INVITED PAPER *Special Section on Advanced Information and Communication Technologies and Services in Conjunction with Main Topics of APCC2015*

Design and Deployment of Enhanced VNode Infrastructure — Deeply Programmable Network Virtualization

Kazuhisa YAMADA[†a)], **Akihiro NAKAO**[††], ***Members***, **Yasusi KANADA**[†††], **Yoshinori SAIDA**[††††], **Koichiro AMEMIYA**[†††††], ***Nonmembers***, ***and*** **Yuki MINAMI**[†], ***Member***

SUMMARY We introduce the design and deployment of the latest version of the VNode infrastructure, VNode-i. We present new extended VNode-i functions that offer high performance and provide convenient deep programmability to network developers. We extend resource abstraction to the transport network and achieve highly precise slice measurement for resource elasticity. We achieve precise resource isolation for VNode-i. We achieve coexistence of high performance and programmability. We also enhance AGW functions. In addition, we extend network virtualization from the core network to edge networks and terminals. In evaluation experiments, we deploy the enhanced VNode-i on the JGN-X testbed and evaluate its performance. We successfully create international federation slices across VNode-i, GENI, and Fed4FIRE. We also present experimental results on video streaming on a federated slice across VNode-i and GENI. Testbed experiments confirm the practicality of the enhanced VNode-i.

key words: *network virtualization, deep programmability, resource abstraction, resource isolation, testbed, international federation*

1. Introduction

The current communications infrastructure has flourished as it is based on the Internet Protocol (IP), which is best known for its basis in the Internet. However, there are limits to the Internet, and to address all the various problems facing the current communications infrastructure, an information and communications base that has an innovative design must be constructed [1]. Network virtualization [2], [3] has recently attracted attention and is one technology that can help achieve this innovative communications infrastructure. We are promoting an advanced network virtualization technology [4] with the aim of constructing an information and communications base that incorporates innovative design ideas. From a general viewpoint, a communication infrastructure consists of "links" that provide network resources for transmitting data and "nodes" that provide computing resources and storage resources to execute programs for interpreting protocols and processing data. Advanced network virtualization technology virtualizes whole networks based on this general viewpoint and offers many advantages to network users. Specifically, advanced network virtualization technology meets five requirements: resource abstraction, resource isolation, resource elasticity, deep programmability, and authentication, authorization and accounting (AAA). This technology enables the creation and design of new generation networks from a clean slate and so inspires innovative thinking. In addition, advanced network virtualization technology provides a new generation information and communications infrastructure that can host multiple virtual networks at the same time. To achieve advanced network virtualization technology, we designed and developed a network virtualization infrastructure called VNode-i [5]–[7].

In this paper, we introduce the design and deployment of the latest version of VNode-i. We present new functions that offer high performance and provide convenient deep programmability to network developers. We extend resource abstraction to the transport network and achieve highly precise slice measurement for resource elasticity. We achieve precise resource isolation for VNode-i. We achieve coexistence of high performance and programmability. We also enhance AGW functions. In addition, we extend network virtualization from the core network to edge networks and terminals. In evaluation experiments, we deploy the enhanced VNode-i on the JGN-X [8] testbed and evaluate its performance. We also successfully create an international federation slice across VNode-i, GENI [9] in the US, and Fed4FIRE [10] in the EU. Finally, we show the results of a video streaming experiment on a federated slice across VNode-i and GENI. Through some experiments on the testbed, we confirm the feasibility of the enhanced VNode-i. Table 1 shows an abbreviation and acronym list.

2. VNode-i

2.1 Characteristics of VNode-i

One of the features of VNode-i is deep programmability. Here, we describe a comparison between VNode-i and state-of-the-art technology for network programmability. Including Software Defined Networking (SDN) and Network Function Virtualization (NFV), research and development related to the introduction of network programmability has been promoted actively in recent years. SDN has achieved flexible network flow control by introducing programmability

Manuscript received November 26, 2015.
Manuscript revised March 18, 2016.

[†]The authors are with NTT Network Innovation Laboratories, NTT Corporation, Yokosuka-shi, 239-0847 Japan.
[††]The author is with The University of Tokyo, Tokyo, 105-0011 Japan.
[†††]The author is with Hitachi Ltd., Kokubunji-shi, 185-8601 Japan.
[††††]The author is with NEC Corporation, Tokyo, 108-8001 Japan.
[†††††]The author is with Fujitsu Laboratories Ltd., Kawasaki-shi, 211-8588 Japan.
a) E-mail: yamada.kazuhisa@lab.ntt.co.jp
DOI: 10.1587/transcom.2015CCI0002

Table 1 Abbreviation and acronym list.

Abbreviation	Description
AGW	Access Gateway
Fed4FIRE	Federation for Future Internet Research and Experimentation
GENI	Global Environment for Network Innovations
JGN-X	Japan Gigabit Network eXtreme
LS	Link Sliver
NMM	Network Monitoring Manager
NS	Node Sliver
NVMS	Network Virtualization Management System
SNC	Service Network Controller
TNC	Transport Network Controller
UT	User Terminal
VNode-i	VNode Infrastructure

Table 2 Programmability comparison.

