants and reaction rate coefficients

Step 3 Determine the number of executions of biochemical reactions during time τ, using a Poisson random variable

Step 4 Execute biochemical reactions as many times as the number determined in Step 3, and update the concentrations of substances

Step 5 Progress simulation time by τ

Step 6 Return to Step 2

The value of τ should be chosen to balance the trade-off relationship between simulation accuracy and simulation speed. As the value of τ increases, the results become different from the actual behavior while the simulation can proceed faster. In [37], the authors determined the optimal value of τ especially for the simulation accuracy, at the sacrifice of the computational cost of the simulation. In [14], the value of τ was detemined by performing some preliminary experiments.

In the simulation experiments in this section, the value of τ is set to 0.6 [msec], which is identical to that in [14].

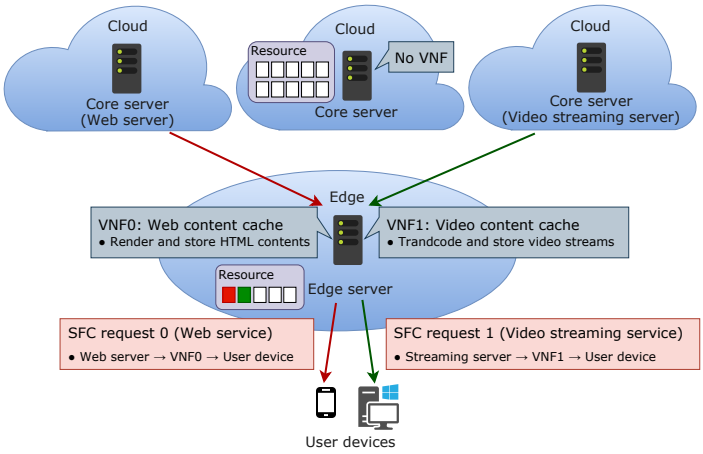
4.2 Common Parameter Settings

The initial values of the concentrations of substances VNF for the VNFs placed on servers are set to 2,000. The initial values of the concentrations of other chemical substances except $RSRC$ are set to 0. Unless otherwise specified, the reaction rate coefficients of Reaction Equations (3)-(10) are set as $r_{u1} = 0.0003$, $r_{v1} = 0.278$, $r_{u2} = 0.1$, $r_{v2} = 0.001$, $r_w = 0.05$, $r_d = 0.01$, $r_{mf} = 0.003$, $r_{rg} = 0.0001$, $r_{dg} = 0.03$, $r_{mg} = 0.005$, $r_{mp} = 0.3$, as used in [14].

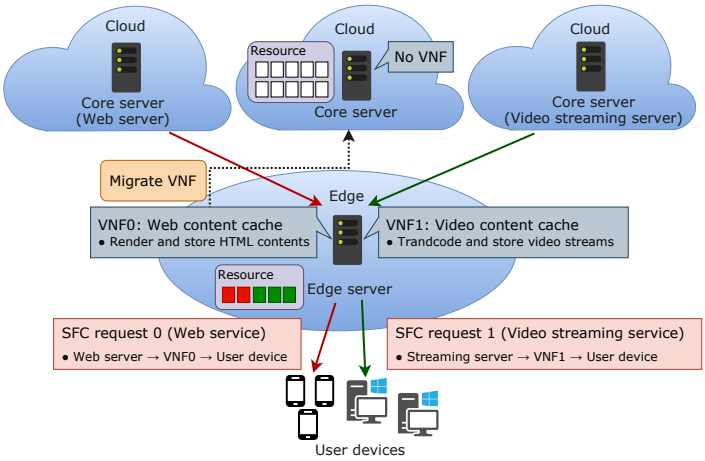
4.3 Scenario 1: Placement of VNFs Considering Flow Priorities

4.3.1 Application Scenario

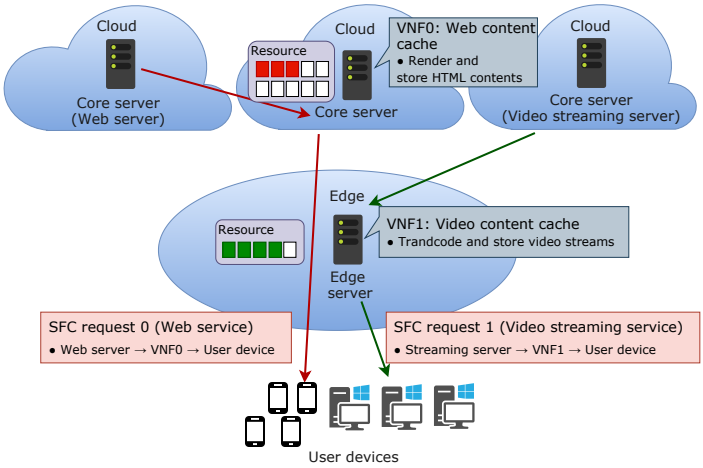
In this scenario, we consider the situation where there are two kinds of application flows with SFC requests that have different priorities on the latency requirements. Figure 5 depicts this scenario. In the figure, Web service and video streaming service are provisioned in edge and cloud computing environments. In the beginning, a Web server and a video streaming server are placed in the cloud. The both servers receive requests from user devices, and send content packets to the user devices. Flows between the Web server and user devices require functions of rendering and caching in the network. On the other hand, transcoding and caching functions are applied to flows between the video streaming server and the user devices. Consequently, these VNFs are deployed in the network, that are called as VNF 0 and VNF 1, respectively. Furthermore, the SFC requests of the flows in the NFV system are {Web server→VNF 0→User device} and {Streaming server→VNF 1→User device}. These SFCs are respectively denoted by SFC 0 and SFC 1. We assume that the flows for the video streaming service have higher priority in being executed at the edge server to meet the latency requirements.



(a) Edge server being not busy



(b) Edge server being busy



(c) Migrating the Web service VNF to the cloud server

Figure 5: Scenario1: Placement of VNFs considering flow priorities

As depicted in Figure 5(a), in case of edge server being not busy, VNF 0 and VNF 1 are deployed on the edge server to realize contents caching for both services near the user devices. In Figure 5(b), since the number of user devices increases, the edge server becomes busy and all packets cannot be processed only at the edge server. Then, as depicted in Figure 5(c), VNF 0 is migrated to the cloud server, while VNF 1 remains on the edge server to avoid the degradation of the quality of both services.

4.3.2 Network Topology and Parameter Settings for Simulation Experiments

Figure 6 depicts the network topology for simulation experiments of Scenario 1, that consists of two nodes and a link. Node 0 and node 1 correspond to the edge server and the cloud server, respectively. VNF_{f_0} and VNF_{f_1} correspond to VNF 0 and VNF 1 in Figure 5, respectively. There are two flows with SFC requests $c_0 = \{f_0\}$ and $c_1 = \{f_1\}$, corresponding to the flows with SFC 0 and SFC 1. These flows are denoted by flow 0 and flow 1, respectively.

The simulation time is 2,000 [msec]. VNF_{f_0} and VNF_{f_1} are initially deployed on node 0, and their initial concentrations are set to 2,000. The initial concentrations of $RSRC$ at node 0 and node 1 are set to 500 and 1,000, respectively, which means that the cloud server has larger and sufficient resource than the edge server. Table 3 shows the temporal change in flow rates. In the table, t is defined as simuation time. For $0 \leq t \leq 1,000$, packets of flow 0 and flow 1 arrive at node 0 at 5 packets per time step, corresponding to 8.3 [Kpps]. At $t = 1,000$, the rates of both flows are increased to 20 packets per time step, corresponding to 33.3 [Kpps]. Note that for $0 \leq t \leq 1,000$, the edge server can process all incoming packets, and for $1,000 < t \leq 2,000$, the edge server cannot process all packets.

To prioritize the execution of VNF 1 at node 0, the rate coefficient r_{u1} in Reaction Equation (3) is adjusted. We utilize $r_{u1} = 0.0003$ for the VNF_{f_0}, and $r_{u1} = 0.003$ for the VNF_{f_1} at node 0, to prioritize flow 1. We also perform simulation experiments with $r_{u1} = 0.0003$ for VNF_{f_0} and VNF_{f_1} at node 0 for comparison purposes.

