Summary: A method for providing functions of VNode infrastructure switches, such as switching or routing, to slices is proposed. The plug-in interfaces and the interfaces for providing layer-3/VLAN switch functions to slices were designed, implemented, and evaluated.

1. Introduction

Background: A component of VNode (i.e., redirector) contains an L3 switch, but slices could not use its functions, such as switching or routing.

Proposal: A method for supporting L3 switch functions by extending the VNode plug-in architecture [1, 2] is proposed [3].

2. Outline of plug-in architecture

- New types of virtual nodes and links can be added to VNode.
- New types are implemented by a combination of two types of plug-ins:
  - Data plug-ins extend data-plane functions such as packet forwarding.
  - Control plug-ins extend control-plane functions: manages data plug-ins.
- New types can be specified in a slice definition (RSpecs).
  - All the implementation parameters can be specified by the developer, or
  - The implementation parameters can be hidden from the developer (can be supplied by control/management components of the VNode).
3. Application of plug-in architecture
The interfaces and the control-plug-in are extended or newly designed.

- **Data plug-in**: the L3 switch — the same switch as the data-plane component of the redirector, but they must be isolated.
- **Control plug-in**: The control software must be newly developed.
- **Data-plane interface** (DPII) is extended: Original DPII is MAC-address-based, but new DPII is VLAN-based — L3-switch requirements.

(1) VLAN-ID-based DPII

<table>
<thead>
<tr>
<th>SDMAC</th>
<th>SSMAC</th>
<th>SEType</th>
<th>VLAN ID etc.</th>
<th>Payload</th>
</tr>
</thead>
</table>

(b) MAC-address-based DPII

<table>
<thead>
<tr>
<th>PDMAC</th>
<th>PSMAC</th>
<th>PEType</th>
<th>SDMAC</th>
<th>SSMAC</th>
<th>SEType</th>
<th>Payload</th>
</tr>
</thead>
</table>

- Inter-plug-in interface: CLI of the L3 switch may be used.

4. L3 switch functions to be provided to slices
L3 switch functions are provided by new node types.

- **Routing function** (VRF function) is supported by “virtual_router” type.
- **Switching function** (of Ethernet) is supported by “virtual_switch” type.

Implementation parameters can be specified, or can be hidden from the developer.

Example: Slice definitions
(a) w/o implementation parameters

```
<nodeSliver name="vrf1" ...>
  <instance type="virtual_router">
    <params>
      <param key="Vf/" value="Vf" /> ...
      <param key="Pn/" value="Vn" />
    </params>
  </instance>
  <vports>
    <vport name="p1" ...> vport name="pm" /
  </vports>
</nodeSliver>
```

(b) with implementation parameters

```
<nodeSliver name="vrf1" ...>
  <instance type="virtual_router">
    <params>
      <!-- Plug-in name -->
      <param key="PlugInName" value="intSw" />
      <!-- Specification of data plug-in interface (DPII) -->
      <param key="DataPort" value="vlan" />
      <!-- Specification of control plug-in interface (CPII) -->
      <param key="ControlPort" value="192.168.110.61" />
    </params>
  </instance>
  <vports>
    <vport name="p1" ...> vport name="pm" /
  </vports>
</nodeSliver>
```

5. Prototyping of routing function

- The proposed method was partially implemented in NACE (NC).
  - NACE is a type of VNode, which is used for federations between VNode and ProtoGENI.
  - The control plug-in, which communicates with the L3 switch by CLI, was implemented as a program written in Perl.
  - OSPF-based IP routing and Ethernet switching functions were implemented.

Acknowledgments
Part of the research results is an outcome of the Advanced Network Virtualization Platform Project A funded by NICT. The authors thank Akihiro Nakao from the University of Tokyo, Satoshi Kamiya from NEC, and other members of the VNode Project for their discussions on virtual-switch interfaces.

References