PAPER

Elastic and Adaptive Resource Orchestration Architecture on 3-Tier Network Virtualization Model[**]

Masayoshi SHIMAMURA[†,††*a)], ***Member***, **Hiroaki YAMANAKA**[†††b)], ***Nonmember***, **Akira NAGATA**[††††c)], ***Member***, **Katsuyoshi IIDA**[†d)], ***Senior Member***, **Eiji KAWAI**[†††e)], ***and* Masato TSURU**[††††f)], ***Members***

SUMMARY Network virtualization environments (NVEs) are emerging to meet the increasing diversity of demands by Internet users where a virtual network (VN) can be constructed to accommodate each specific application service. In the future Internet, diverse service providers (SPs) will provide application services on their own VNs running across diverse infrastructure providers (InPs) that provide physical resources in an NVE. To realize both efficient resource utilization and good QoS of each individual service in such environments, SPs should perform adaptive control on network and computational resources in dynamic and competitive resource sharing, instead of explicit and sufficient reservation of physical resources for their VNs. On the other hand, two novel concepts, software-defined networking (SDN) and network function virtualization (NFV), have emerged to facilitate the efficient use of network and computational resources, flexible provisioning, network programmability, unified management, etc., which enable us to implement adaptive resource control. In this paper, therefore, we propose an architectural design of network orchestration for enabling SPs to maintain QoS of their applications aggressively by means of resource control on their VNs efficiently, by introducing virtual network provider (VNP) between InPs and SPs as 3-tier model, and by integrating SDN and NFV functionalities into NVE framework. We define new north-bound interfaces (NBIs) for resource requests, resource upgrades, resource programming, and alert notifications while using the standard OpenFlow interfaces for resource control on users' traffic flows. The feasibility of the proposed architecture is demonstrated through network experiments using a prototype implementation and a sample application service on nation-wide testbed networks, the JGN-X and RISE.

key words: *network virtualization environment, software-defined networking, network function virtualization, resource orchestration, OpenFlow*

Manuscript received September 30, 2014.
Manuscript revised December 16, 2015.
Manuscript publicized January 18, 2016.

[†]The authors are with the Global Scientific Information and Computing Center, Tokyo Institute of Technology, Tokyo, 152–8550 Japan.
[††]The author was with the Network Application Engineering Laboratories Ltd., Fukuoka-shi, 812–0011 Japan.
[†††]The authors are with the Network Testbed Research and Development Promotion Center, National Institute of Information and Communications Technology, Tokyo, 100–0004 Japan.
[††††]The authors are with the Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan.
[*]Presently, with the Kyushu Research Center, iD Corporation.

[**]An earlier version of this work has been presented in NoF2013, October 2013 [1].

a) E-mail: m-shimamura@intelligent-design.co.jp
b) E-mail: hyamanaka@nict.go.jp
c) E-mail: nagata@infonet.cse.kyutech.ac.jp
d) E-mail: iida@gsic.titech.ac.jp
e) E-mail: eiji-ka@nict.go.jp
f) E-mail: tsuru@cse.kyutech.ac.jp

DOI: 10.1587/transinf.2014EDP7321

1. Introduction

With the growth of the Internet, numerous Internet users are choosing a wide variety of application services in accordance with their individual demands. These application services need to provide outstanding value in order to remain competitive. Consequently, to improve user satisfaction, application services are demanding better network performance.

In this context, new paradigms such as network virtualization environments (NVEs) [2]–[5], software-defined networking (SDN) [6]–[8], and network function virtualization (NFV) [9] have been receiving attention from researchers, operators, and application services. Basic NVE concepts have been proposed [2] in which infrastructure providers (InPs) possess substrate (physical) network resources consisting of relay nodes (switches, routers, etc.) and end nodes (computational and storage nodes) and service providers (SPs) construct logical networks called virtual networks (VNs) by renting resources from the InPs, allowing them to customize their networks to suit a given application service. SDN divides a control plane from a forwarding plane called a data plane [6], thus enabling the control plane to control network devices centrally and agilely. NFV provides network devices with various functions via software processing by using commodity servers rather than specialized devices [9] so that SPs can define value-added network functions within the servers according to their application services. Note that, in this paper, we consider NFV as an environment that enables SPs to install arbitrary programs in virtual resources (e.g., virtual machines) for advanced relay processing (also called in-network processing). Examples of advanced relay processing include online packet compression [10], data caching such as information centric networks [11], [12] and content centric networks [13], and deep packet inspection.

In the future Internet, diverse SPs will provide application services on their own VNs running across diverse InPs in an NVE. The majority of existing works on NVEs in terms of virtual network embedding assume explicit reservation and isolation of physical resources allocated to each VN [14]. However, to achieve both efficient resource utilization and good QoS of each individual service in such environments, instead of explicit and sufficient reservation of physical resources for their VNs, it is essential to allow

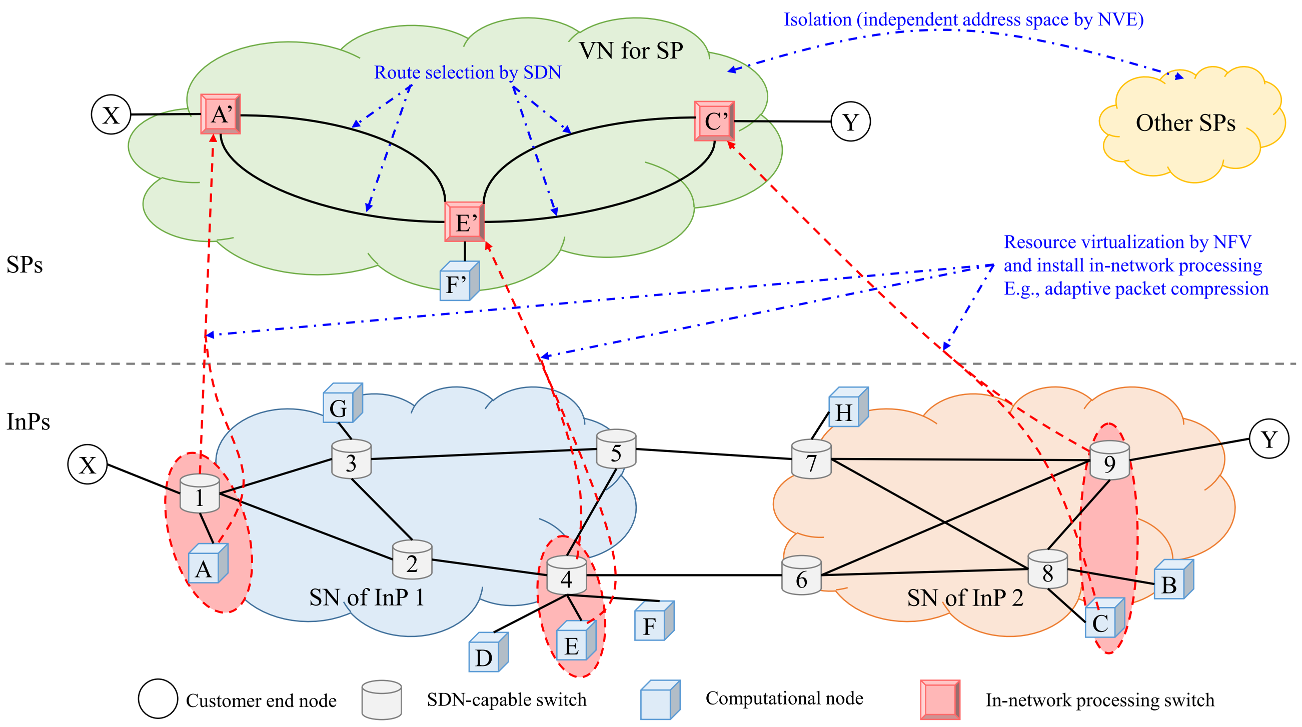

Isolation (independent address space by NVE)
VN for SP
Route selection by SDN
Other SPs
X
A'
C'
Y
E'
F'
SPs
Resource virtualization by NFV
and install in-network processing
E.g., adaptive packet compression
InPs
G
H
X
Y
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
SN of InP 1
SN of InP 2
Customer end node
SDN-capable switch
Computational node
In-network processing switch

Fig. 1 Logical networks for service providers by NVE, SDN, and NFV.