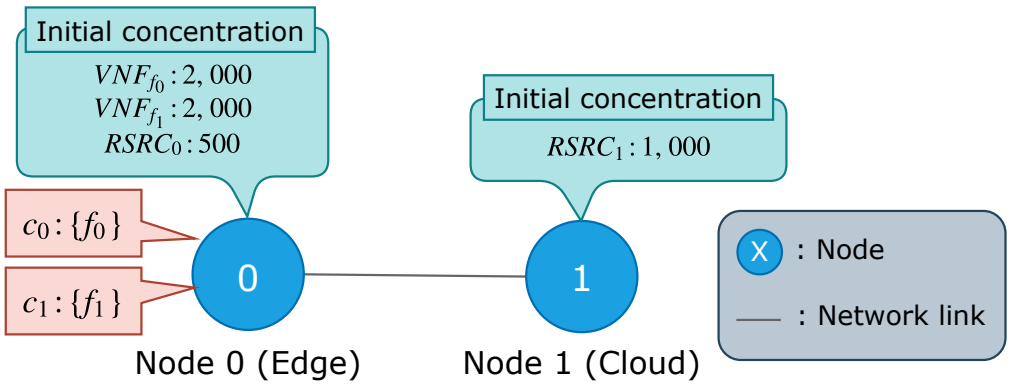


Figure 6: Scenario1: Network topology for simulation experiments

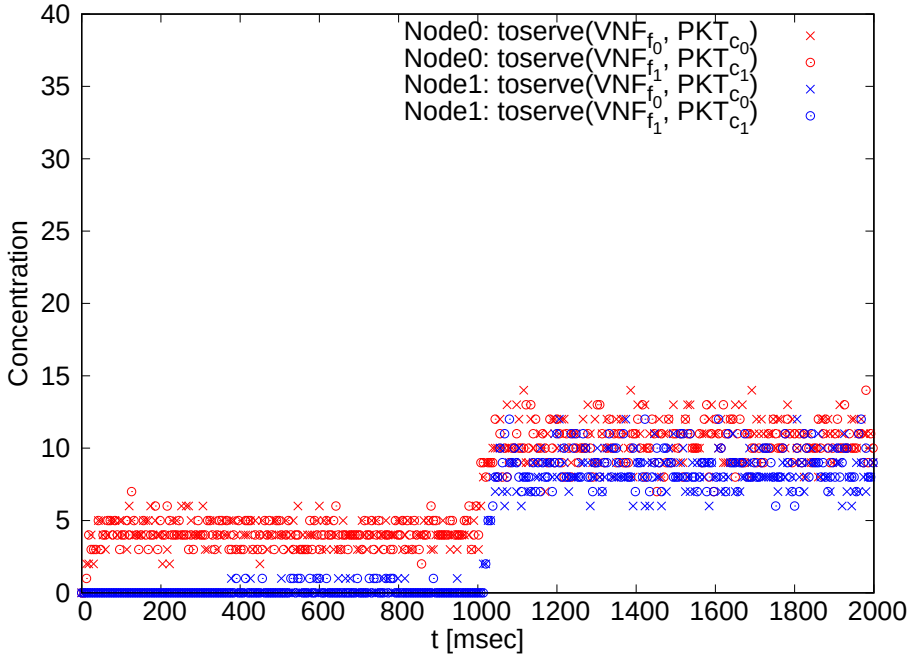
Table 3: Scenario1: Temporal change in rate of flows

Flow	Priority	Rate	
		$0 \leq t \leq 1,000$ [msec]	$1,000 < t \leq 2,000$ [msec]
flow 0	low	8.3 [Kpps]	33.3 [Kpps]
flow 1	high	8.3 [Kpps]	33.3 [Kpps]

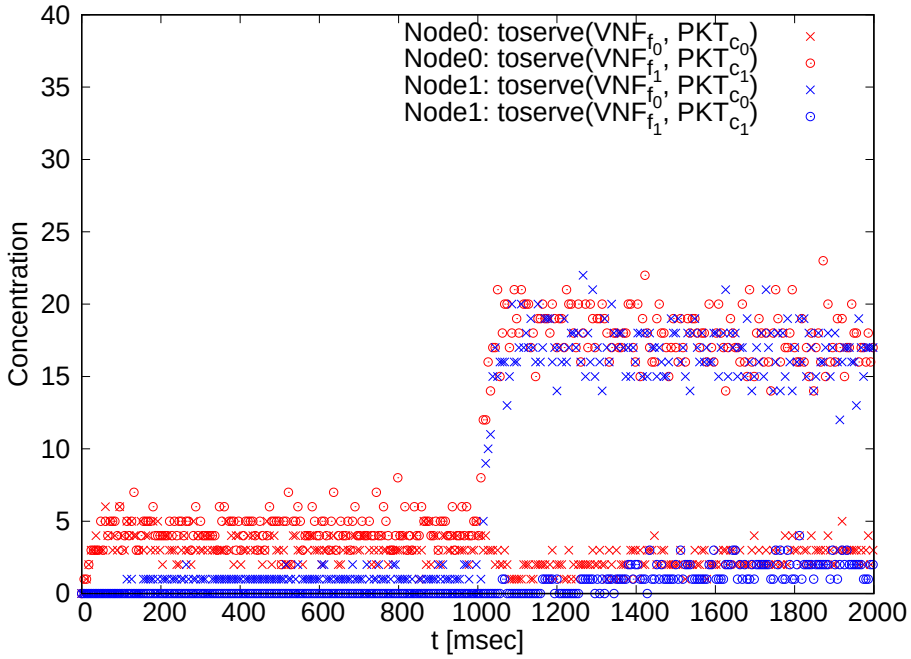
4.3.3 Simulation Results and Discussion

Figure 7 plots the average number of executions of Reaction Equation (4), that corresponds to the executions of VNFs to flow packets, as a function of simulation time step. Figures 7(a) and 7(b) show simulation results with $r_{u1} = 0.0003$ and $r_{u1} = 0.003$ for VNF_{f_0} at node 0, respectively. Figure 8 shows the temporal change in the concentrations of $RSRC$ at node 0 and node 1.

For $0 \leq t \leq 1,000$, VNF_{f_0} and VNF_{f_1} are executed almost only at node 0. This is because node 0 has sufficient resources to execute both VNFs to flow packets. It can be confirmed from the concentration of $RSRC_0$ in Figure 8. For $1,000 < t \leq 2,000$, VNF_{f_0} and VNF_{f_1} are executed at node 0 and node 1 in a distributed manner, when using the same value of r_{u1} for VNF_{f_0} and VNF_{f_1}. This is because node 0 has insufficient resources to execute both VNFs due to the increase in flow rates. It can be confirmed from the concentration of $RSRC_0$ in Figure 8(a). On the other hand, when setting r_{u1} in accordance with flow priorities, VNF_{f_0} and VNF_{f_1} are executed at node 1 and node 0, respectively. This behavior realizes that the VNF in video streaming service with higher priority is preferentially executed at the edge server, as depicted in Figure 5.

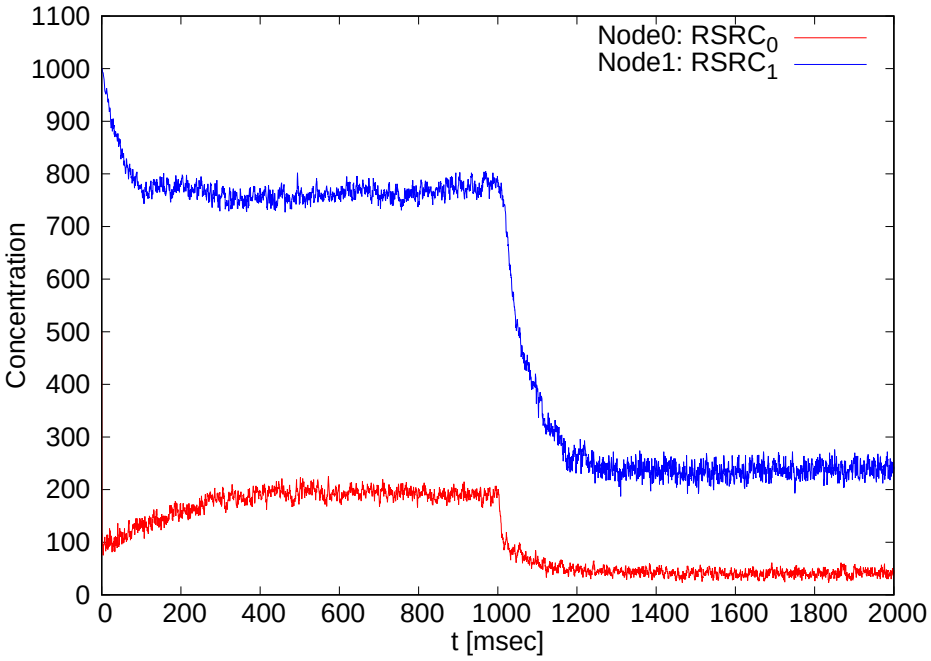


(a) $r_{u1} = 0.0003$

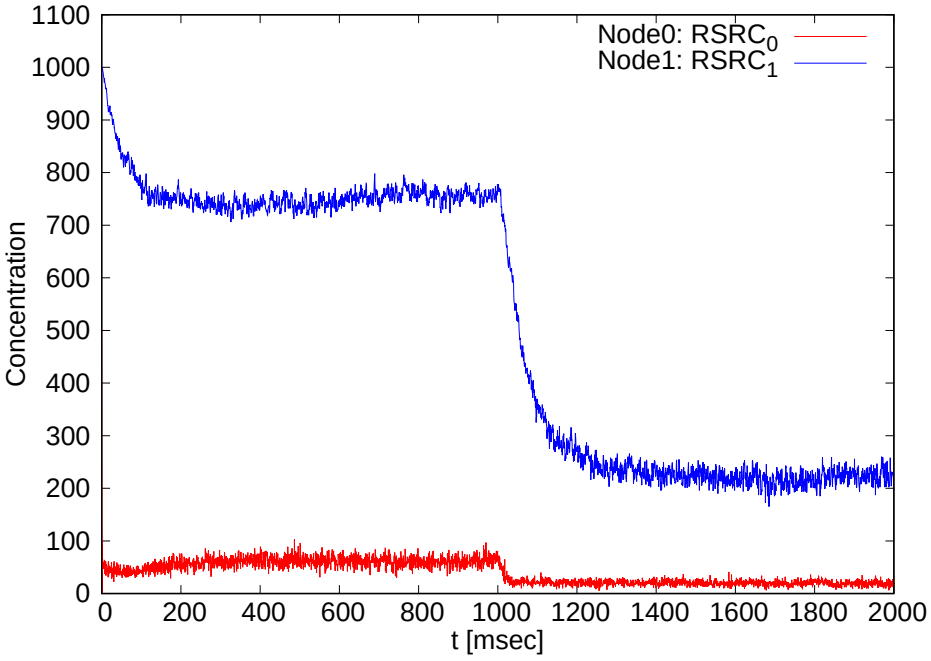


(b) $r_{u1} = 0.003$

Figure 7: Scenario1: Average number of executions of Reaction Equation (4)



