PAPER

Model for Jointly Determining Service Function Chain Routes and Update Scheduling with State Consistency

Tomoki TAKAHASHI[†a)], ***Student Member***, **Takehiro SATO**[†], ***Member***, ***and*** **Eiji OKI**[†], ***Fellow***

SUMMARY Service chaining technology realizes that packets are processed through virtual network functions (VNFs); the set of successive VNFs is called a service function chain (SFC). Routes of SFCs need to be updated in order to resolve a quality of service degradation due to concentration of processing loads on certain VNFs. When updating the SFC routes, states of migrated VNF instances need to be kept consistent to prevent a degradation of processing accuracy of the VNF instances. Existing studies determine SFCs to be updated, their routes, and an update scheduling in multiple phases; each decision does not necessarily minimize the total time required to update the SFC routes. This paper proposes a model that jointly determines SFCs to be updated, their routes, and an update scheduling while guaranteeing state consistency. The proposed model considers the time required to update flow entries of SFC routes, the time required to migrate states of VNF instances, and the delay time required to transmit, propagate, and process packets. The objective function is to minimize the total time required to update SFC routes under a constraint of processing load balancing among VNF instances. The proposed model is formulated as an integer linear programming problem. The proposed model is compared to a benchmark model based on the existing studies. Numerical results show that the proposed model can reduce the total update time compared to the benchmark model.

key words: *network function virtualization, software-defined network, service function chain, network update*

1. Introduction

Service chaining is a technology that enables traffic to pass through a particular set of functions in sequence [1]. Service chaining is achieved through a combination of network function virtualization (NFV) and software-defined networking (SDN). NFV is a technology that allows functions provided by physical network equipment to run as software in a virtual environment. SDN is a technology that provides centralized and flexible control by separating the control and data planes with a software-based controller. In service chaining, VNFs in a virtualized network are chained together on demand. The set of concatenated VNFs is called a service function chain (SFC). Typical examples of SFCs are voice over Internet protocol (VoIP) and online gaming [2]. VoIP is a technology that converts voice signals into digital data and transmits it over the Internet. Traffic of VoIP traverses network address translation (NAT) VNFs, firewall VNFs, and a traffic monitor (TM) VNF. The TM VNF performs monitoring, tracking, and analysis of network traffic data. Online gaming connects dedicated servers and users' client machines via the Internet to play the same game at the same time. Traffic of online gaming passes through a NAT and a firewall VNF first, then a video optimization controller VNF that provides functions such as on-demand video caching, then a WAN optimization controller VNF that increases transfer speeds by data compression, and then an intrusion detection and prevention system VNF that detects and blocks threats. VNF instances of the same type have the same function.

Network situation is always changing. There can be a case where processing load is concentrated on a particular VNF instance whereas the resources of other VNF instances are not used. In this case, the processing delay in the VNF instance with the concentrated load can increase compared to usual cases, which may cause a degradation of the quality of service. To solve this problem, it is necessary to switch an SFC route that passes through the overloaded VNF instance to another route so that the load of the VNF instances can be evenly distributed.

Each VNF instance that makes up an SFC holds a state of the processed packets [3]. For example, an intrusion detection system (IDS) VNF holds information as a state such as transmission control protocol (TCP) sequence numbers, partially reassembled hyper text transfer protocol (HTTP) payloads, and flow connection counters [3]. When an SFC route is updated, the state held by the VNF instance on the route before update and that held by the VNF instance on the route after update need to be consistent. If the state is identical between the VNF instances on the route before and after the SFC update, this condition is called state consistency. When the states of the two VNF instances are consistent, the results of packets processing in the two VNF instances are the same. If state consistency is not guaranteed when updating the SFC route, the accuracy of processing in the VNF instance on the updated route may degrade.

To guarantee state consistency, the state held by the VNF instance on the route before update needs to be migrated to the VNF instance on the new route, which is called state migration. Packets cannot be processed in the VNF instance where the state migration is in progress. New packets arriving at the VNF instance are forwarded to a buffer in the controller and temporarily stored. The time required for the state migration differs for each type of VNF instance [3]. Even among VNF instances of the same type, the size of the migrated state may vary depending on the amount of packets processed by the VNF instance; the state migration time can be different for each VNF instance. If an excessive

Manuscript received December 5, 2023.
Manuscript revised March 6, 2024.
Manuscript publicized June 28, 2024.

[†]Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: ttakahashi@icn.cce.i.kyoto-u.ac.jp

DOI: 10.23919/transcom.2023EBP3203

number of SFC routes are updated, time required for update may affect the quality of service or buffer overflow may occur. When SFC routes are updated for load balancing, time required for update and buffer usage needs to be suppressed.

Existing works [3]–[7] studied SFC updating with state consistency. Gember-Jacobson et al. [3] presented a control plane architecture called OpenNF, which updates SFC routes. This work designed an application programming interface to transfer a state of one VNF instance to another VNF instance. Existing works [4], [5] presented models that select which SFC routes to update (SFC selection) and allocate updated routes to SFCs (routing allocation) in order to perform load balancing among VNF instances. Zhang et al. [4] presented a model named Co-scaler, which performs an SFC selection and a routing allocation so that state migrations of VNF instances that comprise an SFC can be performed in parallel. Sun et al. [5] presented a model that performs SFC selection and routing allocation under VNF scale out, scale in, and load balancing situations. Fan et al. [6] presented a model that determines an SFC selection and a routing allocation, and an update scheduling. The work in [7] presented a scheduling model that minimizes the total time required to update all SFCs (the total update time) while guaranteeing state consistency. When updating an SFC composed of multiple VNF instances, this model performs state migrations in parallel as in [4]. The model in [7] can provide a scheduling that allows multiple SFCs to be updated at the same time.

Since the model in [7] only obtains an update scheduling, an SFC selection and a routing allocation need to be performed in advance by the models such as [4], [5]. The model in [6] determines an SFC selection and an update scheduling after determining a routing allocation; these determinations are separated into multiple phases. When the models in [6], [7] are used to obtain an update scheduling, determinations of an SFC selection, a routing allocation, and an update scheduling are performed not in a single phase but in multiple phases. The decision obtained in a phased manner does not necessarily minimize the total update time.

It is required for a network operator to update SFC routes as soon as possible when the load concentrates on a specific VNF instance and the service quality degrades. If the time required for the SFC update is long, quality of service may continue to degrade; for example, long processing delay and packet loss caused by load imbalance are prolonged. A question arises; how do we obtain an SFC selection, a routing allocation, and an update scheduling that minimize the total update time to achieve load balancing with state consistency?

This paper proposes a model that determines SFCs to be updated, their routes, and an update scheduling at the same time. The objective is to minimize the total update time under a constraint of balancing processing load of VNF instances. The proposed model considers simultaneous state migrations of an SFC composed of multiple VNF instances and simultaneous updates of multiple SFCs, as in [7]. The proposed model is formulated as an integer linear programming (ILP) problem. The proposed model is evaluated in terms of the total update time, the number of flows to be updated, the standard deviation of processing capacity for VNF instances, the amount of used buffer capacity, the number of rounds to be used, and the computation time. Numerical results show that the proposed model can reduce the total update time compared to the benchmark model. As the capacity utilization ratio increases, the standard deviation tends to increase and the total update time and the number of updated flows tend to decrease. Numerical results also show that, as the buffer capacity increases, the total update time and the number of used rounds tend to decrease.

The rest of this paper is organized as follows. Section 2 describes the related works. Section 3 describes the overview and the problem formulation of the proposed model. Section 4 shows numerical results. Finally, Sect. 6 concludes this paper.

2. Related Works

2.1 SFC Update Problem

A problem of updating routes of SFCs is called an SFC update problem. The SFC update problem consists of an SFC selection, a routing allocation, and an update scheduling. SFCs are updated in situations such as VNF scale out, VNF scale in, load balancing, failure recovery, and VNF upgrading [5].

There are existing works on SFC update problem [4]–[8], which are summarized in Table 1. Hereinafter, a VNF on a route of SFC before update is called an original VNF. A VNF on a route of SFC after update is called a target VNF. Wen et al. [8] presented a slice recovery and reconfiguration model that performs a routing allocation. This model does not consider to guarantee state consistency when a slice is updated. Zhang et al. [4] presented a model called Co-scaler, which performs an SFC selection and a routing allocation. Co-scaler considers a situation of VNF scale out; a state is migrated to a newly deployed VNF instance. Co-scaler minimizes the number of packets affected by SFC updates while balancing the load so that the standard deviation of the VNF instance load is below a threshold value. Sun et al. [5] presented a model that performs an SFC selection and a routing allocation. This model minimizes the difference between penalty and benefit while balancing the load so that the load variance of VNF instances is below a threshold value. Here, the penalty is defined as buffer usage and update time. The benefit is defined as saved VM cost obtained by discarding virtual machines with low utilization. Fan et al. [6] presented a model that performs an SFC selection, a routing allocation, and an update scheduling. This model minimizes the usage ratio for all VNF instances so that the total delay of the SFC update and total time required to update flow entries for each VNF instance are below a threshold value. This model performs an SFC update in multiple phases; routing allocation is performed first and then SFC selection and an update scheduling are performed. Takahashi et al. [7] presented a model that performs an update scheduling. The model in [7] minimizes the total time required to update all

Table 1 Summary of existing works related to SFC update problem.

Reference	Problem	Situation	Where to migrate state	Objective	State consistency	Buffer capacity constraints	Others
Wen et al. [8]	Routing allocation	Failure recovery	Non-failed VNF instance	Minimizing total bandwidth consumption			Model considers stochastic demands with robust optimization
Zhang et al. [4]	SFC selection and routing allocation	VNF scale out	Newly deployed VNF instance	Minimizing total traffic of SFCs to be updated	✓		
Sun et al. [5]	SFC selection and routing allocation	VNF scale out, VNF scale in, and load balancing	Depends on situation	Minimizing migration costs, i.e., difference between penalty and benefit	✓	✓	
Fan et al. [6]	SFC selection, routing allocation, and update scheduling	VNF scale out, VNF scale in, and load balancing	Depends on situation	Minimizing usage ratio for all VNF instances	✓	✓	Model solves the problems in a phased manner
Takahashi et al. [7]	Update scheduling	VNF scale out, VNF scale in, load balancing, and VNF upgrading	Depends on situation	Minimizing total time required to update all SFCs	✓	✓	Model considers buffer splitting
This work	SFC selection, routing allocation, and update scheduling	VNF scale out, VNF scale in, and load balancing	Depends on situation	Minimizing total time required to update all SFCs	✓	✓	Model solves the problems simultaneously

SFCs. This model considers a state migration time for each VNF instance, whereas the model in [6] assumes that a state migration time is constant for all VNF instances.

Compared to the models in [4]–[7], the proposed model jointly performs an SFC selection, a routing allocation, and an update scheduling. Similar to the models in [4], [6], [7], the proposed model considers to perform multiple state migrations of an SFC at the same time. Also, similar to the model in [7], the proposed model considers simultaneous updates of multiple SFCs and defines a state migration time for each VNF instance.

2.2 Consistent VNF Migration Methods

A method to migrate a VNF for an SFC update with state consistency is called a consistent VNF migration method. The consistent VNF migration has two properties: loss-freedom and order-preservation. Loss-freedom is the property of not dropping packets when updating an SFC [9]. Order-preservation is the property of processing packets in the same order during updating an SFC as in the case of no SFC update [9].

