Slice Extension of Network Infrastructure into Virtualization-enabled Terminal Device

Coordinating Network Management System with Terminal Device Management System in Network Virtualization

Yohei Katayama
NTT Network Innovation Laboratories

Takehito Yamamoto
the NTT Bizlink, Inc.

Kazuhisa Yamada
NTT Network Innovation Laboratories

Akihiro Nakao
the University of Tokyo

ABSTRACT

This paper proposes a new management scheme that extends the network infrastructure slices into end terminal devices. In our proposal, end terminal device management is independent of network infrastructure management. In addition, we propose a procedure for coordinating the terminal management system with the network infrastructure management system.

Keywords

Network Virtualization, End Terminal Device (End System), Management System

1. INTRODUCTION

Network services provide applications to the network user and many new applications will appear to satisfy new user demands. These new applications will require more advanced functionality from the network infrastructure. To satisfy these requirements, the network infrastructure must become more flexible and support the swift creation and modification of various network functionalities. *Network virtualization* is a key technology for realizing such flexibility due to its *deep programmability*[1]. It enables any functionality to be realized in the network infrastructure as a *slice*, i.e. a set of virtual network resources. However, some functionalities such as naming, addressing, and routing must be implemented into not only the network infrastructure but also end terminal devices such as a desktop computer, which is outside the traditional infrastructure.

Such functionalities can be implemented by preparing a slice that involves the cooperation of end terminal device(s) and the network infrastructure. This approach, *slice extension*, extends slice reach into end terminal devices. Slice extension offers several advantages:

- A new service or application can be realized by preparing the appropriate slice, which includes the resources of end terminal devices as necessary.
- The same program and/or protocol operates over the whole slice including end terminal devices.

The authors propose the basic design and coordination procedure for slice extension. Our proposals enable the developer of a slice to develop and operate functionalities and protocols over the slice in an end-to-end manner.

2. SLICE EXTENSION

2.1 Requirements and architecture

The authors assume an end terminal device that can, via virtualization technologies such as KVM[2] and encapsulation technologies such as GRE[3], provide a virtual node (VN) and a virtual link (VL). An end terminal device that has this capability is called the virtualization-enabled terminal device (VT) in this paper.

A consideration of slice extension shows that VT resource management must be independent from network infrastructure resource management because these two entities have, in general, different owners, and because network infrastructure responsibility does not extend to VTs in practice. On the other hand, the configuration of a VT-supported slice and the programs installed onto the slice must be set by the developer of the slice so that the user of a VT can receive services or applications on the slice with appropriate setups. Therefore, our resource management scheme provides each VT with its own management system (namely terminal device management system, or TMS) that operates independently from the management system of the network infrastructure (namely network management system, or NMS); note that TMS cooperates with NMS. NMS instantiates and manages the original part of the slice that contains only network elements (NEs) (namely NE-part slice), while TMS realizes the extended part of the slice that contains its VT (namely VT-part slice). Fig.1 presents this slice extension architecture.