SPs (especially for elastic services) to efficiently share the resources with other VNs dynamically and competitively. Therefore, we consider an integration of NVE, SDN, and NFV for enabling SPs to perform adaptive resource control on their VNs as a key to resource-efficient QoS management in accommodating diverse application services over diverse multiple InPs.

Figure 1 illustrates a simple example of adaptive resource control. SPs aim to achieve both efficient resource utilization and good QoS by exploiting features of NVE, SDN, and NFV. In the figure, two InPs provide their resources to the SP located at the upper left based on NVE framework. The SP constructs its own VN logically independent of other VNs, meaning that it can use an arbitrary address space without potential conflicts with others. The SP can select appropriate computational resource F' to improve QoS for customers X and Y, e.g., minimizing the delay time for both customers. Moreover, the SP obtains multiple logical links between each customer and F', respectively, for detour link in preparation to congestion on substrate networks (SNs) of InPs. Then, by using SDN (OpenFlow in particular) technology, the SP dynamically chooses appropriate logical links between X and F' and between Y and F', respectively. This kind of link selection can be also seen in recent video application services in which the best links are chosen depending on the pre-buffered playtime of video playback [15]. Furthermore, by leveraging NFV, the SP performs advanced relay processing if needed at appropriate places with additional resources from InPs. The SP obtains special switches A', C' and E' consists of SDN-capable switches and computational resources by resource virtualization from InPs. After that, the SP installs original special functions (e.g., the adaptive packet compression scheme [10]) into the special switches. In those scenarios for SP to control its logical network aggressively, appropriate interfaces called north-bound interfaces (NBIs) should be provided to SP.

In standardization organizations such as the Open Networking Foundation and the European Telecommunications Standards Institute (ETSI), and in communities such as OpenDaylight [16], numerous NBIs have been considered to enable SPs to control the network resources of InPs. However, little work seems to be completed on interfaces and architectures in NVE to allow SPs to perform adaptive resource control. For example, neither NBIs to obtain multiple logical links (including detour links) between the source and destination nodes nor NBIs to newly define advanced network functions, which are essential to SP's resource control.

Developing such NBIs would result in the deployment of a new network environment that enables SPs to perform adaptive resource control. One of the biggest challenges here is considering the resource information to be provided to SPs from InPs from the perspective of a real environment. Related works are roughly classified into two types. In the first type, SPs require a network performance guarantee from the InPs. These works relate to virtual network embedding [14] and are theoretical study. In the second type, InPs disclose all resource information to SPs. These works are considered testbed networks [17]. In a real environment, InPs cannot disclose all information due to security concerns, also known as the topology hiding problem. InPs will inevitably need to hide (i.e., abstract) the resource information, so we need to consider how InPs provide the resource information to SPs and what this information is.

The main contribution of this paper is to propose an architectural design of network orchestration for enabling SPs to perform adaptive resource control on their own VNs efficiently and to maintain QoS of their own applications aggressively where VNs dynamically and competitively share the network and computational resources provided by diverse multiple InPs. This paper also demonstrates its feasibility through network experiments using a prototype implementation and a sample application service on nation-wide testbed networks, the JGN-X and RISE.

In this work, we reconsider NBIs to obtain network resources for SPs and propose a network orchestration architecture that enables SPs to perform advanced relay processing by using the computational and storage functions on server resources rather than the forwarding functions on network resources. Since the demands of each SP are different, we consider approaches that extract the appropriate resources for specific SPs. As a means of meeting SP demands, we introduce a mediator called a virtual network provider (VNP). The VNP has a key role in resource management and orchestration as well as ETSI NFV management and orchestration (MANO) [18] while it has different types of NBIs as key features for achieving both resource efficiency and good QoS for SPs in dynamic and competitive resource sharing.

The rest of this paper is organized as follows. Section 2 provides a scenario description and discusses related works. Section 3 details the design of the proposed architecture. In Sect. 4, we discuss our prototype implementation, a use case scenario, and a demonstration. Section 5 concludes with a brief summary of key points.

2. Scenario Description and Related Work

In this section, we present our scenario description and briefly touch on related works.

2.1 Scenario Description

In the proposed network orchestration architecture, we assume that there is a single mediator VNP, with which each InP and each SP has a direct relationship. Therefore, InPs and SPs are loosely collaborated through the VNP.

We assume each InP is ready for SDN technologies (e.g., OpenFlow) and an network monitoring framework (e.g., perfSONAR [19]). First, InPs prepare virtualized resources from their own substrate resources using virtualization technologies, e.g., FlowVisor [20] and FlowN [21] for network resources and kernel-based virtual machines [22] and Linux containers [23] for server resources. Also, InPs prepare isolated control planes for multiple controllers such as FlowVisor. InPs then provide the VNP with NBIs as a right of access to network resources and resource information. The topology hiding problem [24] is addressed by having the InPs provide the VNP with a subset of all the network resources rather than the entire set. In this paper, we assume that the VNP can receive information from the path between source and destination switches rather than from hop-by-hop links. Furthermore, the InP provides best-effort performance rather than guaranteed performance that strictly satisfies performance demand.

The VNP collects resource information from each InP and manages the information as a single network called a middle virtual network (MVN). Then, as with the InPs, the VNP provides SPs with an NBI as a right of access to network resources and resource information. The VNP also has two important roles that differ somewhat from those of the InPs. First, the VNP defines original resources for SPs, e.g., special switches with advanced relay processing capability consisting of switch and server resources. Second, the VNP provides the SPs with a resource monitoring NBI that enables them to control resources in accordance with changing resource and application information.

SPs independently control the logical resources provided by the VNP in accordance with resource and application information. Therefore, they require resources that can satisfy the demands they make of the VNP. Then, to control an obtained resource, SPs set a trigger to detect performance degradation via the resource monitoring NBI. Finally, SPs determine the appropriate resources for their own traffic flows in accordance with the current needs of the application services and resources.

2.2 Related Work

Here, we discuss research related to our work. First, we describe virtualization technologies, also known as multi-tenancy technologies. Next, we describe topology abstraction in the network virtualization environment. Then, we describe resource orchestration and management. Finally, we focus on an interesting study for improving network performance using application information and SDN technologies.

In NVE, isolation technologies are required. In this paper, we do not focus on the details and simply employ existing technologies. Reference [21] has proposed a multi-tenancy technology called FlowN that can be used by each SP to control its own resources without the interference of resource information, e.g., IP addresses. Another important technology is topology abstraction. VeRTIGO proposed in Ref. [25] provide a single (big) switch abstraction and edge switches abstraction. Abstracted resource information has an advantage of topology hiding because a subset of all the network resources can be hidden. On the other hand, it leads to a disadvantage, coarse-grained resource control, because a network topology must be simple.

Reference [18] also known as ETSI MANO defines management and orchestration framework for provisioning and configuration of virtualized network functions (VNFs). This document defines three types of functional blocks: resource orchestrator, resource manager, and function manager. Moreover, it also defines reference points among functional blocks. The MANO specification can be useful and essential information for all studies of NFV including our study.

For performance improvement, Ref. [15] aims to perform resource control for the application flows of YouTube traffic. The authors proposed an application-aware resource control scheme using deep packet inspection technologies. This scheme, which is based on the pre-buffered playtime of YouTube videos, is used to select an appropriate link for traversing traffic flows.

All the related works are related with our study and these can be employed as useful and essential pieces in the VNP without competition. A key difference between our proposal and the related works is that the VNP provides resource construction and monitoring NBIs to SPs for achieving both resource efficiency and good QoS for SPs where diverse VNs dynamically and competitively share the substrate network and computational resources of InPs.

3. Architectual Design

To realize the scenarios we assume, several new NBIs are necessary. These NBIs are roughly classified into two types. The first type is an NBI for obtaining resources and defining original functions to the resources. The other type is an NBI for obtaining resource information. These NBIs are described in the following subsections.

Table 1 shows all the NBIs designed for the proposed architecture, and Fig. 2 depicts sample sequences of the NBIs. SPs use `SpJoinRequest` when they start constructing their VNs and use `SpLeaveRequest` when they finish. They can show available resource information through `AvailableTopologyRequest` and then obtain the desired resources through `VnResourceRequest`. `EndNodeJoinRequest` can be used to attach end nodes to VNs. When SPs perform advanced relay processing on their VNs, they use `ResourceUpgradeRequest` to obtain special relay nodes and `ResourceProgrammingRequest` to install their programs on these nodes.

