Network-resource Isolation for Virtualization Nodes

Yasusi Kanada, Hitachi, Ltd.
Kei Shiraishi, Hitachi, Ltd.
Akihiro Nakao, University of Tokyo
Outline: Why NRI and how?

- We developed a network-virtualization architecture and platform.
  - In this platform, multiple slices (i.e., virtual networks) can be created on one physical network called a virtualization platform.

- Network-resource isolation (NRI) between slices is necessary for network virtualization.
  - Because resource interference (concerning communication bandwidth, delay, etc.) between slices must be avoided.

- We propose three methods of NRI based on shaping and policing (QoS mechanisms).
  - Per-slice shaping (PSS)
  - Per-link policing (PLP)
  - Combined method (PSS with PLP)
VNode (virtualization nodes) is a component of the network virtualization platform.

- VNode is a physical node.
- VNode forwards packets on the platform, which contain a virtualized packet on a slice (i.e., overlay approach).
- VNodes are connected by tunnels using a protocol such as GRE.
Components of VNode

▶ Programmer
  ◆ is a programmable component that processes packets on the slices.

▶ Redirector
  ◆ forwards (redirects) packets from another VNode to a programmer and forwards packets from a programmer to another VNode.
  ◆ is a component that can forward or route packets on the platform.

▶ VNode Manager
  ◆ is a software component that manages the VNode.
Internal Structure of Redirector in VNode

- The redirector contains a high-end switch (or router) (and a packet encoder/decoder, such as a GRE encoder/decoder).
- This switch has policers and shapers that can be used for implementing NRI.
Specification of NRI

To isolate a slice from other slices, **bandwidth** (and **burst size**) is specified in virtual links in the slice definition.

Example of virtual link specification:

```
<linkSliver type="link" subtype="GRE" name="VirtualLink1">
  <vports><vport name="port0" /><vport name="port1" /></vports>
  <resources>
    <resource key="bandwidth" value="30M" />
    <resource key="burstSize" value="10k" />
  </resources>
</linkSliver>
```

Bandwidth = 30 Mbps, Burst size = 10 kB
Traffic control functions used for NRI

▶ Shaping
- queues packets, and limits and schedules the egress traffic.
- delays the packet, and drops it when the queue is filled.
- is more expensive and less scalable than policing (i.e., requires more memory and scheduling overhead).

▶ Policing
- measures network traffic without accumulating packets and drops packets when the bandwidth (or the burst size) exceeds a limit.
- can be used for guaranteeing bandwidth of virtual links that shares a queue (i.e., divides bandwidth reserved for a queue to slices).
- is less expensive and more scalable than shaping.

<table>
<thead>
<tr>
<th>Policing</th>
<th>Shaping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice s1 (Class c1)</td>
<td>virtual links</td>
</tr>
<tr>
<td>Slice s2 (Class c1)</td>
<td>virtual links</td>
</tr>
<tr>
<td>Slice s3 (Class c2)</td>
<td>virtual links</td>
</tr>
<tr>
<td>Redirector</td>
<td>virtual links</td>
</tr>
<tr>
<td>Policers</td>
<td>Shapers</td>
</tr>
<tr>
<td>Slice s1 (Class c1)</td>
<td>virtual links</td>
</tr>
<tr>
<td>Slice s2 (Class c1)</td>
<td>virtual links</td>
</tr>
<tr>
<td>Slice s3 (Class c2)</td>
<td>virtual links</td>
</tr>
</tbody>
</table>
Three Methods for NRI

- Per-slice shaping (PSS)
- Per-link policing (PLP)
- Combined method (PSS with PLP)
Three Methods for NRI (cont’d)

▶ PSS (Per-slice shaping)

- isolates slices strictly by shaping traffic per-slice instead of per-link (i.e., per virtual-link).
  - Although per-link shaping is required for guaranteeing QoS.
- is **sufficient for NRI** but does not guarantee per-link bandwidth.
- does not use policing (does not intentionally drop packets).
- is **more scalable than per-link shaping** (because it uses 80–90% less queues).

```
Slice s1 (Class c1)  Slice s1 (Class c1)
Slice s2 (Class c1)  Slice s2 (Class c1)
Slice s3 (Class c2)  Slice s3 (Class c2)
```