	C-plane	D-plane	Protocol
SDN	Flow control (OpenFlow)		
NFV		Network appliances	
VNode-i	Flow control (All protocols)	In-network processing	New protocol

into the control plane to apply the OpenFlow [11] technology. For SDN, offer of Open Source Software (OSS) such as the Open Network Operating System (ONOS) [12] and OpenDaylight [13] has also started.

On the other hand, NFV is aimed at flexible provision of network appliances by introducing programmability to the data plane. The European Telecommunications Standards Institute (ETSI) is currently in the process of standardizing NFV [14]. VNode-i can provide the programmability of the control plane and the data plane at the same time. In addition, VNode-i is aimed at achieving deeper programmability. Specifically, VNode-i achieves in-network processing and wide-area generic processing such as cache and transcoding. Furthermore, VNode-i can handle a new protocol to replace the current TCP/IP architecture. Table 2 shows a programmability comparison of SDN, NFV, and VNode-i.

2.2 Original VNode-i

In this section, we briefly describe the original VNode-i. Figure 1 shows its basic construction. VNode-i provides programmability to both the control/management plane and data plane by establishing slices. Each slice can have a different network topology and software functions. VNode-i consists of multiple VNodes, access gateways (AGWs), and a network virtualization management system (NVMS). We describe the VNode, AGW, NVMS, and a slice in the following sections.

2.2.1 VNode

A VNode is the main component of VNode-i. A VNode is an integrated node unit consisting of the Programmer and Redirector. VNodes provide virtualized resources such as a slice according to the slice design specifications.

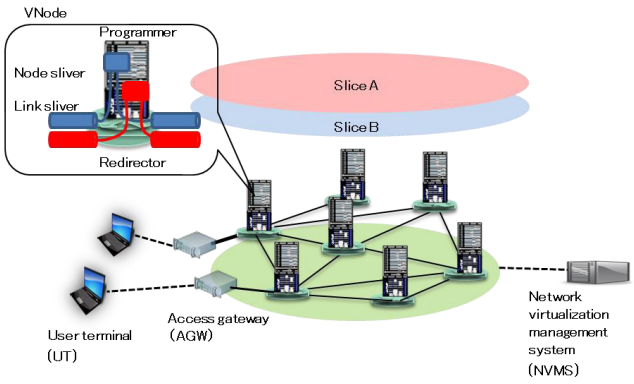

Fig. 1 VNode-i.

1) Programmer: We construct a VNode in two parts, Programmer and Redirector, because the technologies for providing virtualized computing resources and link resources may advance at different speeds. The Programmer provides various processing components such as general-purpose servers, network processors, and OpenFlow switches to yield various combinations of virtualized computing resources. The Programmer provides virtual machines (VMs) as slice nodes that work as processing components. We can deploy various software functions on a VM to enhance service functionalities.
2) Redirector: The Redirector provides virtualized link resources such as bandwidth and buffer size as virtual links to connect slice nodes. We achieve the desired Quality of Service (QoS) by instructing the Redirector to transfer traffic data with particular characteristics such as a streaming service.

2.2.2 AGW

An AGW is a border node between the end-user access side and the VNode-i domain. An AGW authenticates user terminals (UTs) and accommodates UTs into the slice that they are authorized to access.

2.2.3 NVMS

The NVMS manages and controls all VNode-i resources. The NVMS provides a portal site to slice developers/owners as well as AAA functions. Slice developers/owners can reserve resources via the portal with operational security. Slice developers can create slices via a GUI on a web browser.

2.2.4 Slice

A slice is a set of connected computation resources and link resources. These resources are virtualized and isolated. Therefore, we can configure our own network topology and deploy our own software functions into network nodes, i.e., VNode-i provides programmability.
1) Node sliver: A node sliver is a set of virtualized node

resources such as CPU time, memory, and storage. We can assign these resources to nodes of a slice and install an operating system such as Linux on the nodes by writing a slice design specification that respects the limitations of the physical resources. Node slivers work as various processing components and are mapped into the Programmer of the VNodes.

2) Link sliver: A link sliver is a set of link resources such as bandwidth, burst size for QoS control, and queuing delay. The slice design specification determines how resources are assigned to the links of a slice. Link slivers are mapped into the Redirector of the VNodes.

3. Enhanced VNode-i

This section presents new VNode-i functions that achieve high performance and make deep programmability easier for network developers to access while satisfying the five requirements described in the Introduction. In addition, we extend network virtualization from the core network to edge networks, terminals, and other network virtualization infrastructures. The specific functions are described below.

3.1 Edge Network Virtualization

Network virtualization technology that covers the entire gamut from edge networks to terminals as well as the core network is necessary to achieve the slice concept. In general, edge and core networks have different requirements. In network virtualization, sufficient resources and a highly precise band guarantee are required so that as many slices as possible can share the resources of the core network. On the other hand, various types of terminals must be contained in a scalable slice and virtualization technology must be easy to introduce. Therefore, we decided to use the core and edge networks to constitute the infrastructure.

We developed FLARE, a light-weight and low-power consumption network virtualization node for edge networks. Figure 2 shows the FLARE node architecture. The hardware of the FLARE node connects many-core processors that execute packet processing on the data plane and x86 architecture Intel processors that control packet processing on the control plane with a PCIe interface. Using a VM constructed using the lightweight Linux container (LXC) technique, virtualization is executed on both types of processors, and an outside FLARE node management server creates a slice in the FLARE node. Since the slice is created on many-core processors, isolating the processing between slices is achieved by assigning a core to each slice. In addition, the FLARE node offers a slicer-slice, which is a special slice for allocating packets in a slice based on tag information that is added to the I/O port or a packet. Using the slicer-slice, individual network processing is enabled in each slice. Figure 2 shows an Ethernet switch established on slice 1, a switch for packets whose MAC address length is extended is established on slice 2, and an OpenFlow switch is established on programmable slice n.

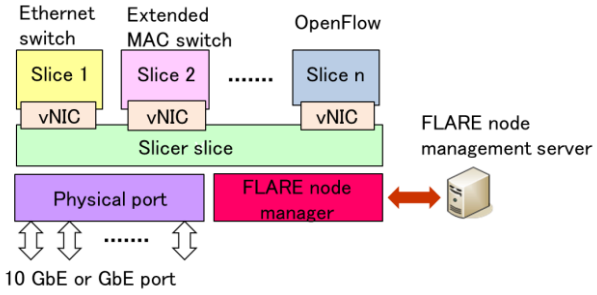

Fig. 2 FLARE node.

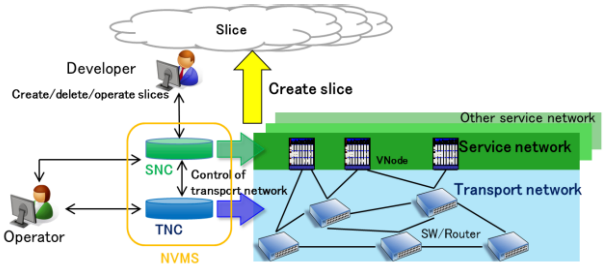

Fig. 3 Service network and transport network control.

3.2 Extended Resource Abstraction and Elasticity

VNode-i achieves flexible and integrated resource management by abstraction and virtualization of physical resources. We extend resource abstraction to the transport network and achieve integrated control and management of slices and the transport network.

The network infrastructure that provides general network services consists of a "service network" (service nodes that provide network service) and a "transport network" that connects and transmits packets between service nodes. The transport network transfers packets independent of the service contents and is used in common by multiple services. The service network also consists of VNodes and the transport network, which consists of existing switches and routers. Therefore, integrated control and management of the service and transport networks are necessary in VNode-i. Figure 3 shows an integrated control and management system for VNode-i. We designed a service network controller (SNC) that controls and manages the service network and a transport network controller (TNC) that controls and manages the transport network as the NVMS. The TNC automatically allocates physical resources to the transport path that is necessary to establish a link sliver at slice creation in cooperation with the SNC. Transport network resources are managed as a transport path in the SNC, and the SNC notifies the TNC of the resource requirements such as the bandwidth to create a slice. The TNC allocates the required transport path with the prepared physical resources and offers it to the SNC.

Due to resource elasticity, VNode-i can dynamically change the resources allocated to a slice and can maintain ser-

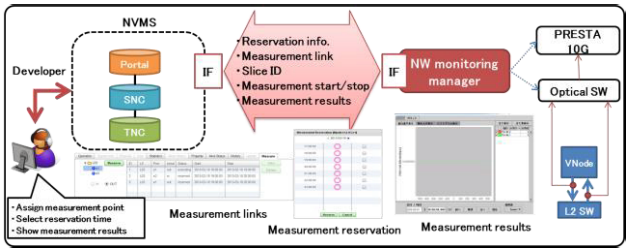

Fig. 4 Highly precise slice measurements based on NVMS-NMM cooperation.

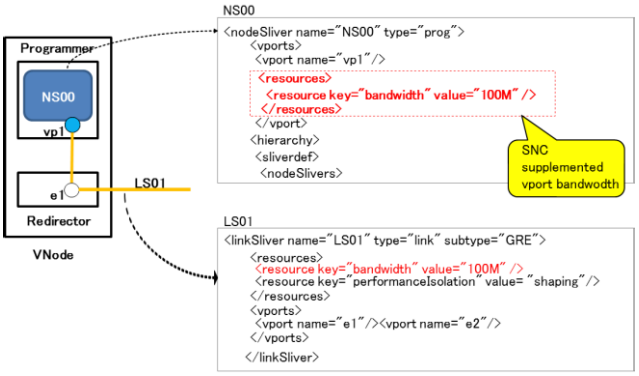

Fig. 5 Resource isolation between Programmer and Redirector.

vice quality and resource optimization. To achieve resource optimization, the performance of each slice in VNode-i must be precisely measured. To this end, we developed the Network Monitoring Manager (NMM). Figure 4 shows the highly precise slice measurement system based on NVMS-NMM cooperation. The Developer, the slice owner, indicates the ID of the slice to be measured, the measurement point in the slice, and the date and time for measurement reservation from the NVMS Portal screen. The NVMS offers measurement link information and slice identification information to the NMM. Then, the NMM configures the optical switches to set the measurement point and the filter in the existing network monitoring equipment, PRESTA 10G [15]. The NMM notifies the developer of the measurement results through the NVMS.

3.3 Precise Resource Isolation

Preventing resource interference between slices is critical for network virtualization. For example, a slice must not consume bandwidth to the extent that it imposes an unfair delay on other slices. A function to avoid such resource interference is resource isolation. An early version of VNode-i offered a basic resource isolation function in the Redirector. Its basic function is described below. The Redirector incorporates an L3 switch with a VLAN function, IP routing function, and QoS functions, which are Weighted Fair Queues (WFQs) and policing. WFQs can achieve high-performance bandwidth control for each slice, but since it employs relatively expensive high-speed memory, the number of supported slices is limited.

The latest version of Redirector provides resource isolation based on WFQs as well as more precise resource isolation control for each link sliver. It includes a hierarchical shaper, a new hardware module. The previous Redirector used three broadband WFQs and offered per-slice resource isolation. The hierarchical shaper prepares up to 4000 narrowband WFQs in addition to the three broadband WFQs, so resource isolation for each link sliver is achieved. The latest Redirector uses the broadband WFQs for slices with more than 1 Gbps bandwidth and the narrowband WFQs for all other slices. The hierarchical shaper provides precision resource isolation that is scalable from 10 kbps to over 1 Gbps. Another problem is to achieve resource isolation throughout VNode-i including the Programmer, AGW, and transport network. We establish cooperation in resource management among the Programmer, Redirector, and AGW to address this problem. Specifically, the SNC supplements the link sliver specifications to ensure isolation in the slice design file and orders resource isolation for the Programmer and AGW. The Programmer and AGW achieve the indicated resource isolation, so resource isolation is achieved throughout VNode-i. Figure 5 shows resource isolation cooperation between the Programmer and Redirector. The Slice developer only defines the link sliver (LS01) bandwidth for resource isolation using the slice design file. The SNC automatically supplements the bandwidth of the vport (vp1) on the node sliver (NS00). The Programmer sets the bandwidth of vp1 in accordance with the supplemented slice design file, which achieves resource isolation between the Programmer and Redirector.

3.4 Coexistence of High Performance and Programmability

The Programmer, which is one of the VNode components, provides programmability for network processing. In addition, high-speed packet forwarding is also required for the Programmer. To achieve both functions, we actualized various network functions as programs by preparing two kinds of mechanisms: one, the slow path, provides a flexible programing environment using VMs on general-purpose servers; the other, the fast path, provides a programming environment in which network processors transfer packets at high speed. With regard to the slow path, we address the problem of achieving high computing performance given the gap in the network I/O performance.

Figure 6 shows the Programmer architecture. We use OpenFlow switches to connect the computer resources in the device. This allows us to build a network node that combines various computer resources freely. The packet converter converts the packet format (MAC-in-MAC to VLAN format) between that for the Redirector and the Programmer. We use a software switch (vSwitch) in a slow path processor for VM communications. In conventional slow path processors, software switching performance and processor I/O are performance bottlenecks. To achieve coexistence between

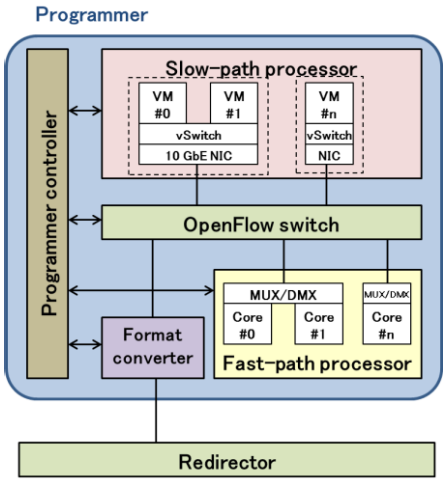

Fig. 6 Programmer architecture.

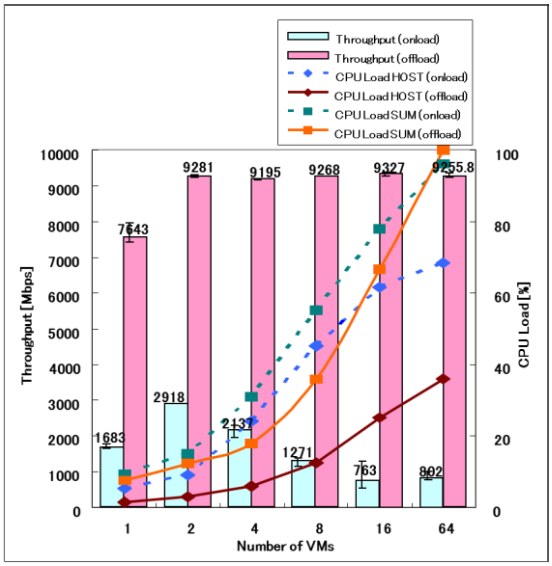

Fig. 7 Performance evaluation results for slow path.

high performance and programmability, we apply the hardware offload technique to the slow path. In addition, we enhance the network I/O performance by applying 10 GbE NIC hardware. We add new 10 GbE NICs that support the Single Root I/O Virtualization (SR-IOV) function to slow path processors; this ensures high performance switching and forwarding for VM communications. If the Offload operation is specified, the VM communicates with the NIC directly, without passing through the vSwitch. We achieve dynamic onload/offload switching technology per slice and integrated management of the vSwitch and NIC for flexible resource assignment. Figure 7 shows performance evaluation results for the slow path. Offload represents results when applying the hardware offload technique, and onload represents results without the offload technique. The SUM of the CPU Load is the total CPU utilization of the guest OS, and HOST is only for the host OS. The wire-speed of 10 Gbps can be achieved with offload. On the other hand, the throughput is less than 3 Gbps and the performance degrades when more than 2 VMs are active using I/O without offload. In addition, the CPU usage is lower with offload in all cases. The figure shows that the throughput performance improves and that the CPU utilization of the host OS decreases when using the offload technique. The results show that the proposed method is effective in improving the performance.

3.5 Enhancement of AGW

The AGW is gateway equipment for slices in VNode-i. The AGW is deployed at VNode-i edges and provides connectivity and authentication between slices and physical devices or networks using various protocols including proprietary ones. The enhanced functions of the AGW are described hereafter.

The first enhanced function is connectivity. The AGW identifies the users and connects user devices or networks with slices using various protocols. We provide VLAN accommodation in addition to the conventional Security Architecture for Internet Protocol (IPsec) accommodation.

Table 3 AGW performance.

Throughput with IPsec	Throughput with VLAN	Slow Path Node Sliver Throughput
1.3 Gbps*1	4.7 Gbps*1	1.5 Gbps*2

*1. 1372 byte frame, Intel Xeon X5690 (3.46 GHz/6 cores) x 2
*2. Using Intel Xeon X5690 (3.46 GHz/6 cores) and allocate vCPUx2 and 2 GB mem. for VM, vCPUx4 for vhostnet

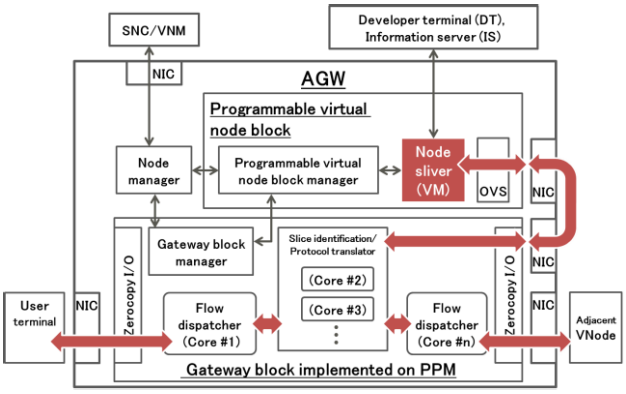

Fig. 8 AGW architecture.

The second enhanced function is easy-opt-in. The AGW authenticates the users and their corresponding packets using the IPsec protocol when users connect their terminals to the slices through the gateway. However, it takes labor to set the IPsec information manually, so the enhanced AGW uses Internet Key Exchange protocol version 1 (IKEv1) to perform automatic key exchange in the network.

The third enhanced function is programmability. The latest version of the AGW provides programmable virtual nodes (Node Sliver VM) on programmable virtual node blocks and enables execution of network/data processing applications at the edge of the slices.

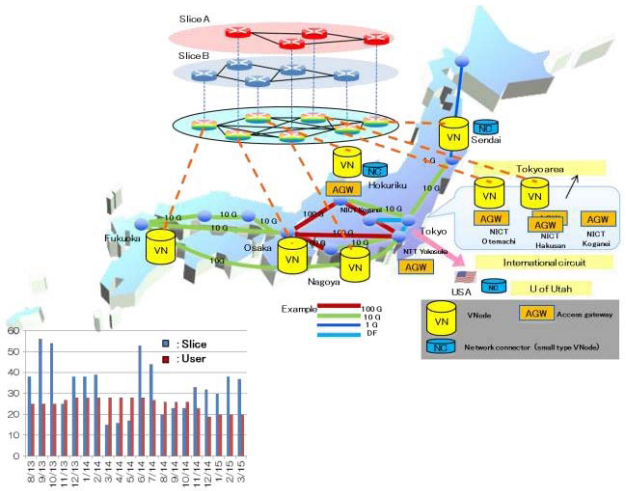

Fig. 9 Deployment of VNode on JGN-X testbed.

The fourth enhanced function is a customizable protocol stack on the gateway block. This functionality allows configuration of the gateway interface in accordance with the protocols used in user terminals and/or networks.

Additionally, the latest AGW provides higher performance for frame transfer even on commodity Intel architecture (IA) servers (Table 3). Higher performance is achieved through the Packet Processing Middleware (PPM) functionalities, Zero Copy I/O, and parallel processing framework utilizing multiple/multi-core processors.

Figure 8 shows the internal architecture and the communication path in the AGW. The AGW must have a small footprint with emphasis on economy rather than high performance. This is because VNode-i has more AGWs than VNodes as AGWs will be widely distributed at VNode-i edges including near or at user locations.

In order to satisfy these requirements, we adopted the following policies in designing the AGW. First, all AGW functionalities, the programmable virtual node, and management functionalities, should be provided on a single commodity IA server. Second, the number of programmable virtual nodes provided by this AGW should be scalable as needed by adding IA servers running the programmable virtual node provider.

4. Deploying VNode-i on Testbed

We deployed the latest version of VNode-i on the JGN-X testbed by establishing 7 VNodes and 6 AGWs. Approximately 40 slices were used in the evaluation experiment as shown in Fig. 9.

We evaluated the throughput performance on the JGN-X testbed. Figure 10(a) shows the configuration for performance evaluation. We created node slivers on VNodes at Nagoya (NS01) and Osaka (NS02), and test packets were transmitted from NS01 to NS02. Figure 10(b) shows the configuration for performance evaluation for the previous version of VNode-i. We created node slivers on 2 VNodes

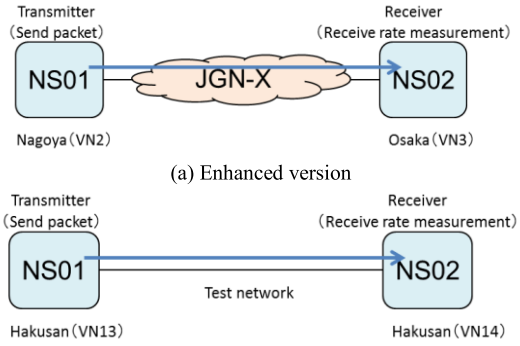

Fig. 10 Configuration for performance evaluation.

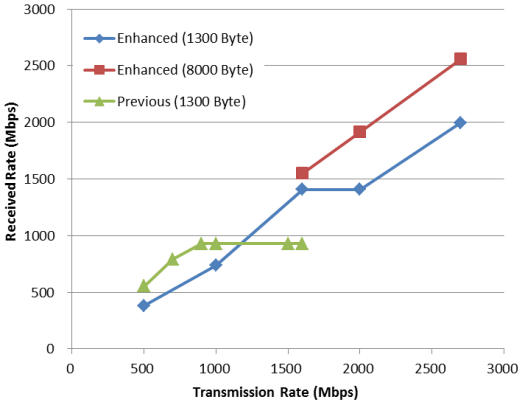

Fig. 11 Throughput performance evaluation.

in the test environment. Figure 11 shows the evaluation results. Since the Programmer Offload functionality is limited, 1500 Byte length following the packet transfer is limited to the value of the discrete bandwidth. Specifically, in the 1300 Byte length of the packet transfer, the transfer bandwidth is limited to 390 Mbps, 737 Mbps, 1387 Mbps, and 2340 Mbps. When performing a packet transmission exceeding these transfer bandwidths, the received packet rate exceeds the limited transfer bandwidth. On the other hand, for a packet transfer of 1500 Byte length or more, because there are no constraints, the transmission packet rate and packet receive rate are consistent. From the evaluation results, for the packet length of 1300 Byte, although up to the transmission rate of 2 Gbps receive rate is as expected, when the transmission rate of 2.7 Gbps the receive rate is 2 Gbps, which is below expectation. Therefore, the throughput performance for 1300 Byte packet length is confirmed as 2 Gbps. On the other hand, for the packet length 8000 Byte, since the receive rate was as expected for the transmission rate 2.7 Gbps, the throughput performance of 2.7 Gbps or more has been confirmed. In a throughput performance comparison, the previous version of VNode-i exhibited a limit of 900 Mbps, and so we confirmed an improvement in performance.

In another evaluation, we installed one small VNode

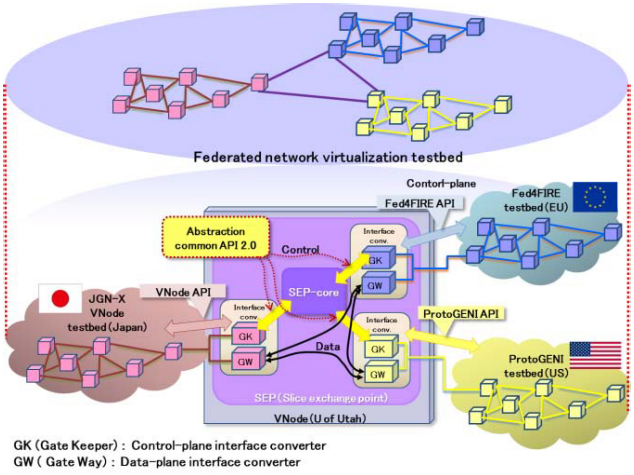

Fig. 12 International federation experiment among Japan, US, and EU.

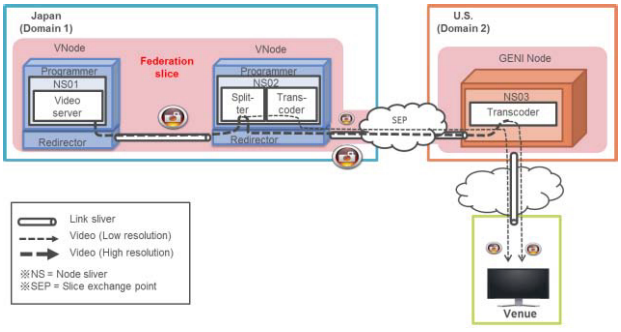

Fig. 13 Streaming over federated slice.

at the University of Utah and established an international federation with ProtoGENI, which is a network virtualization testbed of the GENI project in the US. Federation was also achieved with Fed4FIRE, which is a European network virtualization testbed and demonstrates slice construction in a global multi-domain environment. Figure 12 shows the construction of the international federation experiment. Federation was established using the Slice Exchange Point (SEP) [16], [17] within the small VNode at the University of Utah. We successfully created and evaluated large-scale international federation slices.

To confirm feasibility, we actually created a federation slice on VNode-i and GENI, and conducted some experiments on the slice. One example is introduced below. We transmitted a video stream from Japan to the US via a federation slice and measured resource utilization of both infrastructures under different conditions. We achieved multicast streaming and adaptive bit rate streaming by deploying the packet copy and transcoding functions.

The experimental configuration is shown in Fig. 13. It has three node slivers. NS01 is the video server and NS02 is a relay node; both were created on VNode-i. NS03, a relay node, is created on GENI. Table 4 shows the physical specifications and node sliver specifications for both domains. Because of practical limitations, these specifications are not equal.

Table 4 Physical specifications and node sliver specifications.

Specification	Node Sliver in VNode	Node Sliver in GENI
Physical server	Express5800/R120a-2	HP ProLiant DL 360e G8
Physical CPU	Intel Xeon X5550	Intel Xeon E5-2430L v2
Physical CPU clock rate	2.66 GHz	2.4 GHz
Allocated CPU	4 cores	4 cores * 2 thread
Allocated memory	4 GB	12 GB
OS	Ubuntu server 12.04	Ubuntu server 12.04

Table 5 Comparison between VNode and GENI.

	VNode (NS02)	GENI (NS03)
CPU utilization	32.9%	10.2%
Memory usage	419 MB	578 MB

(Average value)

We transmitted streaming content from Japan to the US and multicasting was achieved by installing a splitter in NS02. We achieved multicasting by deploying a very simple function that copies the packets and sets additional destination IP addresses. We call the multicasting function the splitter. In this case, the splitter is a simple program written in C. One stream was transcoded in NS02 and the other was transcoded in NS03. We installed a generic OS, Ubuntu server 12.04, on relay nodes and ran VideoLAN Client (VLC) [18] 2.0.8 on them as the transcoding function. These node slivers are virtual machines virtualized by KVM in both domains but other generic OSs can be used. Therefore, the same software functions were deployed on VNode and GENI. We used dstat to measure the resource utilization of both infrastructures [19]. In VNode-i, we deployed a QoS function, based on policing and shaping, for link slivers, so as to guarantee the link bandwidth of 50 Mbps (burst size is 50 kB). The QoS and guaranteed link bandwidth functions were achieved by the Redirector.

We show the measured results in Table 5. CPU utilization was measured by dstat on the node sliver. These values represent the measured load of the deployed functions. Table 5 gives the memory usage, which was also measured by dstat. These values show the processor memory used by all node processes and therefore do not include buffer memory or cache memory. CPU utilization of the VNode node sliver includes the multicasting function as well as the transcoding function. CPU loads created by the multicasting function are not large. We measured it separately and found it to be approximately 5%. We note, however, that CPU utilization was higher in the VNode domain since the CPU in the GENI node sliver works in Hyper-Threading mode. Therefore, it implements transcoding with a higher degree of parallelism. The GENI node sliver used more memory and so had lower CPU utilization. We confirmed that both streams were transcoded correctly and that the content played smoothly at the client.

5. Conclusion

In this paper, we introduced the design and deployment of the latest version of VNode-i. It offers new functions to

achieve high performance and to provide convenient deep programmability to network developers. We extended resource abstraction to the transport network and achieved highly precise slice measurement for resource elasticity. We achieved precise resource isolation for VNode-i. We achieved coexistence of high performance and programmability. We also enhanced AGW functions. In addition, we extended network virtualization from the core network to edge networks and terminals. We deployed the enhanced VNode-i on the JGN-X testbed in evaluation experiments and evaluated its performance. We successfully created an international federation slice among VNode-i, GENI and Fed4FIRE. We also described a video streaming experiment using a federated slice across VNode-i and GENI. Through testbed experiments, we were able to confirm the practicality of the enhanced VNode-i.

Acknowledgments

The research results have been achieved as part of the "New generation network R&D program for innovative network virtualization platform and its applications", ordered by Commissioned Research of the National Institute of Information and Communications Technology (NICT). The authors thank the VNode project members.

References

[1] "AKARI" Architecture Design project. [Online] Available: http://www.nict.go.jp/en/photonic_nw/archi/akari/akari-top_e.html
[2] N.M.M.K. Chowdhury and R. Boutaba, "A survey of network virtualization," Computer Networks, vol.54, no.5, pp.862–876, April 2010.
[3] N.M.M.K. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," IEEE Commun. Mag., vol.47, no.7, pp.20–26, July 2009.
[4] Network Virtualization Study Group (NVSG). (2012, June) Advanced Network Virtualization: Definition, Benefits, Applications, and Technical Challenges. [Online]. Available: https://nvlab.nakao-lab.org/nv-study-group-white-paper.v1.0.pdf
[5] A. Nakao, "VNode: A deeply programmable network testbed through network virtualization," 3rd IEICE Technical Committee on Network Virtualization, 2012.
[6] Y. Kanada, K. Shiraishi, and A. Nakao, "Network-resource isolation for virtualization nodes," IEICE Trans. Commun., vol.E96-B, no.1, pp.20–30, 2013.
[7] Y. Katayama, T. Yamamoto, Y. Tsukishima, K. Yamada, N. Takahashi, A. Takahara, and A. Nakao, "Design and implementation of network virtualization management system," IEICE Trans. Commun., vol.E97-B, no.11, pp.2286–2301, 2014.
[8] New Generation Network Testbed JGN-X. [Online]. Available: http://www.jgn.nict.go.jp/english/index.html
[9] Global Environment for Network Innovations (GENI) Project. [Online]. Available: http://www.geni.net
[10] Federation for Future Internet Research and Experimentation (Fed4FIRE). [Online]. Available: http://www.fed4fire.eu
[11] "Openflow." [Online] Available: http://archive.openflow.org
[12] "ONOS." [Online] Available: http://onosproject.org/
[13] "OpenDaylight." [Online] Available: http://www.opendaylight.org/
[14] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, J. Benitez, U. Michel, H. Damker, K. Ogaki, T. Matsuzaki, M. Fukui, K. Shimano, D. Delisle, Q. Loudier, C. Kolias, I. Guardini, E. Demaria, R. Minerva, A. Manzalini, D. López, F.J.R. Salguero, F. Ruhl, and P. Sen, "Network functions virtualisation — Introductory white paper," [Online] Available: http://portal.etsi.org/nfv/nfv_white_paper.pdf
[15] K. Shimizu, K. Sebayashi, and M. Maruyama, "Successful high-precision QoS measurement in widely deployed 10-Gbit/s networks based on general-purpose personal computers," NTT Technical Review, vol.7, no.3, 2009.
[16] T. Tarui, Y. Kanada, M. Hayashi, and A. Nakao, "Federating heterogeneous network virtualization platforms by slice exchange point," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp.746–749, 2015.
[17] Common API 2.0 [Online] Available: https://nvlab.nakao-lab.org/Common_API_V2.0.pdf
[18] VideoLAN Client (VLC). [Online]. Available: https://www.videolan.org/vlc/
[19] dstat. [Online]. Available: http://dag.wiee.rs/home-made/dstat/

Kazuhisa Yamada received the B.E. degree in electronics engineering from Gunma University, Gunma, Japan, in 1987. In 1987, he joined NTT Electrical Communication Laboratories, Yokosuka-shi, Japan, where he has been engaged in research on a programmable transport system. Since 1998, he has been developing carrier network operation systems. Since 2003, he has been developing a digital television relay network operation system. His current research interests include network virtualization and network management technologies for future networks. He is a senior research engineer at the NTT Network Innovation Laboratories.

Akihiro Nakao received B.S. degree (1991) in Physics and M.E. degree (1994) in Information Engineering from the University of Tokyo. He was at IBM Yamato Laboratory, Tokyo Research Laboratory, and IBM Texas Austin from 1994 to 2005. He received the M.S. degree (2001) and Ph.D. degree (2005) in Computer Science from Princeton University. He has been teaching as an associate professor (2005–2014) and as a professor (2014–present) in Applied Computer Science, at Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies, the University of Tokyo.

Yasusi Kanada received the B.E. degree in mathematical engineering from the University of Tokyo in 1979 and M.E. degree in information engineering from the University of Tokyo in 1981. He has been working for Hitachi, Ltd. since 1981. He stayed in Carnegie Mellon University from 1988 to 1990, and stayed in Tsukuba Laboratory of Real World Computing Partnership (RWCP) from 1992 to 1995. He received the Ph.D. degree from the University of Tokyo in 1992.

Yoshinori Saida received the B.S. and M.S. degrees in Electronic Engineering from Kyushu University in 1991 and 1993, respectively. In 1993, he joined NEC Corporation, Tokyo, Japan, where he has been engaged in research on a software platform for mobile terminals. He is a manager in the NEC Corporation.

Koichiro Amemiya received the B.S. and M.S. degrees in applied physics from the University of Tokyo, Japan, in 2000 and 2002 respectively. In 2002 he joined Fujitsu Laboratories Ltd., Kawasaki, Japan, where he has been engaged in the research of distributed computing. He is a senior researcher in the Software Laboratories in Fujitsu Laboratories Ltd. He is a member of the IPSJ.

Yuki Minami received the Bachelor of Information Science and Master of Information Science degrees from Osaka University, Japan, in 2009 and 2011, respectively. He joined the NTT Network Innovation Laboratories in 2011 and he engaged in research and development on network virtualization technology.