For monitoring VNs, SPs use `VnTopologyRequest` and `VnFlowListRequest` to obtain the current topology of their VNs and flow list on the VNs, respectively. Moreover, they use `VnLinkQualityRequest`, `VnFlowQualityRequest`, and `VnResourceQualityRequest` to obtain quality information of the links, flows, and resources, respectively. When SPs need to detect performance degradation of flows, links, and resources, they use `VnMonitoringRequest` to request monitoring and receive alerts if the monitored flows, links, or resources degrade.

3.1 Resource Request NBI

The VNP provides SPs with a resource request NBI. First, SPs require disclosure of available resources satisfying their demands to the VNP. SPs set source and destination switches, desired performance, and the number of detour paths. After the VNP responds with the available resources, SPs select which ones they need.

Figure 2 (a) shows an example sequence of resource request NBIs. `AvailableTopologyRequest(vm_stat, max_vm_num, src_node_id, dst_node_id, link_stat, max_link_num)` is used for obtaining available resources including computational resources representing virtual machine resources and relay node resources representing switches. The first two parameters are specified for virtual machine resources and the last four parameters are specified for switch resources. Parameter `vm_stat` specifies the conditions of virtual machines, such as available storage size and computational load. Parameter `max_vm_num` specifies the number of desired virtual machines satisfying the specifications of `vm_stat`. Parameters `src_node_id` and `dst_node_id` specify the identifier of the source and destination relay nodes, respectively. Parameter `link_stat` specifies the link conditions, such as available bandwidth and current averaged delay time. Parameter `max_link_num`

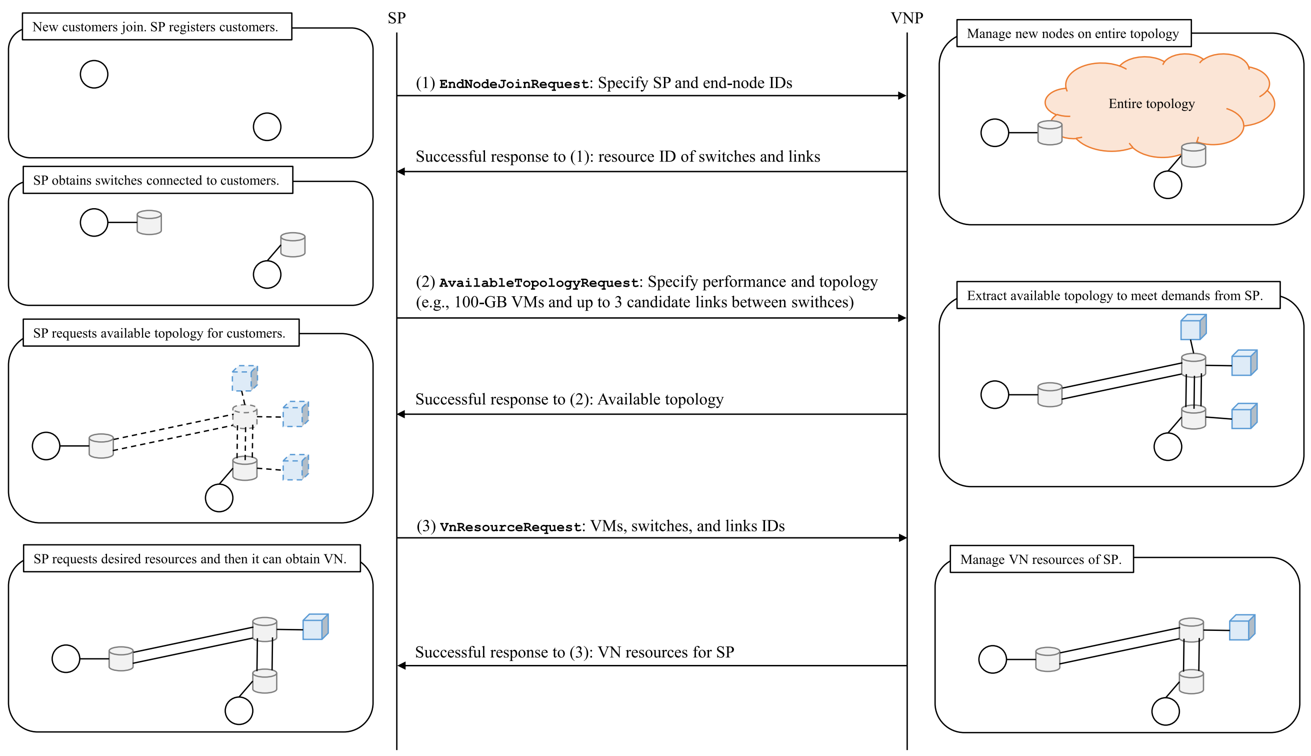
Table 1 NBIs of the proposed network orchestration architecture.

Construction NBIs

NBI name	Description
`SpJoinRequest`	Join for initiation
`SpLeaveRequest`	Leave for termination
`AvailableTopologyRequest`	Obtain available resources including switches and virtual machines
`VnResourceRequest`	Request desired resources
`EndNodeJoinRequest`	Attach end node to VN
`ResourceUpgradeRequest`	Upgrade resources for advanced relay processing
`ResourceProgrammingRequest`	Install programs to upgraded resources

Monitoring NBIs

NBI name	Description
`VnTopologyRequest`	Obtain current VN topology
`VnFlowListRequest`	Obtain existing flow list
`VnLinkQualityRequest`	Obtain link quality
`VnFlowQualityRequest`	Obtain flow quality
`VnResourceQualityRequest`	Obtain resource status
`VnMonitoringRequest`	Request resource monitoring
`AlertNotificationRequest`	Receive alert notification


New customers join. SP registers customers.
SP
VNP
Manage new nodes on entire topology
Entire topology
(1) EndNodeJoinRequest: Specify SP and end-node IDs
Successful response to (1): resource ID of switches and links
SP obtains switches connected to customers.
(2) AvailableTopologyRequest: Specify performance and topology (e.g., 100-GB VMs and up to 3 candidate links between swithces)
Extract available topology to meet demands from SP.
SP requests available topology for customers.
Successful response to (2): Available topology
(3) VnResourceRequest: VMs, switches, and links IDs
SP requests desired resources and then it can obtain VN.
Manage VN resources of SP.
Successful response to (3): VN resources for SP

(a) Resource request NBIs

Fig. 2 Sample sequence of NBIs usage.

specifies the number of desired links satisfying the specification of `link_stat` between `src_node_id` and `dst_node_id` for detour paths.

Regarding return values, SPs receive `switch_list`, `vm_list`, and `link_list`. `switch_list` includes switch identifier, monetary cost, and neighbor switch identifiers. `vm_list` includes virtual machine identifier, processing speed, storage size, and monetary cost. `link_list` includes link identifier, link bandwidth, link performance, source switch identifier, and destination switch identifier. After receiving the return values, SPs specify their desired resources through `VnResourceRequest`.

3.2 Resource Upgrade NBI

The VNP provides SPs with a resource upgrade NBI. When SPs attempt to employ advanced relay processing using logical resources, they can use special relay nodes capable of advanced relay processing, such as advanced relay nodes [26].

Figure 2 (b) shows an example sequence of resource upgrade NBIs. `ResourceUpgradeRequest(sw_id, function)` upgrades a relay node specified in `sw_id` with a special capability of `function`. Return values are information of upgraded resources including identifier, processing speed, storage size, and link information, as with regular switches.