There are existing works on consistent VNF migration methods [3], [10]–[15]. Gember-Jacobson et al. [3] presented a control plane architecture called OpenNF. OpenNF forwards newly-arriving packets during a state migration to the buffer in the controller. After the state migration is completed, newly-arriving packets are stored in a buffer of a target VNF. The target VNF first processes packets from the controller and then processes packets that directly arrive at the target VNF. Gember-Jacobson et al. [10] presented an improved OpenNF framework that performs peer-to-peer transfers and packet reprocessing. States and packets are transferred directly from an original VNF to a target VNF on a peer-to-peer basis. During the state migration, new packets arriving at the original VNF are copied and forwarded to the buffer in the target VNF for reprocessing. Sherry et al. [11] presented a system called Fault-Tolerant MiddleBox that recovers a middlebox to its pre-failure state in the event of the middlebox failure. A snapshot and packet logs are stored in a stable storage and used to obtain outputs from the backup middlebox. The packet logs are made from the time the snapshot is stored until a middlebox failure occurs. Clark et al. [12] presented a design for VM live migration. The state migration is performed by iteratively transferring snapshots. Nobach et al. [13] presented a system called slim migration (SliM) that transfers states by using a statelet that represents information about changes in a VNF state. SliM transfers a snapshot and a statelet from an original VNF to a target VNF. Packets arriving at the original VNF during the transfer of a snapshot are processed to create statelets. Wang et al. [14] presented a framework for protecting a VNF in a secure operating environment based on software guard extensions (SGX) and Click, which is a modular software architecture for making network applications. This framework provides a security-aware state synchronization procedure. Wang et al. [15] presented two state update schemes, which are called a controller-forwarding based scheme and a tagging-based scheme. These schemes suppress migration time when updating an SFC. Similar to OpenNF [3], the controller-forwarding based scheme uses a buffer of the controller and a target VNF instance. In the tagging-based scheme, newly-arriving packets during a state migration are forwarded to the buffer in the target VNF without using the buffer in the controller.

The proposed model guarantees state consistency by migrating a state of an original VNF to a target VNF as the methods in [3], [10], [12]–[15]. The proposed model uses buffers for newly-arriving packets at the controller and VNF instance; the proposed model adopts the method in [3] and the method of controller-forwarding based scheme in [15]. Different from the method in [10] and the method of tagging-based scheme in [15], the proposed model forwards the state via the controller.

3. Proposed Model

3.1 Overview

The proposed model jointly determines SFCs to be updated, their updated routes, and the update scheduling. The objective is to minimize the total time required to update the SFCs with the constraint that a processing load of each VNF instance is distributed. The SFCs can be simultaneously updated with state consistency. The update scheduling is expressed as an allocation of the SFCs to rounds. SFC updates allocated to the same round are completed within the duration time of that round. All SFC updates are completed by the final round. The time required to update an SFC is defined as the time from the start of its update until when a packet arrives at the destination host via the updated route, which will be explained precisely later. When multiple SFC updates are allocated to a round, the duration time of that round is determined by the time of the SFC update which is completed last.

A system structure considered in the proposed model is shown in Fig. 1. Note that Fig. 1 focuses on the control plane of the system; physical links connecting physical nodes to each other are omitted. Each physical node has a physical machine and an SDN switch. The physical node is simply called *host* hereafter. On the physical machine, a VM runs on virtualization software. A VNF runs on the VM. The status of each physical machine is sent to the controller, and the controller keeps track of the entire network. The controller decides SFCs to be updated, their updated routes, and update scheduling of those SFCs by the proposed model. Then, the controller performs the update operation according to the obtained scheduling; state migrations of the VNFs and flow entry updates of the SDN switches are performed.

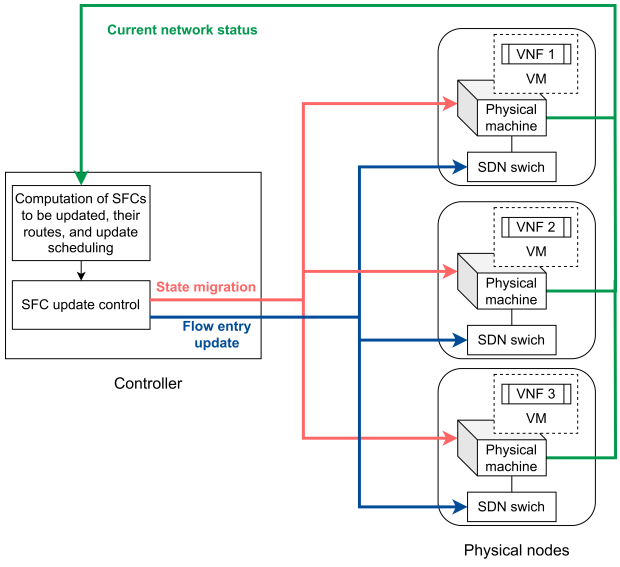
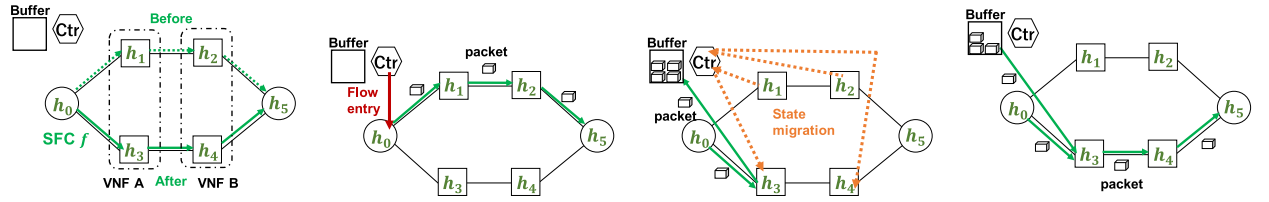


Fig. 1 System structure.

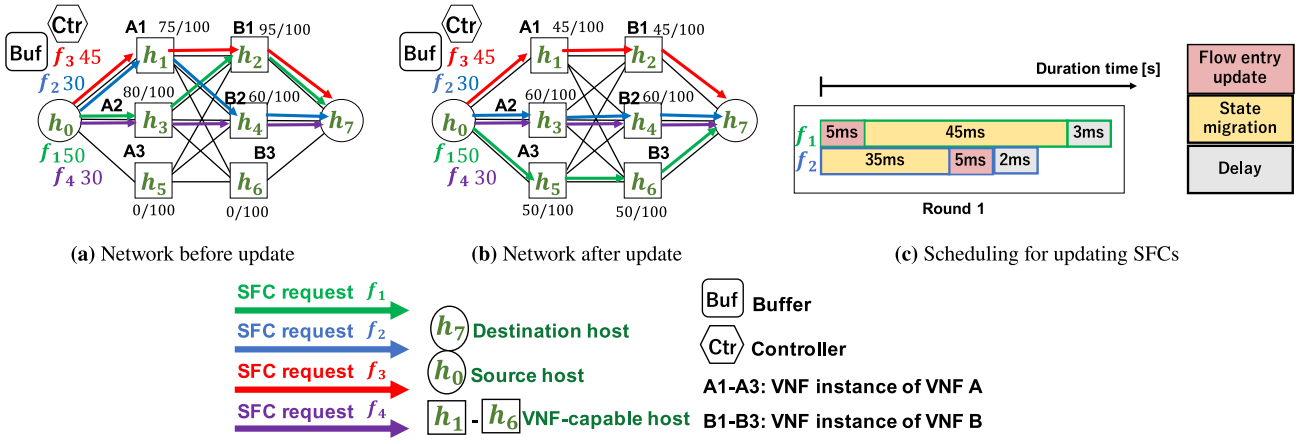
The following scenarios are considered in the proposed model. An SFC consists of one or more VNFs and provides a service by traversing VNF instances. One or more flows pass through an SFC. Each flow maintains its state in a VNF instance that makes up an SFC. When a state migration is performed for the VNF instance, the states of all flows passing the SFC are migrated. The time it takes to push a packet into a link is considered as a transmission delay for the link. Traffic arrives at the SFC at a fixed rate [packets/s]. A VNF instance immediately processes a packet in a fixed time [s] and forwards the packet to the next host; this fixed time for the VNF instance is considered as a processing delay. The reasons for fixing the values of transmission rate and packet processing time are as follows. We consider that an SFC has a maximum transmission rate. In the proposed model, the transmission rate is set to this maximum transmission rate to tolerate variations in the amount of actual network traffic. We also consider that the minimum processing capacity to run a VNF is reserved at each host; the time required for packet processing is guaranteed to be less than a certain value. The time it takes for a packet to propagate on a link is considered as a propagation delay. A controller is located on one of the hosts on the physical network. Communications between the controller and each host, such as flow entry updates, state migrations, and forwarding of packets from hosts to a buffer, are performed in a control plane. Packets pass through an SFC in a data plane. The control plane uses the same physical link as the data plane; for each link, a propagation delay is the same for the control plane and the data plane. On the other hand, for each link, a transmission capacity is allocated separately for the control plane and the data plane. The proposed model uses buffers of the controller and VNF instances in the same way as the consistent migration methods in [3], [15]. The proposed model assumes that buffer overflow at the VNF instance does not occur. The buffer of the VNF instance is used to temporarily store newly-arriving packets to achieve packet order preservation. If the buffer capacity of the VNF instance is insufficient, the new packets can be dropped; state consistency cannot be guaranteed. Therefore, we assume that the VNF instance has a buffer with sufficient capacity to prevent the buffer overflow. The proposed model considers a buffer capacity constraint of the controller.

An example of an SFC update procedure by the proposed model is shown in Fig. 2. "Ctr" represents the controller and h_0–h_5 represent hosts. h_0 and h_5 are a source and destination hosts, respectively. Instances of VNF A are installed at h_1 and h_3. Instances of VNF B are installed at h_2 and h_4. There is one SFC, named f, in the network. Figure 2(a) shows the routes of f before and after the update. The route of SFC f is updated from the dotted green route to the solid green route. Figure 2(b)–2(d) show packet flow on the network. Figure 2(b) shows the packet flow at the time of starting the SFC update. Before the SFC update, packets are forwarded via the route h_0-h_1-h_2-h_5. A flow entry is installed from the controller to host h_0; it is the time when the SFC update begins. As a result of the flow entry



(a) Routes before and after update **(b)** Start of SFC update (flow entry update) **(c)** During state migration **(d)** Completion of SFC update (Buffered packet reaches destination host)

Fig. 2 Example of SFC update procedure by proposed model.



(a) Network before update **(b)** Network after update **(c)** Scheduling for updating SFCs

Fig. 3 Example of determination of SFCs to be updated, their routes, and update scheduling for multiple SFCs.

update, the route of SFC f is switched from h_0-h_1-h_2-h_5 to h_0-h_3-h_4-h_5. Figure 2(c) shows the packet flow during state migration of VNF A and VNF B. The states of VNFs A and B are migrated from h_1 to h_3 and h_2 to h_4, respectively, via the controller. During the state migration, new packets of f arriving at host h_3 on the updated route are forwarded and stored in the buffer until the state migration of f is completed. Figure 2(d) shows the packet flow at the completion of the SFC update. Packets of f stored in the buffer are forwarded from the buffer to host h_3, processed by hosts h_3 and h_4, and then forwarded to destination host h_5. The time when a buffered packet of the controller reaches destination host h_5 for the first time is the end of the SFC update; it is the time when packet processing is resumed at destination host h_5. While the buffered packets are being processed at host h_3, new packets arrive at h_3. These new packets are buffered in the VNF instance of host h_3 until the end of the processing for packet order preservation. After the SFC update is completed, the new arriving packets from host h_0 are sent to destination host h_5 along the solid green route in Fig. 2(a). Note that when updates of multiple SFCs are performed simultaneously, one or more SFCs are updated in a procedure such that the flow entry update is performed after the state migration. Instead of waiting for the completion of other SFCs' flow entry update, the state migration is performed in advance. In this case, the time when the SFC update starts is the time when the state migration starts.

