

CoreLab and VNode

Approach of Cluster Japan ☺

Aki Nakao
University of Tokyo
2010/7/21

Goals of This Talk

- ✪ Introducing our (similar) research efforts
 - ▣ GENI is cool, but we also do cool stuff ☺
- ✪ Seeking opportunities for collaboration
 - ▣ Especially on Federation etc.

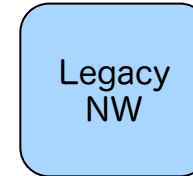
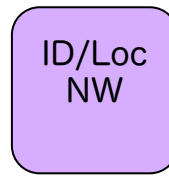
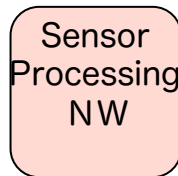
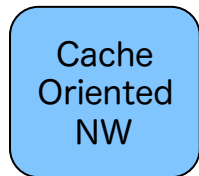


Net-Virt Infrastructure for diverse NWs

Slice 1

Slice 2

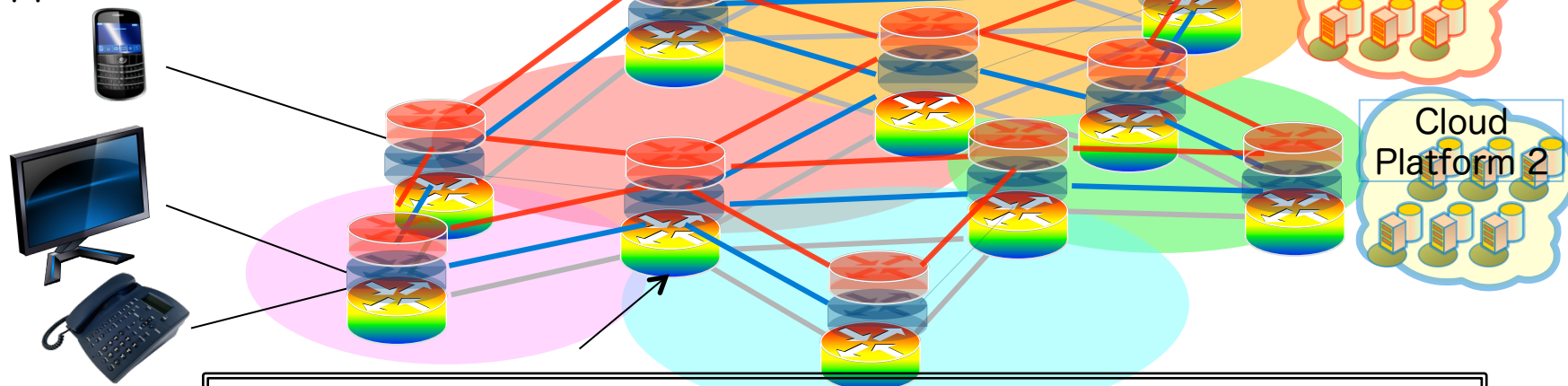
Slice N



Slices accommodate diverse NWs

Slicing Resources (CPU Cycles, Network, Storage, etc)

Handsets, PCs
Appliances, Sensors,...



Network Virtualization Infrastructure

Network Virtualization Research in Japan

✧ NV Research Lab (UTokyo+NICT)

We are trying to understand “Network Virtualization” Infrastructure

✧ CoreLab (UTokyo+NICT)

✧ Enabling net-virt via only S/W on COTS x86 machines

✧ VNode (UTokyo+NICT+NTT+NEC+Hitachi+Fujitsu)

✧ Enabling net-virt via designing H/W based on production routers

✧ Applications

✧ In Network Processing

✧ Non-IP Protocols





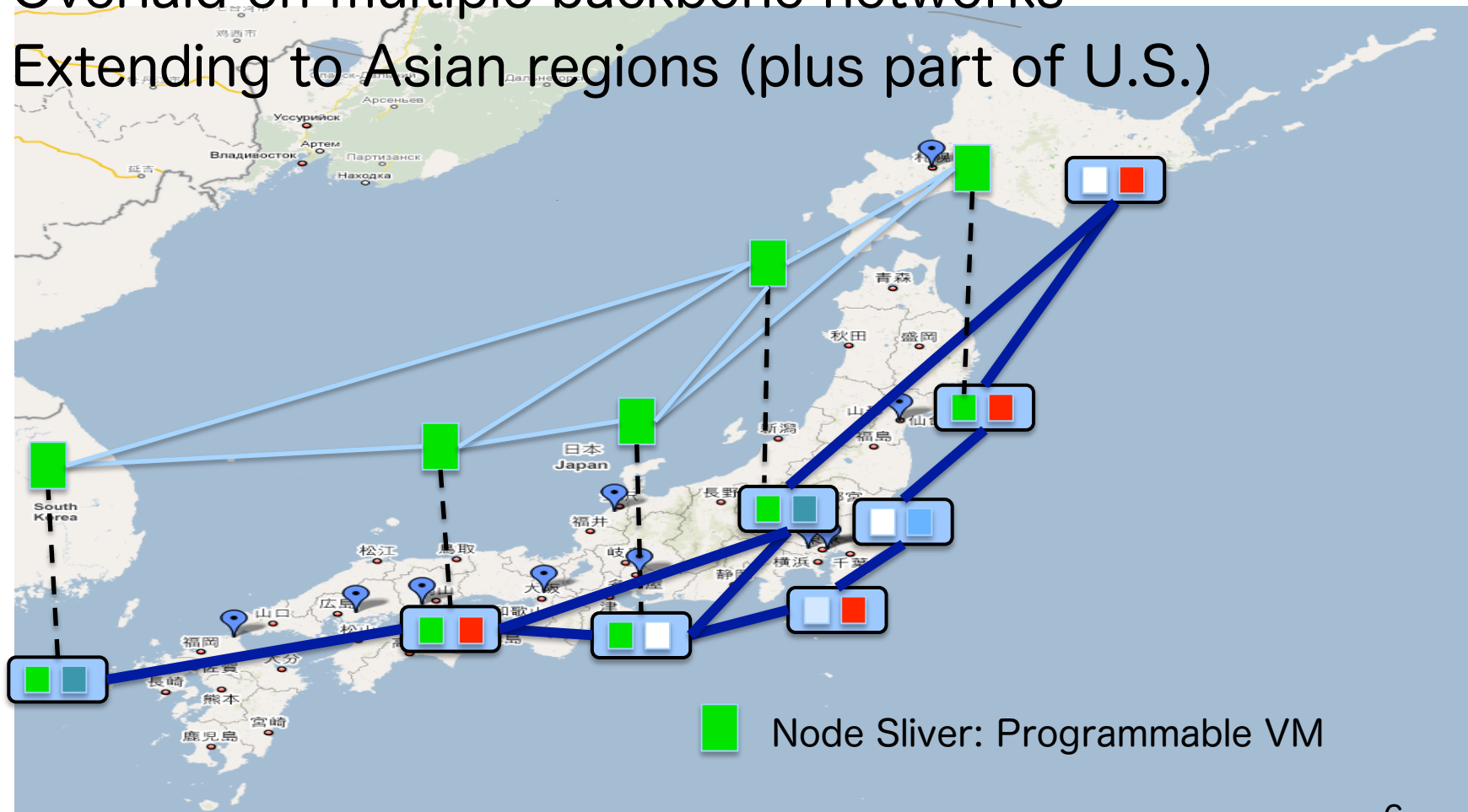
CoreLab

(UTokyo & NICT)

Enabling net-virt *via only S/W* on COTS x86 machines

CoreLab

- Shares DNA with PlanetLab (w/ more Net Virt Support)
- Creates slices across Japan (24+ nodes)
- Overlaid on multiple backbone networks
- Extending to Asian regions (plus part of U.S.)



CoreLab Highlights 1/2

1. Flexible Virtualization

Host-based Virtualization (KVM)
Resource Container (OpenVZ, LXC)

2. FlowSpace Isolation

GRE-tap Tunnels between Slivers
Classify traffic from/to slices via OpenFlow Switch

3. Switch Emulation

OpenFlow Switch In A Slice (OFIAS)




DEMO

4. Multi-Homing

Support multi-homing via multiple physical NICs

Create Slice

Interface:



eth0
eth1
eth2
eth3

CoreLab Highlights 2/2

5. VLAN Support

Assign any number of VLANs to a slice

Add A New Node

VLAN ID

6. DCN Integration

Isolate bandwidth through tagged VLAN and DCN capable L2 switches



DEMO

7. Guest-Host Pass-Through

Support PCI NIC Pass-Through (both Wired and Wireless)
Dedicate 10Gbps NIC to a sliver

Create Slice

PassThrough:

enabled
disabled

Highlight 1 : Flexible Virtualization

Create Slice

You must provide a short description of the new slice as well as against your slice and PlanetLab Operations is unable to determ

There are three possible "instantiation" states for a slice. **plc-in not-instantiated** allows you to reserve a slice name; you may

NOTE: All PlanetLab users are **strongly** encouraged to join the posting to this list. Site administrators often use this list to post well.

Site:	CoreLab Test rev3291 Central->cl
Name:	cl_
URL:	http://
Description:	
VMImage:	f10v1109plc118test
VM Type:	kvm lxc ovz
NIC Model:	virtio e1000 rtl8139
MultiHome:	enabled disabled
Memory Size:	256 512 1024 4096 8192
Allow Suspending:	<input checked="" type="checkbox"/> (What's this?)