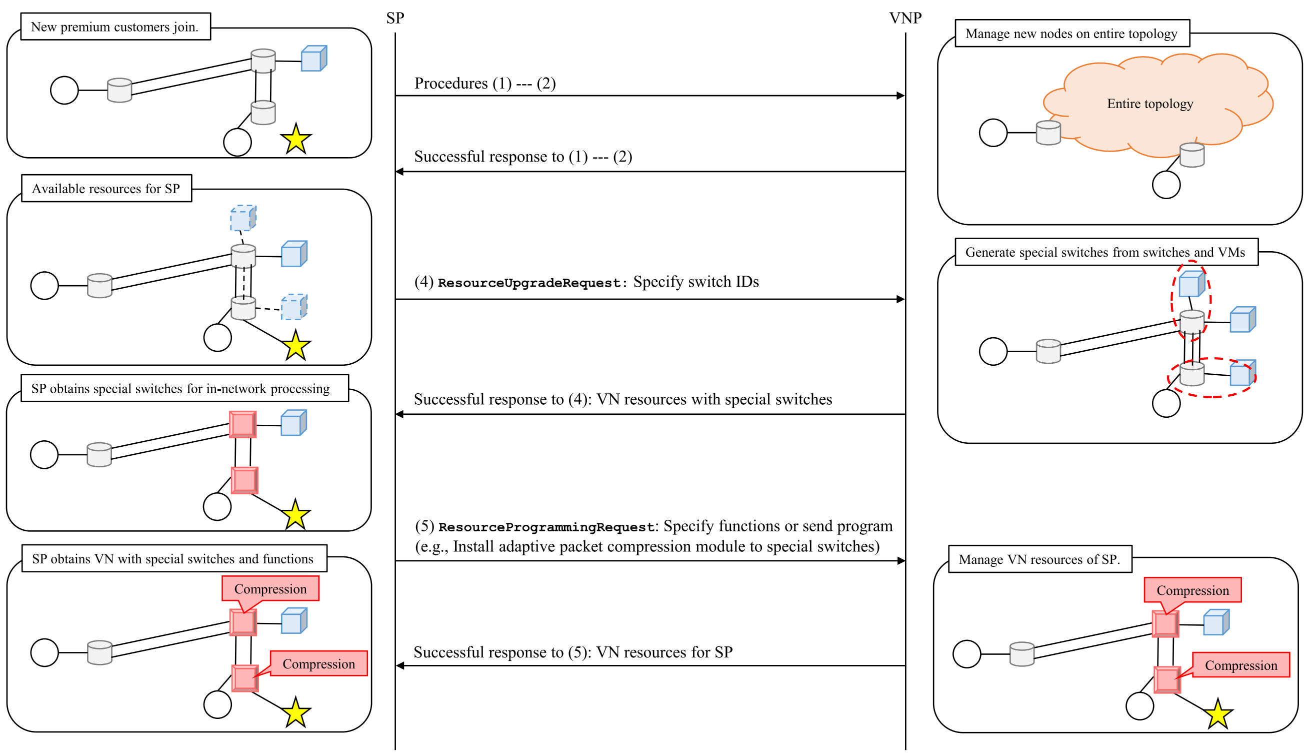
An example of `function` is the adaptive packet compression proposed in Ref. [10]. The VNP prepares capabilities of special relay nodes and SPs choose their desired capabilities. For development purposes, we assume that the VNP prepares capabilities for a free programming environment such as a Linux container [23].

After receiving the request from SPs, the VNP generates a logical special relay node. This special relay node consists of a relay node and a computational resource such as a virtual machine, but SPs recognize only a single special relay node. The VNP sets the route at the relay node: from the relay node to the virtual machine for pre-processing packets and from the virtual machine to the relay node for post-processing packets.

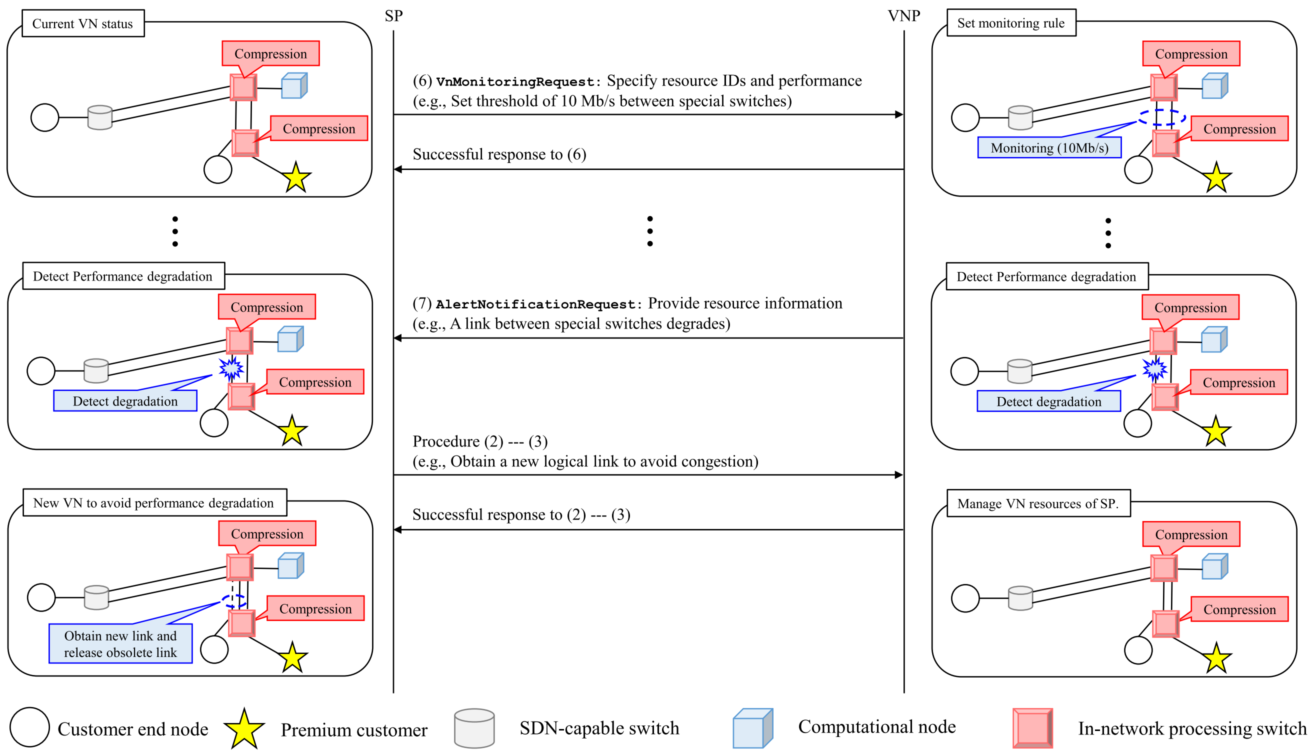
3.3 Resource Programming NBI

The VNP provides SPs with a resource programming NBI. When SPs set parameters of the special relay node or install programs on it, they can use this NBI as depicted in Fig. 2 (c) If SPs choose a prepared capability via `ResourceUpgradeRequest`, the VNP also provides an NBI for parameter setting. For example, in adaptive packet compression [10], the SPs can set the parameters of packet compression, the condition of packet compression, etc. We assume that the VNP provides NBIs for setting parameters in accordance with special capabilities. Furthermore, for development purposes, we designed an NBI for installing programs whereby the SPs can specify the programs and commands to be performed at the special relay nodes.

SPs specify the `program` and `execute` parameters for


SP
VNP
New premium customers join.
Procedures (1) --- (2)
Manage new nodes on entire topology
Entire topology
Successful response to (1) --- (2)
Available resources for SP
(4) ResourceUpgradeRequest: Specify switch IDs
Generate special switches from switches and VMs
Successful response to (4): VN resources with special switches
SP obtains special switches for in-network processing
(5) ResourceProgrammingRequest: Specify functions or send program
(e.g., Install adaptive packet compression module to special switches)
SP obtains VN with special switches and functions
Manage VN resources of SP.
Compression
Compression
Successful response to (5): VN resources for SP

(b) Resource upgrade NBIs


SP
VNP
Current VN status
Set monitoring rule
Compression
Compression
(6) VnMonitoringRequest: Specify resource IDs and performance
(e.g., Set threshold of 10 Mb/s between special switches)
Monitoring (10Mb/s)
Successful response to (6)
Detect Performance degradation
Detect degradation
(7) AlertNotificationRequest: Provide resource information
(e.g., A link between special switches degrades)
Procedure (2) --- (3)
(e.g., Obtain a new logical link to avoid congestion)
New VN to avoid performance degradation
Manage VN resources of SP.
Obtain new link and
release obsolete link
Successful response to (2) --- (3)
Customer end node
Premium customer
SDN-capable switch
Computational node
In-network processing switch

(c) Resource monitoring NBIs

Fig. 2 (Continued)

ResourceProgrammingRequest. program represents the programs to be performed, including source code and binary files, and execute represents the execution procedures. For example, if SPs install a C-language program, they send source codes and scripts for compiling and execution.