Table 2 Time required for state migration of VNF A and VNF B for each SFC.

	f_1	f_2	f_3	f_4
State migration time of VNF A	40 ms	35 ms	40 ms	25 ms
State migration time of VNF B	45 ms	40 ms	65 ms	30 ms

An example of determination of SFCs to be updated, their routes, and an update scheduling for multiple SFCs is shown in Fig. 3. Each VNF instance has a processing capacity of 100 [packets/s]. There are four types of SFCs: f_1, f_2, f_3, and f_4. The transmission rate of f_1, f_2, f_3, and f_4 are 50, 30, 45, and 30 [packets/s], respectively. Time required for the state migration of VNF A and VNF B for each SFC is shown in Table 2. Figure 3(a) shows the network before update. Figure 3(a) shows that SFCs use 75, 80, 0, 95, 60, and 0 of processing capacity [packets/s] for VNF instances A1, A2, A3, B1, B2, and B3, respectively; the processing load is concentrated on VNF instances A2 and B1. Figure 3(b) shows the network after update; SFC routes to be updated and their updated routes are obtained by the proposed model. In Fig. 3(b), SFCs use 45, 60, 50, 45, 60, and 50 of processing capacity [packets/s] for VNF instance A1, A2, A3, B1, B2, and B3, respectively; the processing

load is distributed among the VNF instances by updating routes of f_1 and f_2. State of VNF instances A2 and B1 in f_1 and VNF instance A1 in f_2 needs to be migrated to VNF instances A3, B3, and A2, respectively. Figure 3(c) shows the scheduling for updating SFCs f_1 and f_2 obtained by the proposed model. Updates of SFCs f_1 and f_2 are allocated to one round; SFC updates are performed simultaneously. The time required to update each SFC consists of three parts: the time required to update the flow entry in a switch to the updated route, the time required to perform the state migration, and the delay required to send the packets stored in the buffer to the destination host on the updated route. The time required to perform the state migration is the maximum value of the state migration time among VNF instances that are to be updated. The time to complete the update of f_1 is 53 ms, which is the sum of 5 ms for the flow entry update, 45 ms for the state migration, and 3 ms for delay. 45 ms for the state migration comes from the state migration time for VNF B. The time to complete the update of f_2 is 42 ms, which is the sum of 35 ms for the state migration, 5 ms for the flow entry update, and 2 ms for delay. 35 ms for the state migration comes from the state migration time for VNF A. The duration time for a round is the maximum of the SFC update times allocated to that round. The duration time for round 1 is 53 ms, which comes from f_1.

3.2 Notation

In the proposed model, sets, parameters, and variables are defined. Notations of sets and parameters are shown in Table 3. Notations of variables are shown in Table 4.

3.2.1 Sets

Sets are defined as follows. A set of SFCs is denoted by F. A set of candidate paths that SFC $f \in F$ can use after the update is denoted by P_f. P_f contains a route before the update of SFC $f \in F$. A set of VNF instances is denoted by S. A set of hosts is denoted by H. A set of links is denoted by E. A set of rounds is denoted by $Q \in [0, \sum_{f \in F} l_f]$. In this model, SFCs which are not updated are allocated to round zero; SFCs to be updated are allocated to round one or later. SFC updates start from round one.

3.2.2 Parameters

Parameters for SFCs are as follows. The number of VNF instances which make up SFC $f \in F$ is denoted by l_f. SFC f is represented by a chain of VNF instances in the order of a increasing value of $i \in [1, l_f]$. A transmission rate of SFC $f \in F$ is denoted by r_f.

Parameters for VNF instances are as follows. The time required to migrate a state of SFC $f \in F$ from VNF instance $s \in S$ to $s' \in S$ is denoted by $\tau^{\text{state}}_{fss'}$. A threshold value of capacity utilization ratio for VNF instances is denoted by σ_0, where $0 \leq \sigma_0 \leq 1$. A processing capacity of VNF instance $s \in S$ is denoted by c^{proc}_s. The time required for VNF instance $s \in S$ to process one packet is denoted by τ^{proc}_s.

Table 3 Notations of sets and parameters.

Sets	Definitions
F	Set of SFCs.
Q	Set of rounds.
S	Set of VNF instances.
H	Set of hosts.
P_f	Set of candidate paths that SFC $f \in F$ can use after update.
E	Set of links.
Parameters	**Definitions**
l_f	Number of VNF instances of SFC $f \in F$.
$\tau^{\text{state}}_{fss'}$	Time required to migrate a state of SFC $f \in F$ from VNF instance $s \in S$ to $s' \in S$.
σ_0	Threshold value of capacity utilization ratio for VNF instances.
r_f	Transmission rate of SFC $f \in F$.
τ^{entry}_{fh}	Time required to install a flow entry of SFC $f \in F$ to a switch on host $h \in H$.
α_{efp}	Binary parameter such that $\alpha_{efp} = 1$ if link $e \in E$ belongs to path $p \in P_f$ of SFC $f \in F$ and 0 otherwise.
β_{sfp}	Binary parameter such that $\beta_{sfp} = 1$ if VNF instance $s \in S$ belongs to path $p \in P_f$ of SFC $f \in F$ and 0 otherwise.
γ_{fp}	Binary parameter such that $\gamma_{fp} = 1$ if path $p \in P_f$ of SFC $f \in F$ is the same as the path before update.
c^{trans}_e	Transmission capacity of link $e \in E$.
c^{proc}_s	Processing capacity of VNF instance $s \in S$.
c^{buffer}	Capacity of the buffer.
ψ_{fpis}	Binary parameter such that $\psi_{fpis} = 1$ if path $p \in P_f$ of SFC $f \in F$ passes through VNF instance $s \in S$ as the i-th VNF instance and 0 otherwise.
τ^{prop}_s	Total propagation delay from the controller to VNF instance $s \in S$.
τ^{trans}_s	Total transmission delay from the controller to VNF instance $s \in S$.
$\hat{\tau}^{\text{prop}}_{sfp}$	Total propagation delay from VNF instance $s \in S$ to a destination host of path $p \in P_f$, where $f \in F$.
$\hat{\tau}^{\text{trans}}_{sfp}$	Total transmission delay from VNF instance $s \in S$ to a destination host of path $p \in P_f$, where $f \in F$.
τ^{proc}_s	Time required for VNF instance $s \in S$ to process one packet.
η_{hfp}	Binary parameter such that $\eta_{hfp} = 1$ if path $p \in P_f$ of SFC $f \in F$ switches at host $h \in H$ from the path before update and 0 otherwise.

Parameters of the network related to the update time are as follows. The time required to install a flow entry of SFC $f \in F$ to a switch on host $h \in H$ is denoted by τ^{entry}_{fh}. The total propagation delay from the controller to VNF instance $s \in S$ is denoted by τ^{prop}_s. The total transmission delay from the controller to VNF instance $s \in S$ is denoted by τ^{trans}_s. The total propagation delay from VNF instance $s \in S$ to a destination host of path $p \in P_f$ is denoted by τ^{prop}_{sfp}, where $f \in F$. The total transmission delay from VNF instance $s \in S$ to a destination host of path $p \in P_f$ is denoted by $\tau^{\text{trans}}_{sfp}$, where $f \in F$.

Other parameters related to the network are as follows. A capacity of the buffer is denoted by c^{buffer}. A transmission capacity of link $e \in E$ is denoted by c^{trans}_e. α_{efp} is a parameter such that $\alpha_{efp} = 1$ if link $e \in E$ belongs to path $p \in P_f$ of SFC $f \in F$ and 0 otherwise. β_{sfp} is a parameter such that $\beta_{sfp} = 1$ if VNF instance $s \in S$ belongs to path $p \in P_f$ of SFC $f \in F$ and 0 otherwise. γ_{fp} is a parameter such that $\gamma_{fp} = 1$ if path $p \in P_f$ of SFC $f \in F$ is the same as the path before update. ψ_{fpis} is a parameter such that $\psi_{fpis} = 1$ if path $p \in P_f$ of SFC $f \in F$ passes through VNF instance $s \in S$ as the i-th VNF instance and 0 otherwise. η_{hfp} is a parameter such that $\eta_{hfp} = 1$ if path $p \in P_f$ of SFC $f \in F$ switches at host $h \in H$ from the path before update and 0

Table 4 Notations of variables.

Variables	Definitions
t_q	Time required at round $q \in Q\backslash\{0\}$.
x_{fq}	Binary decision variable such that $x_{fq} = 1$ if SFC $f \in F$ is allocated to round $q \in Q$ and 0 otherwise.
x'_{fq}	Binary decision variable such that $x'_{fq} = 1$ if the update of SFC $f \in F$ is already completed at the start of round $q \in Q\backslash\{0\}$ and 0 otherwise.
$\hat{x}_q$	Binary decision variable such that $\hat{x}_q = 1$ if one or more SFC updates are performed at round $q \in Q\backslash\{0\}$ and 0 otherwise.
y_{fp}	Binary decision variable such that $y_{fp} = 1$ if SFC $f \in F$ is allocated to path $p \in P_f$ and 0 otherwise.
y'_{fis}	Binary decision variable such that $y'_{fis} = 1$ if an updated route of SFC $f \in F$ passes through VNF instance $s \in S$ in the i-th order and 0 otherwise, where $i \in [1, l_f]$.
t_f	Time when an update of SFC $f \in F$ is completed, by regarding the time at which each round starts as zero.
t_f^{entry}	Time required to update a flow entry for SFC $f \in F$.
t_f^{state}	Time required to migrate all states of SFC $f \in F$.
t_f^{delay}	Sum of propagation delay, transmission delay, and processing delay caused by updating SFC $f \in F$.
t_f^{start}	Time when a flow entry update for SFC $f \in F$ starts, by regarding the time at which each round starts as zero.
$\hat{t}_{qk}^{\mathrm{start}}$	Time when the k-th flow entry update for SFC $f \in F$ starts at round $q \in Q\backslash\{0\}$, by regarding the time at which each round starts as zero, where $k \in [1, \lvert F\rvert]$.
a_{fqk}	Binary decision variable such that $a_{fqk} = 1$ if a flow entry for SFC $f \in F$ is updated in the k-th order at round $q \in Q\backslash\{0\}$ and 0 otherwise, where $k \in [1, \lvert F\rvert]$.
b_{fi}	Binary decision variable such that $b_{fi} = 1$ if a state migration of the i-th VNF instance of SFC $f \in F$ is performed and 0 otherwise, where $i \in [1, l_f]$.
w_{sf}	Binary decision variable such that $w_{sf} = 1$ if VNF instance $s \in S$ is the first VNF instance when a state migration of SFC $f \in F$ is performed and 0 otherwise.
u_{fi}	Binary decision variable such that $u_{fi} = 1$ if the i-th VNF instance of SFC $f \in F$ is the first VNF instance and 0 otherwise.
σ	Maximum capacity utilization ratio among all VNF instances, where $0 \le \sigma \le 1$.
L_s	Used amount of processing capacity of VNF instance $s \in S$.
c_f^{used}	Total used buffer capacity by all SFCs at the time when an update of SFC $f \in F$ is completed.
$\hat{c}_{f\hat{f}}^{\mathrm{used}}$	Amount of buffer capacity used by SFC $\hat{f} \in F$ at the time when packets of SFC $f \in F$ start to be forwarded from the buffer.
$\hat{y}_{fpq}^{\mathrm{orig}}$	Binary decision variable such that $\hat{y}_{fpq}^{\mathrm{orig}} = 1$ if SFC $f \in F$ uses the same path $p \in P_f$ as before an update at round $q \in Q\backslash\{0\}$ and 0 otherwise.
$\hat{y}_{fpq}^{\mathrm{target}}$	Binary decision variable such that $\hat{y}_{fpq}^{\mathrm{target}} = 1$ if SFC $f \in F$ uses path $p \in P_f$ which is used after an update at round $q \in Q\backslash\{0\}$ and 0 otherwise.

otherwise.

3.2.3 Variables

Variables related to round allocation in an update scheduling are as follows. x_{fq} is a binary decision variable such that $x_{fq} = 1$ if SFC $f \in F$ is allocated to round $q \in Q$ and 0 otherwise. x'_{fq} is a binary decision variable such that $x'_{fq} = 1$ if the update of SFC $f \in F$ is already completed at the start of round $q \in Q\backslash\{0\}$ and 0 otherwise. $\hat{x}_q$ is a binary decision variable such that $\hat{x}_q = 1$ if one or more SFC updates are performed at round $q \in Q\backslash\{0\}$ and 0 otherwise. a_{fqk} is a binary decision variable such that $a_{fqk} = 1$ if a flow entry for SFC $f \in F$ is updated in the k-th order at round $q \in Q\backslash\{0\}$ and 0 otherwise, where $k \in [1, |F|]$. w_{sf} is a binary decision variable such that $w_{sf} = 1$ if VNF instance $s \in S$ is the first VNF instance (the VNF instance that forwards packets to the buffer) when a state migration of SFC $f \in F$ is performed and 0 otherwise. u_{fi} is a binary decision variable such that $u_{fi} = 1$ if the i-th VNF instance of SFC $f \in F$ is the first VNF instance and 0 otherwise.