Add Slice

VM Image
Arbitrary OS images registered

VM Type
KVM, LXC, OVZ, and more coming

NIC Model
Vanilla, Para-Virtualized Drivers

Multihome
SINET, JGN2Plus, or both

Memory Size
256, 512, 1024, 4096, 8192 MB

Suspend Feature
Suspend the slice when idol

Registering/Selecting VM Images

VMImages

Remove	ID	Name	VMImages OS Type	Arch	Description
<input type="checkbox"/>	1	f10v1109plc118test	redhat	i386	f10v1109plc118test

Select All Select None

Remove vmimages

Name:

Base URL:

Disk image torrent URL: will be Base URL + "base-" + Name + ".img.gz.torrent"

Disk image pattern URL: will be Base URL + "base-" + Name + ".img.gz.pattern"

Rootfs torrent URL: will be Base URL + "base-" + Name + ".tar.gz.torrent"

OS Type:
redhat
debian
windows

Arch:
i386
x86_64

Description:

Add vmimage

Fedora10 (RedHat Type/ i386)

We can add arbitrary VM images to disseminate to nodes...

Graphical Topology Management

- GUI for editing a slice topology
- Web-based VNC to login a slice

The screenshot displays a web browser window titled "Topology Assignment for cl_gec | CoreLab Test rev3165". The address bar shows the URL "https://plc119.nvlab.org:443/db/slices/slice_topology.php?id=105". The browser's search bar contains "Google". The page content is divided into a left sidebar and a main grid area. The sidebar contains a menu with the following items:

- **Create Slice**
- Attribute Types
- Sirius
- **Admin**
 - Peers
 - Node Downtimes
- **Users**
 - My Account
 - Log out of CoreLab Test rev3165
- VMImages
- Downloads
- NodeLogs
- About

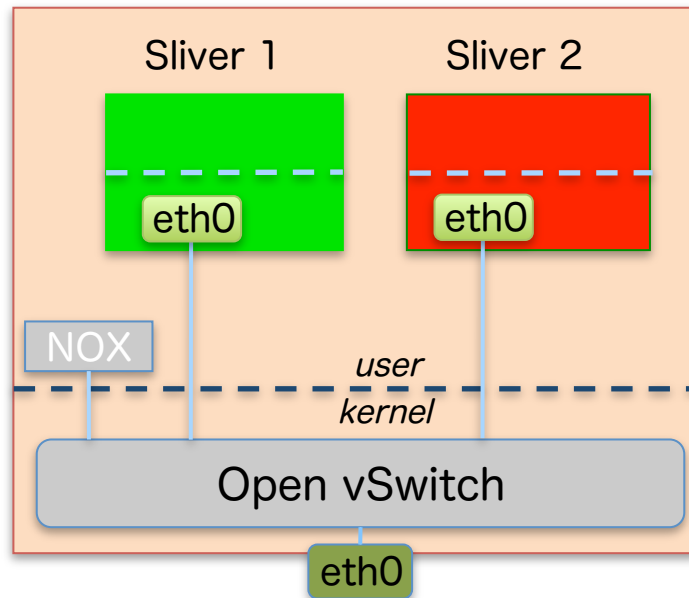
Below the sidebar, there is a "drupal" section with the following links:

- create content
- my account
- administer
- log out

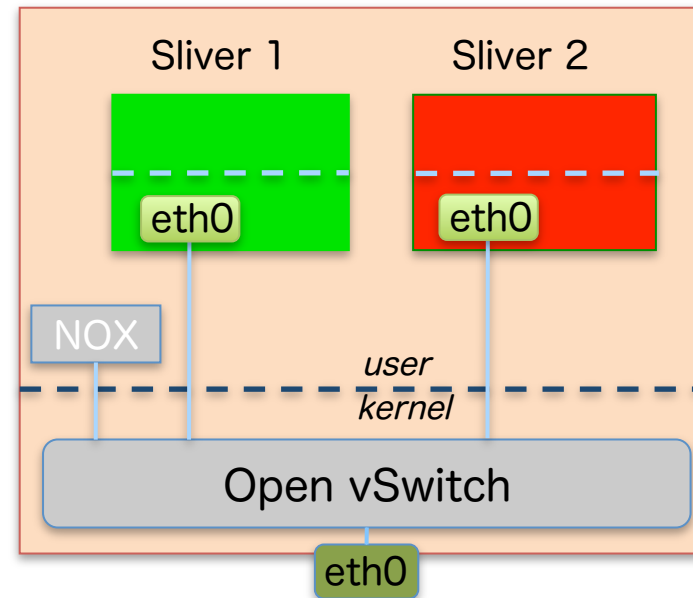
The main grid area displays a map of Japan overlaid on a green grid. The map is divided into several colored regions: Hokkaido (yellow), Tohoku (orange), Kanto (green), Chubu (light blue), and Kyushu (purple). A specific node is highlighted in red and labeled "node130". To the right of the map, there is a vertical color scale legend with a yellow box and a slider, ranging from 50 to 275.

Highlight 2 : FlowSpace Isolation

CoreLab node1



CoreLab node2

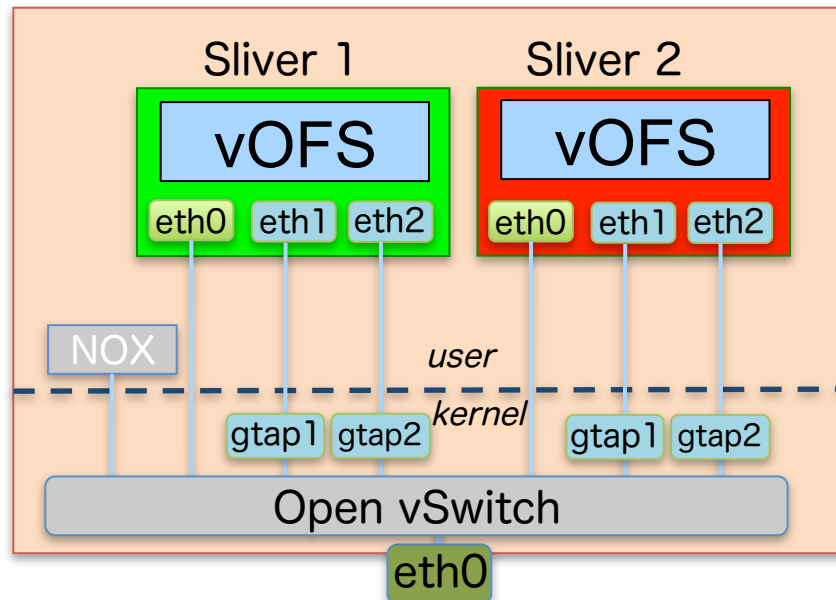


- Classify traffic from/to slices via OpenFlow Switch
- Sliver1 and Sliver2 (guests) may have **the same IP Address** as Host
- In general, we can **assign “flowspace” per slice.**

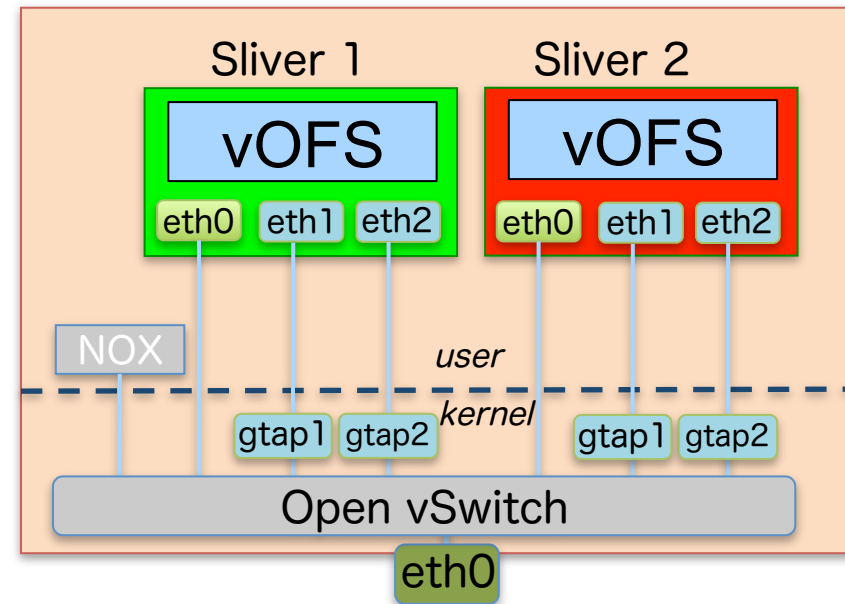
Port-space isolation with Open vSwitch [Tridentcom2010]

Highlight 3 : Switch Emulation

CoreLab node1



CoreLab node2



- Establish L2 GRE-TAP tunnels
- Install virtual OpenFlow switches (vOFS's) in each sliver