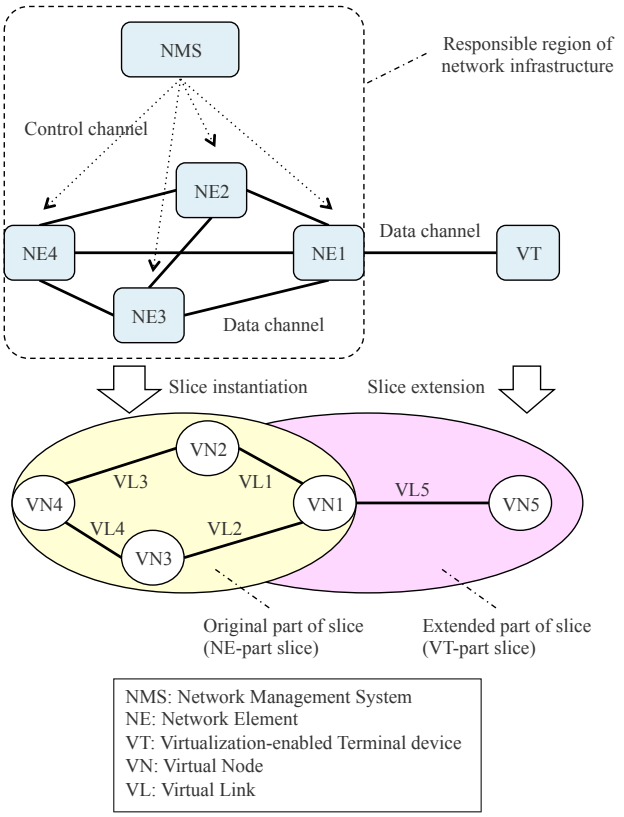

Figure 1: Architecture of slice extension

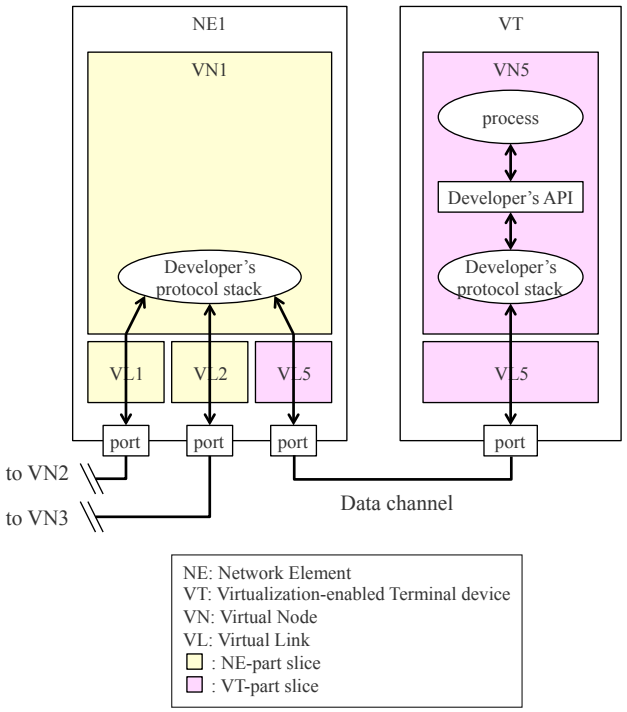

Figure 2: Installed programs in slice extension (following Fig.1)

Fig.2 shows a brief example of installed programs ("Developer's API" and "Developer's protocol stack") in slice extension following Fig.1. The programs in the VT-part slice are programmed by a slice developer so that the programs in the NE-part slice can cooperate with them.

An example of developer's protocol stack is an end-to-end label switched path. In traditional approaches, a label switched path is terminated at the edge of the providers network. In the slice extension approach, label switched paths can be extended up to an end terminal device and may be controlled with a process in a VT-part slice through the developer's API.

A process in a VT-part slice can also be programmed and delivered by a slice developer. Such processes can cooperate with those of another VT, therefore it is easy to implement distributed computation using the resources of multiple VTs.

2.2 Basic design and coordination procedure

Fig.3 presents our proposed basic design and coordination procedure for slice extension. The figure consists of function blocks and message sequence arrows. Table 1 summarizes the message sequence in Fig.3. TMS controls and manages the hypervisor (HV), such as KVM[2], in its VT, while NMS controls and manages the HVs in the NEs, such as VNode[4], of the network infrastructure. TMS manages the resource block that can be assigned to a slice. TMS instantiates the VT-part slice in cooperation with NMS using the interface defined between NMS and TMS (Seq:2-2, 2-4 in the figure). TMS follows the specification of the VT-part of the slice which is defined by the slice developer (e.g. XML format text in [4]). The specification includes a locator for the installed program that is programmed and stored in the network infrastructure by the slice developer.