Variables for the duration time of each round in an update scheduling are as follows. The time required at round $q \in Q\backslash\{0\}$ is denoted by decision variable t_q. The time when an update of SFC $f \in F$ is completed, by regarding the time at which each round starts as zero, is denoted by t_f. The time required to update a flow entry for SFC $f \in F$ is denoted by decision variable t_f^{entry}. The time required to migrate all states of SFC $f \in F$ is denoted by decision variable t_f^{state}. The sum of propagation delay, transmission delay, and processing delay caused by updating SFC $f \in F$ is denoted by decision variable t_f^{delay}. The time when a flow entry update for SFC $f \in F$ starts, by regarding the time at which each round starts as zero, is denoted by decision variable t_f^{start}. The time when the k-th flow entry update for SFC $f \in F$ starts at round $q \in Q\backslash\{0\}$, by regarding the time at which each round starts as zero, is denoted by decision variable $\hat{t}_{qk}^{\mathrm{start}}$, where $k \in [1, |F|]$.

Other variables related to an update scheduling are as follows. The total used buffer capacity by all SFCs at the time when an update of SFC $f \in F$ is completed is denoted by decision variable c_f^{used}. The amount of buffer capacity used by SFC $\hat{f} \in F$ at the time when packets of SFC $f \in F$ start to be forwarded from the buffer is denoted by decision variable $\hat{c}_{f\hat{f}}^{\mathrm{used}}$. $\hat{y}_{fpq}^{\mathrm{orig}}$ is a binary decision variable such that $\hat{y}_{fpq}^{\mathrm{orig}} = 1$ if SFC $f \in F$ uses the same path $p \in P_f$ as before an update at round $q \in Q\backslash\{0\}$ and 0 otherwise. $\hat{y}_{fpq}^{\mathrm{target}}$ is a binary decision variable such that $\hat{y}_{fpq}^{\mathrm{target}} = 1$ if SFC $f \in F$ uses path $p \in P_f$ which is used after an update at round $q \in Q\backslash\{0\}$ and 0 otherwise.

Variables related to an SFC selection are as follows. y_{fp} is a binary decision variable such that $y_{fp} = 1$ if SFC $f \in F$ is allocated to path $p \in P_f$ and 0 otherwise. y'_{fis} is a binary decision variable such that $y'_{fis} = 1$ if an updated route of SFC $f \in F$ passes through VNF instance $s \in S$ in the i-th order and 0 otherwise, where $i \in [1, l_f]$. b_{fi} is a binary decision variable such that $b_{fi} = 1$ if a state migration of the i-th VNF instance of SFC $f \in F$ is performed and 0 otherwise, where $i \in [1, l_f]$. The maximum capacity utilization ratio among all VNF instances is denoted by decision variable σ, where $0 \le \sigma \le 1$. The used amount of processing capacity of VNF instance $s \in S$ is denoted by decision variable L_s.

3.3 Problem Formulation

3.3.1 Formulation of Optimization Problem

The problem that determines SFCs to be updated, their routes, and an update scheduling at the same time is for-

mulated. This problem is called SFC routing selection and scheduling problem (SFC-RS) hereinafter.

The objective function is formulated as follows:

$$\min \sum_{q \in Q \backslash \{0\}} t_q + \epsilon_0 \sum_{f \in F} \hat{c}_{ff}^{\text{used}}. \tag{1}$$

The first term of (1) represents the total update time. The second term of (1) represents the sum of buffer capacity used by all SFCs. ϵ_0 is a parameter that is set to such a small positive value that a value of the second term is sufficiently smaller than that of the first term; it ensures that SFCs to be updated and their routes that use the minimum amount of buffer capacity are determined in the case where there exists more than one solution such that the first term is minimized.

$$t_q = \max_{f \in F} t_f x_{fq}, \forall q \in Q \backslash \{0\}, \tag{2a}$$

$$t_f = \begin{cases} t_f^{\text{entry}} + t_f^{\text{state}} + t_f^{\text{delay}} & \text{if} \quad t_f^{\text{start}} = 0, \\ t_f^{\text{start}} + t_f^{\text{entry}} + t_f^{\text{delay}} & \text{otherwise}, \end{cases}$$

$$\forall f \in F, \tag{2b}$$

$$t_f^{\text{entry}} = \sum_{p \in P_f} y_{fp} \sum_{h \in H} \eta_{hfp} \tau_{fh}^{\text{entry}}, \forall f \in F, \tag{2c}$$

$$t_f^{\text{delay}} = \sum_{s \in S} w_{sf} (\tau_s^{\text{prop}} + \hat{c}_{ff}^{\text{used}} \tau_s^{\text{trans}} + \sum_{p \in P_f} y_{fp} (\hat{\tau}_{sfp}^{\text{prop}} + \hat{\tau}_{sfp}^{\text{trans}})) + \sum_{i \in [1, l_f]} u_{fi} \sum_{p \in P_f} y_{fp} \sum_{j \in [i, l_f]} \sum_{s \in S} \psi_{fpjs} \tau_s^{\text{proc}}, \forall f \in F, \tag{2d}$$

$$t_f^{\text{state}} = \max_{i \in [1, l_f]} \sum_{p \in P_f} \sum_{s \in S} \gamma_{fp} \psi_{fpis} \sum_{p' \in P_f} \sum_{s' \in S} y_{fp'} \psi_{fp'is'} \tau_{fss'}^{\text{state}}, \forall f \in F, \tag{2e}$$

$$t_f^{\text{start}} = \sum_{k \in [1, |F|]} \sum_{q \in Q \backslash \{0\}} \hat{t}_{qk}^{\text{start}} a_{fqk}, \forall f \in F, \tag{2f}$$

$$\hat{t}_{q1}^{\text{start}} = 0, \forall q \in Q \backslash \{0\}, \tag{2g}$$

$$\hat{t}_{qk}^{\text{start}} = \begin{cases} \hat{t}_{q(k-1)}^{\text{start}} + \sum_{f \in F} a_{fq(k-1)} t_f^{\text{entry}} \\ \quad \text{if} \quad \hat{t}_{q(k-1)}^{\text{start}} + \sum_{f \in F} a_{fq(k-1)} t_f^{\text{entry}} \\ \qquad > \sum_{f \in F} a_{fqk} t_f^{\text{state}}, \\ \sum_{f \in F} a_{fqk} t_f^{\text{state}} \quad \text{otherwise}, \end{cases}$$

$$\forall k \in [2, |F|], q \in Q \backslash \{0\}. \tag{2h}$$

The duration time of each round t_q is expressed as follows. The time at which round q starts is set to 0. Equation (2a) shows that t_q is expressed as the time when the last update is completed among the SFCs allocated to round $q \in Q \backslash \{0\}$. Equation (2b) shows that, if the flow entry update of SFC $f \in F$ is performed at the round start time, the time when SFC f finishes update t_f is the sum of the time taken for the flow entry update t_f^{entry}, the state migration t_f^{state}, and the delay t_f^{delay}; otherwise, t_f is the sum of the start time of the flow entry update t_f^{start}, t_f^{entry}, and t_f^{delay}. If the flow entry update of SFC $f \in F$ is not performed at the round start time, the state migration is completed before the start of the flow entry update of SFC f. Equation (2c) shows that the time required to update the flow entry for SFC $f \in F$ is represented by the time required to update the flow entry for host $h \in H$ where the path of SFC f switches to path $p \in P_f$. Equation (2d) shows that the delay caused by updating SFC $f \in F$ is represented by the sum of the delay due to packet forwarding and processing between the controller and the destination host on the updated path. Equation (2e) shows that the time required to migrate the state of SFC $f \in F$ is the maximum state migration time among VNF instances that make up SFC f. Equation (2f) shows the time when the flow entry update of SFC $f \in F$ starts. Equation (2g) shows that the time to start the first flow entry update is round start time 0. The *if* statement in (2h) represents whether the sum of $\hat{t}_{q(k-1)}^{\text{start}}$ and the time taken for the $(k-1)$-th flow entry update is greater than the time taken for the state migration of SFC that performs the k-th flow entry update. When the *if* statement is true, the start of the k-th flow entry update is waited until the time when the $(k-1)$-th flow entry update is completed. Otherwise, the k-th flow entry update starts at the time when the state migration of the SFC that performs the k-th flow entry update is completed.

$$\sum_{k \in [1, |F|]} a_{fqk} = x_{fq}, \forall f \in F, q \in Q \backslash \{0\}, \tag{3a}$$

$$\sum_{f \in F} a_{fqk} \leq 1, \forall k \in [1, |F|], q \in Q \backslash \{0\}, \tag{3b}$$

$$\sum_{f \in F} a_{fq(k-1)} \geq \sum_{f \in F} a_{fqk}, \forall k \in [2, |F|], q \in Q \backslash \{0\}. \tag{3c}$$

Equation (3a) guarantees that the flow entry update for SFC $f \in F$ is performed once at round $q \in Q \backslash \{0\}$. Equation (3b) guarantees that flow entry updates are not performed simultaneously, but one by one at round $q \in Q \backslash \{0\}$. Equation (3c) guarantees that the order of flow entry updates k at round $q \in Q \backslash \{0\}$ is allocated starting with the smallest order of k.

$$b_{fi} = \begin{cases} 0 & \text{if} \quad \sum_{s \in S} y'_{fis} \sum_{p' \in P_f} \gamma_{fp'} \psi_{fp'is} = 1, \\ 1 & \text{otherwise}, \end{cases}$$

$$\forall f \in F, i \in [1, l_f], \tag{4a}$$

$$y'_{fis} = \sum_{p \in P_f} y_{fp} \psi_{fpis}, \forall f \in F, i \in [1, l_f], s \in S. \tag{4b}$$

Equation (4a) guarantees that no state migration is required if the path used after the update ($p \in P_f$) and that used before the update ($p' \in P_f$) pass through the same VNF instance. Otherwise, a state migration is required. Equation (4b) shows that y'_{fis} is a binary variable that is 1 if the path used after the update of SFC $f \in F$ passes through VNF instance $s \in S$ as the i-th VNF instance and 0 otherwise.

$$w_{sf} = \sum_{i \in [1, l_f]} u_{fi} \sum_{p \in P_f} y_{fp} \psi_{fpis}, \forall s \in S, f \in F, \tag{5a}$$

$$u_{f1} = b_{f1}, \forall f \in F, \tag{5b}$$

$$u_{fi} = \begin{cases} 1 & \text{if} \quad \sum_{j\in[2,i]} b_{f(j-1)} = 0, b_{fi} = 1, \\ 0 & \text{otherwise}, \end{cases}$$

$$\forall f \in F, i \in [2, l_f]. \tag{5c}$$

Equations (5a), (5b), and (5c) determine the values of w_{sf} and u_{fi}. Equation (5c) means that the VNF instance with the smallest value of i among the VNF instances performing the state migration of SFC $f \in F$ is the first VNF instance.

$$\sum_{q\in Q} x_{fq} = 1, \forall f \in F. \tag{6}$$

Equation (6) guarantees that the state migration of SFC $f \in F$ is performed at only one round.

$$x_{f0} = \sum_{p\in P_f} y_{fp}\gamma_{fp}, \forall f \in F. \tag{7}$$

Equation (7) shows that if the path used after the update of SFC $f \in F$ is identical to that used before the update, the update of SFC f is allocated to round 0; the path of SFC f is not updated. Note that SFC updates start from round 1.

$$\sum_{p\in P_f} y_{fp} = 1, \forall f \in F. \tag{8}$$

Equation (8) guarantees that only one path in the set of candidate paths P_f of SFC $f \in F$ is selected as the updated path.

$$L_s \le \sigma c_s^{\text{proc}}, \forall s \in S, \tag{9a}$$

$$\sigma \le \sigma_0, \tag{9b}$$

$$L_s = \sum_{f\in F}\sum_{p\in P_f} \beta_{sfp} y_{fp} r_f, \forall s \in S. \tag{9c}$$