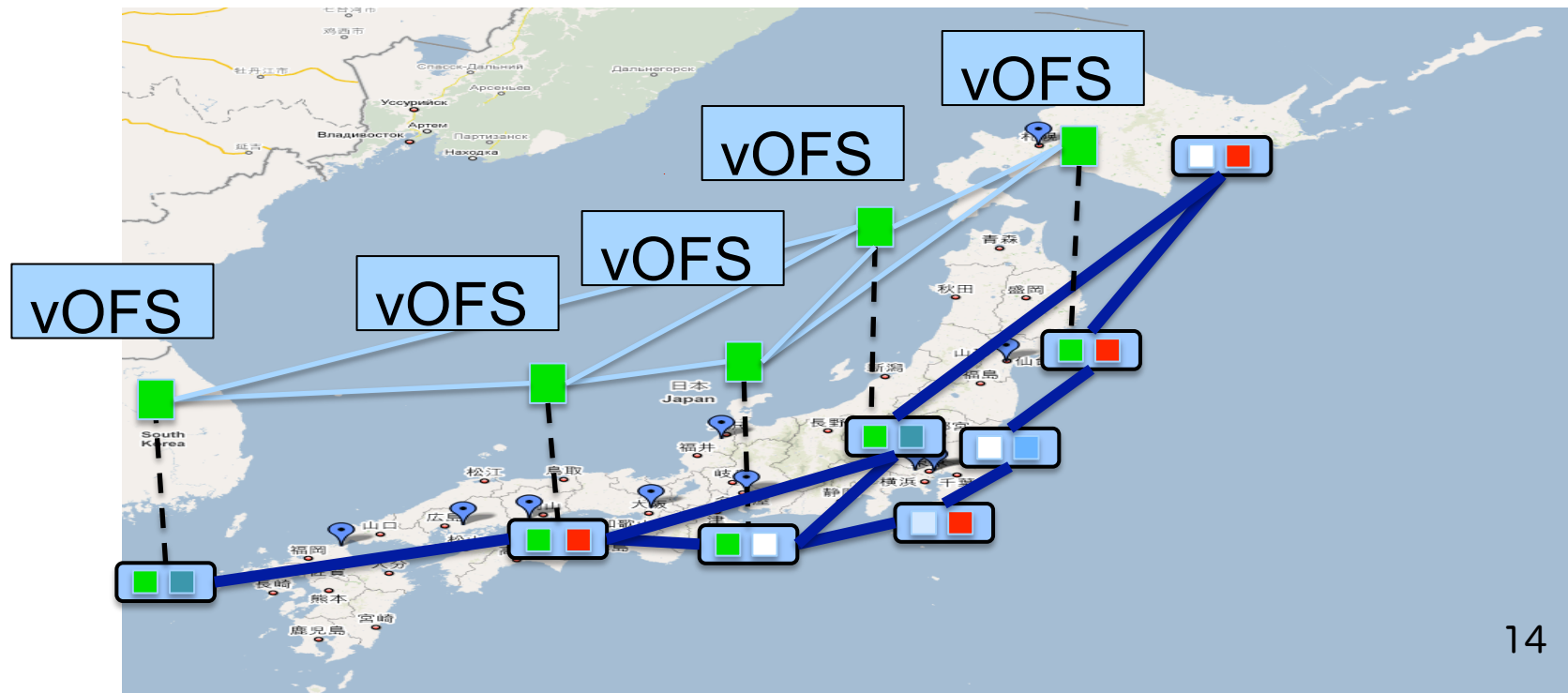
OpenFlow In A Slice 1/2

Benefit

- Allow a slice to experiment with OpenFlow networking

Ingredients

- Slices created on CoreLab
- vOFS**: Virtual OpenFlow Switch within a sliver

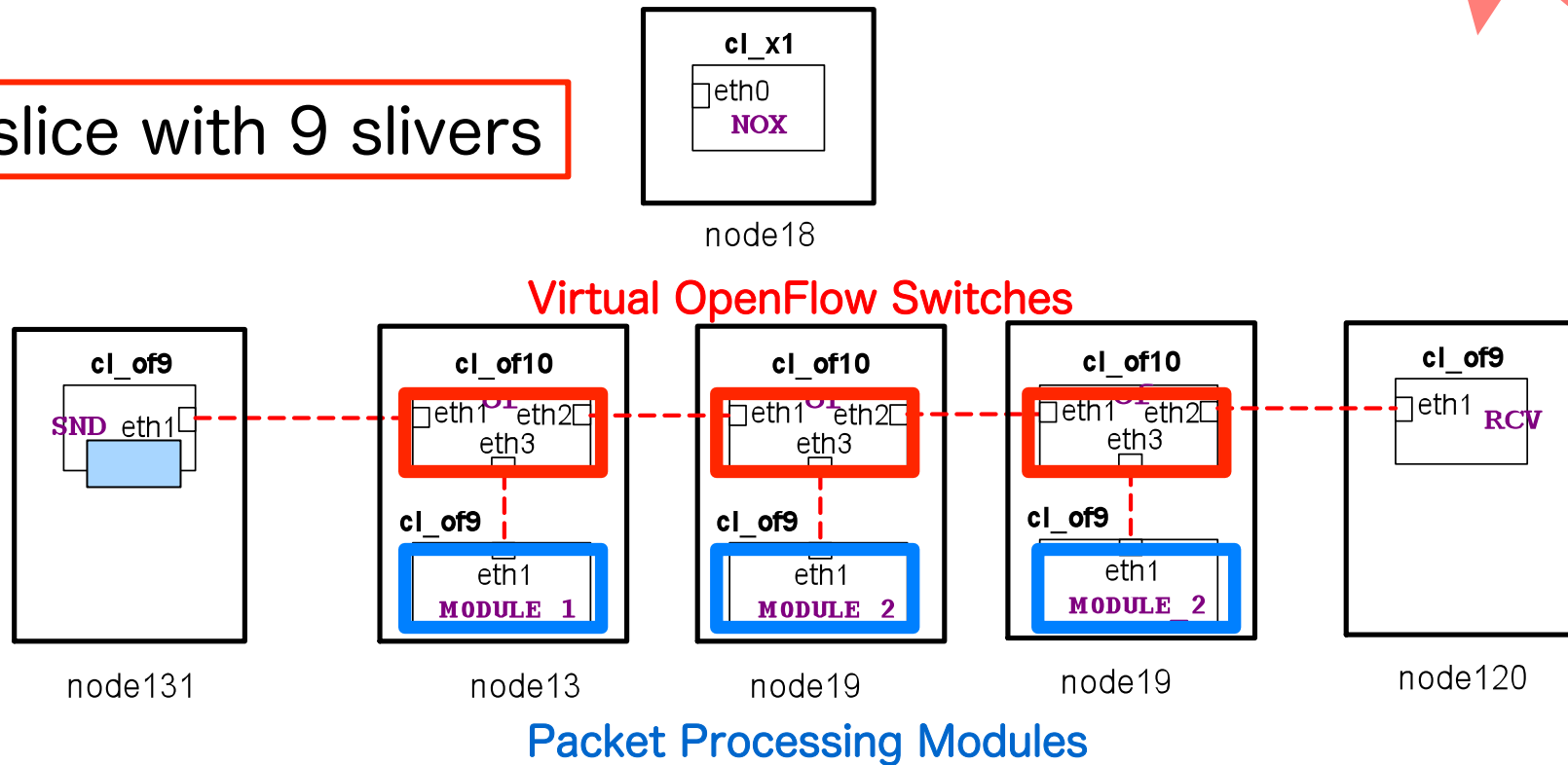


Openflow In A Slice 2/2

DEMO: OpenPipes In A Slice (OPIAS)