3.4 Event Notification NBI

The VNP provides SPs with an event notification NBI. VnMonitoringRequest(resource_id, condition) is used for receiving notification when the performance of a certain resource degrades. Parameter `resource_id` specifies an identifier of resources such as links and virtual machines. Parameter `condition` specifies the condition of the performance, e.g., "the available bandwidth is lower than 100 Mb/s". The response includes the resource identifier, the condition, the notification information such as current performance, and a timestamp.

After the request, SPs receive alerts from the VNP through `AlertNotificationRequest` if the monitored resources degrade. Received information includes monitored resource identifier, condition, current performance, and timestamp.

4. Prototype Implementation and Demonstration

We used the architectural design presented above to develop a prototype implementation. In this section, we describe a use case scenario and a demonstration of the implementation in nation-wide JGN-X and RISE testbed networks [27].

4.1 Prototype Implementation

The prototype of the proposed network orchestration architecture is depicted in Fig. 3. One InP, one VNP, and one SP are shown. The InP and VNP each contain three types of controllers for VN construction, monitoring, and control. We use the Linux OS for the construction and monitoring controllers of the VNPs and InPs, OpenFlow switches for forwarding devices, Trema [28] for the OpenFlow controller, KVM for virtual machines, and perfSONAR for monitoring tools. Although the implementation depends on the development policy of the InPs, in this paper we assume that all implementations are the same as the prototype implementation.

The VNP provides three types of NBI suites for construction, monitoring, and control. The VN construction NBI suite includes resource request, resource upgrade, and resource programming NBIs. The VN monitoring NBI suite includes resource monitoring and event notification NBIs. The VN control NBI suite includes all of the OpenFlow control interfaces. The VNP provides all NBIs in terms of construction and monitoring excepting a resource programming NBI using a RESTful interface via HTTP. The VNP provides the interface of a secure shell for the resource programming NBI so that the SP can install Linux programs on the special relay node.

4.2 Use Case Scenario

We consider a premium software telephony service for our use case scenario. The telephony service is sensitive to delay and jitter, so delay time and packet loss need to be prevented. In this use case scenario, the SP performs concurrent multipath transfer for redundancy using the special relay nodes rather than simple packet transfer using regular relay nodes. Moreover, the SP removes duplicated packets caused by the multipath transfer before arriving at end nodes

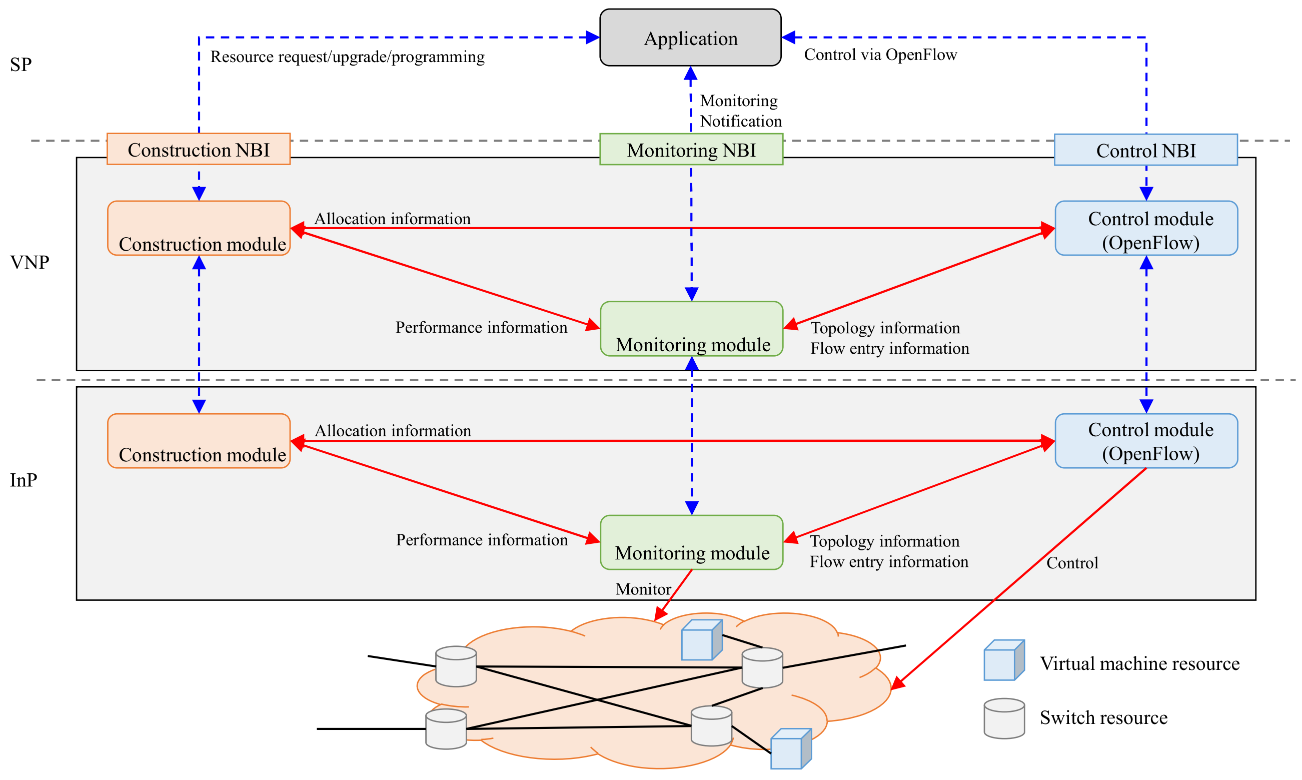


Fig. 3 Prototype implementation of proposed network orchestration architecture.