Equations (9a)–(9c) guarantee that the processing load is distributed so that the capacity utilization ratio is below the threshold value for all VNF instances. L_s is the used processing capacity at VNF instance $s \in S$ after all SFC updates are completed.

$$c_f^{\text{used}} \le c^{\text{buffer}}, \forall f \in F, \tag{10a}$$

$$c_f^{\text{used}} = \sum_{\hat{f}\in F} \hat{c}_{f\hat{f}}^{\text{used}}, \forall f \in F, \tag{10b}$$

$$\hat{c}_{f\hat{f}}^{\text{used}} = \begin{cases} r_{\hat{f}} \sum_{q\in Q\setminus\{0\}} x_{fq}(t_f - t_f^{\text{delay}} - a_{\hat{f}q1} t_{\hat{f}}^{\text{entry}}) \\ \quad \text{if} \quad t_f - t_f^{\text{delay}} < \sum_{q\in Q\setminus\{0\}} x_{fq} x_{\hat{f}q} t_{\hat{f}}, \\ 0 \quad \text{otherwise}, \end{cases}$$

$$\forall f \in F, \hat{f} \in F. \tag{10c}$$

Equation (10a) guarantees that the used buffer capacity by all SFCs at the time when the update of SFC $f \in F$ is completed does not exceed the maximum buffer capacity. Equation (10b) shows that c_f^{used} is the sum of the used buffer capacity by all SFCs at the time when the update of SFC $f \in F$ is completed. The *if* statement in (10c) is whether SFC $\hat{f} \in F$ completes its update after the time SFC $f \in F$ starts forwarding packets from the controller to the first VNF instance. When the *if* statement is true, SFC $\hat{f}$ uses the buffer of the controller at the time when SFC f starts forwarding packets. Otherwise, SFC $\hat{f}$ does not use the buffer at the time. The amount of buffer capacity used by SFC $\hat{f}$ at the time when SFC f starts forwarding packets is the product of transmission rate of SFC $\hat{f}$ and the amount of time SFC $\hat{f}$ uses the buffer before the time SFC f starts forwarding packets.

$$\hat{y}_{fpq}^{\text{orig}} = \gamma_{fp}(1 - x'_{fq}), \forall f \in F, p \in P_f, q \in Q\setminus\{0\}, \tag{11a}$$

$$\hat{y}_{fpq}^{\text{target}} = y_{fp}(x_{fq} \vee x'_{fq}), \forall f \in F, p \in P_f, q \in Q\setminus\{0\}, \tag{11b}$$

$$\sum_{f\in F} r_f \sum_{p\in P_f} \alpha_{efp}(\hat{y}_{fpq}^{\text{orig}} \vee \hat{y}_{fpq}^{\text{target}}) \le c_e^{\text{trans}},$$

$$\forall e \in E, q \in Q\setminus\{0\}, \tag{11c}$$

$$\sum_{f\in F} r_f \sum_{p\in P_f} \beta_{sfp}(\hat{y}_{fpq}^{\text{orig}} \vee \hat{y}_{fpq}^{\text{target}}) \le c_s^{\text{proc}},$$

$$\forall s \in S, q \in Q\setminus\{0\}. \tag{11d}$$

Equations (11a) and (11b) determine the values of $\hat{y}_{fpq}^{\text{orig}}$ and $\hat{y}_{fpq}^{\text{target}}$, respectively. Equation (11c) guarantees that the link capacity used in each round does not exceed the upper bound. Equation (11d) guarantees that the amount of used capacity of VNF instance in each round does not exceed the upper limit. If a path of SFC is updated in a round, it is assumed that the link capacity and VNF instance capacity are allocated to the paths used before and after update in that round.

$$x'_{f1} = 0, \forall f \in F, \tag{12a}$$

$$x'_{fq} = \coprod_{q'\in[1,q-1]} x_{fq'}, \forall f \in F, q \in Q\setminus\{0,1\}. \tag{12b}$$

Equations (12a) and (12b) determine the value of x'_{fq}. Equation (12a) guarantees that no SFCs to be updated are completed by the start of round 1. Note that $\coprod_{q'\in[1,q-1]} x_{fq'}$ represents taking all *OR*s of $x_{fq'}$ for $q' \in [1, q-1]$.

$$\hat{x}_q = \coprod_{f\in F} x_{fq}, \forall q \in Q\setminus\{0\}, \tag{13a}$$

$$\hat{x}_{(q-1)} \ge \hat{x}_q, \forall q \in Q\setminus\{0,1\}. \tag{13b}$$

Equation (13a) determines the value of $\hat{x}_q$. Equation (13b) guarantees that SFC updates are allocated to rounds in ascending order of round number. Note that $\coprod_{f\in F} x_{fq}$ represents taking all *OR*s of x_{fq} for $f \in F$.

3.3.2 Formulation in ILP Form

The problem in (1)–(13b) is non-linear since it involves *if*

statements, a *max* statement, *OR* operators, and products of decision variables. The non-linear equations need to be transformed into linear equations to formulate the problem as an ILP problem.

if and *max* statements are linearized by following (14)–(20). In (15), we can replace a *max* statement as $\geq$ to get the largest value on the left-hand side [7]. This is because t_f^{state} is set to such a small value that it can satisfy the constraint of $\geq$ by minimizing the objective function in (1). In (18) and (19), since the decision variables that make up *if* statements are all binary variables and their conditions contain 0 or 1, *if* statements can be expressed using *OR* operations. An *OR* operation can be linearized by an existing method [16]. In (14), (16), and (20), in order to linearize *if* statements, we introduce a sufficiently large value, B_1, that is larger than or equal to the upper bound of the duration time of a single round and binary variables. Using such a sufficiently large value and binary variables to linearize mathematical statements is a typical approach, which existing studies [16]–[19] adopted, for example. How to express linear formulation depends on each mathematical statement. We use this approach to fit our specific situation with customization.

$$t_f = (1-\theta_f)t_f^{\text{state}} + \theta_f t_f^{\text{start}} + t_f^{\text{entry}} + t_f^{\text{delay}}, \forall f \in F, \tag{14a}$$

$$t_f^{\text{start}} > -(1-\theta_f), \forall f \in F, \tag{14b}$$

$$t_f^{\text{start}} \leq B_1\theta_f, \forall f \in F. \tag{14c}$$

Equations (14a)–(14c) correspond to (2b). θ_f is a binary decision variable such that $\theta_f = 1$ if a condition $t_f^{\text{start}} > 0$ is satisfied and 0 otherwise.

$$t_f^{\text{state}} \geq \sum_{p\in P_f}\sum_{s\in S}\gamma_{fp}\psi_{fpis}\sum_{p'\in P_f}\sum_{s'\in S}y_{fp'}\psi_{fp'is'}\tau_{fss'}^{\text{state}}, \forall f \in F, i \in [1, l_f]. \tag{15}$$

Equation (15) corresponds to (2e).

$$\hat{t}_{qk}^{\text{start}} = \mu_{qk}(\hat{t}_{q(k-1)}^{\text{start}} + \sum_{f\in F} a_{fq(k-1)}t_f^{\text{entry}}) + (1-\mu_{qk}) \sum_{f\in F} a_{fqk}t_f^{\text{state}}, \forall k \in [2, |F|], q \in Q\backslash\{0\}, \tag{16a}$$

$$\zeta_{qk} = \hat{t}_{q(k-1)}^{\text{start}} + \sum_{f\in F}(a_{fq(k-1)}t_f^{\text{entry}} - a_{fqk}t_f^{\text{state}}), \forall k \in [2, |F|], q \in Q\backslash\{0\}, \tag{16b}$$

$$\zeta_{qk} \geq -B_1(1-\mu_{qk}), \forall k \in [2, |F|], q \in Q\backslash\{0\}, \tag{16c}$$

$$\zeta_{qk} \leq B_1\mu_{qk}, \forall k \in [2, |F|], q \in Q\backslash\{0\}. \tag{16d}$$

Equations (16a)–(16d) correspond to (2h). μ_{qk} is a binary decision variable such that $\mu_{qk} = 1$ if $\zeta_{qk} > 0$ and 0 if $\zeta_{qk} < 0$; the value of μ_{qk} is 'don't care' if $\zeta_{qk} = 0$. Let t^{upper} denote the upper bound. The duration time of a single round does not exceed t^{upper}. t^{upper} is obtained by following (17a)–(17d):

$$t^{\text{upper}} = \sum_{f\in F}(t'^{\text{entry}}_f + t'^{\text{state}}_f + t'^{\text{delay}}_f), \tag{17a}$$

$$t'^{\text{entry}}_f = \max_{h\in H}\tau_{fh}^{\text{entry}}, \forall f \in F, \tag{17b}$$

$$t'^{\text{state}}_f = \max_{s,s'\in S}\tau_{fss'}^{\text{state}}, \forall f \in F, \tag{17c}$$

$$t'^{\text{delay}}_f = \max_{s\in S, p\in P_f}(\tau_s^{\text{prop}} + c^{\text{buffer}}\tau_s^{\text{trans}} + \hat{\tau}_{sfp}^{\text{prop}} + \hat{\tau}_{sfp}^{\text{trans}} + \tau_s^{\text{proc}}), \forall f \in F. \tag{17d}$$

Equation (17a) shows that t^{upper} is set to the time required in the case where all SFC updates are performed in serial. Equations (17b), (17c), and (17d) determine the maximum values of t'^{entry}_f, t'^{state}_f, and t'^{delay}_f for the time required for the flow entry update, the state migration, and the delay, respectively.

$$b_{fi} = 1 - \coprod_{s\in S} y'_{fis}\sum_{p'\in P_f}\gamma_{fp'}\psi_{fp'is}, \forall f \in F, i \in [1, l_f]. \tag{18}$$

Equation (18) corresponds to (4a).

$$u_{fi} = (1-\phi_{fi})b_{fi}, \forall f \in F, i \in [2, l_f], \tag{19a}$$

$$\phi_{fi} = \coprod_{j\in[2,i]} b_{f(j-1)}, \forall f \in F, i \in [2, l_f]. \tag{19b}$$

Equations (19a)–(19b) correspond to (5c). ϕ_{fi} is a binary decision variable represented by $\phi_{fi} = \coprod_{j\in[2,i]} b_{f(j-1)}$.

$$\hat{c}_{f\hat{f}}^{\text{used}} = r_{\hat{f}}\sum_{q\in Q\backslash\{0\}} x_{fq}t_{f\hat{f}q}^{\text{used}}\nu_{f\hat{f}q}, \forall f \in F, \hat{f} \in F, q \in Q\backslash\{0\}, \tag{20a}$$

$$t_{f\hat{f}q}^{\text{used}} = t_f - t_f^{\text{delay}} - a_{\hat{f}q1}t_{\hat{f}}^{\text{entry}}, \forall f \in F, \hat{f} \in F, q \in Q\backslash\{0\}, \tag{20b}$$

$$\sum_{q\in Q\backslash\{0\}} x_{fq}x_{\hat{f}q}t_{\hat{f}} - t_f + t_f^{\text{delay}} > -B_1(1-\nu_{f\hat{f}q}), \forall f \in F, \hat{f} \in F, q \in Q\backslash\{0\}, \tag{20c}$$

$$\sum_{q\in Q\backslash\{0\}} x_{fq}x_{\hat{f}q}t_{\hat{f}} - t_f + t_f^{\text{delay}} \leq B_1\nu_{f\hat{f}q}, \forall f \in F, \hat{f} \in F, q \in Q\backslash\{0\}. \tag{20d}$$

Equations (20a)–(20d) correspond to (10c). $\nu_{f\hat{f}q}$ is a binary decision variable such that $\nu_{f\hat{f}q} = 1$ if a condition $\sum_{q\in Q\backslash\{0\}} x_{fq}x_{\hat{f}q}t_{\hat{f}} - t_f + t_f^{\text{delay}} > 0$ is satisfied and 0 otherwise.

