Federating Heterogeneous Network Virtualization Platforms by Slice Exchange Point

Toshiaki Tarui and Yasusi Kanada
Hitachi, Ltd., Central Research Laboratory
Yokohama, Japan
Toshiaki.Tarui.my@hitachi.com and Yasusi.Kanada.yq@hitachi.com

Michiaki Hayashi
KDDI R&D Laboratories
Saitama, Japan
mc-hayashi@kddilabs.jp

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

***Abstract* – An architecture called the slice-exchange-point (SEP) has been designed for federating heterogeneous network-virtualization platforms by creating and managing slices (virtual networks). SEP enables whole inter-domain resources to be managed by the network manager of any single domain. Slice-operation commands are propagated to other domains through SEP by using a common API. SEP introduces the following four features: infrastructure neutrality, single interface federation, abstract and clean federation, and extensibility of capabilities. SEP's functions to achieve these features are discussed. SEP was partially implemented on two VNode domains and one ProtoGENI domain and was verified to function effectively.**

I. Introduction

Network virtualization technology enables a physical network to be sliced into multiple virtual networks (slices). To execute network applications globally, it is necessary to federate multiple parts of slices created on different domains of virtualization platforms, which have wide varieties of architecture and capabilities. So federation mechanism has to handle heterogeneous environment.

This paper describes features and implementation of the slice-exchange-point (SEP) [1], which supports federation between heterogeneous virtualization platforms. The SEP consists of a federation manager (SEP core) with a universal federation API (common API [2]), and interworking functions between domains and the SEP core.

The goal of SEP is to federate various heterogeneous virtualization platforms, enabling developers to manage whole federated slices easily using the control framework with which they are familiar, and also to use original capabilities of other virtualization platforms without constraints. To achieve above goal, this paper introduces the following four features of SEP: *infrastructure neutrality*; *single interface federation*, *abstract and clean federation* and *extensibility of capabilities*. SEP's functions to achieve them are studied.

A SEP-based federation method with the above features has been partially implemented on the VNode [3] platform and ProtoGENI [5][4] platform, and slice creation between multiple virtualization platforms was tested.

II. Related Work

To utilize resources between heterogeneous platforms, Management by Delegation [6] uses application-level federation to enable distribution processing. SEP uses platform-level federation to support wide range of applications. Previous platform-level federation was conducted between G-lambda and EnLIGHTened [7], which utilize compute and network resources each other in heterogeneous environment. The federation requires wrappers to directly convert commands between domains, which is difficult to scale. The common API enables us to support multiple domains in scalable manner.

Federation between clouds or grids [8] has been widely investigated to utilize resources between heterogeneous environments. The SEP's federation supports different capability required for network virtualization; slice configuration has to be propagated during federation.

GENI introduces the slice-based federation architecture (SFA) [13]. Each testbeds introduces the model and capabilities required for SFA, and SFA wrappers [14][15] translate testbed specific APIs and resource specifications to and from SFA-based ones. This paper proposes an alternative method where all virtualization platforms can utilize their own management framework.

Federating multiple networks is not supported in current SDN or NFV. Software Defined Internet Exchange (SDX) [16][17] is proposed to address this issue. Current SDX focuses on SDN in IXPs, and it does not yet support interoperability of heterogeneous virtualization platforms.

There have been efforts to unify or standardize control framework, control APIs, and resource definitions (information and data models) [9][10][11][12]. However, single specification may not be accepted in all domains universally. The SEP accepts multiple control frameworks with different control APIs and methods for resource definitions.

III. Outline of Slice Exchange Point

A. Purpose

SEP's control plane employs exchange model to achieve scalability required to support multiple virtualization platforms [1]. A universal federation API called the 'common API' and the 'common slice definition' (resource specification) included in it are used to exchange federation requests. If a federation mechanism uses a peer-to-peer translation model, the number of bridging interfaces required increases combinatorially with the number of virtualization platforms. With SEP, each virtualization platform has to introduce a bridging interface only to the common API.

B. Features

In order for developers to manage heterogeneous virtualization platforms, SEP has to introduce the followings:

(1) Infrastructure Neutrality

The common API and the common slice definition should not depend on a particular virtualization platform, so that virtualization platforms with different slice architectures and control frameworks can be federated together. Neutrality is crucial because each virtualization platform has its own slice model and capabilities and SEP has to enable their maximum use.