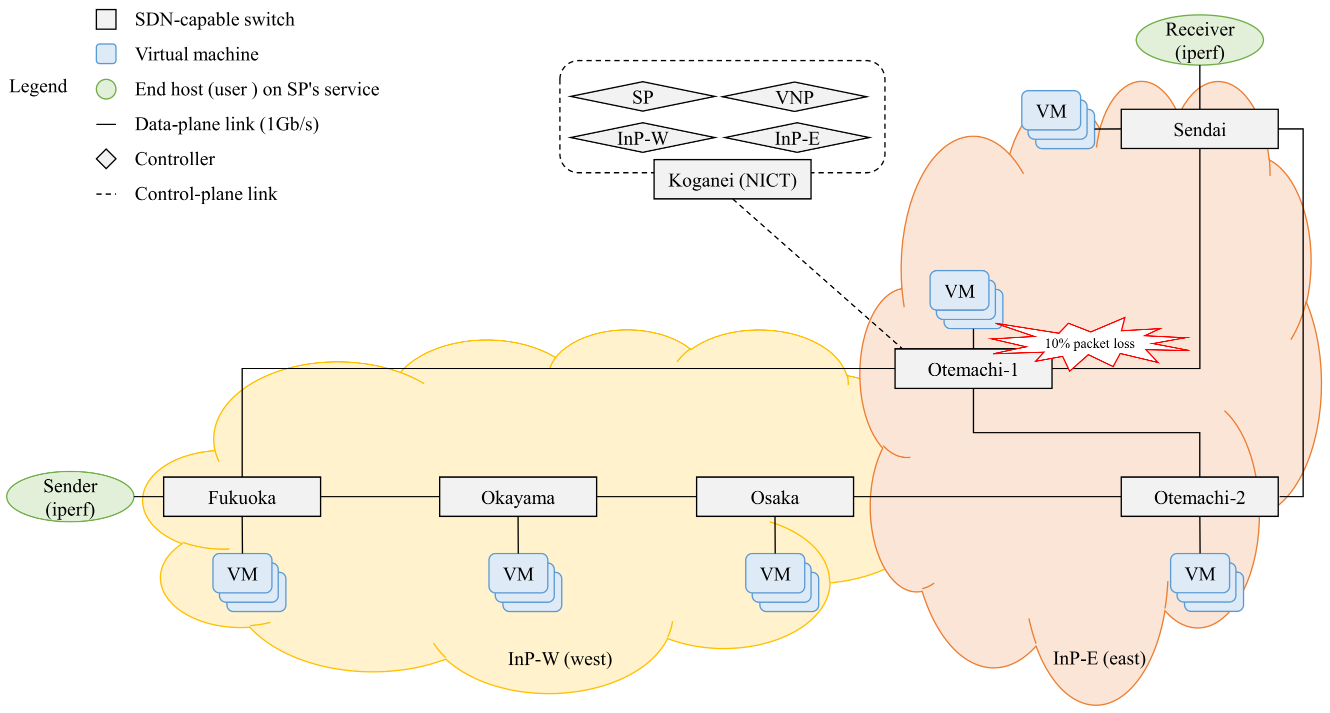
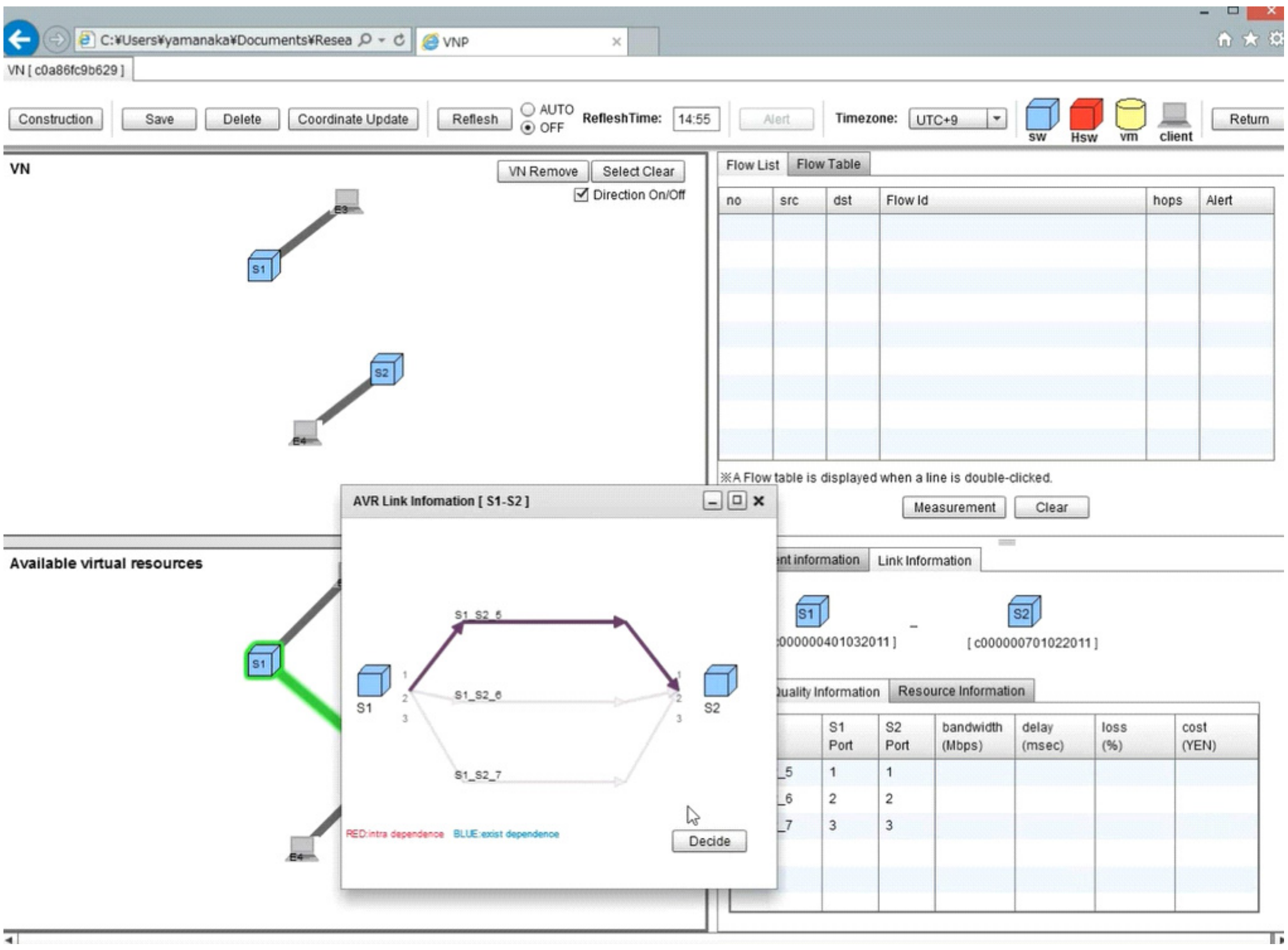


Fig. 4 Network topology for demonstration.

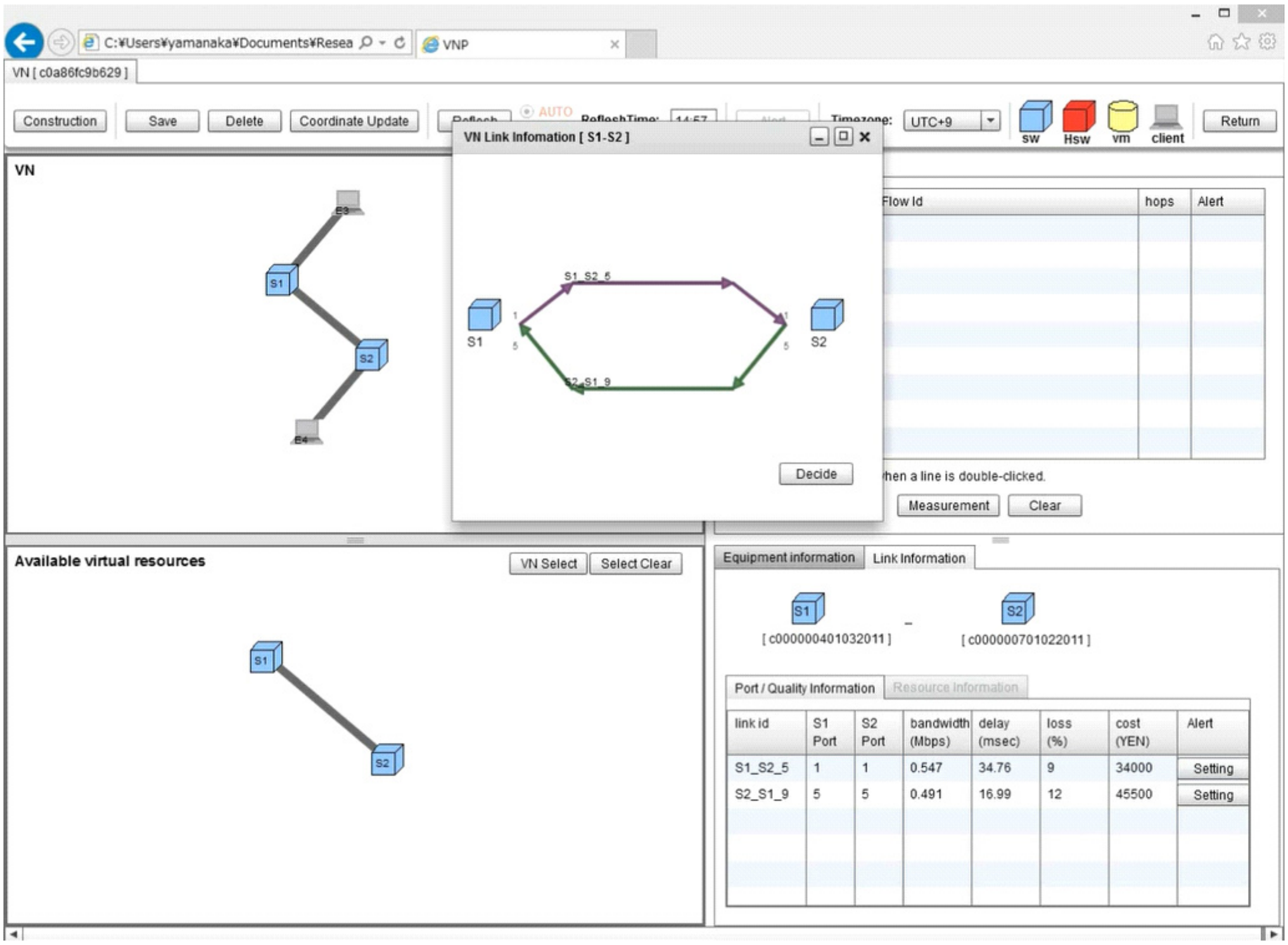


no	src	dst	Flow Id	hops	Alert

※A Flow table is displayed when a line is double-clicked.