(a) $r_{u1} = 0.0003$



(b) $r_{u1} = 0.003$

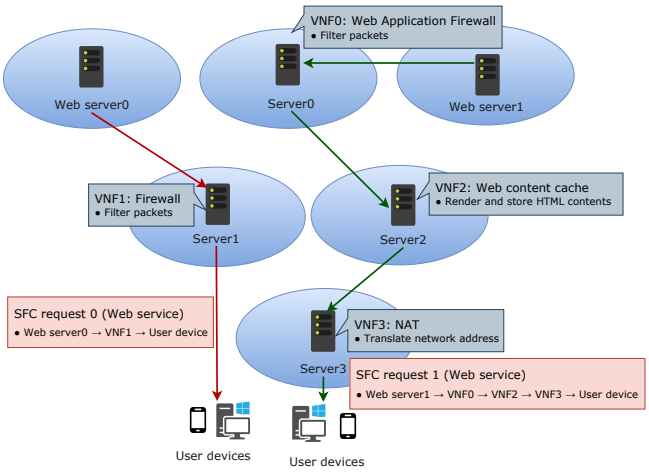
Figure 8: Scenario1: Temporal change in the concentrations of $RSRC$ at node 0 and node 1

4.4 Scenario 2: Route changes and VNF migrations on network failures

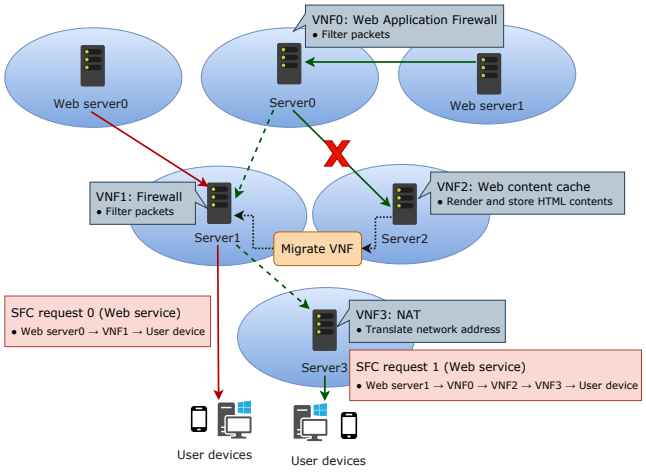
4.4.1 Application Scenario

In this scenario, we consider the situation where failures of network link occur. Figure 9 depicts this scenario. In the figure, two Web services are provisioned in cloud computing environment. The both of Web server 0 and Web server 1 receive requests from user devices, and send content packets to the user devices. Flows between the Web servers and user devices require a function for filtering, monitoring, and blocking HTTP traffic, a function for monitoring and controling incoming and outgoing network traffic, a function for rendering and caching in the network, and a function for translating network addresses. Consequently, four VNFs exist in the network, that are denoted by VNF 0, VNF 1, VNF 2, and VNF 3. Initially, VNF 0, VNF 1, VNF 2, and VNF 3 are respectively deployed on server 0, server 1, server 2, and server 3. VNF 1 is applied to the flow between Web server 0 and user devices, and VNF 0, VNF 2, VNF 3 are sequentially applied to the flow Web server 1 and user devices. The SFC requests of the two flows are $\{$Web server$\rightarrow$VNF1$\rightarrow$User device$\}$ and $\{$Web server$\rightarrow$VNF0$\rightarrow$VNF2$\rightarrow$VNF3$\rightarrow$User device$\}$, that is called SFC 0 and SFC 1.

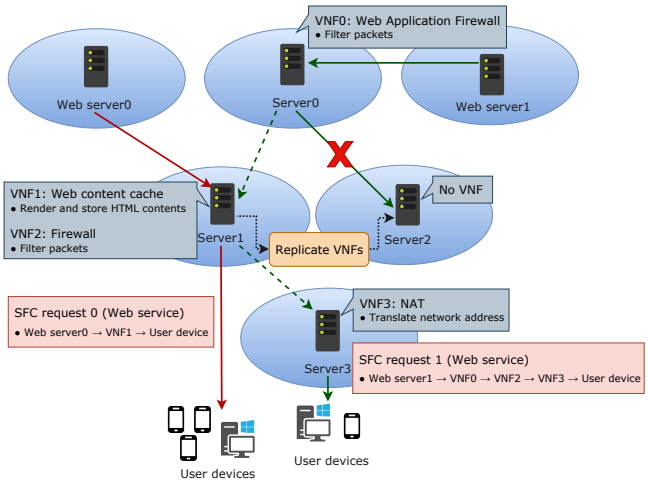
Figure 9(a) shows the situation where the system is operated normally. In Figure 9(b), a network link between server 0 and server 2 is disconnected due to network failures. Then, server 0 forwards flow packets via server 1 to continue the service. Additionally, VNF 2 is migrated to server 1 to reduce the number of hops. In Figure 9(c), the number of user devices increases and server 1 becomes busy, that means all packets cannot be processed only at server 1. Then, VNF 1 and VNF 2 are executed at server 1 and server 2 in a distributed manner.



(a) System being operated normally



(b) System failures occur



(c) The server being busy

Figure 9: Scenario2: Route changes and VNF migrations on network failures

4.4.2 Network Topology and Parameter Settings for Simulation Experiments

Figure 10 depicts the network topology for Scenario 2, that consists of four nodes and five links. Node 0, node 1, node 2, and node 3 correspond to server 0, server 1, server 2, and server 3 in Figure 9, respectively. VNF_{f_0}, VNF_{f_1}, VNF_{f_2}, and VNF_{f_3} correspond to VNF 0, VNF 1, VNF 2, and VNF 3, and are initially deployed on node 0, node 1, node 2, and node 3, respectively. There are two flows with SFC requests $c_0 = \{f_1\}$ and $c_1 = \{f_0, f_2, f_3\}$, corresponding to the flows with SFC 0 and SFC 1, that are called flow 0 and flow 1, respectively. When VNF_{f_0} is executed to the flow with c_1, c_1 changes as $c_2 = \{f_2, f_3\}$. When VNF_{f_2} is executed to the flow with c_2, c_2 changes as $c_3 = \{f_3\}$. The flows with SFC requests c_2 and c_3 are called flow 2 and flow 3, respectively.

The simulation time is 3,000 [msec]. The initial concentrations of VNF_{f_0}, VNF_{f_1}, VNF_{f_2}, and VNF_{f_3} are set to 2,000. The initial concentrations of $RSRC$ at all nodes are set to 1,000. Table 4 shows the temporal change in flow rates. In the table, t is defined as simulation time. For $0 \leq t \leq 2,000$, packets of flow 0 arrive at node 1 at 10 packets per time step, corresponding to 16.6 [Kpps]. Packets of flow 1 arrive at node 0 at 20 packets per time step, corresponding to 33.3 [Kpps]. At $t = 2,000$, the rate of flow 0 is increased to 30 packets per time step, corresponding to 50 [Kpps]. Note that for $0 \leq t \leq 2,000$, node 1 processes all incoming packets, and for $2,000 < t \leq 3,000$, node 1 cannot process all packets. In addition, at $t = 1,000$, a network link between node 0 and node 2 is disconnected.

When we configure the diffusion of VNFs so that all VNFs can be diffused to any other nodes, the concentrations of VNF_{f_1} and VNF_{f_2} increase at node 0 by executions of Reaction Equation (4) because packets of flow 1 arrive at node 0. Then, VNF_{f_1} and VNF_{f_2} are executed at node 0, and we cannot confirm the behaviors in Scenario 2. Therefore, the diffusion areas of VNF_{f_1} and VNF_{f_2} are limited to node 1 and node 2, and the reaction rate coefficient r_{mf} for VNF_{f_0} and VNF_{f_3} of Reaction Equation (5) is set to 0, so that the route of flow 1 is adequately changed on network failure, and that VNF_{f_1} and VNF_{f_2} are executed at node 1 and node 2 in a distributed manner, when the amount of traffic increases.