(2) Single Interface Federation

SEP has to offer single inter-domain interface, so that

domains can communicate each other through a single interface to SEP. Without this, domains have to provide different interfaces each corresponds to other domains.
Furthermore, federation has to be performed through single developer interface, which allow developers to use the familiar control-frameworks of their virtualization platforms to create and operate federated slices.

(3) Abstract and Clean Federation
SEP has to abstract virtual resources on slices from physical resources on the substrate for both intra- and inter-domain resources. Virtual resources have to be specified in the common slice definition without physical-resource information, although virtual-physical relationships could be specified if the developer wants. SEP can be used with any substrate networks and any virtual resources.
SEP also has to introduce clean federation, which enables intra- and the inter-domain network to select substrate protocols feely; without interfering with each other. In the case of inter-domain virtual link (subsection III-E), each domain can select any method to implement intra-domain parts of the virtual link; VLAN, GRE, etc., and the SEP can implement the inter-domain part independently.

(4) Extensibility of Capabilities
SEP has to handle new capabilities of virtualization platforms that it does not support directly. Even if the common API and the common slice definition are designed so as to support wide range of virtualization platforms, not all capabilities can be supported. The extensibility feature has to be transparent, i.e., any information can be tunneled through SEP. This feature is indispensable for virtualization testbeds, where new capabilities are tested frequently.

C. Architecture

SEP (Figure 1) consists of a conceptually centralized federation manager (SEP core) and interworking functions between domains and the SEP core. Each domain has a Gatekeeper (GK) for control plane to translate domain-specific federation commands and resource specifications into the common API and the common slice definition and vice versa, and a Federation Gateway (GW) for data plane to convert packet formats (protocols, network parameters, etc.) between intra- and inter-domain substrate. The GW is managed by the GK. In Figure 1, left domain is the source domain to which the developer issues slice creation request and the right domains are destination domains that receive common-API commands, however, as SEP's function is bidirectional, any domain can be a source domain.

This realizes the SEP features in the following way:

- *Infrastructure Neutrality*
  SEP's commands and resource specifications are neutral and they can be translated into those of any domain. The extensibilities feature also loosens restrictions.
- *Single Interface Federation*
  GK and GW provide conceptually single interface to destination domains, regardless of physical inter-domain configuration. Bidirectional function enables slice creation and operation commands to be submitted to any domains.
- *Abstract and Clean Federation*
  The common slice definition is abstract and can be specified independently of the intra- and inter-domain substrate's implementation, and each domain's control framework can select resource abstraction method freely. Packet conversions at GWs enable network protocol and parameter on the SEP's inter-domain network to be selected independently of each domain's internal network.

D. Common API and Common Slice Definition

The common API [2] commands are classified into three categories: resource enquiry ('ResourceInfo'), slice handling ('CreateSlice', 'Add', 'Delete', 'Run', 'Stop', and 'DeleteSlice'), and obtaining information ('Status' and 'Statistics'). A common-API command is transferred by a request-and-reply message pair of XML-RPC.

The common slice definition consists of the followings:

- Each-domain's part of the slice (virtual nodes, links)
- Inter-domain-network part of inter-domain virtual links

E. Data Plane

An inter-domain virtual link consists of three parts: two intra-domain parts and an inter-domain part. These are connected by GWs. Although an intra-domain virtual link is logically one link, intra-domain parts and the inter-domain part are physically isolated by GWs.

IV. Slice Exchange Point Functions

A. Translation of Command and Resource Specifications

To achieve this translation, SEP provides the following functions, which contribute to infrastructure neutrality.

a) Command-granularity translation
Slice handling commands on each virtualization platform are not correspond one-to-one; a single command in a source domain could cause multiple common-API commands, or multiple commands could be merged into a single common-API command. GKs adjust command granularity. E.g., a slice creation command ('CreateSliver') on GENI's AM API [18] Version 2 is translated into two consecutive common-API commands: 'CreateSlice' and 'Run', while two consecutive AM API Version 3 commands ('Allocate' and 'Provision') are merged into a single common-API command: 'CreateSlice'.

b) Node-hierarchy conversion
Virtualization platforms have different hierarchical node architecture, e.g., virtual node VMs in VNode domain are defined inside physical enclosure; while, ProtoGENI's virtual nodes are defined independently. SEP abstracts node hierarchy differences. In the common slice definition virtual node can be specified with any hierarchical level. Thus, the source GK can translate resource specifications of source domains directly (without hierarchy conversion) into the common slice definition. Destination GKs are responsible for converting the hierarchical differences.