	S1 Port	S2 Port	bandwidth (Mbps)	delay (msec)	loss (%)	cost (YEN)
_5	1	1				
_6	2	2				
_7	3	3				

Fig. 5 Available resource request.



link id	S1 Port	S2 Port	bandwidth (Mbps)	delay (msec)	loss (%)	cost (YEN)	Alert
S1_S2_5	1	1	0.547	34.76	9	34000	Setting
S2_S1_9	5	5	0.491	16.99	12	45500	Setting

Fig. 6 Logical network.

because the duplicated packets may cause unintended behavior at end nodes, e.g., the end nodes may misunderstand the duplicated packets as an occurrence of retransmission. To perform duplicated packet removal, arrival packets need to be properly stored and managed, so regular relay nodes cannot perform the removal processing. Consequently, the SPs use special relay nodes consisting of a combination of regular relay nodes and virtual machines.

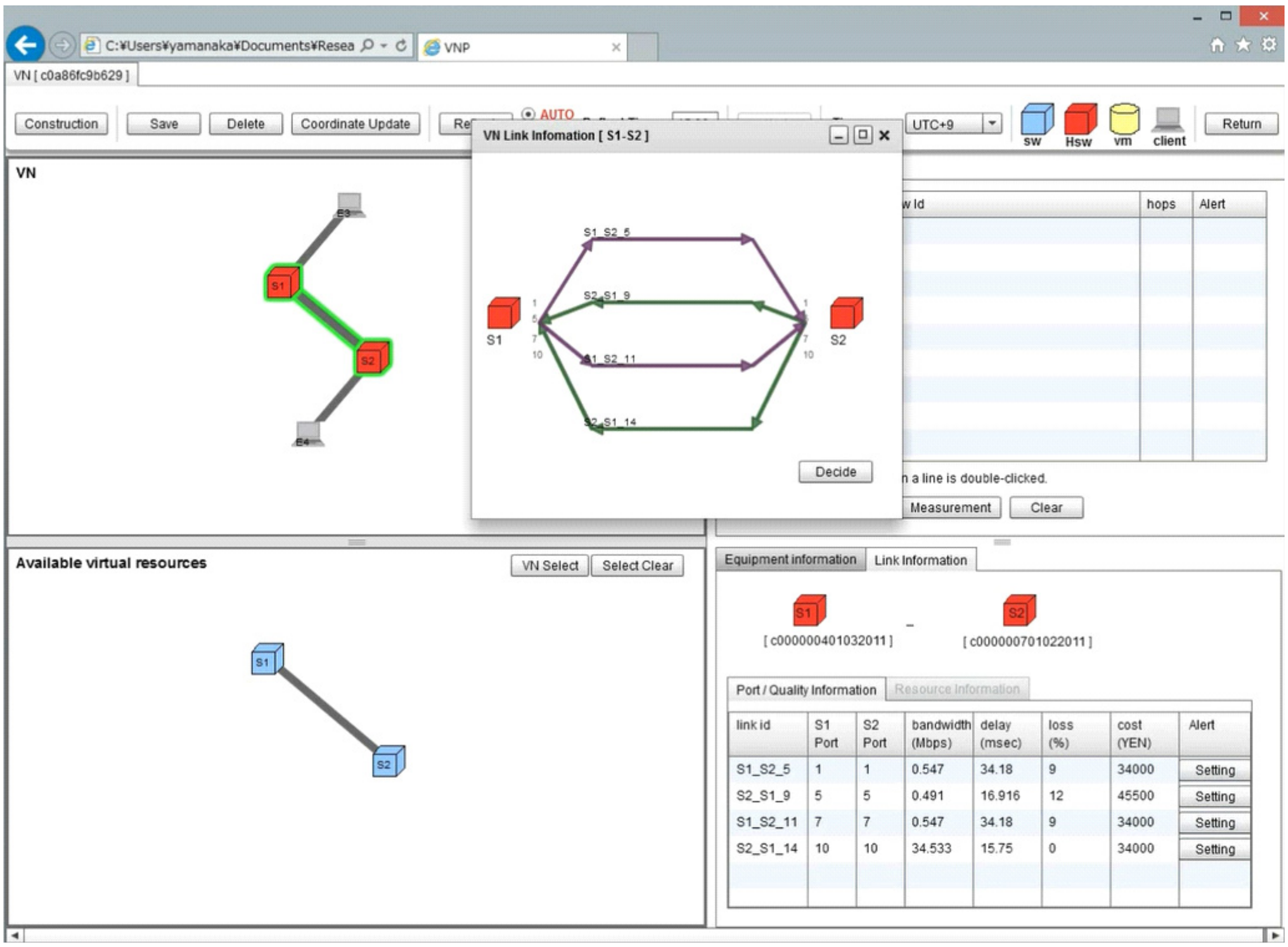
In this operation, the SP first sets the desired performance, such as a delay time of 5 ms, and obtains regular relay nodes and links via the resource request NBI. Next, the SP sets the condition of notification, e.g., "the delay time is larger than 10 ms", via the event notification NBI. When the SP receives the notification, it obtains the special relay nodes via the resource upgrade NBI and multiple links for the detour path via the resource request NBI. The SP then installs programs on the special relay nodes via the resource programming NBI. Note that, for the implementation in this paper, the SP removes a packet containing the same identification and header checksum fields of the IP header, i.e., a total of 32 bit fields.

4.3 Demonstration

Here, we investigate the proposed architecture using the prototype implementation and a sample application service. Figure 4 shows a network topology constructed on nationwide testbed networks, the JGN-X and RISE. There are one SP, one VNP, two InPs, and two end hosts on SP's service. Since there are detour paths between two end hosts, the SP can obtain several logical links. The SP attempts to provide delay and loss tolerant logical network for the premium software telephony service to its customers as described in Sect. 4.2.

As shown in Fig. 5, the SP requests available link resources between two OpenFlow switches connected to end nodes. The SP receives information of three candidate simplex links and chooses a single link from among them. After that, the SP obtains two simplex links between two OpenFlow switches, as shown in Fig. 6.

For this experiment, we use not only an OpenFlow physical switch but also Open vSwitch on the Linux OS. We set a 10% packet loss ratio of the links using Linux tc command on the Linux OS containing Open vSwitch. Then, the source end node sends data with UDP packets toward the



[c000000401032011] – [c000000701022011]

link id	S1 Port	S2 Port	bandwidth (Mbps)	delay (msec)	loss (%)	cost (YEN)	Alert
S1_S2_5	1	1	0.547	34.18	9	34000	Setting
S2_S1_9	5	5	0.491	16.916	12	45500	Setting
S1_S2_11	7	7	0.547	34.18	9	34000	Setting
S2_S1_14	10	10	34.533	15.75	0	34000	Setting

Fig. 7 Resource upgrade and detour path.

destination end node using iperf over the course of 30 seconds. Under this condition, the measured packet loss ratio is equal to 9.2%. To improve the performance, the SP upgrades obtained switches to special relay nodes and obtains two links from source to destination relay nodes and vice versa, as shown in Fig. 7. The source-side relay node duplicates incoming packets and forwards the original and duplicated packets to two respective links. When the destination-side relay node receives the packets, it checks the duplication and removes any subsequent duplicated packets. Under this condition, the packet loss ratio is decreased to 2.8%.

5. Concluding Remarks

We argued that it is essential to allow SPs to perform adaptive resource control in accommodating diverse application services over diverse multiple InPs, and considered that an integration of NVE, SDN, and NFV is a key to resource-efficient QoS management. This motivated us to design an elastic and adaptive resource orchestration architecture with NBIs for resource requests, resource upgrades and resource programming for VN construction and NBIs for resource monitoring and event notification for VN management, which is implemented on 3-tier network virtualization model. A mediator named VNP has a key role in resource management and orchestration, and it provides the above NBIs to SPs. The proposed NBIs in the VNP enable SPs to control their logical network and computational resources in dynamic and competitive resource sharing for achieving both resource efficiency and good QoS, instead of explicit and sufficient reservation of physical resources for their VNs. The demonstration of a premium software telephony service using a prototype implementation had shown the feasibility of our proposed architecture as well as the effectiveness of advanced relay processing performed by the SP.