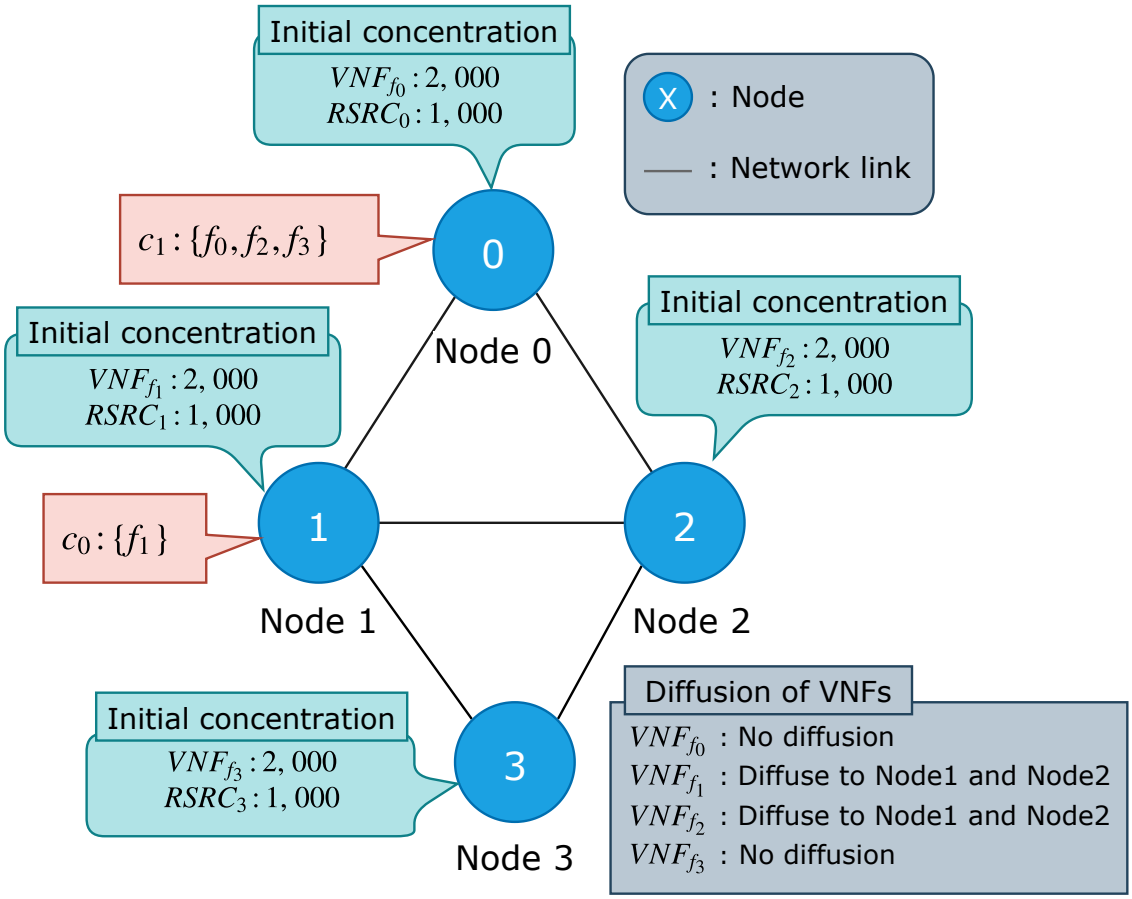


Figure 10: Scenario2: Network topology for simulation experiments

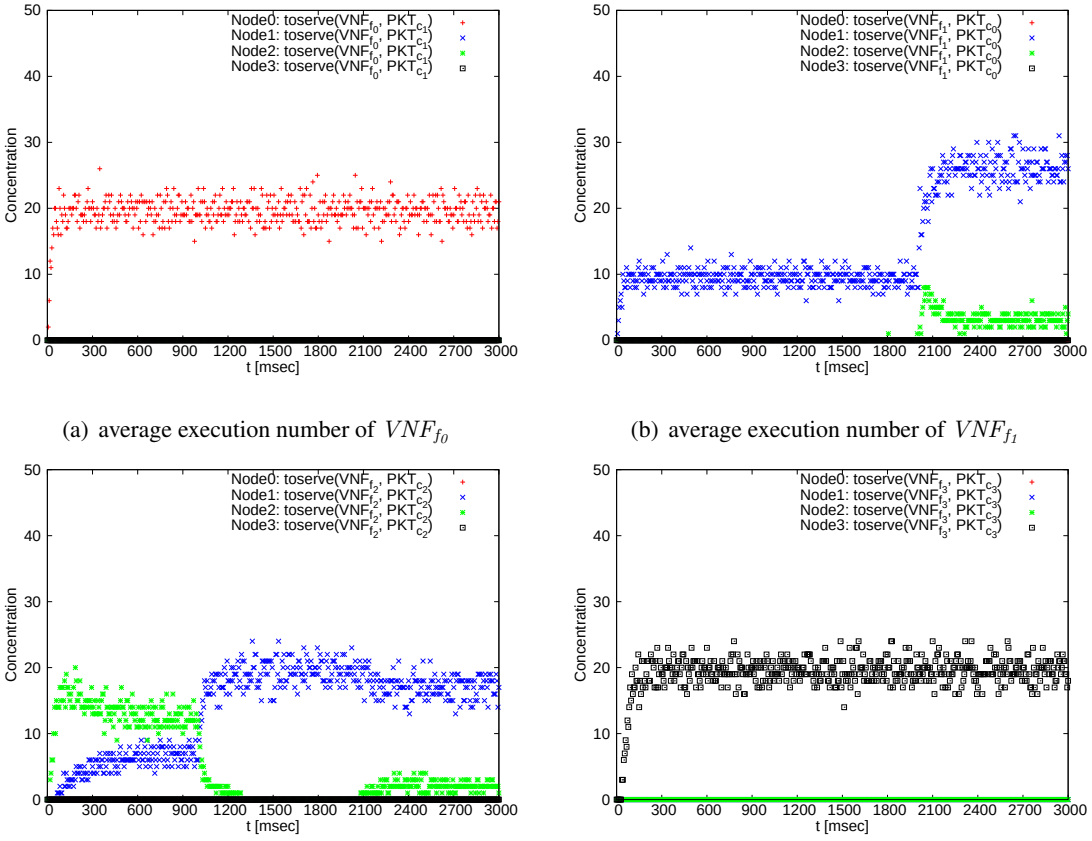
Table 4: Scenario2: Temporal change in rate of flows

Flow	Rate	
	$0 \leq t \leq 2,000$ [msec]	$2,000 < t \leq 3,000$ [msec]
flow 0	16.6 [Kpps]	50 [Kpps]
flow 1	33.3 [Kpps]	33.3 [Kpps]

4.4.3 Simulation Results and Discussion

Figure 11 plots the average number of executions of Reaction Equation (4). Figures 11(a), 11(b), 11(c), and 11(d) are results for VNF_{f_0}, VNF_{f_1}, VNF_{f_2}, and VNF_{f_3}, respectively. Figure 12 shows the temporal change in the concentrations of VNF_{f_0}, VNF_{f_1}, VNF_{f_2}, and VNF_{f_3}. Figure 13 shows the temporal change in the concentrations of $RSRC$ at all nodes.

For $0 \leq t \leq 1,000$, VNF_{f_1} is executed at node 1 in Figure 11(b). It can be confirmed from the concentration of VNF_{f_1} in Figure 12(b). VNF_{f_0}, VNF_{f_2}, and VNF_{f_3} are executed at node 0, node 2, and node 3 in Figures 11(a), 11(c), and 11(d). For $1,000 < t \leq 2,000$, VNF_{f_2} is executed at node 1. This is because VNF_{f_2} is migrated to node 1, where packets of flow 1 arrive after VNF_{f_0} is applied. This behavior realizes that server 0 forwards flow packets via server 1 and VNF 2 is migrated to server 1, as depicted in Figure 9. For $2,000 < t \leq 3,000$, VNF_{f_1} and VNF_{f_2} are executed at both of node 1 and node 2 in a distibuted manner. This is because node 1 has insufficient resources to execute VNF_{f_1} and VNF_{f_2}. It can be confirmed from the concentration of $RSRC_1$ in Figure 13. From the above results, we confirmed that the behaviors in Scenario 2 can be achieved.