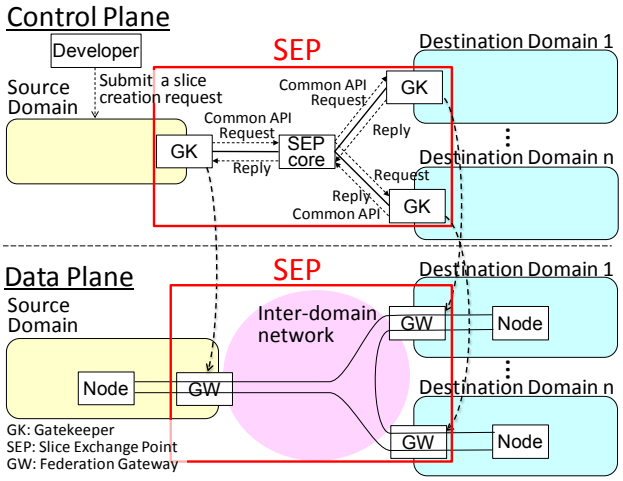

Figure 1 SEP (Slice-Exchange-Point) Architecture

B. Inter-Domain Virtual-Link Management

SEP can determine inter-domain substrate-network parameters (link type, MAC addresses, VLAN number, etc.) freely, without interfering with the domain's internal network, which help achieving abstract and clean federation. SEP introduces the following methods to determine network parameters for inter-domain virtual link:

a) Inter-GK negotiation.
   GKs manage inter-domain network resources, and GKs on both side of the inter-domain link negotiate parameters through common-API requests and replies.

b) Assigned by the SEP core
   The SEP core or a broker manages inter-domain-network resources, and assigns them into the common-API request. Virtual links between destination domains has to be defined with this method, because the GK of source domain does not have information (parameters, etc.) of inter-domain network between destination domains.

SEP's inter-domain link management helps achieving single inter-domain interface; developer can create inter-domain links with single interface of the GW, regardless of destination domain, and SEP can create and optimize them freely.

C. Transparent Tunneling using Syntactic Extension

'Extension' parts are introduced to the common slice definition to enable extensibility of capabilities. Syntactic extension fields can contain any new information. GKs and the SEP core do not check contents and forwards information to destination domains. Extension fields do not negatively influence SEP execution; if destination domains cannot handle extensions, they ignore extensions safely.

V. Implementation

A. Overview of the Experimental System

The proposed federation architecture is implemented in a real virtual-network environment spread across multiple continents (Figure 2). Three domains are connected by SEP: one ProtoGENI domain (located at The University of Utah, USA) and two VNode domains (located in Tokyo, Japan).

A federation-less federation method [19][20] is used by the source domain to describe and submit resource specifications of destination domains.

B. Control Planes

Control-plane is implemented by three GKs and the SEP core, represented by VMs. The SEP core delivers a common-API request to destination domains, and also combines common-API replies (execution results, etc) returned from destination domains into a single common-API reply, which realizes single inter-domain interface. Although the SEP specification enables common-API requests between the SEP core and destination GKs to be executed concurrently, the current implementation executes them sequentially.

C. Data Planes

Each VNode domain has GW; L2 switch with a network processor card to accommodate inter-domain virtual-link VLANs to VNode virtual links (GREs) [21]. GW is controlled from GK by CLI.

In current implementation, ProtoGENI side GW is omitted, and GWs on VNode domains are connected to the ProtoGENI's internal network directly. Because VNode-side GWs have VLAN-translation functionality and can accommodate ProtoGENI's virtual-link VLANs directly, no other VLAN-translation mechanism (GW) is required on the ProtoGENI side. VLAN numbers used for ProtoGENI virtual links are extracted from the ProtoGENI slice-creation result (manifest), and then transmitted through the common-API command and set to the VNode-side GW.

D. Inter-Domain Virtual-Link Management

Inter-VNode virtual links created in ProtoGENI-to-VNode federation using method b) of subsection IV-B is discussed. Inter-VNode virtual links support MAC encapsulated non-IP (any-frame) packets [19]. The following method allows SEP to create non-IP virtual links when a slice-creation is requested from the ProtoGENI domain, which has no information about inter-VNode non-IP network.