Fig.4 is an example message format of VT-part slice specification. At first, we describe the elements in the format. "Slice-design" element determines its scope. "Slice-spec" element has slice specification. "Sliverdef" element describes the specification of VN ("nodeSliver" in the example) and VL ("linkSliver" in the example). "Structure" element describes topological information, i.e. bindings of virtual ports ("vport" in the example). In this example, the "nodeSliver" element whose "name" attribute is "VN1" has the value "_super" as "type" attribute. The value means that the component specification is defined in the NE-part slice. The example has a "bootImage" parameter so that the developer of the example can designate from where VT gets the OS image which the developer designed. "Resources" element describes the amount of resources to be assigned to a VT-part slice. In the example, a cre-

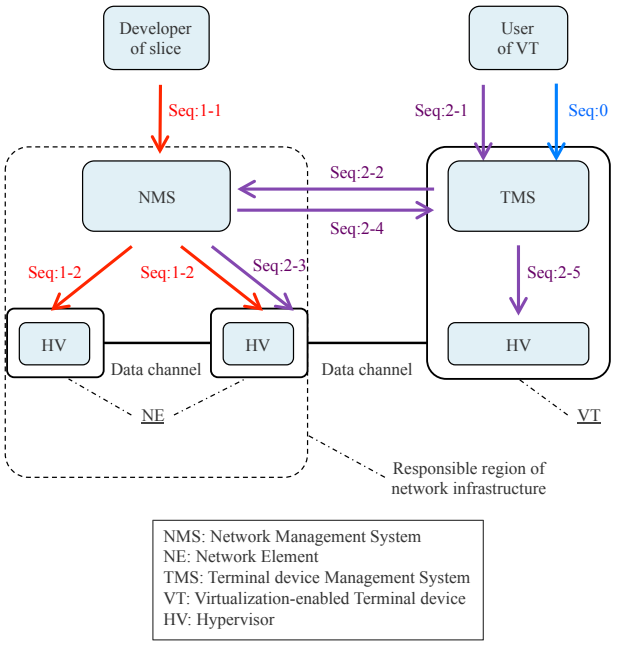

Figure 3: Basic design and coordination procedure

Table 1: Summary of message sequence in Fig.3

Phase	Seq.#	Description
Preparation	Seq: 0	User of VT sets up resource block that can be assigned to a VT-part slice.
Instantiation of NE-part slice.	Seq:1-1	Developer of the network infrastructure defines the specification of a NE-part slice and that of a VT-part slice, and puts them into NMS.
	Seq:1-2	NMS issues command to HV(s) to instantiate a NE-part slice.
Instantiation of VT-part slice.	Seq:2-1	User of VT sends request to TMS triggering TMS cooperation with NMS.
	Seq:2-2	TMS sends to NMS a request for obtaining the specification of a VT-part slice.
	Seq:2-3	NMS issues a command to a HV to modify the NE-part slice at the edge of the network infrastructure.
	Seq:2-4	NMS submits to TMS a response that includes the specification of the VT-part slice.
	Seq:2-5	TMS issues a command to a HV to instantiate the VT-part slice.

ated VN ("VN5") has 1 CPU core, x86_64 instruction set architecture, 2G-size memory, boot OS downloaded from a bootImage server and connectivity to the VN in the NE-part slice "VN1" through a VL ("VL5"). A created VL ("VL5") has 1G bandwidth, 1024 byte buffer for burst traffic and shaping function. The VL is implemented by GRE, IPSec, IPv4 and Ethernet.

3. RELATED WORK

To the best of our knowledge, no existing work has proposed a slice extension mechanism for implementing any functionalities into end terminal devices.

Work by Wilson et. al [5] introduced a mechanism to install protocol stacks into an end terminal device by using a kernel module and a user space daemon. The mechanism takes the approach of not instantiating any virtual node. Instead, it specifies the functionalities implemented into the protocol stack of the end terminal device.

Other research[6] proposed a VT control mechanism for network virtualization. In the context of the research, network virtualization means virtualizing several physical routers into a single logical router and the goal of the study is a logical router that can track VM migration across multiple networks in an efficient manner by using a centralized flow controller, like Openflow, in the network infrastructure.