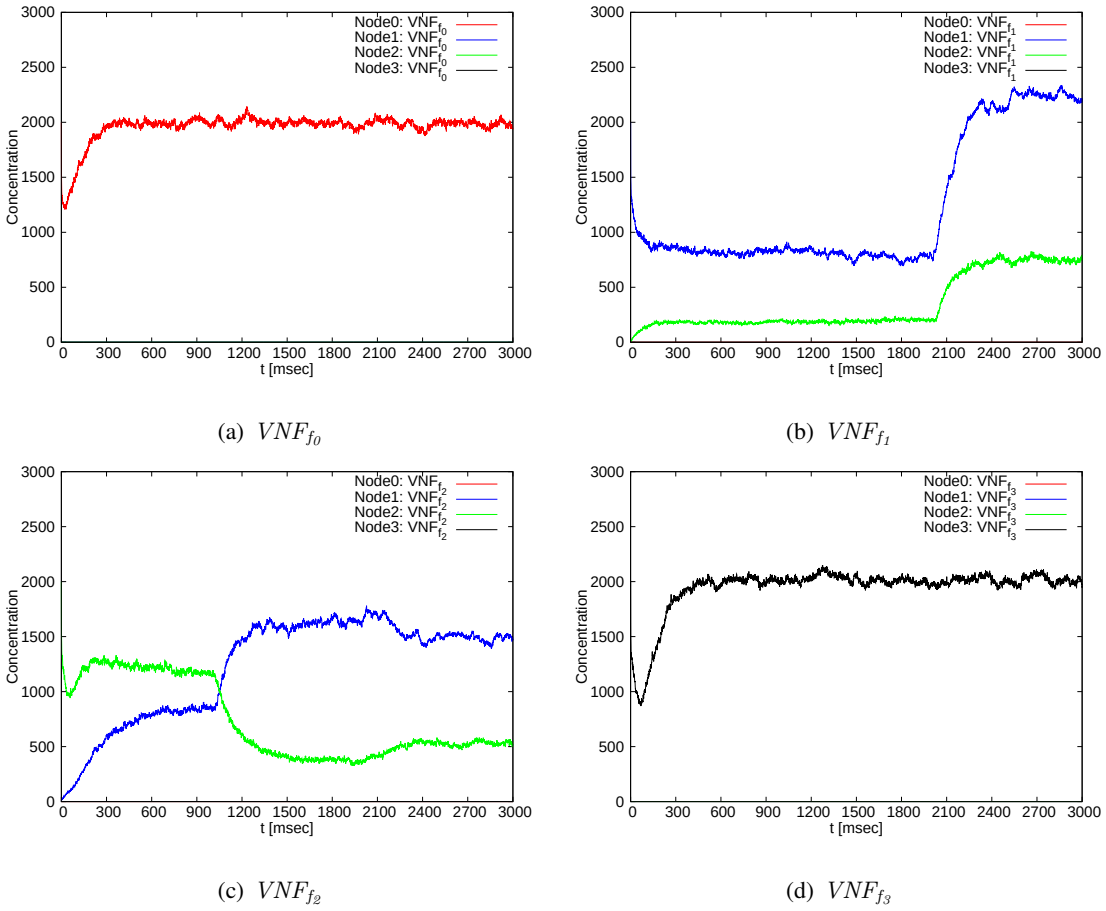
(a) average execution number of VNF_{f_0}

(b) average execution number of VNF_{f_1}

(c) average execution number of VNF_{f_2}

(d) average execution number of VNF_{f_3}

Figure 11: Scenario2: Average number of executions of Reaction Equation (4)



(a) VNF_{f_0}

(b) VNF_{f_1}

(c) VNF_{f_2}

(d) VNF_{f_3}

Figure 12: Scenario2: Temporal change in the concentrations of VNF

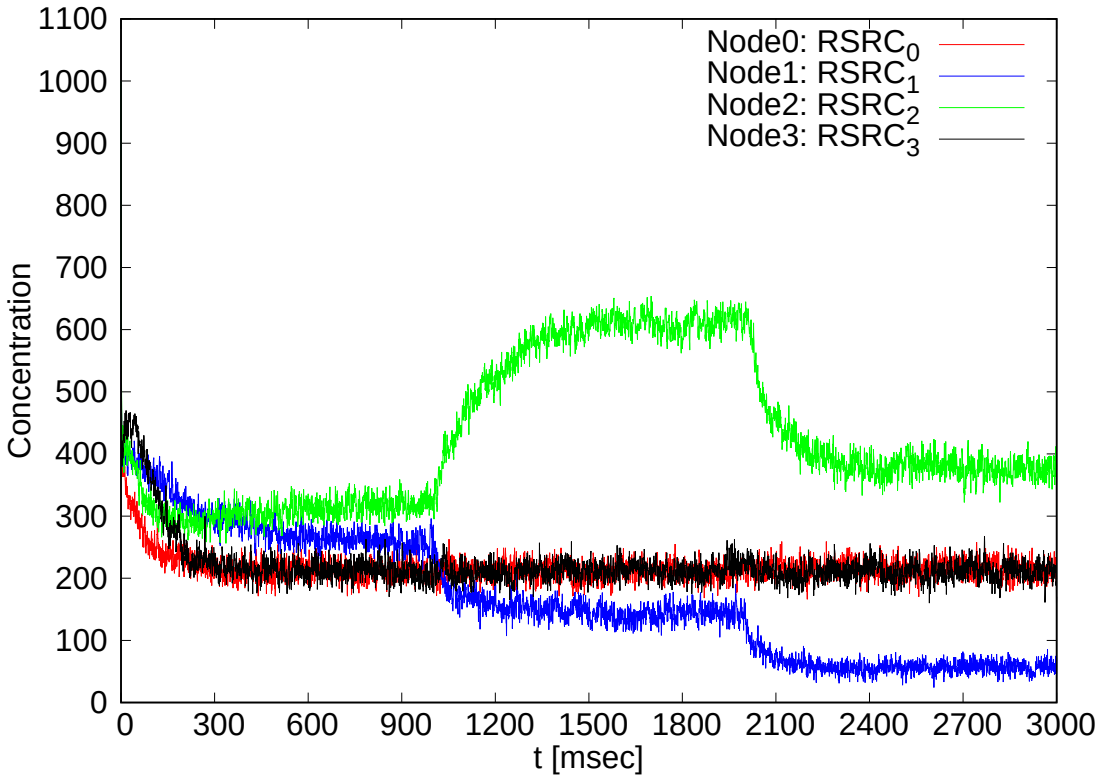


Figure 13: Scenario2: Temporal change in the concentrations of $RSRC$ at all nodes

5 Implementation Design of the Proposed Method with the NFV Framework

In this section, we describe the implementation design of the method proposed in Section 3, based on the NFV framework [15] and its integration with SDN [38] proposed by ETSI ISG.

5.1 NFV Framework and its Integration with SDN

Figure 14 depicts NFV framework and its integration with SDN. In the NFV framework, there are three main components: VNF, NFV Infrastructure (NFVI), and NFV Management and Orchestration (NFV MANO). VNF is the software implementation of a network function and is deployed on a virtual machine (VM). NFVI is an infrastructure to execute VNFs and manages physical and virtual resources. Physical resources are virtualized at virtualization layer such as hypervisor, and provided as a VM that deploys VNFs. NFV MANO manages the lifecycle and orchestration of resources in NFVI and VNFs. NFV MANO includes three functions: NFV Orchestrator (NFVO), VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM). NFVO manages SFC and orchestrates VNFs. VNFM manages the lifecycle (create, update, and delete) of VNFs. VIM manages computing, network, and storage resources in NFVI, and allocates resources to VNFs.

In [38], the integration of SDN in the NFV framework is discussed. SDN Controller is utilized to manage the physical resources in NFVI. SDN Switch is included in network resources in NFVI.

5.2 Positioning of the Proposed Method

As depicted in Figure 14, VIM is extended to include the management function of biochemical reaction equations ('Biochemical Reactions" in the figure) for proposed method. In Figure 14, the communication interfaces between functions are also illustrated. Vn-Nf is used to execute VNFs in the environment provided by NFVI. Or-Vnfm is used to acquire the state of VNFs for SFC by NFVO. Vi-Vnfm is used for resource allocation request by VNFM, and exchanging the state information of resources in NFVI. Or-Vi is used for resource allocation request by NFVO, and exchanging the state information of resources in NFVI. Nf-Vi is used to monitor the state of NFVI and biochemical reaction equations by VIM. Ve-Vnfm is used to manage the lifecycle of VNFs. Ss-Sc is used to control SDN Switch by SDN Controller. Or-Sc is used to receive flow routes by SDN Controller.