References

[1] M. Shimamura, H. Yamanaka, Y. Uratani, A. Nagata, S. Ishii, K. Iida, E. Kawai, and M. Tsuru, "Architecture for resource controllable NVE to meet service providers' dynamic QoS demands," Proc. IEEE 4th International Conference on the Network of the Future (NoF 2013), pp.1–6, Oct. 2013.
[2] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," ACM SIGCOMM Comput. Commun. Review, vol.37, no.1, pp.61–64, Jan. 2007.
[3] Y. Zhu, R. Zhang-Shen, S. Rangarajan, and J. Rexford, "Cabernet:

Connectivity architecture for better network services," Proc. 2008 ACM CoNEXT Conference, Dec. 2008.
[4] J. Carapinha and J. Jiménez, "Network virtualization: A view from the bottom," Proc. 1st ACM workshop on Virtualized Infrastructure Systems and Architectures (VISA '09), pp.73–80, Aug. 2009.
[5] N.M.M.K. Chowdhury and R. Boutaba, "A survey of network virtualization," Comput. Networks, vol.54, no.5, pp.862–876, April 2010.
[6] Open Networking Foundation (ONF), March 2011. Information available at http://www.opennetworkingfoundation.org/
[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Comput. Commun. Review, vol.38, no.2, pp.69–74, April 2008.
[8] OpenFlow, "OpenFlow switch specication version 1.3.1," Sept. 2012. Information available at https://www.opennetworking.org/sdn-resources/technical-library
[9] European Telecommunications Standards Institute (ETSI), "Network functions virtualisation (NFV); terminology for main concepts in NFV," Oct. 2013. Information available at http://www.etsi.org/technologies-clusters/technologies/nfv
[10] M. Shimamura, H. Koga, T. Ikenaga, and M. Tsuru, "Compressing packets adaptively inside networks," IEICE Trans. Commun., vol.E93-B, no.3, pp.501–515, March 2010.
[11] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," IEEE Commun. Mag., vol.50, no.7, pp.26–36, July 2012.
[12] G. Xylomenos, C.N. Ververidis, V.A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K.V. Katsaros, and G.C. Polyzos, "A survey of information-centric networking research," IEEE Commun. Surveys & Tutorials, vol.16, no.2, pp.1024–1049, May 2014.
[13] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," Commun. ACM, vol.55, no.1, pp.117–124, Jan. 2012.
[14] A. Fischer, J.F. Botero, M.T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," IEEE Commun. Surveys & Tutorials, vol.15, no.4, pp.1888–1906, Nov. 2013.
[15] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of YouTube video streaming," Proc. 2013 European Workshop on Software Defined Networking (EWSDN), pp.87–92, Oct. 2013.
[16] OpenDaylight, April 2013. Information available at http://www.opendaylight.org/
[17] "GENI: Global environment for network innovations." Information available at http://www.geni.net/
[18] European Telecommunications Standards Institute (ETSI), "Network functions virtualisation (NFV); management and orchestration," Dec. 2014. Information available at http://www.etsi.org/technologies-clusters/technologies/nfv
[19] perfSONAR, "Performance service-oriented network monitoring architecture," Information available at http://www.perfsonar.net/
[20] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the test-bed?," Proc. 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10), June 2010.
[21] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," IEEE Internet Comput., vol.17, no.2, pp.20–27, March–April 2013.
[22] Kernel Based Virtual Machine (KVM). Information available at http://www.linux-kvm.org/
[23] LXC — Linux Containers. Information available at https://linuxcontainers.org/
[24] J. Seedorf and E. Burger, "Application-layer traffic optimization (ALTO) problem statement," RFC 5693 (Informational), Oct. 2009. Information available at http://www.ietf.org/rfc/rfc5693.txt
[25] R.D. Corin, M. Gerola, R. Riggio, F. de Pellegrini, and E. Salvadori, "VeRTIGO: Network virtualization and beyond," Proc. 2012 European Workshop on Software Defined Networking (EWSDN), pp.24–29, Oct. 2012.
[26] M. Shimamura, T. Ikenaga, and M. Tsuru, "A design and prototyping of in-network processing platform to enable adaptive network services," IEICE Trans. Inf. & Syst., vol.E96-D, no.2, pp.238–248, Feb. 2013.
[27] National Institute of Information and Communications Technology (NICT), "New generation network testbed JGN-X," April 2011. Information available at http://www.jgn.nict.go.jp/english/
[28] Trema: Full-Stack OpenFlow Framework. Information available at https://github.com/trema/trema

Masayoshi Shimamura received the M.E. and Ph.D. degrees from the Graduate School of Information Science of Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2005 and 2009, respectively. From 2008 to 2011, he was a postdoctoral researcher at the Network Design Research Center of Kyushu Institute of Technology. From 2011 to 2014, he was an assistant professor at the Global Scientific Information and Computing Center of Tokyo Institute of Technology. From 2014 to 2015, he was a researcher at Network Application Engineering Laboratories Ltd. Since 2016, he has been a researcher at Kyushu Research Center of iD Corporation. His research interests include performance evaluation of networking systems, new generation networks, software-defined networking, and network virtualization.

Hiroaki Yamanaka received M.E. and Ph.D. from Osaka University in 2008 and 2011, respectively. From 2011, he is a researcher at Network Testbed R&D Laboratory of National Institute of Information and Communications Technology (NICT). His current research work is focusing on network virtualization and OpenFlow.

Akira Nagata received the B.E. and M.E. degrees from Osaka University in 2000 and 2002, respectively, and then received his Ph.D. degree from Kyushu Institute of Technology, Japan in 2014. He worked at Fujitsu Laboratories Ltd. from 2002 to 2007 and Network Application Engineering Laboratories Ltd. from 2007 to 2015. In 2015, he moved to iD Corporation. He is also a research member of Kyushu Institute of Technology, Japan. He has been engaged mainly in the research of network architecture and network system.

Katsuyoshi Iida received the B.E., M.E. and Ph.D. degrees in Computer Science and Systems Engineering from Kyushu Institute of Technology (KIT), Iizuka, Japan in 1996, in Information Science from Nara Institute of Science and Technology (NAIST), Ikoma, Japan in 1998, and Computer Science and Systems Engineering from KIT in 2001, respectively. Since Oct. 2000, he was an Assistant Professor in the Graduate School of Information Science, NAIST. Currently, he is an Associate Professor in the Global Scientific Information and Computing Center, Tokyo Institute of Technology, Tokyo, Japan. His research interests include network systems engineering such as network architecture, performance evaluation, QoS, and mobile networks. He is a member of the WIDE project and IEEE. He received the 18th TELECOM System Technology Award, the Telecommunications Advancement Foundation, Japan, the IEICE Communications Society's distinguished service award, and Tokyo Tech young researcher's award in 2003, 2005 and 2010, respectively.

Eiji Kawai received Ph.D. from Nara Institute of Science and Technology in 2001. He worked for many years on a wide variety of information system technologies such as operating systems, Internet, performance evaluation, and virtualization. Since 2009, he has been working for National Institute of Information and Communications Technology, and now he is director of network testbed research and development laboratory and leading the RISE OpenFlow/SDN testbed project.

Masato Tsuru received B.E. and M.E. degrees from Kyoto University, Japan in 1983 and 1985, respectively, and then received his D.E. degree from Kyushu Institute of Technology, Japan in 2002. He worked at Oki Electric Industry Co., Ltd. (1985–1990), Information Science Center, Nagasaki University (1990–2000), and Japan Telecom Information Service Co., Ltd./Telecommunications Advancement Organization of Japan (2000–2003). In 2003, he moved to the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology as an Associate Professor, and then has been a Professor in the same department since April 2006. His research interests include performance measurement, modeling, and management of computer communication networks. He is a member of the IEEE, ACM, IPSJ, and JSSST.