The SEP core maintains a pool of pre-allocated VLANs and MAC addresses to be assigned to VNode GWs. When a common-API slice-creation request is first issued by the ProtoGENI-side GK, no network parameters are assigned for inter-VNode virtual links. The SEP core assigns VLAN numbers and MAC addresses into the common-API request.

VI. Experimental Results

Evaluation results are discussed focusing on SEP's features and efficiency. *Extensibility of capabilities* has been implemented but not yet tested, because federation between VNode and ProtoGENI does not require it.

A. Single Interface Federation

Single developer interface was verified by submitting slice from both domains with their slice-creation API.

For ProtoGENI-to-VNode federation, three-domain-slice creation, between a ProtoGENI domain and two VNode domains, was achieved. Three domains, each has three to five virtual nodes, were connected by inter-domain virtual links each other. VNode GW accommodated ProtoGENI's virtual-link VLANs, as well as the inter-VNode virtual link.

Single inter-domain interface for control plane was verified; domains were able to send and receive federation commands through their GK. However, in current implementation, single inter-domain interface for data plane was achieved only on VNode domains. In VNode domains, all inter-domain virtual links can be connected to the GWs regardless of destination domain. On the other hand, in the ProtoGENI domain, developer has to specify a device (VNode GW) corresponds to virtual link's destination domain, because ProtoGENI-side GW is omitted.

VNode-to-ProtoGENI slice creation was also conducted between a VNode domain and a ProtoGENI domain (based on common API V1.0 [22]). Three virtual nodes per each

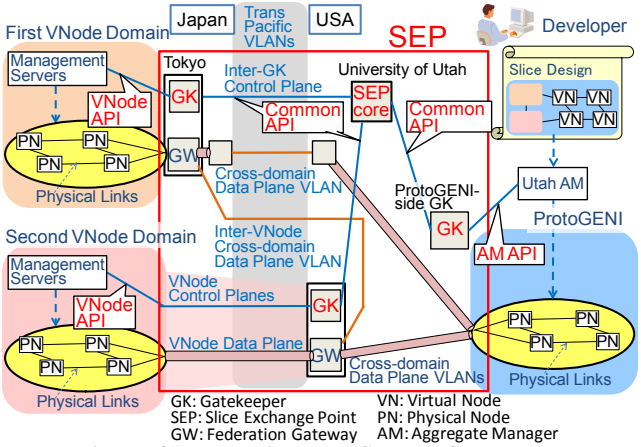

Figure 2 Experimental System Structure

domain and one inter-domain virtual link were created.

B. Infrastructure Neutrality

Infrastructure neutrality was verified by creating slice between heterogeneous domains; ProtoGENI and VNode.

Node hierarchy difference was converted, e.g., during ProtoGENI-to-VNode federation, VNode-side GKs converted ProtoGENI's one-level node specifications into VNode's two-level node specifications, by adding a hierarchy to virtual nodes. Command granularity was also converted. The ProtoGENI side GK translated a 'CreateSliver' command in ProtoGENI domain into two common-API commands: 'CreateSlice' and 'Run', and vise versa.

C. Clean and Abstract Federation

Even though VNode and ProtoGENI have different substrate implementation, *clean and abstract federation* allows SEP to creating virtual links between them.

For virtual links between ProtoGENI and VNode, VNode-side GWs converted ProtoGENI's VLANs into VNode's GREs (subsection V-C). For inter-VNode virtual link, SEP core maintained link parameters set to GWs, and non-IP virtual link was created even though source domain (ProtoGENI) did not provide them (subsection V-D).

Virtual nodes and links on VNode domains are abstracted, on the other hand, those on ProtoGENI are not fully abstracted, e.g., virtual links depend on substrate VLANs. Abstract federation enabled developer to create slice between domains with different resource abstraction method.

D. SEP's Efficiency

SEP's efficiency was investigated using common-API execution sequence extracted from SEP's log in the case of three-domain ProtoGENI-to-VNode federation. SEP took an average of 8 minutes 3 seconds (seven results with 27-second standard deviation) to create and start remote domain part of the slice, i.e., from the source GK receives slice creation request until slice startup complete. If VM startup time (7 minutes 8 seconds) which is inevitable is excluded (it could be reduced by parallel VM startup, as SEP core submits a common-API command to each domain sequentially in current implementation), it took about 55 seconds to create the slice. SEP is able to provisioning resources globally in reasonable time (around one minute).