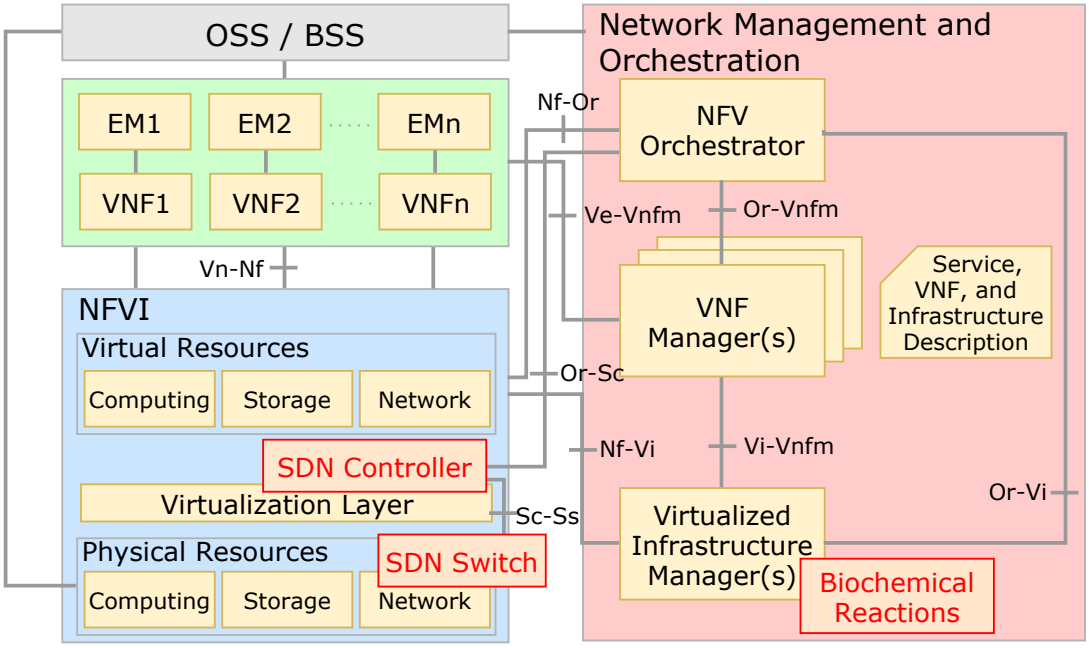


Figure 14: NFV framework and its integration with SDN

We adopt OPNFV [16] to implement the NFV framework. OPNFV aims at implementing the whole of NFV framework by integrating OSS as OpenStack [17], OpenDaylight [18], Open vSwitch [39], and KVM [19]. In OPNFV, since OpenFlow is used as a southbound protocol for SDN, we introduce the implementation design of the proposed method with OpenFlow.

5.3 Implementation Environment

Figure 15 depicts the system configuration of OPNFV and the placement of functions shown in Figure 14. In Figure 15, the NFV framework is implemented on a single physical machine. Multiple VMs and virtual switches are implemented on the physical machine. There are three types of VMs: Jump Server, Controller, and Compute. Jump Server is utilized for installing and maintaining Controller, Compute, and networking environment in the system. Controller is utilized for implementing NFV MANO nd SDN Controller. Compute is utilized for implementing VNFs and NFVI. VMs for deploying VNFs can be placed on Compute. There are four types of networks in the system: Admin, Tenant, Public, and Storage. Admin Network is used for installing and maintaining the NFV system. Tenant Network is used for network traffic generated by tenants on the NFV system. Public Network is used to connect to external networks. Storage Network is used for I/O processing of storage.

NFV MANO in Figure 14 is implemented on Controller using OpenStack in Figure 15. SDN Controller and Switch are respectively implemented on Controller with OpenDaylight and Open vSwitch. VNFs are deployed on VMs on Compute. NFVI is constructed with Compute, and virtual resources are provided with KVM.

In Figure 15, BR is a program that implements the tuple space of the proposed method, and runs as one process on each VM deploying VNFs on Compute. BR creates a tuple space and executes biochemical reaction equations. The concentration of PKT can be updated in accordance with the flow rate monitored at VNF. However, considering the software implementation of existing network functions and implementation difficulties, one possible alternative is to monitor the flow rate at the corresponding port of the SDN Switch. Resource allocation to VNFs, and activation and deactivation of VNFs can be performed by each VM on Compute executing VNFs in a distributed manner, in accordance with the concentrations of $MEDIATE$ and VNF of the corresponding tuple space. For ease of implementation, such VNF control can be conducted at VIM on the Controller in a centralized manner.

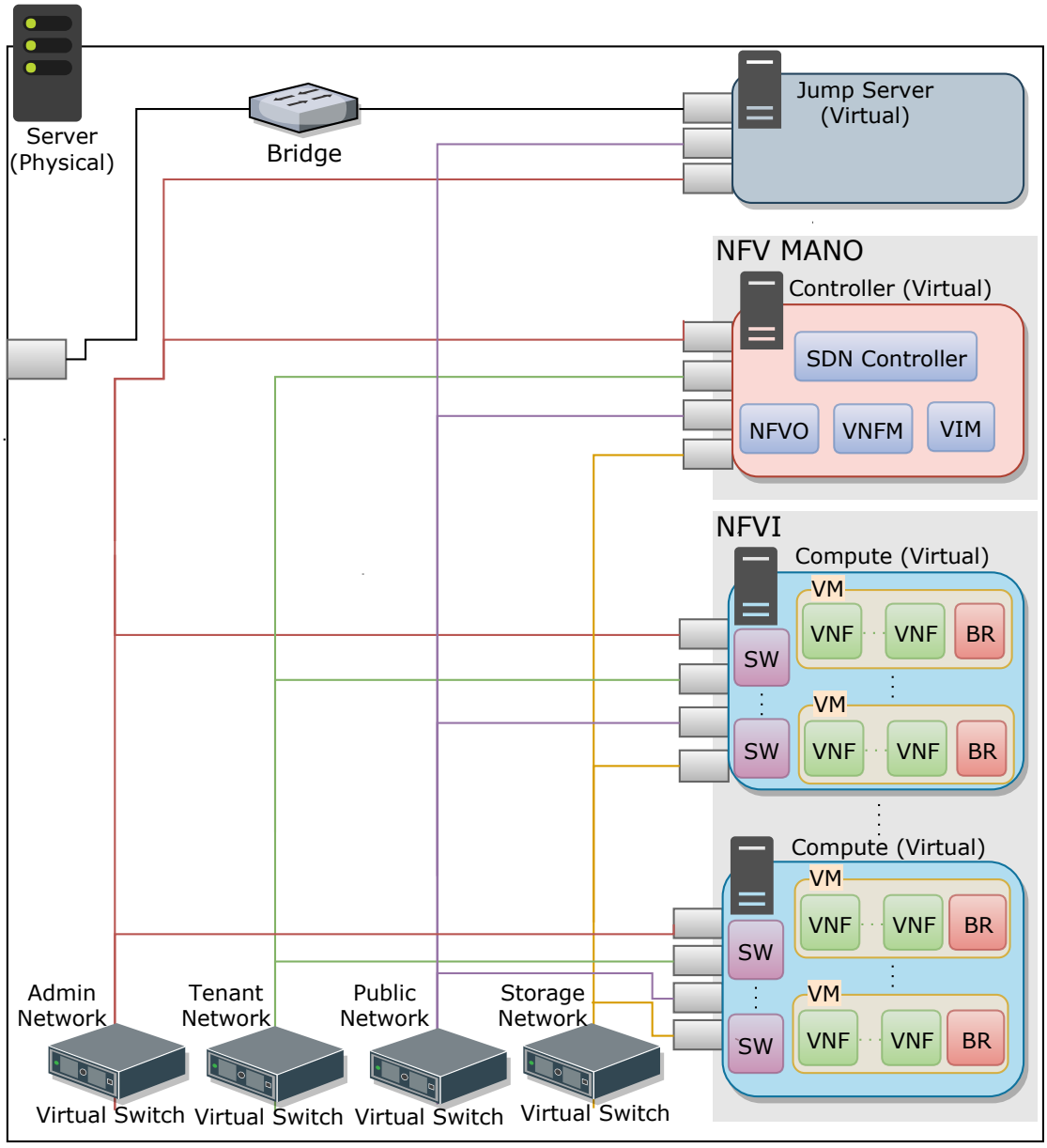


Figure 15: System configuration of OPNFV

In the proposed method, the moving direction of packets is stochastically determined as explained in SubSection 3.2.3. However, such stochastic behavior may cause a routing loop in the actual network environment. Therefore, in our implementation, flow routes are determined by SDN Controller in a centralized manner. In detail, NFVO collects the information of the concentration of $GRAD$ at each tuple space, and determines the active flow routes. The flow routes are then installed in SDN Switches via the SDN Controller.

5.4 Handle of Service Function Chaining

The mechanism of Network Service Header (NSH) [20] proposed by IETF is briefly explained. The implementation design of SFC using NSH is then described.

5.4.1 Utilization of Network Service Header

NSH is a header added to flow packets to control the flow with an SFC request in the NFV system. Figure 16 depicts the format of NSH. NSH is composed of three fields: Base Header, Service Path Header, and Context Header. Base Header contains the basic information of NSH such as version, header length, and payload information. Service Path Header contains the identifier of a flow route and the state of SFC of the flow. Context Header contains the metadata. Service Path Header is composed of Service Path Identifier (SPI) and Service Index (SI). SPI has an ID of Service Function Path (SFP), which is described below. SI has the remaining number of functions to be executed to the flow. Therefore, with a pair of SPI and SI, the next function to be executed to the flow can be identified.

Figure 17 depicts the implementation design of SFC using NSH, as described in RFC 8300 [20]. In the figure, Service Function (SF) means the function to be executed to flow packets. Service Function Forwarder (SFF) forwards flow packets to the specified SF or another SFF. SFP represents a flow route with detailed location of servers in which required SFs exist. SFP is used for forwarding packets to the designated server in accordance with the SFC request. Service Classifier (SC) is located at the entrance of the NFV system, which determines an SFP for a flow, and inserts an NSH into the packets.

