Providing Infrastructure Functions for Virtual Networks by Applying Node Plug-in Architecture

Yasusi Kanada
Hitachi, Ltd., Japan
Background

► VNode and VNode Infrastructure was developed in a collaborative project.

► VNode is a physical node with network-virtualization functions.

► VNode Infrastructure is a network architecture and testbed with network-virtualization function.
  ■ On this infrastructure, multiple developers can create and use slices (i.e., virtual networks) concurrently and independently.
Background (cont’d)

► VMs (virtual nodes) in VNodes can implement routing and switching, but the performance is limited.
► VNode contains a layer-3 (L3) switch, which has high-performance routing and switching functions.
► Slices can not use its functions, such as switching or routing.
Proposal

► A method for supporting L3 switch functions to slices is proposed.
► This method is based on the node plug-in architecture, which was proposed in previous papers.
A developer can create a slice by sending a slice definition (RSpecs) to the management server (i.e., DC).

A slice definition contains definitions of virtual nodes and links of *predefined types*.

**Slice definition**

```
<nodeSliver name="..." >
  <instance type="SlowPath_VM">...
  </instance>
</nodeSliver>
...
<linkSliver name="..." type="link" >
  ...
</linkSliver>
...
```

**Predefined virtual-node type**

**Predefined virtual-link type**

**DC**: Domain Controller

**Virtualization platform**

(IP network)
Plug-in Architecture for VNode (Previous Work)

▶ Plug-ins enable new types of virtual nodes/links to be added to VNode.

▶ New types can be specified in a slice definition.
  ■ 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).

```xml
<nodeSliver name="vrf1" ...
  <instance type="virtual_router">
    <params>
      ...<param key="DataPort" value="vlan" />
      ...<param key="ControlPort" value="192.168.110.61" />
      ...<param key="Command-runNodeSliver" value="run_vrf" />
      ...<param key="Command-stopNodeSliver" value="stop_vrf" />
    </params>
    </instance>
  </nodeSliver>
```
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.
Proposal: Plug-ins and Interfaces for L3 Switch Functions

► Data plug-in: the L3 switch
  ■ The same switch as the data-plane component of the VNode,
  ■ The data plug-in must be isolated from the VNode.

► Data plug-in interface (DPII) is extended:

- Original DPII is MAC-address-based
  - | PDMAC | PSMAC | PEType | SDMAC | SSMAC | SEType | Payload |
  - MAC address M1
  - MAC address M2
  - Virtual node

- New DPII is VLAN-based — L3-switch requirements
  - | SDMAC | SSMAC | SEType | VLAN tag | Payload |
  - VLAN V1
  - VLAN V2
  - VLAN V3
  - Virtual node
Control plug-in must be developed.

- It assigns VLAN IDs for the DPII.

Control plug-in interface (CPII)

- CLI is used for CPII.
- Command names for this CLI must be specified.
  - E.g., when they are specified in slice definitions:
    
    ```xml
    <param key="Command-runNodeSliver" value="run_vrf" />
    <param key="Command-stopNodeSliver" value="stop_vrf" />
    ```

- Parameters for the commands must be specified.
L3 Switch Functions to be Provided to Slices

► Switching function (of Ethernet)
  ■ Number of ports is arbitrary.
  ■ No plug-in parameters are required.

► Routing function (VRF function)
  ■ Number of ports is arbitrary.
  ■ Routing parameters are specified as parameters in the slice definition.

```xml
<param key="routing_protocol" value="ospf" />  
<param key="ospf_subnet" value="192.168.0.0" />  
<param key="ospf_mask" value="0.0.15.255" />   
<param key="ospf_area" value="110" />          
<param key="ospf_domain" value="1" />          
<param key="router_ip" value="192.168.101.1" />
```
Prototyping and Evaluation

The plug-in interfaces were partially implemented to a type of VNode called NACE (NC).

The control plug-in was implemented in Perl.
  ■ It communicates with the L3 switch through CLI,

OSPF-based IP routing and Ethernet switching plug-ins were implemented.
  ■ Routing and switching among three terminals were tested.
  ■ Rerouting between two virtual routers were tested.
Prototyping and Evaluation (cont’d)

► **Virtual-node development cost** (when plug-in parameters are embedded to control components)
  - **Ethernet switching**: only 8 lines are required for specifying a virtual switch.
    - 16 lines with plug-in parameters.
  - **OSPF routing**: only 19 lines (including OSPF parameters) are required for specifying a virtual router
    - 25 lines with plug-in parameters.

```xml
<nodeSliver name="sw1" ...>
  <instance type="virtual_switch" />
  <vports>
    <vport name="p1"/>
    ...
    <vport name="pm"/>
  </vports>
</nodeSliver>

<nodeSliver name="vrf1" ...>
  <instance type="virtual_router">
    <params>
      <param key="P1" value="V1" />
      ...
      <param key="Pn" value="Vn" />
    </params>
  </instance>
  <vports>
    <vport name="p1"/>
    ...
    <vport name="pm"/>
  </vports>
</nodeSliver>
```
Summary and Conclusion

► Summary
  ■ A method for supporting L3 switch functions, which is based on the node plug-in architecture, is proposed.
  ■ OSPF routing and Ethernet switching functions were prototyped for VNode by this method and evaluated.

► Conclusions
  ■ Plug-ins for routing and switching can easily be developed.
  ■ Slice developers can easily use the plug-in functions in slices.

► Future work
  ■ Extending switch/router plug-ins and implementing new plug-ins (e.g., switching non-Ethernet addresses).