Another work[7] showed a design to federate multiple networks between ends in order to provide a slice across networks. It relies on centralized coordinators, namely brokers, and the paper made no mention of end terminals.

End-to-end connectivity was proposed for multiple federated networks including end terminals, but a functionality implementation scheme for end terminals was not[8].

4. CONCLUSIONS

We proposed a new management scheme the allows slices in a network infrastructure to be extended into virtualization-enabled terminal devices. We introduced a coordination mechanism for slice extension. Our proposals also offer slice extension into application servers in a data center, mobile phones in wireless system and other types of terminal devices and systems.

In future work, we will implement the proposals and evaluate their effectiveness in a comparison against existing procedures.

5. ACKNOWLEDGMENTS

A part of this research has been executed under the Commissioned Research of National Institute of Information and Communications Technology (NICT).

6. REFERENCES

[1] Akihiro Nakao. Network Virtualization as Foundation for Enabling New Network Architectures and Applications. *IEICE Trans. Commun.*, E93.B(3):454–457, 2010.

```
<?xml version="1.0" encoding="UTF-8"?> <!-- delivered from NMS to TMS -->
<slice-design>
  <slicespec class="VT-part_slice">
    <sliverdef>
      <linkSlivers>
        <linkSliver name="VL5">
          <vports><vport name="e1"/><vport name="e2"/></vports>
          <resources>
            <resource key="bandwidth" value="1G"/>
            <resource key="burstSize" value="1024"/>
            <resource key="performanceIsolation" value="shaping"/>
            <resource key="protocol" value="802.3,IPv4,GRE,IPsec"/>
          </resources>
        </linkSliver>
      </linkSlivers>
      <nodeSlivers>
        <nodeSliver name="VN5">
          <vports><vport name="vp1"/></vports>
          <instance type="SlowPath_VM">
            <resources>
              <resource key="cpumode" value="dedicated"/>
              <resource key="cpu" value="1"/>
              <resource key="arch" value="x86_64"/>
              <resource key="memory" value="2048"/>
            </resources>
            <params>
              <param key="bootImage" value="http://{bootImageServer}/bootImage.img"/>
            </params>
          </instance>
        </nodeSliver>
        <nodeSliver name="VN1" type="_super">
          <vports><vport name="vp1"/></vports>
        </nodeSliver>
      </nodeSlivers>
    </sliverdef>
    <structure>
      <bind>
        <vport slivername="VN5" portname="vp1"/>
        <vport slivername="VL5" portname="e1"/>
      </bind>
      <bind>
        <vport slivername="VL5" portname="e2"/>
        <vport slivername="VN1" portname="vp1"/>
      </bind>
    </structure>
  </slicespec>
</slice-design>
```

Figure 4: A specification example

[2] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm : the Linux Virtual Machine Monitor. *Proceedings of the Linux Symposium, Ottawa, Ontario, 2007*, pages 225–230, July 2007.

[3] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784 (Proposed Standard), March 2000. Updated by RFC 2890.

[4] Akihiro Nakao. Corelab and VNode. 8th GENI Engineering Conference (GEC8), July 2010.

[5] Michael Wilson, Fred Kuhns, and Jonathan Turner. Network Access in a Diversified Internet. In Ian Akyildiz, Raghupathy Sivakumar, Eylem Ekici, Jaudelice Oliveira, and Janise McNair, editors, *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4479 of *Lecture Notes in Computer Science*, pages 1204–1207. Springer Berlin / Heidelberg, 2007.

[6] Fang Hao, T. V. Lakshman, Sarit Mukherjee, and Haoyu Song. Enhancing dynamic cloud-based services using network virtualization. *SIGCOMM Comput. Commun. Rev.*, 40(1):67–74, January 2010.

[7] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, January 2007.

[8] N. M. Mosharaf Kabir Chowdhury, Fida-E Zaheer, and Raouf Boutaba. iMark: an identity management framework for network virtualization environment. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, IM'09, pages 335–342, Piscataway, NJ, USA, 2009. IEEE Press.