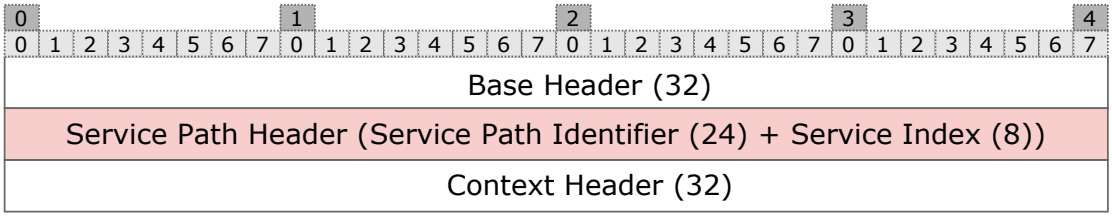


Figure 16: Network Service Header (NSH)

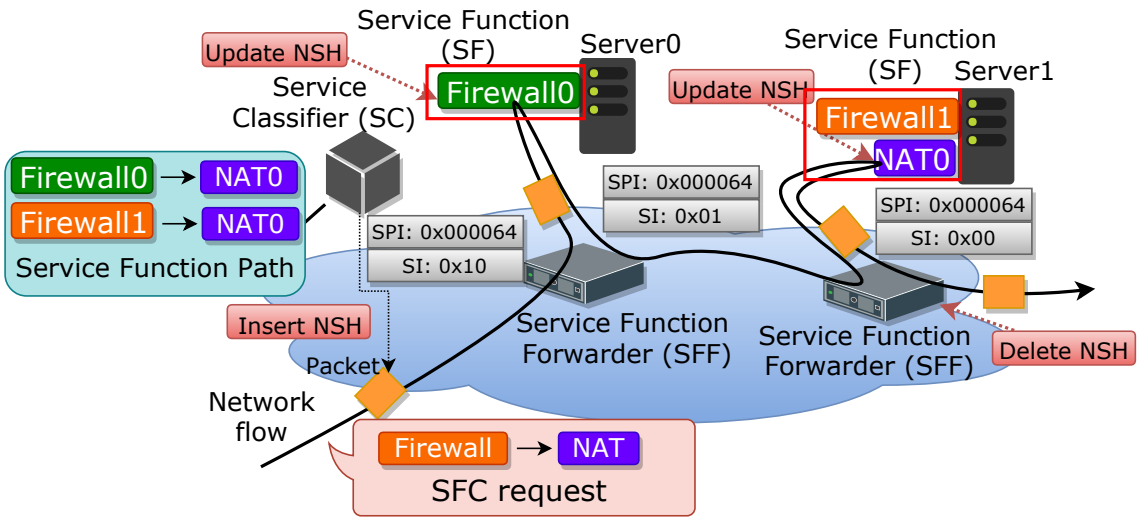


Figure 17: Implementation design of SFC using NSH in RFC 8300

In Figure 17, we consider the situation where flow packets having an SFC request of {Firewall → NAT} arrive at the NFV system. Since there are two SFs of firewall ("Firewall0" and "Firewall1") and one SF of NAT ("NAT0") in the system, the SFP for the flow could be {Firewall0 → NAT0} or {Firewall1 → NAT0}. There candidates are managed by the SC. The SC determines an SFP from the candidates for the flow and inserts NSH into the flow packets arriving at the SC. Here, NSH includes SPI of 0x000064 and SI of 0x10. When an SF is executed to a flow packet, the SF decrements the value of SI by one and forwards the packet to corresponding SFF. When SI becomes zero, the SFF deletes the NSH from the packet. Otherwise, the SFF forwards the packet to the next SF.

For ease of implementation, Service Path Header is only utilized and other two fields (Base Header and Context Header) are not implemented. To handle NSH in the NFV system, encapsulation and decapsulation functions of OpenFlow are exploited. Note that the latest version of Open vSwitch can encapsulate and decapsulate packets with NSH.

Table 5 describes samples of flow entries of inserting, updating, and deleting NSH with OpenFlow. In the table, nsh_spi and nsh_si respectively correspond to SPI and SI in NSH. The flow entry for insertion indicates that packets arriving at port 1 of SDN Switch are encapsulated with NSH including nsh_spi of 0x000064 and nsh_si of 0x10, and output to port2. The flow entry for update indicates that for packets including nsh_spi of 0x000064 and nsh_si of 0x10, the value of nsh_si is decremented to 0x01, and the packets output to port3. The flow entry for deletion indicates that packets including nsh_si of 0x00 are decapsulated and output to port 1.

Figure 18 depicts the implementation of SFC using NSH in our implementation. SFs and SFFs are respectively provided as VNFs on the servers and SDN Switches. SFP is managed by NFVO on NFV MANO. SC is implemented in NFVO and SDN Controller.

5.4.2 Stochastic Selection of Flow Routes

In NFV, a route of packets is generally detemined in a flow-by-flow manner. On the other hand, in the proposed method, as described in SubSection 3.2.3, it is stochastically detemined in a packet-by-packet manner. Therefore, the stochastic determination of flow routes is proposed to fill the gap.

Table 5: Flow entries for handling NSH

	Match Field	Action
Insertion	in_port=1, ip	encap(hdr=nsh), set_field=0x000064→ nsh_spi, set_field=0x10→nsh_si, output:2
Update	nsp_spi=0x000064, nsh_si=0x10, ip	set_field=0x01→nsh_si, output:3
Deletion	nsh_si=0x00, ip	decap(), output:1

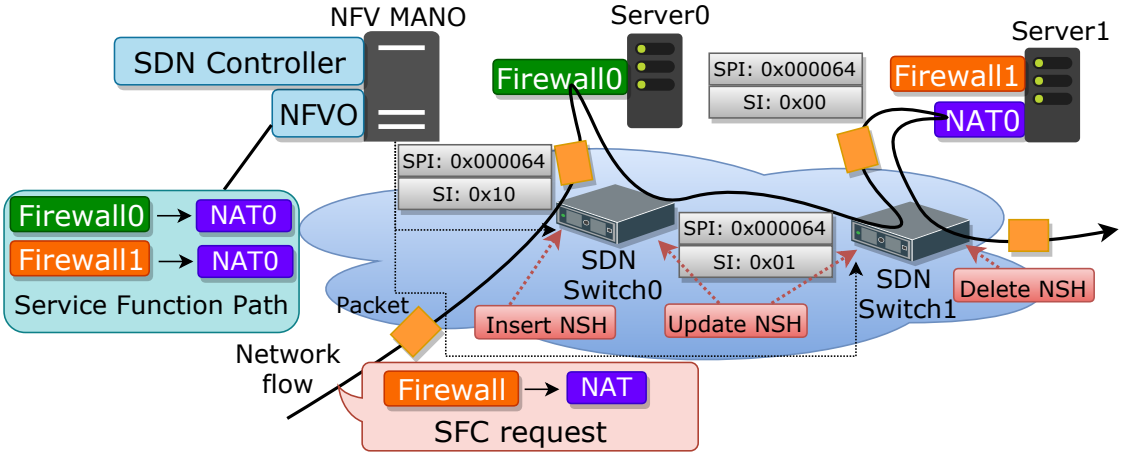


Figure 18: Implementation design of SFC using NSH

Figure 19 depicts the stochastic determination of flow route with the proposed method. In the figure, we consider the situation where flow packets have an SFC request of $\{\text{Firewall} \rightarrow \text{NAT}\}$ arrive at the NFV system. BRs are runnning on servers ("Server0" and "Server1") to create tuple spaces. Since there are two VNFs for firewall ("Firewall0" and "Firewall1") and one VNF of NAT ("NAT0"), the SFP for a flow could be $\{\text{Firewall0} \rightarrow \text{NAT0}\}$ or $\{\text{Firewall1} \rightarrow \text{NAT0}\}$. When flow packets arrive at the SDN Switch, the SFP for the flow is stochastically determined on the basis of the concentrations of $GRAD$ at both VNFs. In the figure, since the concentrations of $GRAD$ at Firewall0 and Firewall1 are respectively 2,000 and 1,000, $\{\text{Firewall0} \rightarrow \text{NAT0}\}$ or $\{\text{Firewall1} \rightarrow \text{NAT0}\}$ is respectively assigned to the flow with the probability of 0.66 and 0.33.