The product of binary variables $w_{sf}y_{fp}$, $u_{fi}y_{fp}$, $b_{fi}\phi_{fi}$, $x_{fq}\nu_{f\hat{f}q}$, $y_{fp}x_{fq}$, and $y_{fp}x'_{fq}$ can be linearized. The *OR* operations of (11b), (11c), (12b), (13a), (18), and (19b) can be linearized. The product of a non-negative real variable and a binary variable can be linearized by using a constant value. $w_{sf}\hat{c}_{ff}^{\text{used}}$ can be linearized by using B_2, which is a sufficiently large value that satisfies $B_2 \geq c^{\text{buffer}}$. $t_f x_{fq}$, $\theta_f t_f^{\text{state}}$, $\theta_f t_f^{\text{start}}$, $\hat{t}_{qk}^{\text{start}}a_{fqk}$, $a_{fq(k-1)}t_f^{\text{entry}}$, $a_{fqk}t_f^{\text{state}}$, $\hat{t}_{q(k-1)}^{\text{start}}\mu_{qk}$, $a_{fq(k-1)}t_f^{\text{entry}}\mu_{qk}$, $a_{fqk}t_f^{\text{state}}\mu_{qk}$, $a_{\hat{f}q1}t_{\hat{f}}^{\text{entry}}$, $x_{fq}x_{\hat{f}q}t_{\hat{f}}$, $x_{fq}x_{\hat{f}q}$, and $t_{f\hat{f}q}^{\text{used}}x_{fq}\nu_{f\hat{f}q}$ can be linearized by

using B_1. For example, let ι, κ, and λ be binary variables and ρ and ω be non-negative real variables. $\lambda = \iota\kappa$ is linearized by: $\lambda \leq \iota$, $\lambda \leq \kappa$, and $\lambda \geq \iota + \kappa - 1$. $\lambda = \iota \vee \kappa$ is linearized by: $\lambda \leq \iota + \kappa$ and $\lambda \geq \frac{1}{2}(\iota + \kappa)$ [16]. $\omega = \rho\iota$ is linearized by: $\omega \geq \rho + B_{\mathrm{L}}(\iota - 1)$, $\omega \leq \rho$, and $\omega \leq B_{\mathrm{L}}\iota$, where B_{L} is a sufficiently large value to ensure that its value is not lower than ρ [17].

4. Numerical Results

4.1 Evaluation Environment

This section evaluates the proposed model in terms of the total update time, the number of rounds to be used, the amount of used buffer capacity, the standard deviation of processing load for VNF instances, and the computation time required to obtain the optimal solution. As a benchmark model, we use a strategy that determines SFCs to be updated, their routes, and an update scheduling in two separate phases.

Results are obtained by changing the number of SFCs deployed on a network. The number of VNF types that make up an SFC is set to two: IDS VNF and firewall VNF. The time required for IDS VNF and firewall VNF to process one packet is set to 40 and 71 μs, respectively [20]. IDS and firewall VNF instances have a processing capacity of 25000 and 14000 [packets/s], respectively [20]. The transmission rate for each SFC is randomly set in the range of 2000 to 4000 [packets/s] [7]. The capacity of each link is set to 5 Gbps [21] in both of the data plane and the control plane.

In this evaluation, we assume that the packets flowing on each link follow the distribution in Table 5 [22]. Let c_e^{band} [bit/s] denote the capacity of link $e \in E$. The capacity of link $e \in E$ can be converted to c_e^{trans} [packets/s] using the following (21):

$$c_e^{\mathrm{trans}} = \frac{c_e^{\mathrm{band}}}{\sum_{m \in M} g_m \frac{v_m}{100}}, \forall e \in E, \tag{21}$$

where M is a set of packet size ranges shown in Table 5, g_m is the average of the minimum and maximum values of packet size range $m \in M$, and v_m is the percentage of packet size range $m \in M$.

The state migration time of the i-th VNF instance of SFC $f \in F$ is obtained by:

$$\tau_{fss'}^{\mathrm{state}} = \begin{cases} 0 & \text{if} \quad s = s', \\ n_f(\delta_s^{\mathrm{state}} + \delta_{ss'}^{\mathrm{trans}}) + \delta_{ss'}^{\mathrm{prop}} & \text{otherwise}, \end{cases}$$
$$\forall f \in F, s \in S, s' \in S, \tag{22}$$

where n_f is the number of flows passing through SFC $f \in F$, $\delta_s^{\mathrm{state}}$ is the time required to migrate the state of one flow in VNF instance $s \in S$, $\delta_{ss'}^{\mathrm{trans}}$ is the transmission delay required to migrate a state of one flow from VNF instance $s \in S$ to $s' \in S$, and $\delta_{ss'}^{\mathrm{prop}}$ is the propagation delay required to migrate a state from VNF instance $s \in S$ to $s' \in S$. n_f is randomly set in the range of 10–100 [5]. We set the state migration time of one flow in the IDS and the firewall VNFs based on that of PRADS, which is considered in [3] for an asset monitor VNF. The time required to perform a state migration of one flow in PRADS is 0.4 ms. Bro IDS and iptables which are different types of VNFs considered in [3] take 5 and 0.5 times longer than PRADS, respectively; we set the value of $\delta_s^{\mathrm{state}}$ assuming that the time required to migrate a state of one flow in IDS and firewall VNFs is 2 and 0.2 ms, respectively. $\delta_{ss'}^{\mathrm{trans}}$ is obtained by dividing the size of a state for one flow by the link capacity. The size of the state in IDS and firewall VNFs is set as follows. The size of the state for one flow in the firewall VNF is 2 KB [23]. We assume that the time required to migrate a state of one flow is proportional to the size of the migrated state; that of the state of one flow in the IDS VNF is set to $2\,\mathrm{KB} \times 5 \times \frac{1}{0.5} = 20\,\mathrm{KB}$.

The time it takes to install a flow entry from the controller to each host is obtained by:

$$\tau_{fh}^{\mathrm{entry}} = n_f(\hat{\delta}^{\mathrm{entry}} + \hat{\delta}_h^{\mathrm{trans}}) + \hat{\delta}_h^{\mathrm{prop}}, \forall f \in F, h \in H, \tag{23}$$

where $\hat{\delta}^{\mathrm{entry}}$ is the time required to install one flow entry on a switch, $\hat{\delta}_h^{\mathrm{trans}}$ is the transmission delay required to send a packet from the controller to host $h \in H$ for one flow entry update, and $\hat{\delta}_h^{\mathrm{prop}}$ is the propagation delay required for a packet to propagate from the controller to host $h \in H$. $\hat{\delta}^{\mathrm{entry}}$ is set to 0.7 ms [24]. $\hat{\delta}_h^{\mathrm{trans}}$ is obtained by dividing the packet size for a flow entry update by the link capacity. The packet size for a flow entry update is assumed to be equal to the size of one flow entry in a switch; the packet size is set to 356 bits [25].

Since the proposed model considers scenarios where processing load is concentrated on a particular VNF instance and load balancing is required, SFC routes before an update, given as a parameter, are determined to satisfy following (24a)–(24b):

$$\sigma_0 < \max_{s \in S} \frac{L_s^{\mathrm{before}}}{c_s^{\mathrm{proc}}} \leq 1, \tag{24a}$$

$$L_s^{\mathrm{before}} = \sum_{f \in F} \sum_{p \in P_f} \beta_{sfp} \gamma_{fp} r_f, \forall s \in S. \tag{24b}$$

Equation (24a) guarantees that the maximum capacity utilization ratio of VNF instances is greater than threshold value σ_0. Equation (24b) represents that L_s^{before} is the processing load of VNF instance $s \in S$ before the update. In addition, the SFC routes before the update are determined so that an used link capacity does not exceed the upper limit.

The number of flows to be updated, the standard deviation of processing capacity for VNF instances, and the amount of used buffer capacity are obtained by $\sum_{f \in F} n_f \sum_{q \in Q \setminus \{0\}} x_{fq}$, $\sqrt{\frac{1}{|S|} \sum_{s \in S} (L_s - \bar{L}_s)^2}$, and $\sum_{f \in F} \hat{c}_{ff}^{\mathrm{used}}$, respectively. Note that $\bar{L}_s$ is the average value of L_s.

Table 5 Packet size distribution [22].

Packet size range [byte]	64–79	80–159	160–319	320–639	640–1279	1280–1518
Percentage	28.24	7.30	1.15	1.20	0.92	61.19

The results in this section are obtained with a 95% confidence interval that is less than 15% of the mean in terms of the total update time. Trials that obtain an infeasible solution are omitted. The ILP problems of the proposed model and benchmark model are solved with CPLEX Interactive Optimizer 20.1.0.0 [26] on the computer equipped with AMD EPYC 7502P and 128 GB RAM. The upper limit of the computation time is set to 20000 s.

4.2 Benchmark Model

The benchmark model is developed based on [4], [7]. The benchmark model finds a solution of SFC-RS in two phases by dividing it into two problems. In the first phase, the benchmark model solves a problem that determines SFCs to be updated and their routes. This problem is called the SFC routing selection problem (SFC-R) hereinafter. In the second phase, the benchmark model solves a problem that determines an update scheduling with the solution obtained by SFC-R as input. This problem is called the SFC scheduling problem (SFC-S) hereinafter. Both SFC-R and SFC-S are formulated as ILP problems. The computation time for the benchmark model is the sum of the time to solve SFC-R and SFC-S.

4.2.1 SFC-R

SFC-R is based on the model presented in [4]. In the presented model, the objective function to be minimized is the number of packets affected by SFC updates. In SFC-R, the objective function is changed to the total update time it takes to update the SFCs one by one, which is the sum of the flow entry update time, the state migration time, and the delay time. The delay time consists of transmission, propagation, and processing delays for SFCs to be updated. With this change in the objective function, constraints that represent the state migration time and the delay time are considered in SFC-R.

The formulation of SFC-R is given by:

$$\min \sum_{f \in F} x_f (t_f^{\text{state}} + t_f^{\text{entry}} + t_f^{\text{delay}}), \tag{25a}$$

$$\text{subject to} \quad \text{(2c), (2e), (4a)–(4b), (5a)–(5c), (8), (9a)–(9c),} \tag{25b}$$

$$x_f = 1 - \sum_{p \in P_f} y_{fp} \gamma_{fp}, \forall f \in F, \tag{25c}$$

$$t_f^{\text{delay}} = \sum_{s \in S} w_{sf} (\tau_s^{\text{prop}} + c_f^{\text{used}} \tau_s^{\text{trans}} + \sum_{p \in P_f} y_{fp} (\hat{\tau}_{sfp}^{\text{prop}} + \hat{\tau}_{sfp}^{\text{trans}})) + \sum_{i \in [1, l_f]} u_{fi} \sum_{p \in P_f} y_{fp} \sum_{j \in [i, l_f]} \sum_{s \in S} \psi_{fpjs} \tau_s^{\text{proc}}, \forall f \in F, \tag{25d}$$

$$c_f^{\text{used}} = r_f t_f^{\text{state}}, \forall f \in F, \tag{25e}$$

$$\sum_{f \in F} \sum_{p \in P_f} \alpha_{efp} y_{fp} r_f \leq c_e^{\text{trans}}, \forall e \in E, \tag{25f}$$

$$c_f^{\text{used}} \leq c^{\text{buffer}}, \forall f \in F. \tag{25g}$$

Equation (25c) represents that x_f is a binary decision variable such that $x_f = 1$ if the path of SFC $f \in F$ is to be updated and 0 otherwise. The model presented in [4] has a constraint that, when a state migration is performed, states must be migrated to a newly-installed VNF instance. To compare the proposed and benchmark models under the same condition, SFC-R omits this constraint. Equation (25d) corresponds to (2d) of SFC-RS. Equation (25e) represents that c_f^{used} is the amount of packets that are stored in the buffer when the update of SFC $f \in F$ is completed. Equation (25f) guarantees that the link capacity used after SFC update does not exceed the upper limit. Equation (25g) guarantees that the SFC route is not updated to a route that uses more capacity than the available buffer capacity.