VII. Conclusion

Slice-exchange-point (SEP) architecture was described that supports federation between heterogeneous virtualization platforms. SEP's features and functions to achieve them were discussed in detail.

Command and resource-specification translation functions achieve *infrastructure neutrality* and *single developer interface*; enabling developers to manage whole federated slices with their familiar control framework. Resource-specification translation and data-plane conversion, along with inter-domain virtual-link management, achieve *abstract and clean federation*; enabling virtualization platforms to create virtual resources without constraints. Inter-domain virtual-link management function also realizes *single inter-domain interface*; inter-domain virtual links to different domains can be created through the SEP interface..

SEP was implemented between three domains of real virtualization platforms: two VNode domains and a ProtoGENI domain. Federation between multiple virtualization platforms was tested, and the proposed SEP architecture was confirmed to achieve multi-way federation between multiple heterogeneous domains.

Acknowledgments

We thank Nozomu Nishinaga from the National Institute of Information and Communications Technology (NICT) and Robert Ricci and Gary Wong from The University of Utah for setting up the federation node. Part of the research results is an outcome of the Advanced Network Virtualization Platform Project B funded by NICT and experiments using JGN-X testbed deployed by NICT.

References

[1] Okamoto, S., et al., "Design of Network Slice Exchange for Bridging Future Internet Testbeds", 18th OptoElectronics and Communications Conference (OECC/PS 2013), June-July 2013

[2] Federation Architecture and Common API / Common Slice Definition, http://nvlab.nakao-lab.org/Common_API_V2.0.pdf

[3] 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

[4] Berman, M., et al., "GENI: A federated testbed for innovative network experiments", Computer Networks, Special Issue on Future Internet Testbeds, 2014.

[5] Ricci, R., et al., "Designing a Federated Testbed as a Distributed System", TridentCom 2012, June 2012.

[6] Goldszmidt, G., Yemini, Y., "Distributed management by delegation", 15th International Conference on Distributed Computing Systems, May-June 1995

[7] Thorpe, S. R., et al., "G-lambda and EnLIGHTened: wrapped in middleware co-allocating compute and network resources across Japan and the US", 1st International Conference on Networks for Grid Applications, October 2007.

[8] Rochwerger, B., et al., "The reservoir model and architecture for open federated cloud computing", IBM Journal of Research and Development 53-4, 2009.

[9] GEC 14 Slice Around the World Demo, http://groups.geni.net/geni/wiki/SAW

[10] OGF NSI, "Network Services Framework v1.0", GFD.173

[11] "Common YANG Data Types", RFC6021

[12] OpenStack API, http://api.openstack.org/

[13] Peterson, L., Ricci, R., Falk, A., and Chase, J., "Slice-based Federation Architecture, Version 2.0", http://git.planet-lab.org/?p=sfa.git;a=blob;f=docs/sfa.pdf, July 2010.

[14] Wahle, S., et al., "Conceptual Design and Use Cases for a FIRE Resource Federation Framework", in Tselentis, G., et al. ed., "Towards the Future Internet", IOS Press, 2010.

[15] Augé, J., et al., "Tools to Foster a Global Federation of Testbeds", Computer Networks, Special Issue on Future Internet Testbeds, 2014.

[16] Proto-SDX Demo and Background, http://groups.geni.net/geni/attachment/wiki/GEC19Agenda/Plenary/2014-03-18%20Proto-SDX%20Demo%20and%20Background.pdf

[17] Gupta, A., et al., "SDX: A Software Defined Internet Exchange", SIGCOMM 2014, August 2014.

[18] GENI Aggregate Manager API, http://groups.geni.net/geni/wiki/GAPI_AM_API

[19] Kanada, Y., et al., "Federation-less-federation of Network-virtualization Platforms", IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), May 2013.

[20] Kanada, Y., and Tarui, T., "Federation-less-federation of ProtoGENI and VNode", 29th International Conference on Information Networking, January 2015.

[21] Kanada, Y., et al., "High-performance Network Accommodation into Slices and In-slice Switching Using A Type of Virtualization Node", 2nd International Conference on Advanced Communications and Computation, October 2012.

[22] Federation Architecture and Sequence,http://groups.geni.net/geni/attachment/wiki/GEC18Agenda/FedToolSupport/Federation_Architecture_and_Sequence20130829r1.pdf