<table>
<thead>
<tr>
<th>No policing</th>
<th>Shaping per slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual links</td>
<td>virtual links</td>
</tr>
<tr>
<td>Redirector</td>
<td>Redirector</td>
</tr>
<tr>
<td>No policers</td>
<td>Shapers</td>
</tr>
<tr>
<td>Slice s1 (Class c1)</td>
<td>Slice s1 (Class c1)</td>
</tr>
<tr>
<td>Slice s2 (Class c1)</td>
<td>Slice s2 (Class c1)</td>
</tr>
<tr>
<td>Slice s3 (Class c2)</td>
<td>Slice s3 (Class c2)</td>
</tr>
</tbody>
</table>
Three Methods for NRI (cont’d)

**PLP (Per-link policing)**
- isolates slices (and virtual links) statistically (in a less-strict way) by policing traffic per-link; that is, guarantees per-link bandwidth by measuring and dropping packets link-by-link.
- uses shaping per slice-class (that is, slices share a queue).
- is more scalable than per-slice shaping (is applicable to hundreds of slices).
- may be influenced more by other slices than PSS (may be worse in delay and jitter).

![Diagram of Policing per link and Shaping per class]

- **Policing per link**
  - Slice s1 (Class c1)
  - Slice s2 (Class c1)
  - Slice s3 (Class c2)

- **Shaping per class**
  - Slice s1 (Class c1)
  - Slice s2 (Class c1)
  - Slice s3 (Class c2)
Three Methods for NRI (cont’d)

▶ Combined method (PSS with PLP)
- isolates slices by shaping traffic per slice and policing traffic per-link.
- is as strict as PSS in isolation from other slices.
- statistically guarantees per-link bandwidth (QoS).

Combined method

Policing per link

Shaping per slice

Slice s1 (Class c1)

Slice s2 (Class c1)

Slice s3 (Class c2)
We implemented the three methods for NRI.

Evaluation of slow-path and fast-path virtual nodes

- **Method**: Three slices are used: one for foreground traffic to be measured and two for background cross traffic.
- **Result**: Slow-path (Linux VM) virtual nodes

<table>
<thead>
<tr>
<th>Isolation type</th>
<th>Delay (mS)</th>
<th>Jitter (mS)</th>
<th>Drop ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Std dev</td>
<td>Average</td>
</tr>
<tr>
<td>PLP</td>
<td>1.60</td>
<td>0.12</td>
<td>0.10</td>
</tr>
<tr>
<td>PSS</td>
<td>1.30</td>
<td>0.08</td>
<td>0.11</td>
</tr>
<tr>
<td>Combined</td>
<td>1.33</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>No isolation</td>
<td>12.08</td>
<td>4.28</td>
<td>0.12</td>
</tr>
<tr>
<td>(Congestion-less)</td>
<td>1.31</td>
<td>0.15</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Conditions: Link sliver bandwidth = 100 Mbps, traffic = 90 Mbps. Cross traffic fills the link.

- **Result**: Fast-path virtual node (using a network processor)
  - Slices have been isolated when the foreground traffic is 4.0 Gbps or less. (The physical link bandwidth is 10 Gbps.)
Three methods for NRI for virtualization networks are proposed in this paper.

- PSS enables NRI with 80–90% less queues compared to the per-link shaping.
- PLP enables less strict isolation between tens or hundreds of slices using only one queue.
- A combination of PSS and PLP.

Evaluations of these methods show that PSS performs slightly better in terms of delay and packet-drop ratio.

Applications of PSS and PLP:

- PSS and the combined method are effective for delay-sensitive services.
- PLP may be sufficiently used for the other types of services.
Suppl: Two Types of Slice Components in VNP

- **Node Sliver (or virtual node)**
  - represents computational resources that exist in a VNode (in a programmer).
  - is used for node control or protocol processing with an arbitrary packet format.
  - is generated by slicing physical computational resources.

- **Link Sliver (or virtual link)**
  - represents resources of a virtual link that connects two node slivers.
  - is generated by slicing physical network resources such as bandwidth.
Suppl: Components of Redirector

Redirector

Redirector Manager (RM)

Control Plane (C-Plane)

Redirector Body (RB)

Service Module Card (SMC)

Data Plane (D-Plane)

Internal Data Plane

Redirector Body (High-end L3 switch)
(Packet encoder/decoder)
A (human) **slice developer** writes a slice definition in XML.

The slice definition is sent to **DC**, distributed to each **VNode Manager**, and sent to the **programmer** and the **redirector**.