4.2.2 SFC-S

SFC-S is based on the model presented in [7]. Since SFC-RS does not consider split buffers and the number of simultaneous buffer reads and writes, which are considered in the model in [7], these constraints are omitted in SFC-S. The scheduling of SFC-S is performed in the same way as the scheduling of SFC-RS; an SFC update is completed in one round and the duration time of a round consists of the time required to update flow entries, the time required to migrate states, and the delay required to send packets to the destination host. The solution obtained by SFC-R is used as the parameter of SFC-S.

The formulation of SFC-S is given by:

$$\min (1), \tag{26a}$$

$$\text{subject to} \quad \text{(2a)–(3c), (6), (7), (10a)–(13b).} \tag{26b}$$

Note that y_{fp}, w_{sf}, and u_{fi}, which are variables in SFC-RS, are given in SFC-S as parameters obtained by solving SFC-R.

4.3 Evaluation in 8-Node Network

In this evaluation, we use a network topology called 8-node network shown in Fig. 4 with eight nodes and 15 links. h_0 is a source host and h_7 is a destination host in each SFC route; the 8-node network has nine routes. A propagation delay for each link is set to 5 ms [21]. The controller is deployed on

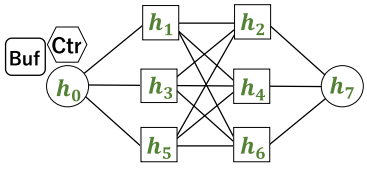


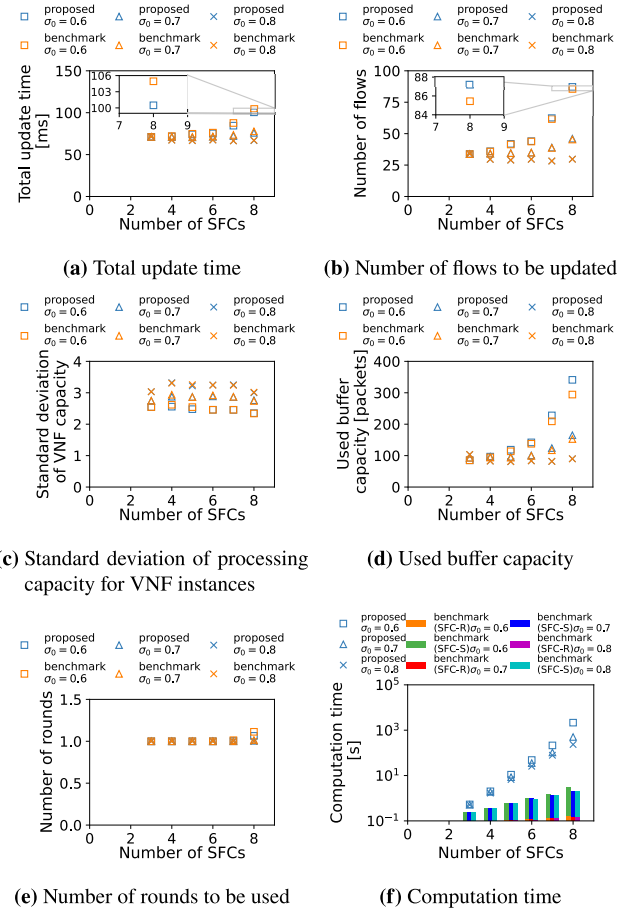
Fig. 4 8-node network.

host h_0.

4.3.1 Dependency on Threshold of Capacity Utilization Ratio σ_0

We evaluate the proposed and benchmark models in terms of the dependency on the threshold of capacity utilization ratio, σ_0. Figure 5 shows the results when buffer capacity is $c^{\text{buffer}} = 10000$ [packets] in the 8-node network.

Figures 5(a), 5(b), and 5(c) show the total update time, the number of flows to be updated, and the standard deviation, respectively. As the value of the capacity utilization ratio σ_0 increases, the standard deviation tends to increase and the total update time tends to decrease. The reason for these results is that, the larger the value of σ_0, the lower the degree of load balancing demanded; the necessity of updating SFCs is smaller. The number of updated flows has the same tendency as the total update time. This indicates that the flow entry update time and the state migration time increase corresponding to the number of updated flows. Compared to the benchmark model, the proposed model reduces the total update time by at most 4.25%. In this case, the number of updated flows increases by 2.05%. This is because the proposed and benchmark models use different route selections. The proposed model performs route selection considering simultaneous update scheduling of multiple SFCs (parallel scheduling). On the other hand, when the benchmark model performs route selection, the benchmark model assumes scheduling of updating SFCs one by one (serial scheduling). The route selection that minimizes the total update time when assuming serial scheduling can differ from that of parallel scheduling. For example, consider that there are two route selections A and B. Route selection A takes 20 ms and 40 ms for updating SFCs 1 and 2, respectively. Route selection B takes 35 ms each for updating SFCs 1 and 2. With serial scheduling, the total update time is $20 + 40 = 60$ ms for route selection A and $35 + 35 = 70$ ms for route selection B; the total update time of route selection A is smaller than that of route selection B. On the other hand, with parallel scheduling, the total update time is $\max(20, 40) = 40$ ms for route selection A and $\max(35, 35) = 35$ ms for route selection B; the total update time of route selection A is larger than that of route selection B. In this example, the proposed model performs route selection B, but the benchmark model performs route selection A.



(a) Total update time **(b)** Number of flows to be updated

(c) Standard deviation of processing capacity for VNF instances **(d)** Used buffer capacity

(e) Number of rounds to be used **(f)** Computation time

Fig. 5 Results when buffer capacity is $c^{\text{buffer}} = 10000$ [packets] in 8-node network.

Figure 5(d) shows the amount of used buffer capacity. In comparison to the benchmark model, the proposed model increases the amount of used buffer capacity at most 15.92%. This indicates that the number of updated SFCs is larger in the proposed model than in the benchmark model.

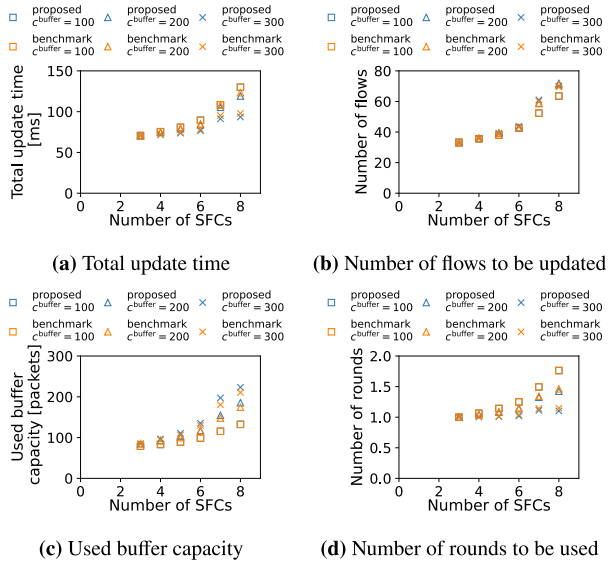
Figure 5(e) shows the number of rounds to be used. When the number of SFCs is between three and six, the number of rounds to be used is one. In these cases, all SFC updates can be done at once, since sufficient buffer capacity c^{buffer} is available. In the case where the number of SFCs is larger than seven, multiple rounds are used due to link capacity constraint (11c) and constraint of processing capacity for VNF instance (11d).

Figure 5(f) shows the computation time. Note that the computation times to solve SFC-R and SFC-S of the benchmark model are plotted in a stacked bar chart. The computation time increases as the number of SFCs increases. The proposed model requires more computation time compared to the benchmark model.

4.3.2 Dependency on Buffer Capacity c^{buffer}

We evaluate the proposed and benchmark models in terms of the dependency on the buffer capacity, c^{buffer}. Fig. 6 shows the results when the threshold of capacity utilization ratio is $\sigma_0 = 0.6$ in the 8-node network.

Figure 6(a) and Fig. 6(b) show the total update time and the number of flows to be updated, respectively. When the number of SFCs is between four and eight, as the buffer capacity c^{buffer} increases, the total update time decreases. This is because a large buffer capacity allows more packets to be buffered within a round and more SFCs to be updated simultaneously. The relationship of the total update time and the number of updated flows is different from that in Fig. 5. This is because the number of used rounds becomes



(a) Total update time **(b)** Number of flows to be updated

(c) Used buffer capacity **(d)** Number of rounds to be used

Fig. 6 Results when threshold of capacity utilization ratio is $\sigma_0 = 0.6$ in 8-node network.

larger than in Fig. 5 due to buffer capacity constraint (10); the total update time increases according to the number of used rounds. When $c^{\text{buffer}} = 200, 300$ and the number of SFCs is between four and eight, the total update time of the proposed model is smaller than that of the benchmark model. On the other hand, when $c^{\text{buffer}} = 100$ and the number of SFCs is between three and seven, the total update time and the number of updated flows of the proposed model and those of the benchmark model are the same. The reason for this is that the buffer capacity constraint (10a) makes a solution infeasible when a large value of n_f is generated; an optimal solution is only obtained when a small value of n_f is generated. A small value of n_f results in no difference of the number of flows and the total update time between the proposed and benchmark models.

Figure 6(c) and Fig. 6(d) show the amount of used buffer capacity and the number of rounds to be used, respectively. When the number of SFCs is between four and eight, as the buffer capacity c^{buffer} increases, the amount of used buffer capacity increases and the number of used rounds decreases. This reason is the same as that for the total update time. Similar to the dependency on σ_0 shown in Fig. 5(d), the proposed model uses more buffer capacity than the benchmark model when $c^{\text{buffer}} = 200, 300$ and the number of SFCs is between four and eight.

4.4 Evaluation in NSFNET

In this evaluation, we use NSFNET [27] which consists of 14 nodes and 21 links. NSFNET is shown in Fig. 7. The controller is deployed on node 5. Three firewalls and three IDSs are randomly deployed in the network nodes except for node 5. The source and destination hosts are randomly selected from among the nodes for each SFC except for node 5. P_f consists of up to ten paths, which are sorted in the

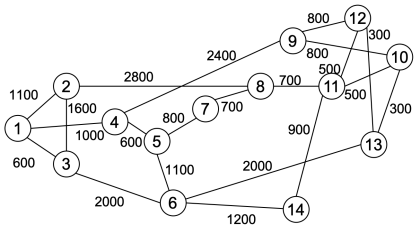
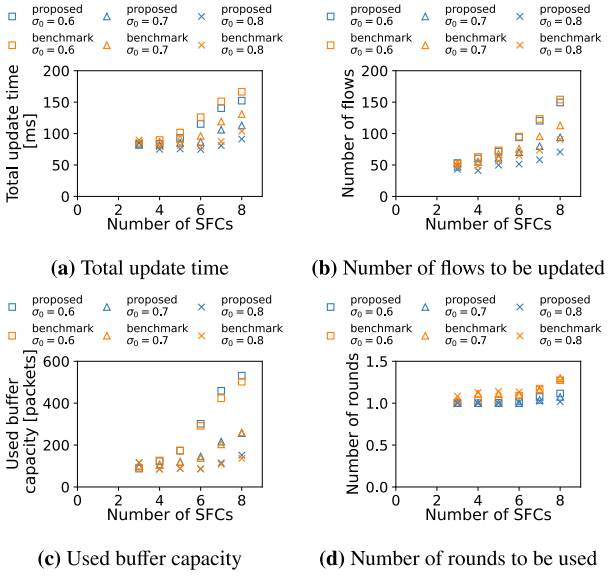


Fig. 7 NSFNET (distances in km) [28].



(a) Total update time **(b)** Number of flows to be updated

(c) Used buffer capacity **(d)** Number of rounds to be used

Fig. 8 Results when buffer capacity is $c^{\text{buffer}} = 10000$ [packets] in NSFNET.

order of path length, for each SFC $f \in F$. A propagation delay for each link is obtained by its distance [28]. Figure 8 shows the results when the buffer capacity is $c^{\text{buffer}} = 10000$ [packets] in NSFNET.