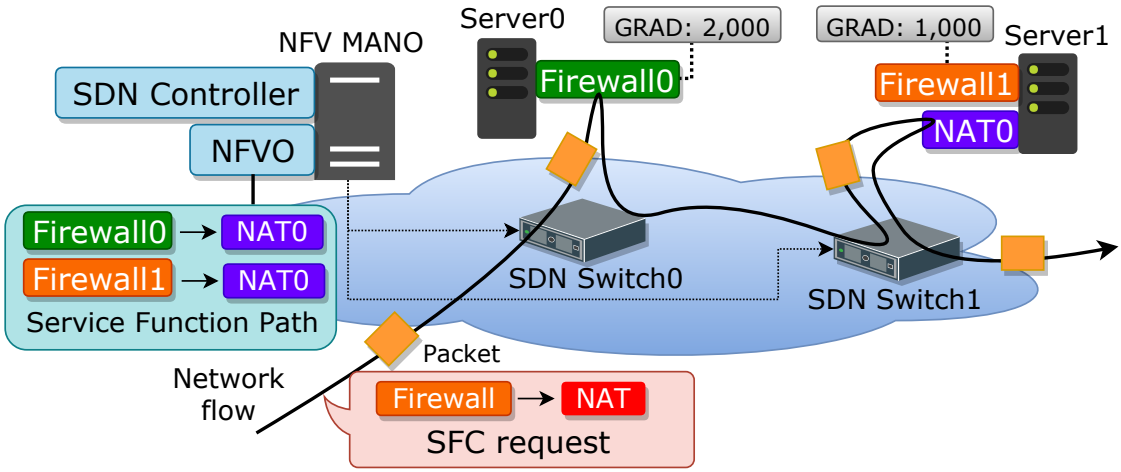


Figure 19: Stochastic determination of flow route with the proposed method

6 Conclusion and Future Work

In this thesis, we evaluated the performance of the NFV system based on biochemically-inspired tuple space model, and presented its implementation design. Specifically, we explained the tuple space model using biochemical reactions and how to apply the model to NFV system. We then performed computer simulation experiments assuming two situations in the NFV system. We confirmed that the proposed method can cope with dynamical environmental changes in the NFV system. Furthermore, we presented the implementation design of the proposed method with the NFV framework. In detail, we described the function placement of the proposed method in the NFV framework and showed the detailed implementation environment. We finally presented an example of the implementation of SFC using NSH.

For future work, we plan to extend the proposed method to include more factors of the actual network environment, such as the effect of the propagation delay and the link bandwidth between tuple spaces. It is also necessary to achieve discrete resource allocation to VNFs to accommodate the CPU core-based resource control in the current virtualized computing environment. Furthermore, it is also important to implement and evaluate the NFV system based on the proposed method based on the described design in this thesis.

Acknowledgments

I want to thank so many people for helping me during master's degree studies. I would like to express my deepest gratitude to my supervisor, Professor Morito Matsuoka. He taught my attitude towards research and also gave my support and useful comments in various situations. And I would like to show my greatest appreciation to Professor Masayuki Murata. He gave me insightful advice, guidance and encouragement. It is thanks to him that I have worked so far with trial and error in my first research. Furthermore, I would like to express the deepest appreciation to Associate Professor Go Hasegawa. He gave me elaborated guidance and invaluable firsthand advice. It is thanks to him that I remember my interest and positive feelings in research and I am able to keep my motivation for my research. I would like to appreciate to Assistant Professor Yuya Tarutani. He gave me beneficial comments about my research and life in the laboratory. I would like to students of Matsuoka Laboratory for their support of my laboratory life. Finally, I truly thank my friends and colleagues in Graduate School of Information Science and Technology of Osaka University, for their great encouragement and support.

References

[1] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, Dec. 2017.

[2] ETSI, "Network Function Virtualisation - White Paper 1." available at `https://portal.etsi.org/nfv/nfv_white_paper.pdf`.

[3] K. Neupane, R. Haddad, and L. Chen, "Next Generation Firewall for Network Security: A Survey," in *Proceedings of SoutheastCon 2018*, pp. 1–6, April 2018.

[4] D. Wing, "Network Address Translation: Extending the Internet Address Space," *IEEE Internet Computing*, vol. 14, no. 4, pp. 66–70, July 2010.

[5] A. Borkar, A. Donode, and A. Kumari, "A Survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and Protection System (IIDPS)," in *Proceedings of 2017 International Conference on Inventive Computing and Informatics (ICICI)*, pp. 949–953, Nov. 2017.

[6] V. Nguyen, A. Brunstrom, K. Grinnemo, and J. Taheri, "SDN/NFV-based Mobile Packet Core Network Architectures: A Survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1567–1602, thirdquarter 2017.

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

[8] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[9] A. Engelmann and A. Jukan, "A Reliability Study of Parallelized VNF Chaining," in *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[10] Z. Allybokus, K. Avrachenkov, J. Leguay, and L. Maggi, "Multi-path Alpha-fair Resource Allocation at Scale in Distributed Software-Defined Networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2655–2666, Dec. 2018.

[11] M. Viroli, M. Casadei, S. Montagna, and F. Zambonelli, "Spatial Coordination of Pervasive Services through Chemical-inspired Tuple Spaces," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 6, no. 2, pp. 1–24, June 2011.

[12] Marco Piraccini, "BioTuCSoN: Biochemical Extension of TuCSoN to Support Self-organising Coordination," Master's thesis, University of Bologna, Mar. 2013.

[13] G. Hasegawa, S. Sakurai, and M. Murata", "Biochemically-inspired Method for Constructing Service Space in Virtualized Network System," in *Proceedings of ICIN 2016*, Mar. 2016.

[14] Koki Sakata, "Adaptive and Autonomous Placement Method of Virtualized Network Functions based on Biochemical Reactions," Master's thesis, Osaka University, Feb. 2018.

[15] ETSI GS NFV 002, "Network Functions Virtualisation (NFV); Architectural Framework." available at `http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf`.

[16] "OPNFV." available at `https://www.opnfv.org`.

[17] "OpenStack." available at `https://www.openstack.org`.

[18] "OpenDaylight." available at `https://www.opendaylight.org`.

[19] "KVM." available at `https://www.linux-kvm.org`.

[20] "Network Service Header (NSH)." available at `https://www.rfc-editor.org/rfc/pdfrfc/rfc8300.txt.pdf`.

[21] X. Li and C. Qian, "A Survey of Network Function Placement," in *Proceedings of IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 948–953, Jan. 2016.

[22] W. Ma, C. Medina, and D. Pan, "Traffic-aware Placement of NFV Middleboxes," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Dec. 2015.

[23] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, Latency-optimal vNF Placement at the Network Edge," in *Proceedings of IEEE Conference on Computer Communications*, pp. 693–701, April 2018.

[24] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards Edge Slicing: VNF Placement Algorithms for a Dynamic & Realistic Edge Cloud Environment," in *Proceedings of IEEE Global Communications Conference*, pp. 1–6, Dec. 2017.

[25] F. B. Jemaa, G. Pujolle, and M. Pariente, "Analytical Models for QoS-driven VNF Placement and Provisioning in Wireless Carrier Cloud," in *Proceedings of ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 148–155, Nov. 2016.

[26] C. Sun, J. Bi, Z. Meng, X. Zhang, and wngxin Hu, "OFM: Optimized Flow Migration for NFV Elasticity Control," in *Proceedings of IWQoS*, pp. 1–10, June 2018.

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

[28] C. Ghribi, M. Mechtri, and D. Zeghlache, "A Dynamic Programming Algorithm for Joint VNF Placement and Chaining," in *Proceedings of ACM Workshop on Cloud-Assisted Networking*, CAN '16, (New York, NY, USA), pp. 19–24, ACM, Dec. 2016.

[29] X. Zhong, Y. Wang, X. Qiu, and S. Guo, "Cost-aware Service Function Chain Orchestration across Multiple Data Centers," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, pp. 1–7, April 2018.

[30] P. T. A. Quang, K. D. Singh, A. Bradai, and A. Benslimane, "QAAV: Quality of Service-aware Adaptive Allocation of Virtual Network Functions in Wireless Network," in *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[31] Y.-F. Wu, Y.-L. Su, and C. H.-P. Wen, "TVM: Tabular VM Migration for Reducing Hop Violations of Service Chains in Cloud Datacenters," in *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2017.

[32] S. Kim, S. Park, Y. Kim, S. Kim, and K. Lee, "VNF-EQ: Dynamic Placement of Virtual Network Functions for Energy Efficiency and QoS Guarantee in NFV," *Cluster Computing*, vol. 20, no. 3, pp. 2107–2117, Sep. 2017.

[33] M. Huang, W. Liang, Y. Ma, and S. Guo, "Throughput Maximization of Delay-sensitive Request Admissions via Virtualized Network Function Placements and Migrations," in *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 1–7, May 2018.

[34] R. Goldberg, Y. B Tewari, and T. Bhat, "Thermodynamics of Enzyme-catalyzed Reactions," *Science Direct*, vol. 20, no. 16, pp. 2874–2877, Dec. 2004.

[35] L. Michaelis, M. Leonora Menten, K. A Johnson, and R. Goody, "The Original Michaelis Constant: Translation of the 1913 Michaelis-Menten Paper," *Biochemistry*, vol. 50, no. 39, pp. 8264–8269, Sep. 2011.

[36] H. Li, Y. Cao, L. Petzold, and D. T Gillespie, "Algorithms and Software for Stochastic Simulation of Biochemical Reacting Systems," *Biotechnology Progress*, vol. 24, pp. 56–61, Feb. 2008.

[37] C. V. Rao and A. Arkin, "Stochastic Chemical Kinetics and the Quasi-steady-state Assumption: Application to the Gillespie Algorithm," *Journal of Chemical Physics*, vol. 118, no. 11, pp. 4999–5010, Aug. 2002.

[38] ETSI GS NFV 005, "Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework." available at `https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_NFV-EVE005v010101p.pdf`.

[39] "Open vSwitch." available at `https://www.openvswitch.org`.