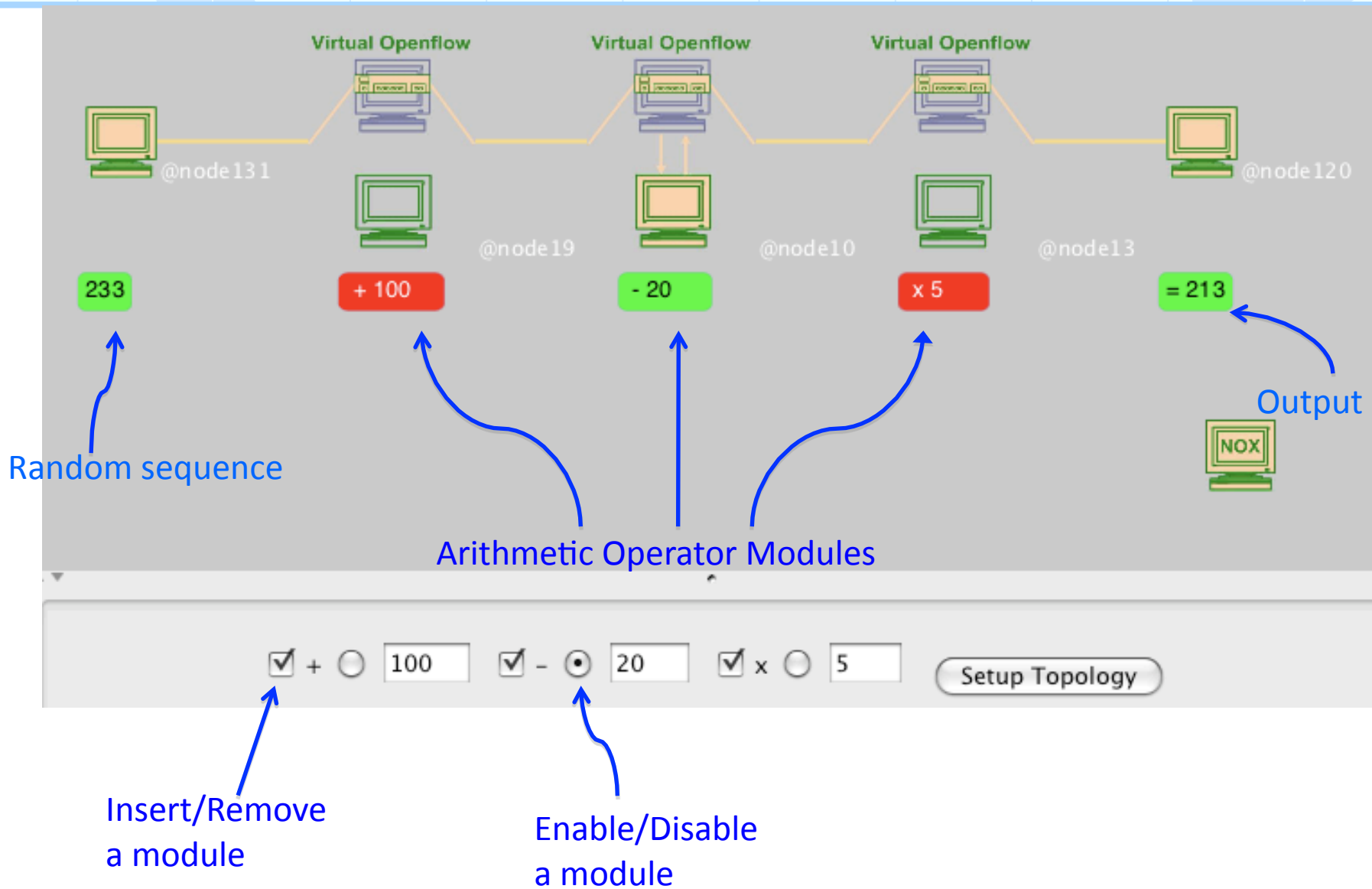
DEMO

A slice with 9 slivers



- ✦ Chaining Packet Processing Modules Along Data Path
- ✦ Virtualize the original OpenPipes demo at SIGCOMM 2009

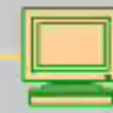
OpenPipes Demo Screenshot



OpenPipes Demo



@node131



@node120

548

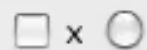
= 548



100



100



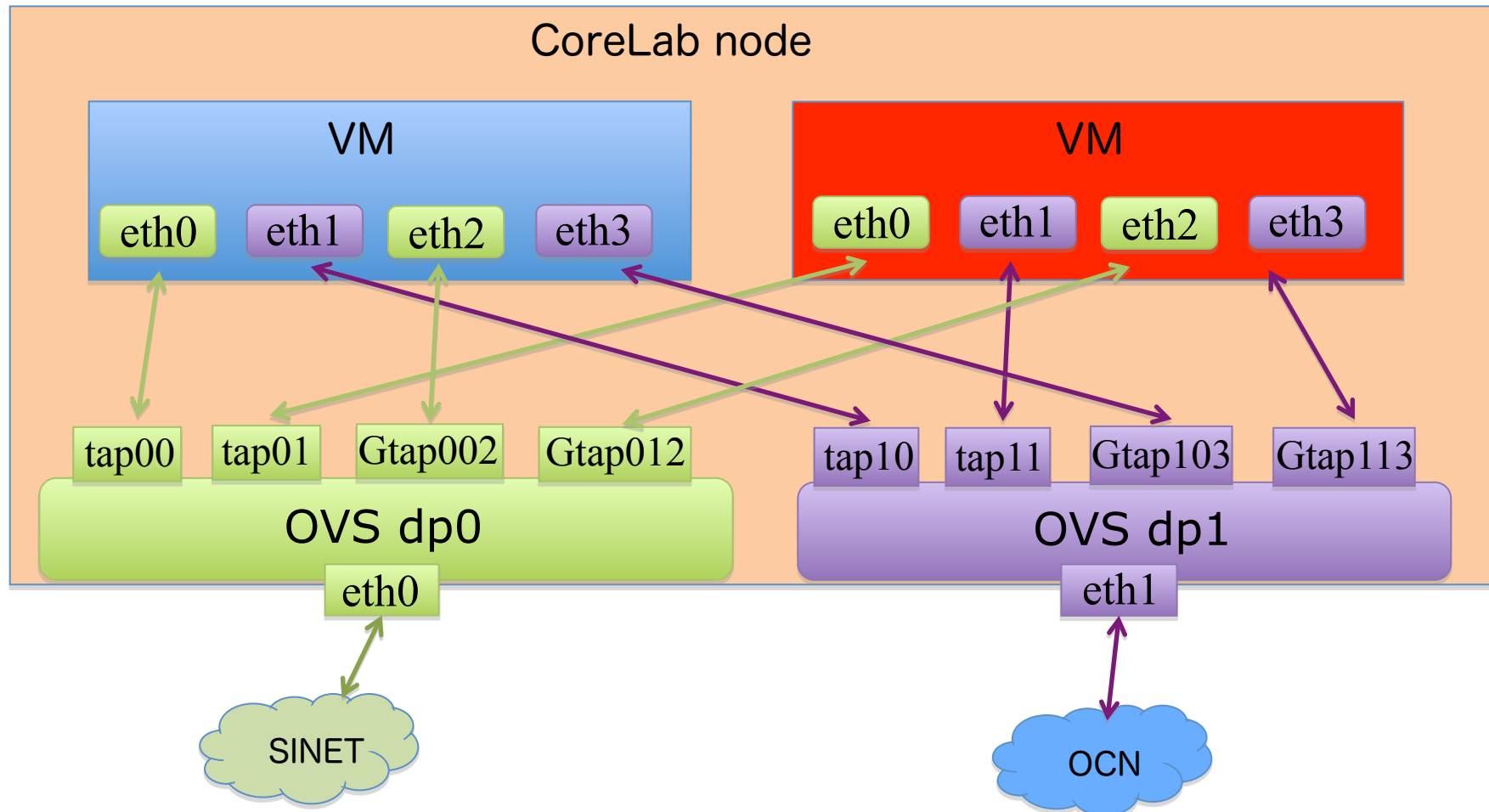
10

Setup Topology

Interpreter

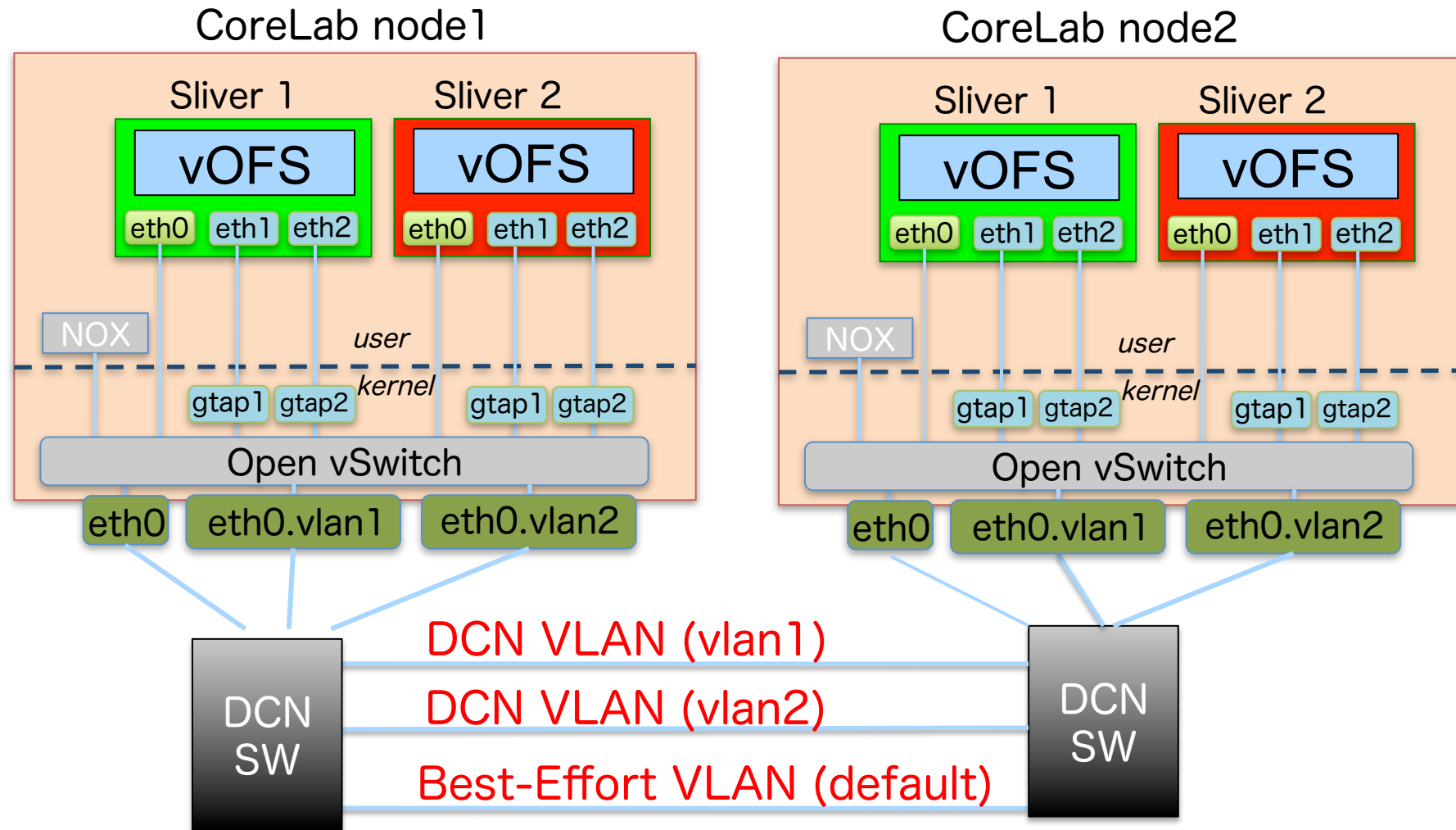
OpenFlow In A Slice (OFIAS)

Highlight 4 : Multi-homing L2 Tunnels



- Virtual Interface **eth0** shares the global IP address with Physical interface **eth0** (SINET)
- Virtual Interface **eth1** shares the global IP address with Physical interface **eth1** (OCN)
- Virtual Interfaces **eth2, eth4, eth $2n$** attach to Ethernet tunnels through **SINET**.
- Virtual interfaces **eth3, eth5, eth $2n+1$** attach to Ethernet tunnels through **OCN**.

Highlight 5&6: DCN+VLAN Integration



Creating Two DCN paths: 2Mbps/10Mbps

Diagram of CoreLab and DCN/ION joint Experiment

-  OSCARS/IDC
-  DRAGON/VLSR
-  GMPLS Control Plane
-  CoreLab Node

CoreLab Node1



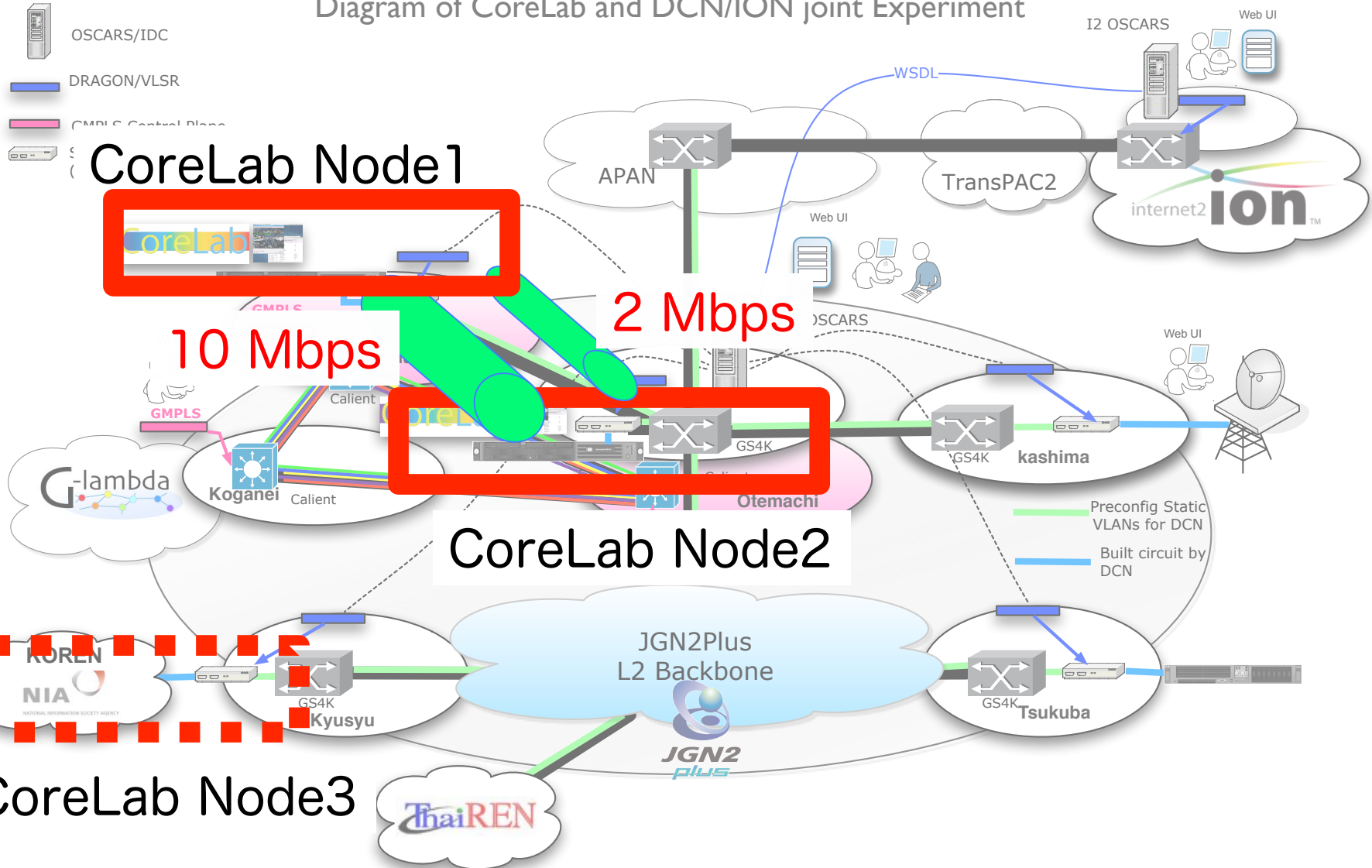
10 Mbps

2 Mbps

CoreLab Node2



CoreLab Node3



Streaming Video b/w Two CoreLab slivers

Screenshots at Hakusan node through VNC :

Poor Video

The screenshot shows a VLC media player window with a severely corrupted video stream. The video frame is filled with a dense, multi-colored digital noise pattern, making the content unrecognizable. The VLC interface includes a menu bar (Media, Playback, Audio, Video, Tools, View, Help), a playback area with controls (play, stop, previous, next, full screen, playlist, extended settings, repeat, shuffle, volume), and a status bar showing 'udp://:1234' and '1.00x' speed. A 'Media Information' dialog box is open, displaying statistics for the current media or stream. The statistics table is as follows:

Category	Value
Input bitrate	1868 kb/s
Demuxed	183294 kB
Stream bitrate	1355 kb/s
Corrupted	238
Discontinuities	0

Below the statistics, a log window shows network-related messages:

```
[ 3] local 192.168
[ ID] Interval
[ 3] 0.0-10.0 sec
[ ID] Interval
[ 3] 10.0-20.0 sec
~[ ID] Interval
[ 3] 0.0-21.7 sec
cl dcn.node1-dcn.ha
```

DCN VLAN bandwidth: **2Mbps**

Fine Video

The screenshot shows a VLC media player window displaying a clear and smooth video stream of a person's face. The video quality is high, with no visible artifacts or noise. The VLC interface is identical to the 'Poor Video' screenshot. A 'Media Information' dialog box is open, displaying statistics for the current media or stream. The statistics table is as follows:

Category	Value
Input bitrate	8336 kb/s
Demuxed	38477 kB
Stream bitrate	8128 kb/s
Corrupted	0
Discontinuities	0

Below the statistics, a log window shows network-related messages:

```
[ 3] local 192.168
[ ID] Interval
[ 3] 0.0-10.0 sec
[ ID] Interval
[ 3] 10.0-20.0 sec
~[ ID] Interval
[ 3] 0.0-21.7 sec
cl dcn.node1-dcn.ha
```

DCN VLAN bandwidth: **10Mbps**

2Mbps Not Enough for Replaying Video

The screenshot shows the VLC media player interface. The main video window displays a corrupted video stream with significant digital artifacts. The Media Information dialog box is open, showing the following statistics:

Category	Value
Decoded blocks	4478
Displayed frames	5496
Lost frames	338
Input	
Read at media	197711 kB
Input bitrate	1870 kb/s
Demuxed	188530 kB
Stream bitrate	1671 kb/s
Corrupted	190
Discontinuities	0
Streaming	
Sent packets	0
Sent bytes	0 kB

The 'Input bitrate' value of 1870 kb/s is highlighted with a red box and a red arrow pointing to it. The VLC status bar at the bottom shows the location as 'udp://@:1234' and a playback speed of 1.00x.

Received bitrate limited to 2Mbps

DCN VLAN Bandwidth Limit : 2Mbps

10Mbps Achieves Fine Video Playback

The screenshot shows the VLC media player interface. The main window displays a video of a woman's face. A 'Media Information' dialog box is open, showing the 'Statistics' tab. The 'Input' section of the statistics is highlighted, and the 'Input bitrate' is circled in red. A red arrow points from the text 'Received bitrate is 8Mbps' to the circled value.

Category	Value
Decoded blocks	3067
Displayed frames	3665
Lost frames	110
Input	
Read at media	136130 kB
Input bitrate	8372 kb/s
Demuxed	129710 kB
Stream bitrate	8131 kb/s
Corrupted	100
Discontinuities	0
Streaming	
Sent packets	0
Sent bytes	0 kB

Received bitrate is 8Mbps

DCN VLAN Bandwidth Limit: 10Mbps



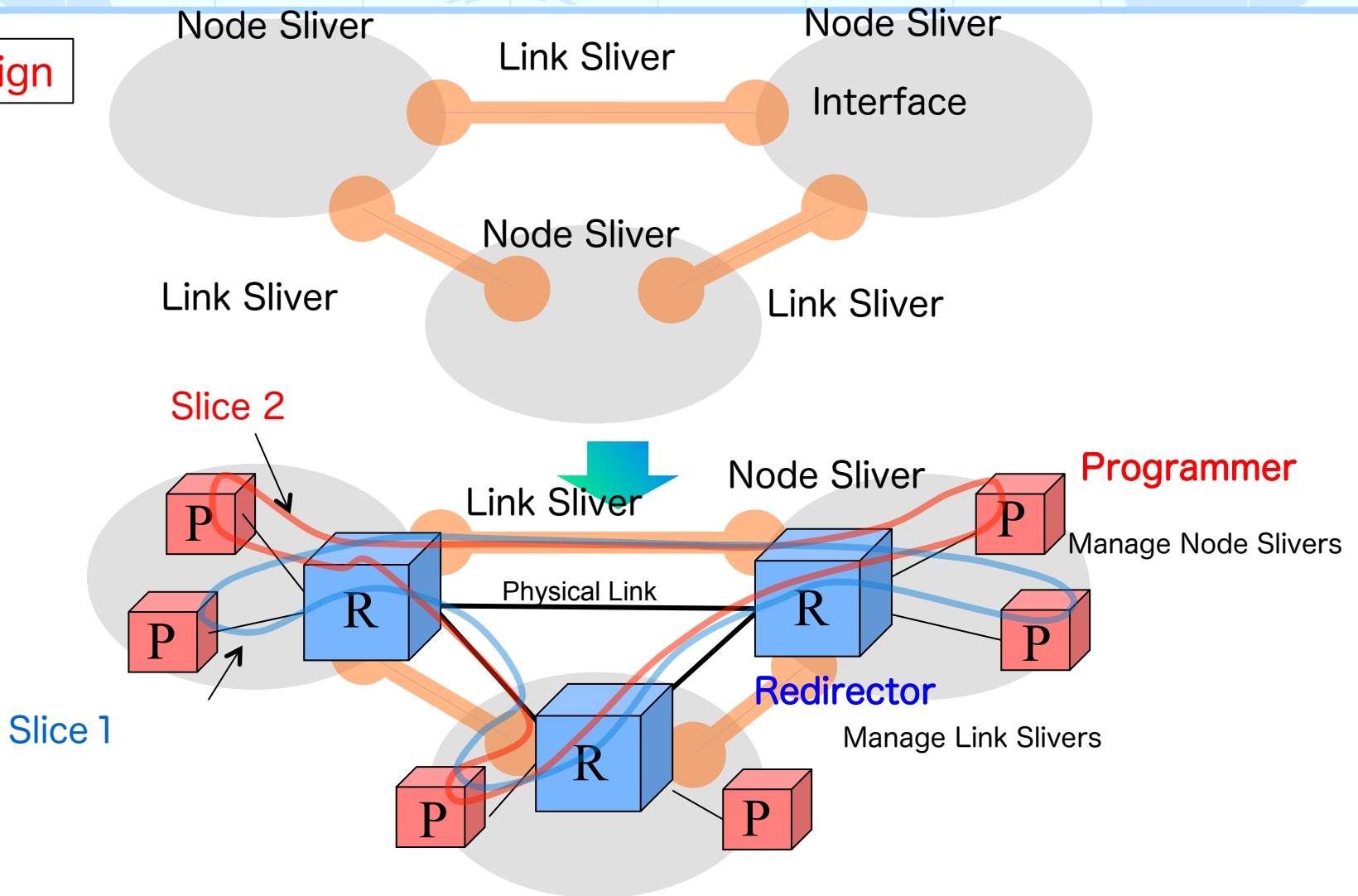
VNode Project

(UTokyo, NICT, NTT, NEC, Hitachi, Fujitsu)

Learned lessons from PlanetLab and CoreLab
Starting over and designing a new “slice” mechanism
Enabling net-virt **via H/W based on production routers**

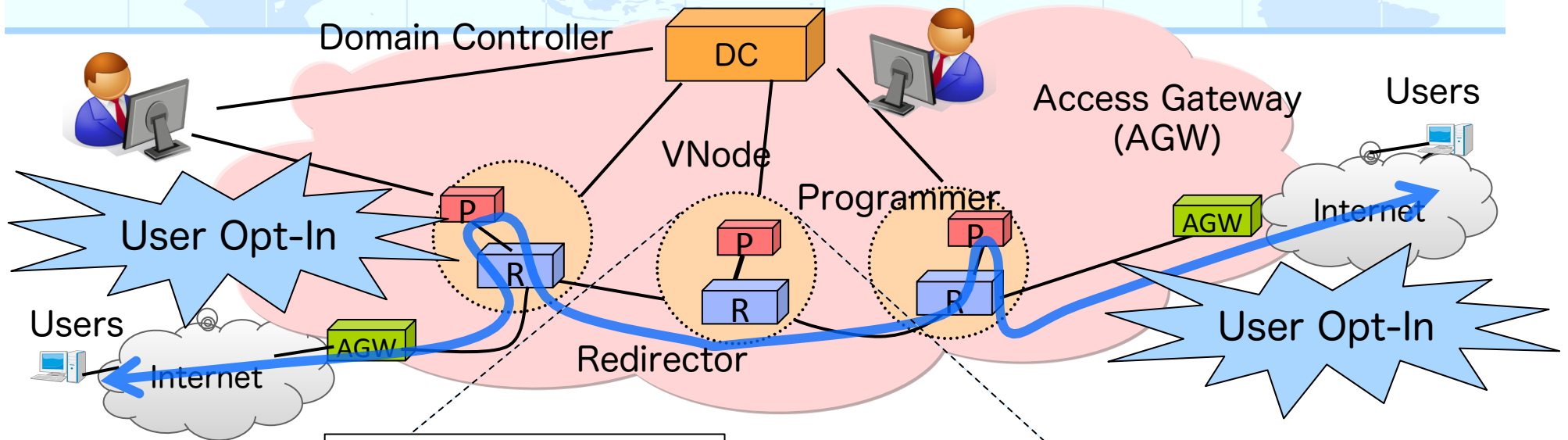
VNode Architecture 1/2

Slice Design

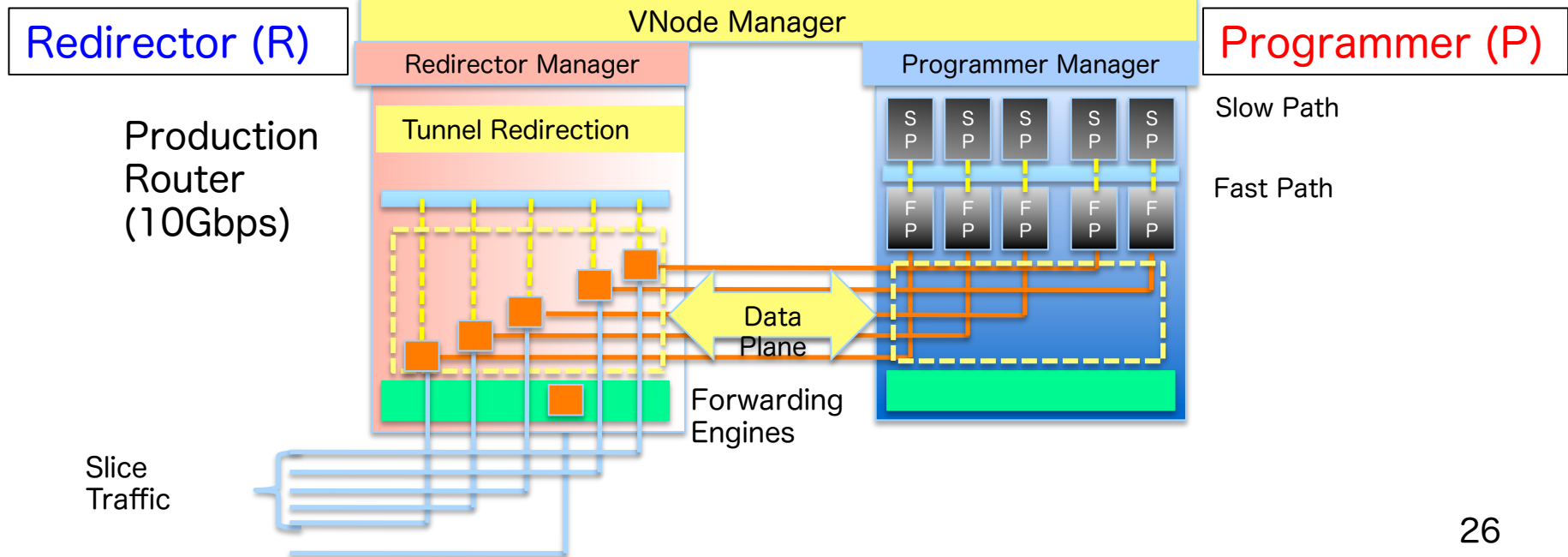


- Separation between Connectivity and Programmability
- Redirector and Programmer may evolve independently

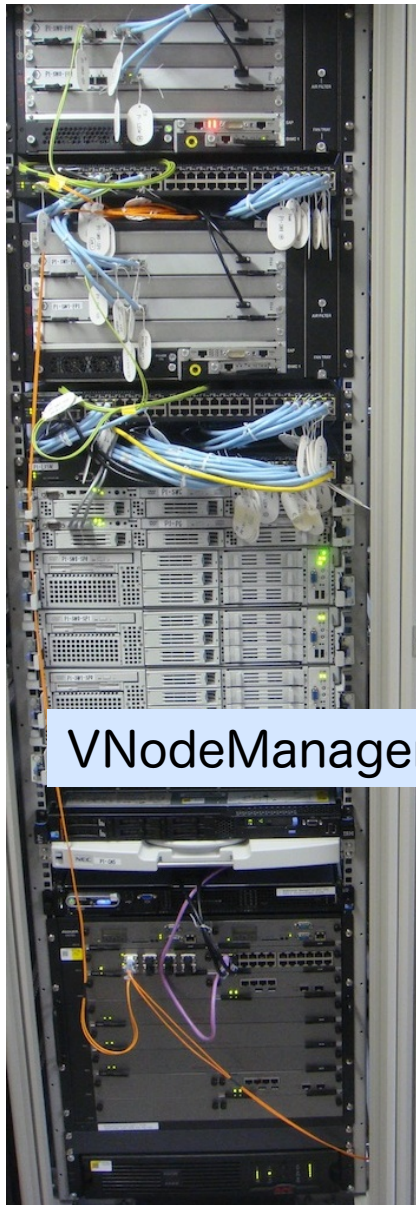
VNode Architecture 2/2



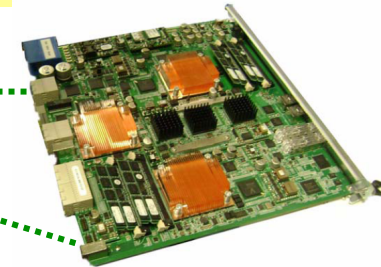
VNode Architecture



VNode: Nuts and Bolts View



Programmer Part
(IA Serverx4 +ATCAx2+OpenFlow SWx2)



OpenFlow Switch

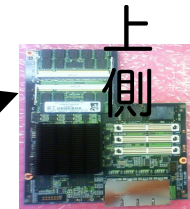
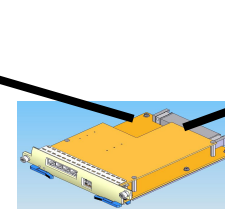
Fast-Path
Network Processor
Card

Redirector Part (AX6700+SMCx2)

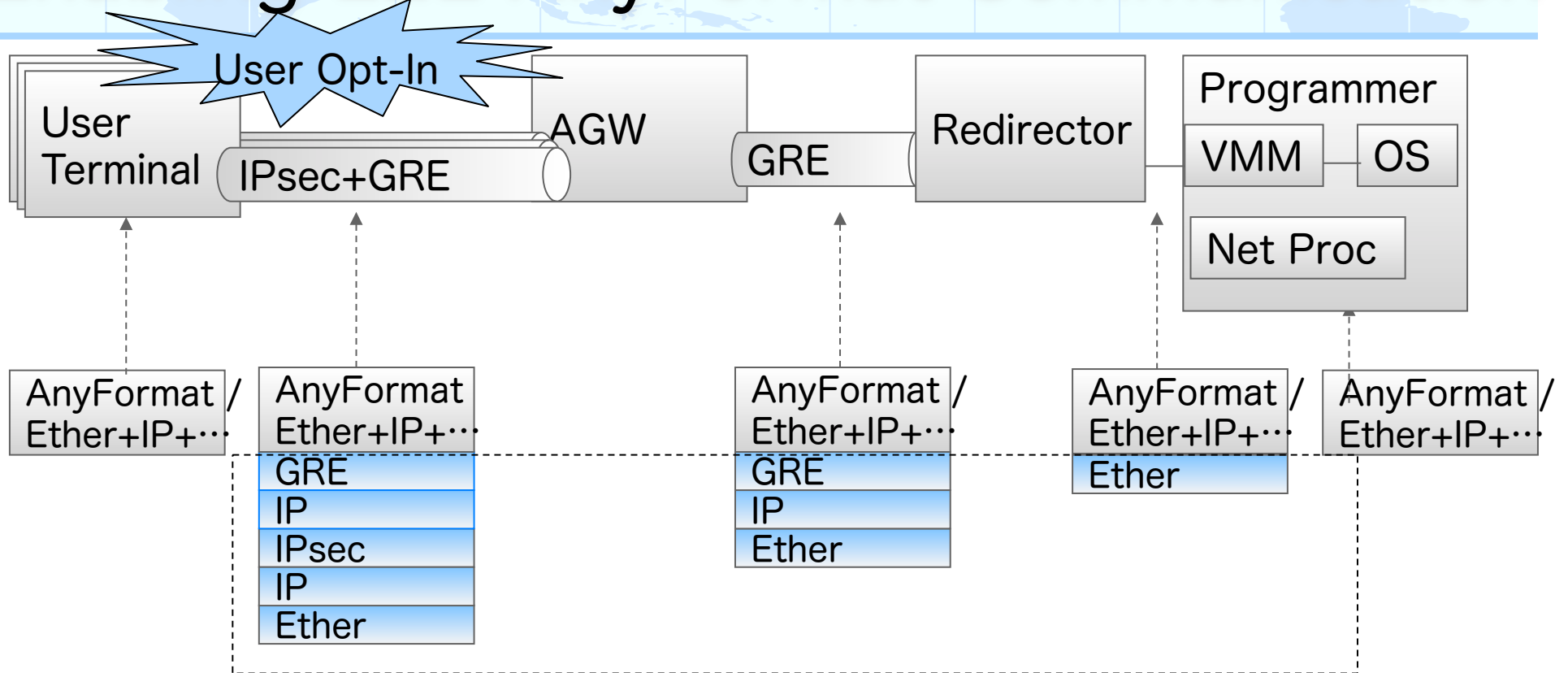
AX6708S



Service Module Card



Enabling E2E AnyFormat Communication

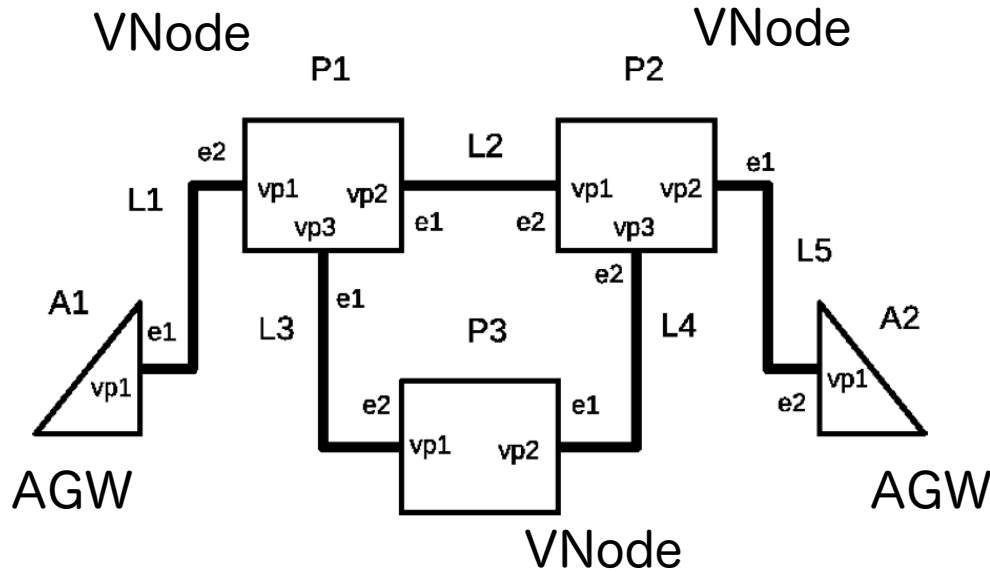


This part could be replaced with any underlay technology by new “Redirectors”

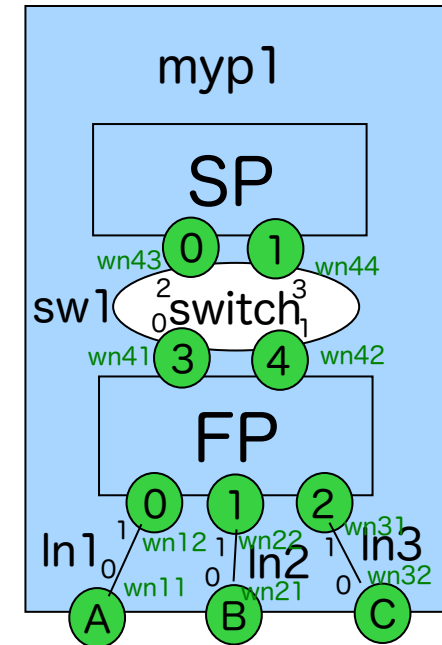
- Enable End-to-End “AnyFormat” Communication
- Use GRE-tap but may extend this to **MPLS, VLAN, and Optical Path**