Figures 8(a), 8(b), and 8(c) show the total update time, the number of flows to be updated, and the amount of used buffer capacity, respectively. The proposed model reduces the total update time by at most 13.77% from the benchmark model. Compared to the results of the 8-node network in Fig. 5, the total update time, the number of flows, and the amount of used buffer capacity tend to be increased. This is because the values of propagation delays are larger than in the 8-node network. The larger the propagation delays, the larger the flow entry update time, the state migration time, and the delay time; the total update time becomes larger. When the total update time becomes larger, the time to accumulate packets in the buffer becomes larger; the amount of used buffer capacity becomes larger. Large propagation delays result in the second term in (22) to be dominant in NSFNET; the number of updated flows becomes larger.

Figure 8(d) shows the number of rounds to be used. The number of used rounds of the proposed model is smaller than that of the benchmark model. This reason is the same

as that in the 8-node network; the proposed model considers link capacity constraint (11c) and constraint of processing capacity for VNF instance (11d) when SFCs to be updated and their routes are determined.

5. Discussions

The state migration can be challenging in some types of states, such as those with a large data size or those having special characteristics. For example, an automated driving application has vehicle trajectory information as a state [29]. Integrated log management software [30] has information as a state such as trained parameters for log analysis by machine learning and standard output results when some programs are executed. Such a large state size results in large state migration time compared to that for other VNF instances. If the state migration time is large, a large buffer capacity is required and the controller's buffer may overflow. It may be possible to address this problem by extending the proposed model so that buffers on the network are used in a flexible manner. For instance, by utilizing not only the controller's buffer but also the buffers of VNF instances cooperatively, the packets arriving during state migration can be distributed. If the buffer capacity is still insufficient even with such an extension, the system itself needs to be changed, such as by duplicating the VNF instances to avoid the need for the state migration, and this is a future work. For another example, an application that manages a login system holds user session information as a state. An application that performs a payment management system maintains information about the amount of money and credit card related to the transaction as a state. There is a case where it is necessary to encrypt such a state that contains sensitive information. To deal with this case, an extension to the state migration procedure is necessary to add complex procedures such as key generation and key sharing among hosts to encrypt the state migration. In addition, extensions that account for delays associated with the encryption are required to be added to the proposed model.

The performance degradation of the proposed model is acceptable in the scenarios we evaluated in Sect. 4 compared to its effectiveness. While the proposed model can reduce the total update time compared to the benchmark model, there are some cases where the used buffer capacity and the computation time increase. When the total update time is reduced by 4.25% in the 8-node network, the used buffer capacity is increased by 15.92% (46.85 packets). When the proposed model in NSFNET reduces the total update time by 13.77%, the used buffer capacity is increased by 5.75% (28.90 packets). In these cases, the increase of used buffer capacity is 0.47% and 0.29%, respectively, of the total buffer capacity of 10000 packets. For the range of values and networks we examined, the computation time of the benchmark model is at most a few seconds, while that of the proposed model is at most several thousand seconds. The proposed model is tolerable for purposes where the SFC routes are reconfigured several times in a day. If the proposed model is applied to a larger number of SFCs or a larger network size, it may not be possible to obtain a solution in a required time. It is necessary to reduce the computation time by using a heuristic method. The development of the heuristic method in the proposed model is a future work.

6. Conclusion

This paper proposed a model that determines SFCs to be updated, their routes, and an update scheduling at the same time. The objective function is to minimize the total time required to update SFC routes under a constraint of processing load balancing for VNF instances. The proposed model is formulated as an ILP problem. The proposed model is compared to the benchmark model which finds a solution of SFC-RS in two phases. The following results are obtained by numerical evaluations. Compared to the benchmark model, the proposed model reduces the total update time by at most 4.25% in the 8-node network. As the value of the capacity utilization ratio σ_0 increases, the total update time, the number of updated flows, and the amount of used buffer capacity tend to decrease. As the buffer capacity c^{buffer} increases, the total update time tends to decrease and the amount of used buffer capacity tends to increase. In NSFNET, the proposed model reduces the total update time by at most 13.77% compared to the benchmark model. The total update time, the number of updated flows, and the amount of used buffer capacity become larger than in the 8-node network.

Acknowledgments

This work was supported in part by JSPS KAKENHI Grant Numbers 21H03426 and 23H03382, and JST, PRESTO Grant Number JPMJPR23P4, Japan.

References

[1] S. Wang, H. Cao, and L. Yang, "A survey of service function chains orchestration in data center networks," 2020 IEEE Globecom Workshops (GC Wkshps), Dec. 2020.
[2] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," 2015 IEEE Conf. Netw. Function Virtualization and Softw. Defined Netw (NFV-SDN), pp.191–197, Nov. 2015.
[3] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling innovation in network function control," ACM SIGCOMM Comput. Commun. Rev., vol.44, no.4, pp.163–174, Oct. 2014.
[4] B. Zhang, P. Zhang, Y. Zhao, Y. Wang, X. Luo, and Y. Jin, "Coscaler: Cooperative scaling of software-defined NFV service function chain," 2016 IEEE Conf. Netw. Function Virtualization and Softw. Defined Netw. (NFV-SDN), pp.33–38, Nov. 2016.
[5] C. Sun, J. Bi, Z. Meng, T. Yang, X. Zhang, and H. Hu, "Enabling NFV elasticity control with optimized flow migration," IEEE J. Sel. Areas Commun., vol.36, no.10, pp.2288–2303, Oct. 2018.
[6] X. Fan, H. Xu, H. Huang, and X. Yang, "Real-time update of joint SFC and routing in software defined networks," IEEE/ACM Trans. Netw., vol.29, pp.2664–2677, Dec. 2021.
[7] T. Takahashi, T. Sato, and E. Oki, "Scheduling model for simultaneous update of multiple service function chains with state consistency," Computer Netw., vol.221, p.109520, Feb. 2023.

[8] R. Wen, G. Feng, J. Tang, T.Q.S. Quek, G. Wang, W. Tan, and S. Qin, "On robustness of network slicing for next-generation mobile networks," IEEE Trans. Commun., vol.67, no.1, pp.430–444, Jan. 2019.
[9] R. Sukapuram, R. Patowary, and G. Barua, "Loss-freedom, order-preservation and no-buffering: Pick any two during flow migration in network functions," 2021 IEEE 29th Int. Conf. Netw. Protocols (ICNP), pp.1–11, Nov. 2021.
[10] A. Gember-Jacobson and A. Akella, "Improving the safety scalability and efficiency of network function state transfers," Proc. 2015 ACM SIGCOMM Workshop Hot Topics in Middleboxes and Netw. Function Virtualization, pp.43–48, Aug. 2015.
[11] J. Sherry, P.X. Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. Martins, S. Ratnasamy, L. Rizzo, and S. Shenker, "Rollback-recovery for middleboxes," Proc. 2015 ACM Conf. Special Interest Group Data Commun., vol.45, no.4, pp.227–240, Oct. 2015.
[12] C. Clark, K. Fraser, S. Hand, J.G. Hansenf, E. Julf, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," Proc. 2nd Conf. Symp. Netw. Syst. Des. and Implementation, pp.273–286, May 2005.
[13] L. Nobach, I. Rimac, V. Hilt, and D. Hausheer, "Statelet-based efficient and seamless NFV state transfer," IEEE Trans. Netw. Service Manage., vol.14, no.4, pp.964–977, Dec. 2017.
[14] J. Wang, S. Hao, Y. Li, C. Fan, J. Wang, L. Han, Z. Hong, and H. Hu, "Challenges towards protecting VNF With SGX," Proc. 2018 ACM Int. Workshop Secur. in Softw. Defined Netw. and Netw. Function Virtualization, pp.39–42, March 2018.
[15] W. Wang, Y. Liu, J. Liu, Y. Li, H. Song, Y. Wang, and J. Yuan, "Consistent state updates for virtualized network function migration," IEEE Trans. Services Comput., vol.13, no.6, pp.999–1006, Nov.-Dec. 2020.
[16] R. Kang, F. He, and E. Oki, "Robust virtual network function allocation in service function chains with uncertain availability schedule," IEEE Trans. Netw. Service Manage., vol.18, no.3, pp.2987–3005, Sept. 2021.
[17] F. He, T. Sato, B.C. Chatterjee, T. Kurimoto, S. Urushidani, and E. Oki, "Robust optimization model for primary and backup resource allocation in cloud providers," IEEE Trans. Cloud Comput., vol.10, pp.2920–2935, Oct.-Dec. 2022.
[18] K. Yokouchi, F. He, and E. Oki, "Backup resource allocation of virtual machines with two-stage probabilistic protection," IEEE Trans. Netw. Service Manage., vol.20, no.4, pp.5085–5102, Dec. 2023.
[19] M. Johnston, H. Lee, and E. Modiano, "A robust optimization approach to backup network design with random failures," IEEE/ACM Trans. Netw., vol.23, no.4, pp.1216–1228, Aug. 2015.
[20] T. Moufakir, M.F. Zhani, A. Gherbi, M. Aloqaily, and N. Ghrada, "SFCaaS: Service function chains as a service in NFV environments," ITU J. Future and Evolving Technol., vol.3, no.3, pp.679–692, Dec. 2022.
[21] S. Agarwal, V.R. Chintapalli, and B.R. Tamma, "FlexSFC: Flexible resource allocation and VNF parallelism for improved SFC placement," 2022 IEEE 8th Int. Conf. Netw. Softw. (NetSoft), pp.302–306, June-July 2022.
[22] M.A. Kourtis, G. Xilouris, V. Riccobene, M.J. McGrath, G. Petralia, H. Koumaras, G. Gardikis, and F. Liberal, "Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration," 2015 IEEE Conf. Netw. Function Virtualization and Softw. Defined Netw. (NFV-SDN), pp.74–78, Nov. 2015.
[23] "Firewalling fundamentals," https://docs.netgate.com/pfsense/en/latest/firewall/fundamentals.html, (accessed 1 Dec. 2023).
[24] K. Qiu, J. Yuan, J. Zhao, X. Wang, S. Secci, and X. Fu, "FastRule: Efficient flow entry updates for TCAM-based OpenFlow switches," IEEE J. Sel. Areas Commun., vol.37, no.3, pp.484–498, March 2019.
[25] S. Banerjee and K. Kannan, "Tag-In-Tag: Efficient flow table management in SDN switches," 10th Int. Conf. Netw. and Service Manage. (CNSM) and Workshop, pp.109–117, Nov. 2014.
[26] IBM ILOG CPLEX optimization studio, IBM, 2023. https://www.ibm.com/products/ilog-cplex-optimization-studio, (accessed 1 Dec. 2023).
[27] NSFNET, "A partnership for high-speed networking: Final report," https://www.merit.edu/wp-content/uploads/2019/06/NSFNET_final-1.pdf, (accessed 17 Nov. 2023).
[28] A.K. Horota, R.A. Costa, S. Rahman, O. Ayoub, G.B. Figueiredo, M. Tornalore, and B. Mukherjee, "Deferred protection of deadline-driven requests in inter-datacenter elastic optical networks," 2020 Int. Conf. Opt. Netw. Des. and Model. (ONDM), May 2020.
[29] T.V. Doan, G.T. Nguyen, M. Reisslein, and F.H.P. Fitzek, "FAST: Flexible and low-latency state transfer in mobile edge computing," IEEE Access, vol.9, pp.115315–115334, Aug. 2021.
[30] F. Gurcan and M. Berigel, "Real-time processing of big data streams: Lifecycle, tools, tasks, and challenges," 2018 2nd Int. Symp. Multidiscip. Stud. and Innov. Technol. (ISMSIT), Oct. 2018.

Tomoki Takahashi is pursuing the M.E. degree at Graduate School of Informatics, Kyoto University, Kyoto, Japan. He received the B.E. degree from Undergraduate School of Electrical and Electronic Engineering, Kyoto University, Japan, in 2022. His research interests include optimization, software-defined network, and network update.

Takehiro Sato received his B.E., M.E., and Ph.D. in engineering from Keio University in 2010, 2011, and 2016, respectively. He is currently an associate professor in the Graduate School of Informatics at Kyoto University. His research interests include design and control methods for optical and virtualized networks.

Eiji Oki is a Professor at Kyoto University, Kyoto, Japan. He was with Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Tokyo, from 1993 to 2008, and The University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar at Polytechnic University, Brooklyn, New York. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks.