Network-virtualization Nodes that Support Mutually Independent Development and Evolution of Node Components

Yasusi Kanada
Hitachi, Ltd.,
Central Research Laboratory
Yokohama, Japan
Yasusi.Kanada.yq@hitachi.com

Kei Shiraishi
Hitachi, Ltd., Telecommunications &
Network Systems Division
Kawasaki, Japan
shiraishi_kei@itg.hitachi.co.jp

Akihiro Nakao
The University of Tokyo, Graduate School of
Interdisciplinary Information Studies
Tokyo, Japan
nakao@iii.u-tokyo.ac.jp

***Abstract* – "Virtualization nodes" (VNodes) for programmable network-virtualization platforms are being developed. Criteria for "clean" network-virtualization are devised and applied to this platform and slices (virtual networks). These criteria meet one of the challenges targeted by the Virtualization Node Project, that is, to enable mutually independent development and evolution of components (namely, computational components called *programmers* and networking components called *redirectors*) in VNodes. To meet this challenge, the redirector plays the central role in implementing the following two functions of VNodes. The first function is creation of mapping between virtual links to external physical paths and mapping between virtual links to internal physical paths, which makes it possible to hide various alternative computational components in the VNode from the external network and to hide these external-network representations from the internal components. The second function is implementation of high-performance data conversion, which connects the external and internal data formats or mappings, by using an add-on card with a network processor. Two results are obtained from the performance evaluation of these functions. First, the overhead caused by mapping creation can be hidden by other tasks under normal conditions, but the overhead caused by mapping deletion must be reduced. Second, the data-conversion rate is half the wire rate, which should be increased in future work.**

I. INTRODUCTION

In Japan, several projects targeting new-generation networks (NwGNs) have been conducted [Aoy 09][AKA 10]. These projects aim to develop new network protocols and architectures (i.e., the "clean slate" approach [Fel 07]) as well as various applications that are difficult to run on IPs but work well on NwGNs. In the Virtualization Node Project (VNP), Nakao et al. [Nak 10] has developed a virtualization-platform architecture called the VNode architecture and a high-performance fully functional virtualization testbed. The goal of this project is to develop an environment in which multiple slices (virtual networks) with independently and arbitrarily designed and programmed NwGN functions run concurrently, but are logically isolated, on a physical network. To isolate slices and separate each slice from the virtualization platform (substrate) completely, the platform and slices should satisfy certain criteria for "clean" network-virtualization; that is, the network structure of slices and the packet contents in slices (including addresses, if they exist) should be independent of the specific platform, and virtualization-related packet-header contents (such as slice identifiers) must be hidden from slices.

These criteria meet one of the challenges targeted by our project, namely, to enable mutually independent development and evolution of *programmers* and *redirectors* [Nak 12b]. Programmers are computational components in VNodes, and redirectors are networking components in VNodes. It is necessary to establish *modularity* of these components to enable the independence. In future, virtualization platforms will probably consist of computational and networking hardware and software developed by various vendors. A VNode may contain various types of computational components, such as Linux virtual machines (VMs), Microsoft Windows VMs, network processors, and GPGPUs, and a network composed of VNodes may consist of various types of networking components, such as VLAN, WDM, and light paths. If the interface between the computational and networking components is clearly defined and works efficiently, each vendor can develop a component independently from other components and the components are modular. That means, they can be freely chosen and used in combination and can be freely enhanced or replaced by other components in accordance with emergence of new technology.

To address the above-described challenge, the redirector should play the central role in implementing the following two functions of a VNode. The first function is creation of mapping between virtual links to external physical paths and mapping between virtual links to internal physical paths, which makes it possible to hide various alternative implementations of the programmer from the external network and to hide these external-network representations from the internal components. The second function is implementation of high-performance conversion, which connects the external and internal representations or mappings, by using an add-on card with a network processor. These functions will lead to success in meeting the above challenge. The architecture of the redirector, the methods for implementation of mapping and data conversion, and results of evaluation of these implementations are described in the following.

The rest of this paper is organized as follows. Section II describes the method of virtualization, focusing on the criteria for clean network-virtualization. Section III describes the structure of redirector, and Section IV outlines the method for separating internal and external implementations by the redirector and explains its advantages and one disadvantage. Sections V and VI describe the two functions in detail. Sections VII to IX present the results of the evaluation, related work, and some concluding remarks.

II. METHOD FOR CLEAN NETWORK-VIRTUALIZATION

A. Network Virtualization

When many users and systems share a limited amount of resources on computers or networks, virtualization technology creates the illusion that each user or system owns resources of their own. Concerning networks, wide-area networks (WANs) are virtualized by using virtual private networks (VPNs). When VPNs are used, a physical network can be shared by multiple organizations, and these organizations can securely and conveniently use VPNs in the same way as virtual leased lines. Nowadays, networks in data centers are virtualized by using VLANs, while servers are virtualized by using VMs.

Many programmable virtualization-network research projects have been carried out, and many models, including PlanetLab [Tur 07], VINI [Bav 06], GENI [Due 12], and Genesis [Kou 01], have been proposed. Slices are created by network virtualization using a *virtualization platform* (substrate) that operates the slices.

In VNP, Nakao et al. [Nak 10][Nak 12b] has developed network-virtualization technology that makes it possible to build programmable virtual-network environments in which slices are isolated logically, securely, and in terms of

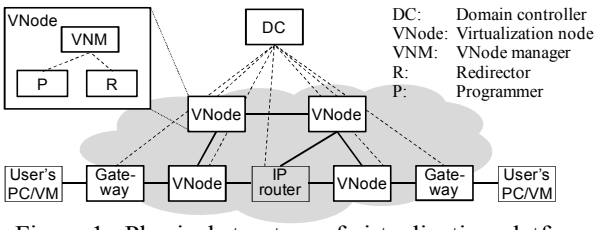

Figure 1. Physical structure of virtualization platform

performance (QoS) from one another [Kan 12]. In these environments, new-generation network protocols can be developed without disrupting the other slices.

B. Structure of Virtualization Platform

In VNP, a physical network consists of one or more domains. Each domain of the virtualization platform is managed by a domain controller (DC), and each domain has two types of nodes: VNode and gateway (**Figure 1**).

VNode (virtualization node) forwards packets on the platform. Each packet on the platform contains a virtual packet in a slice as the payload. Each VNode consists of three components: programmer, redirector, and VNode Manager. *Programmer* is a component that processes packets on slices. Slice developers can inject programs into programmers. *Redirector* is a component that can forward (redirect) packets from another VNode to a programmer or forward packets from a programmer to another VNode. *VNode Manager* (VNM) is a software component that manages the VNode according to instructions from the DC.

C. Structure of Slices

In the virtual-network model developed by VNP, a virtual network is called a *slice*, which consists of the following two components (**Figure 2**) [Nak 10][Nak 12b].

- *Node sliver* (virtual node resource) represents computational resources that exist in a VNode (in a programmer). It is used for node control or protocol processing of arbitrary-format packets. A node sliver is generated by slicing physical computational resources.
- *Link sliver* (virtual link resource) represents resources of a virtual link that connects two node slivers and that any IP and non-IP protocols can be used on. A link sliver is mapped on a physical link between two VNodes or a VNode and a Gateway. A link sliver is generated by slicing physical network resources such as bandwidth.

Both two node slivers and two link slivers are isolated and work concurrently, so two slices that consist of these slivers are also isolated and work concurrently.

The domain controller (DC) of a domain receives a slice design by an XML-based *slice definition*. The DC distributes the slice definition to each VNM, which sends necessary definitions to the programmer and the redirector: the programmer receives information required for node-sliver configuration, and the redirector receives the information required for configuring link slivers.

D. Criteria for Clean Network-Virtualization

The virtualization platform not only aims complete isolation among slices but also aims "clean" network virtualization. Slices are said to be "cleanly" virtualized if the following four criteria are satisfied.

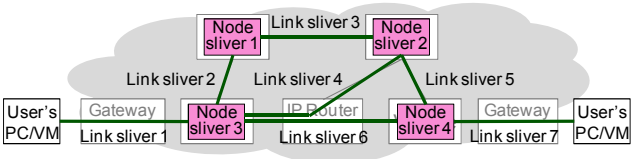

Figure 2. Example of slice design

- *Virtualization-information hiding*: Information related to virtualization including slice identifiers is hidden from slices. Such information is usually contained in the packet header in the platform. For example, if a VLAN ID is used for slice identification, slices are inhibited from accessing the VLAN ID.
- *Address independence*: The addresses (the name space) in the platform, e.g., IP and MAC addresses, can be completely hidden from the slices, and the addresses (the name space) in the slices (if they exist) can be independent from those in the platform.
- *Network-node and -structure independence and hiding*: The mapping between node slivers and physical nodes, i.e., virtual-network embedding [Cho 10], can be hidden from the slice definition, the network structure of the slices can be independently designed from those of the platform, and the network structure of the platform can be hidden from the slices.
- *Free frame-format* [Nak 12a]: Arbitrary (clean-slate) packet formats, which are restricted to neither the internet protocol (IP) nor Ethernet, can be used in slices. Although this criterion does not seem to concern virtualization, it is added because network virtualization platforms strongly require "deep(er) programmability" [Nak 12b].

These four criteria make slices portable, optimizable, and modifiable. Some of them are similar to the principles of machine or memory virtualization [Kha 12][Kan 11]. They are not completely satisfied in the case of most conventional partially-virtualized platforms, such as ProtoGENI [Li 11] [Ric 11], in which the VLAN identifiers used for virtualization can be seen in slices (from slice developers), the packet format is restricted to the IP and Ethernet, the media-access-control (MAC) addresses are shared between the platform and the slices, and physical-node information is embedded in slice definitions.

To satisfy criteria for clean network-virtualization, link slivers are implemented by using a tunneling protocol. GRE (generic routing encapsulation) [Far 00] is used for our virtualization platform. GRE enables free frame format and separation of virtual and physical structures. In addition, if IP and/or Ethernet are used in a slice, GRE/IP/Ethernet packet formats are used to separate the virtual IP and Ethernet address spaces from those of the physical ones completely (see Figure 6).

III. STRUCTURE OF REDIRECTOR

The redirector forms the skeleton of a network on a virtualization platform. It implements link slivers in the slices by physical links in the substrate network. In our IP-based prototype, the redirector implements them by using GRE tunnels. In future versions of the redirector, these slivers will be implemented by other types of physical links such as VLAN paths, MPLS paths, and light paths.

The structure of redirector is shown in **Figure 3**. The redirector consists of the following three components.

- *Redirector body* (RB) is a network node that works as a component of the platform. In the current prototype, the substrate is an IP/Ethernet network. So the RB is an L3 switch, which has both VLAN and IP-routing functions. A carrier-class high-end switch is used to build a VNode.
- *Redirector manager* (RM) is a software component in a Linux-based server. The RM has link-sliver management functions, i.e., it creates, modifies, and deletes link slivers. It also has a range of equipment-management functions for the redirector. The RM has an XML-RPC [XML 04] based control API, and the VNM manages the redirector using this API.

- *Service module card* (SMC) is an add-on card installed in the RB. The SMC is programmable because it contains a 10-Gbps-class network processor. In our VNode prototype, an SMC is used for the external-to-internal (e2i) and internal-to-external (i2e) data conversions.

The redirector is connected to the following networks. *Data plane* (D-plane) is a 10-Gbps network for sending and receiving data packets between VNodes. IP / Ethernet is used in the current implementation of the data plane. *Control plane* (C-plane) is a network (VLAN) between a VNM and a redirector in a VNode and between VNMs. The XML-RPC-based API between VNMs (between VNodes) and a redirector uses the C-plane. *Internal data plane* is a closed network (VLAN) in a VNode. The redirector and the programmers in a VNode communicate using this network.

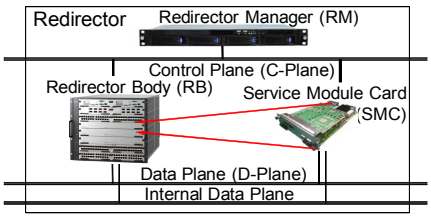

Figure 3. Structure of redirector

IV. Model-Mapping Separation

The method for separating the internal (i.e., computational) and external (i.e., networking) implementations of VNodes is described in the following. A redirector separates these components in cooperation with the programmer and the VNM in the same VNode.

A. Two Functions

VNode has two functions related to the above separation. The redirector plays the central role in implementing them. One function is to create two independent mappings.

- *External model mapping*: The RM and VNM in a VNode cooperatively create one-to-one mapping between an XML-based link-sliver specification and a configuration of an external path between two VNodes. The link sliver is a logical object in a slice, and the path between two VNodes is a physical object in the virtualization platform. GRE tunnels are used in the VNP prototype. However, any types of tunnels or virtual links (e.g., VLAN) and their combinations can be used. This mapping is hidden from the programmer in the VNode.
- *Internal model mapping*: The network interface between a redirector and programmers in the VNode are Ethernet-based or VLAN-based. The RM and a programmer (manager) in a VNode cooperatively create a one-to-one mapping between an XML-based link-sliver (i.e., a logical object) specification and the configuration of this internal network path (i.e., a physical object). This mapping is hidden from the outside of the VNode.

These mappings can be independent because of the criteria for virtualization-information hiding and address independence. The criterion for virtualization-information hiding is important because the two mappings cannot be defined freely if the virtualization information in the link representations can be observed from the slice. The criterion for address independence is important because the two mappings cannot be independent if the internal and external addresses are dependent. In addition, these mappings can be flexible because of the criterion for free frame format. If the frame format is restricted, the mapping separation is less useful. Because this study focuses on a single VNode, the criterion for network-node and -structure independence and hiding is out of the scope.

The other function is to implement the data conversion.

- *Data conversion between external and internal representations*: The external and internal packet data formats used for the link sliver are different because the two model mappings are different. The SMC must therefore convert packets between external and internal representations when they move from a programmer in the VNode to an external node, i.e., a VNode or a gateway, or from another node to a programmer inside.

B. Advantages and a Disadvantage of Mapping Separation

The separation of external and internal model mappings brings several advantages and one disadvantage. The advantages are described as follows. The first advantage is *internal implementation hiding*. The implementation of links between programmers and redirectors are hidden from outside of the VNode. This internal implementation hiding makes programmers modular. Replacement or evolution of a programmer does not affect the redirector and external networks. The second advantage is *external implementation hiding*. The implementation of external links is hidden from programmers. Replacement of an external link by another type of link does not affect programmers. That means, external implementation hiding makes redirectors and external networks modular. These redirectors or networks can be easily replaced or evolved by other types.

The disadvantage is *e2i and i2e conversion overhead*. The separation of mappings makes data conversion between external and internal representations inevitable. If the performance of this conversion is low, it may become a bottleneck. It is a challenge to reduce the overhead.

V. Model Mapping Methods

External and internal model mappings are static; they never vary unless the slice definition is updated. The RM establishes these mappings using the following three types of information. *Link-sliver definitions* included in link-sliver reservation requests sent by the VNM. *Bind information* that is also in the slice definition included in a bind request sent by the VNM to bind node slivers and link slivers. *Sliver-address information* exchanged between the programmer manager (PM) and the RM.

A. External Model Mapping

A VNode maps (RM and VNM in a VNode cooperatively map) a link sliver to a physical link between VNodes (see **Figure 4**) using the parameters negotiated between the VNMs in the VNodes; that is, it maps the port name of the link sliver to the parameters that identifies the physical link, i.e., the addresses of the end points. The VNodes may require some more parameters, such as GRE keys or VLAN identifiers, to interface between the VNodes. Although two VNodes are drawn in this figure, they work independently, except during link-parameter exchange between VNodes.

In the VNP prototype, the RM creates a GRE tunnel that corresponds to a link sliver when it receives a link-sliver definition from the VNM at the slice reservation time. A GRE key, k, and the IP addresses of the both ends of the GRE tunnel, $a1$ and $a2$, must be set. A candidate of a GRE key is generated by the RM, and the VNodes (VNMs) or a gateway of the end-points negotiate the key. Each end-point IP-address is selected by the redirector or the Gateway that contains the end point. The RM records the external mapping between the link sliver and the triple (k, $a1$, $a2$) that identifies the physical link (see Figure 4). The triple is also embedded in the SMC for data conversion.

B. Internal Model Mapping

A VNode maps a link sliver to an internal link between the redirector and the programmer (see **Figure 5**); that is, the RM maps each port name of a link sliver to the port address, and the PM maps each port name of a node sliver to the port address. They may have to specify some more parameters,

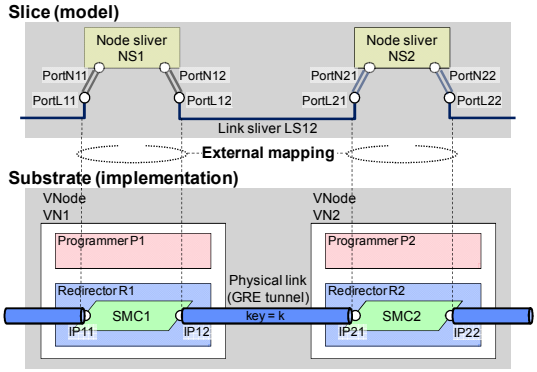

Figure 4. External model mapping

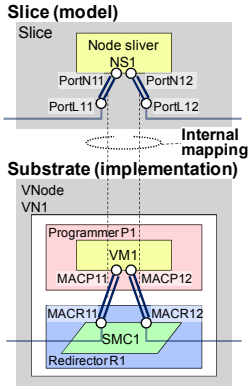

Figure 5. Internal model mapping

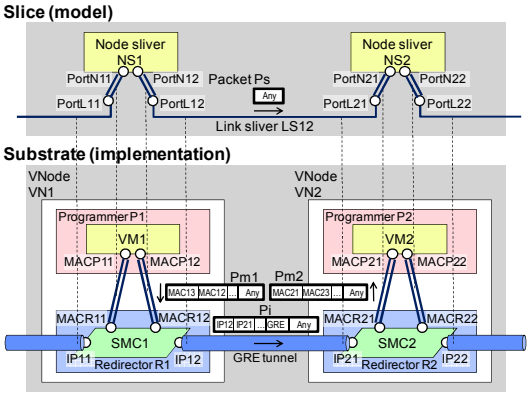

Figure 6. Logical and physical communications and their correspondence

such as VLAN identifiers, to interface between the programmer and the redirector.

The RM uses bind information that contains pairs of node-sliver ports and link-sliver ports that must be bound, and it also uses the sliver-address information that contains the MAC address of the ports. The interface between a redirector and programmers may use VLAN identifiers, but only MAC addresses are used in the current implementation. The virtual MAC addresses of the redirectors (MACR11 and MACR12) that correspond to the link-sliver ports (PortL11 and PortL12) are selected by the RM, and the virtual MAC addresses of the programmers (MACP11 and MACP12) that correspond to the node-sliver ports (PortN11 and PortN12) are selected by the PM. These addresses are exchanged between the RM and PM by using XML-RPC control interface.

VI. Method of Packet Data Conversion

The redirector (SMC) converts data packets between external and internal representations. Communication on the slice and that on the platform are shown schematically in **Figure 6**. The relationship between these communications is also shown in this figure.

A. External Data Transmission

The packets used on the data plane are IP packets, and the redirectors work as IP routers if the destination IP address of the packet is not the end point of the GRE tunnel. However, firstly, if the end point is in an SMC, the SMC converts the packet into the internal format using the network processor (NP), and redirects it to the selected programmer. Secondly, if the redirector (SMC) receives a packet from a programmer, it converts the packet into the external format by using the NP, and sends it to an appropriate VNode or other types of physical nodes by using a GRE tunnel.

B. Necessity of Data Conversion

When the external and internal representations are different, for example, when the external link is GRE-based, the reason that data conversion is necessary is clear. However, the conversion is also required even when both representations are the same; for example, when both of them are Ethernet networks. That is, conversion is required because the internal and external interfaces must be separated for the criteria for clean network-virtualization. In the Ethernet case, the MAC addresses used for the external representation must be different from those used for the internal representation. If the same MAC addresses are used, the advantages described in Section IV B, namely, internal and external implementation hiding, will disappear. Because Ethernet switches do not have such a conversion function, this function must be newly developed.

C. I2e Data Conversion

When a node sliver (e.g., node sliver 1) sends a packet (Ps) through a link sliver 12 in the slice, an Ethernet packet (Pm1) is sent from the programmer that corresponds to the node sliver to the redirector. The SMC in the redirector removes the Ethernet header, generates an IP packet (Pi) by adding an IP and a GRE header that contain the triple, (*k*, *a1*, *a2*), which corresponds to the MAC address in the original packet, and sends it. This program works on a fast path of the SMC, it can achieve 10-Gbps throughput if the packet size is large enough (close to 1500 bytes). However, the performance degrades if the packet size is too small.

D. E2i Data Conversion

A VNode (e.g., VNode 2) receives an IP packet (Pi). The destination address is that of the SMC (SMC 2). The GRE packet contains a triple (*k*, *a1*, *a2*). The SMC finds the link sliver that corresponds to this GRE key, and inserts the destination MAC address (MACP21) corresponding to the port (PortN21) of the node sliver 2 into the decoded packets. This transmission corresponds to transmission of a packet (Ps) from the link sliver to the node sliver. This conversion program can also achieve 10-Gbps throughput if the packet size is large enough.

VII. Evaluation

Performance of the internal and external mappings and that of the data conversion were evaluated in two experiments. First, the performance of redirectors, in particular the performance of these mappings, was evaluated. They are concurrently created by the RMs when creating link slivers (see the time chart in **Figure 7**). When a slice is created, GRE tunnels that correspond to link slivers are created and are connected to node sliver ports at the same time. The time required to run a slice with two node slivers, two gateways, and three link slivers, and the time required to delete a slice were measured by using the logs of the DC and the VNMs in the two VNodes (see **Table 1**). The node slivers were mapped to two VNodes.

The creation and deletion of the tree slivers by redirectors in the two VNodes, which include creation and deletion of external and internal mappings, took 23.4 s and 23.5 s, respectively. Although the creation time is shorter than other set-up time, if there are more link slivers, the overhead may be larger. However, we have already improved the redirector and have been reduced the times to 9.7 s and 9.1 s respectively. As a result, the overhead caused by link creation is mostly hidden if the number of link slivers per

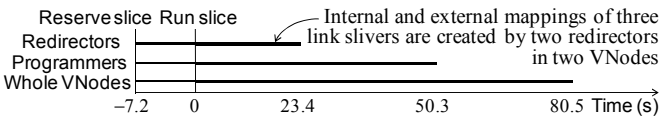

Figure 7. Time-chart to run (start) slice

Table 1. Time required to run slice and to shutdown slice including mapping creation and deletion

VNode parts	Time for "run" (s)		Time for "shutdown" (s)	
	Average	Std dev	Average	Std dev
Redirector	23.4	1.6	23.5	1.5
Programmer	50.3	3.0	0.007	0.002
Whole VNode	80.5	3.0	47.9	2.3

node sliver is five or less (i.e., in normal conditions). In contrast, the overhead can still be seen in link deletion because programmers reply to "shutdown" command immediately. Link deletion should therefore be further optimized. However, time for deletion is less critical because new links can be created while deleting old links if sufficient link resources are available.

Second, the performance of data conversion was evaluated by the following method. The data conversion is performed on the SMC when communicating through a link sliver on a slice. The VNode has one or more 10-Gbps Ethernet interfaces for connection between other VNodes. However, each VNode has only one SMC that is used for both i2e and e2i data conversion. Because an SMC uses only one 10-Gbps Ethernet interface, the sum of i2e and e2i traffic bandwidths must be less than 10 Gbps. As a result, the performance of i2e and e2i conversions for packets with size of 1 kB or more is the wire rate (i.e., 10 Gbps). In future, this performance can be improved by adding SMCs or ASICs and distributing traffic to the hardware.

VIII. Related Work

In GENI projects, a combination of PCs and an OpenFlow switch [McK 08] with GENI APIs is called a GENI Rack [GEN 10], which is similar to a VNode. However, the OpenFlow switch does not have functions for separating computational and networking functions, and the GENI Rack currently does not have link-parameter negotiation functions. In GENI, the same MAC addresses and the same VLAN identifiers are used both for the virtualization platform and the virtual networks, so the model mapping is simpler than in the VNP. In addition, IP and Ethernet are must (i.e., non-IP protocols are not freely used) in a slice in GENI because ARP is required for data communication.

Volker, et al. [Völ 09] proposed a network-virtualization node architecture that allows users to access various networks and protocols using software components called netlets. This is a software-oriented fine-grained approach, while ours is a hardware-oriented coarse-grained approach.

Mosharaf et al. [Cho 09] proposed a solution to the problem of unique naming and addressing, which conforms to the criterion for address independence.

IX. Conclusion

The objective of this study, namely, to enable mutually independent development and evolution of internal and external components on a virtualization platform, can be achieved because criteria for clean network-virtualization are applied. To achieve this objective, redirectors create two separate mappings between virtual links to external and internal physical-paths, and they implement conversion between the external and internal data formats using SMCs.

Performance of the internal and external mappings and that of the data conversion were evaluated. The overhead caused by link-sliver creation (including creation of mapping) is currently not small, but it can be reduced so that it is usually shorter than other set-up times (including that for VM creation). The data conversion can be performed at a rate of 5 Gbps on average, but SMCs can be added to improve this performance. Although these results are not the best, the structure, the functions, and the performance of the redirector mostly satisfy the requirements. In particular, it satisfies criteria for clean network-virtualization and enables mutually independent development and evolution of components.

Future work includes development of a data-conversion method that uses conventional node functions in a better way and that uses less computational resources (such as network processors). A possibility is to use VLAN-based technologies such as MAC-in-MAC (IEEE802.1ah) instead of GRE. Future work also includes reduction of time for link-sliver deletion (including deletion of mapping).

Acknowledgments

We thank Atsushi Takahara, Noriyuki Takahashi, Yohei Katayama, and Takehito Yamamoto from NTT for discussing the design of redirector with us, designing and implementing the NRI management functions in the Domain Controller and VNM, and measuring the management performance. We also thank Akihiro Motoki from NEC, Ken'ichi Abiru from Fujitsu Laboratories, Makoto Kitani from ALAXALA Networks, Yasushi Kasugai from Hitachi, and other members of VNP for discussing the design of redirector. Part of the research results described in this paper is an outcome of the Advanced Network Virtualization Platform Project A funded by National Institute of Information and Communications Technology (NICT).

References

[AKA 10] AKARI Architecture Design Project, "New Generation Network Architecture — AKARI Conceptual Design (ver 2.0)", May 2010.

[Aoy 09] Aoyama, T., "A New Generation Network: Beyond the Internet and NGN", *IEEE Comm. Mag.*, Vol. 47, Vol. 5, pp. 82–87, May 2009.

[Bav 06] Bavier, A., Feamster, N., Huang, M., Peterson, L., and Rexford, J., "In VINI Veritas: Realistic and Controlled Network Experimentation", *SIGCOMM 2006*, pp. 3–14, September 2006.

[Cho 09] Chowdhury, N. M. M. K., Zaheer, F.-E., and Boutaba, R., "iMark: An Identity Management Framework for Network Virtualization Environment", *2009 IFIP/IEEE Int'l Symposium on Integrated Network Management* (*IM 2009*), pp. 335–342, June 2009.

[Cho 10] Chowdhury, M., Samuel, F,. Boutaba, R., "PolyViNE: Policy-based Virtual Network Embedding Across Multiple Domains", *2nd ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architecture* (*VISA*), pp. 49–56, September 2010.

[Due 12] Duerig, J., Ricci, R., Stoller, L., Strum, M., Wong, G., Carpenter, C., Fei, Z., Griffioen, J., Nasir, H., Reed, J., and Wu, X., "Getting Started with GENI: A User Tutorial", *ACM SIGCOMM Computer Communication Review*, Vol. 42, No. 1., pp. 72–77, January 2012.

[Far 00] Farinacci, D., Li, T., Hanks, S., Meyer, D., and Traina, P., "Generic Routing Encapsulation (GRE)", RFC 2784, IETF, March 2000.

[Fel 07] Feldmann, A., "Internet Clean-Slate Design: What and Why?", *ACM SIGCOMM Computer Communication Review*, Vol. 37, No. 3, pp. 59–74, July 2007.

[GEN 10] "Solicitation 3 for GENI Development & Prototyping Document ID", GENI-GPO-SL-03.0, GENI Project, May 2010.

[Kan 11] Kanada, Y., and Tarui, T., "Address-Translation-Based Network Virtualization", *10th Int'l Conference on Networks*, January 2011.

[Kan 12] Kanada, Y., Shiraishi, K., and Nakao, A., "Network-resource Isolation for Virtualization Nodes", *17th IEEE Symposium on Computers and Communications* (*ISCC 2012*), July 2012.

[Kha 12] Khan, A., Zugenmaier, A., Jurca, D., and Kellerer, W., "Network Virtualization: A Hypervisor for the Internet?", *IEEE Communications Magazine*, Vol. 50, No. 1, pp. 136–143, January 2012.

[Kou 01] Kounavis, M., Campbell, A., Chou, S., Modoux, F., Vicente, J., and Zhuang, H., "The Genesis Kernel: A Programming System for Spawning Network Architectures", *IEEE J. on Selected Areas in Commun.*, vol. 19, no. 3, pp. 511–526, 2001.

[Li 11] Li, D. and Hong, X., "Practical Exploitation on System Vulnerability of ProtoGENI", *49th ACM Southwest Conference*, pp. 109–114, March 2011.

[McK 08] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J., "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, pp. 69–74, Vol. 38, No. 2, April 2008.

[Nak 10] Nakao, A., "Virtual Node Project — Virtualization Technology for Building New-Generation Networks", *NICT News*, No. 393, pp. 1–6, June 2010.

[Nak 12a] Nakao, A., et al., "Advanced Network Virtualization: Definition, Benefits, Applications, and Technical Challenges, January 2012", NVSG White Paper v.1.0, https://nvlab.nakao-lab.org/nv-study-group-white-paper.v1.0.pdf

[Nak 12b] Nakao, A., "VNode: A Deeply Programmable Network Testbed Through Network Virtualization", *3rd IEICE Technical Committee on Network Virtualization*, March 2012, http://www.ieice.org/~nv/05-nv20120302-nakao.pdf

[Ric 11] Ricci, R., "Control Framework" (slides), *Geni Experimenters Workshop*, Prinston, NJ, June 2010.

[Tur 07] Turner, J., Crowley, P., Dehart, J., Freestone, A., Heller, B., Kuhms, F., Kumar, S., Lockwood, J., Lu, J.,Wilson, M., Wiseman, C., and Zar, D., "Supercharging PlanetLab — High Performance, Multi-Application, Overlay Network Platform", *ACM SIGCOMM Computer Communication Review*, Vol. 37, No. 4, pp. 85–96, October 2007.

[Völ 09] Völker, L., Martin, D., El Khayaut, I., Werle, C., and Zitterbart, M., "A Node Architecture for 1000 Future Networks", *IEEE ICC Workshops*, pp. 1–5, June 2009.

[XML 04] XML-RPC Home Page, http://www.xmlrpc.com/.