Slice Design for a Virtual Network

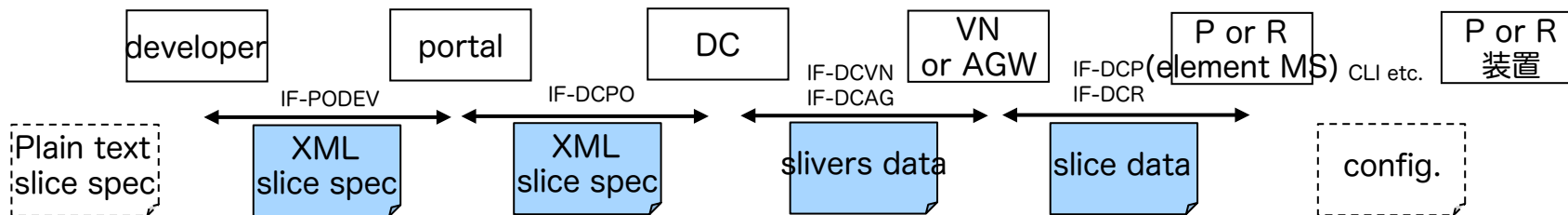
Slice Specification



Node Sliver



Slice Specification Flow



Slice Design XML Format

```
<NodeSliver>
  <sliverID>Sample_Node_Sliver</sliverID>
  <sliceID>100</sliceID>
  <virtualPorts>
    <virtualPort><name>A</name></virtualPort>
    <virtualPort><name>B</name></virtualPort>
    <virtualPort><name>C</name></virtualPort>
  </virtualPorts>
  <components>
    <NodeSliverComponent>
      <name>FP</name>
      <type>FastPath_BareMetal</type>
      <subtype>RMI</subtype>
      <interfaces>
        <numOfInterfaces>5</numOfInterfaces>
      </interfaces>
      <resource>
        <cpu>1</cpu>
        <cpumode>dedicated</cpumode>
        <memory unit="MB">256</memory>
      </resource>
      <parameters>
        <bootImage>http://example.com/RMI_binary </bootImage>
        <bootParams/>
      </parameters>
    </NodeSliverComponent>
    <NodeSliverComponent>
      <name>SP</name>
      <type>SlowPath_VM</type>
      <subtype>KVM</subtype>
      <interfaces>
        <numOfInterfaces>2</numOfInterfaces>
      </interfaces>
      <resource>
        <cpu>1</cpu>
        <cpumode>dedicated</cpumode>
        <memory unit="B">2048</memory>
      </resource>
      <parameters>
        <bootImage>http://example.com/KVM_disk.img </bootImage>
        <bootParams/>
      </parameters>
    </NodeSliverComponent>
  </components>
```

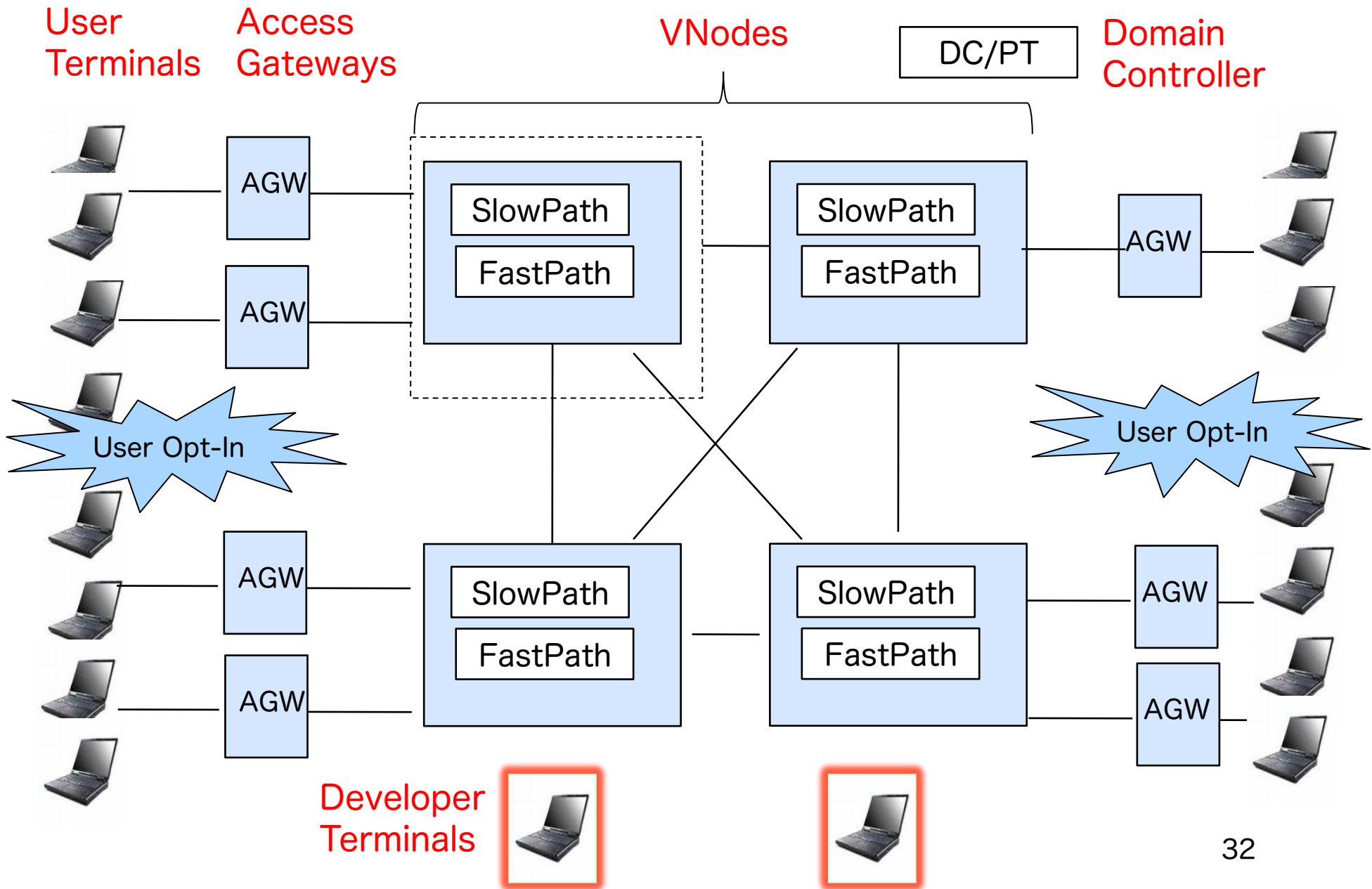
```
<connectionMap>
  <NodeConnection type="P-to-P">
    <source>
      <VirtualInterface>
        <component>_VPORT_</component>
        <interface>A</interface>
      </VirtualInterface>
    </source>
    <destination>
      <VirtualInterface>
        <component>FastPath</component>
        <interface>0</interface>
      </VirtualInterface>
    </destination>
    <bidir>TRUE</bidir>
  </NodeConnection>
  <NodeConnection type="P-to-P">
    <source>
      <VirtualInterface>
        <component>_VPORT_</component>
        <interface>B</interface>
      </VirtualInterface>
    </source>
    <destination>
      <VirtualInterface>
        <component>FastPath</component>
        <interface>1</interface>
      </VirtualInterface>
    </destination>
    <bidir>TRUE</bidir>
  </NodeConnection>
  <NodeConnection type="P-to-P">
    <source>
      <VirtualInterface>
        <component>_VPORT_</component>
        <interface>C</interface>
      </VirtualInterface>
    </source>
    <destination>
      <VirtualInterface>
        <component>FastPath</component>
        <interface>2</interface>
      </VirtualInterface>
    </destination>
    <bidir>TRUE</bidir>
  </NodeConnection>
```

```
<NodeConnection type="Switch">
  <VirtualInterface>
    <component>FastPath</component>
    <interface>3</interface>
  </VirtualInterface>
  <VirtualInterface>
    <component>FastPath</component>
    <interface>4</interface>
  </VirtualInterface>
  <VirtualInterface>
    <component>SlowPath</component>
    <interface>0</interface>
  </VirtualInterface>
  <VirtualInterface>
    <component>SlowPath</component>
    <interface>1</interface>
  </VirtualInterface>
</NodeConnection>
</connectionMap>
<parameters>
  <authKey>Base64_Encoded_SshKeyFile</authKey>
</parameters>
</NodeSliver>
```

A Prototype System (4 VNodes)

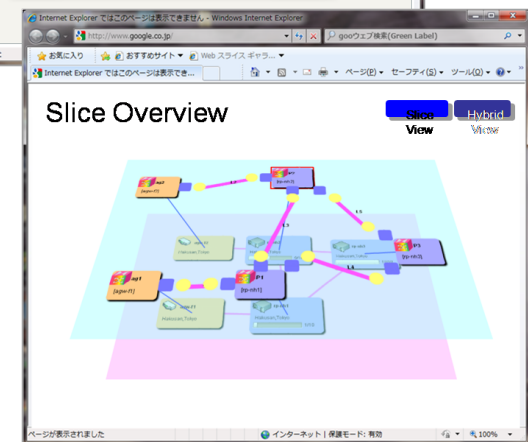
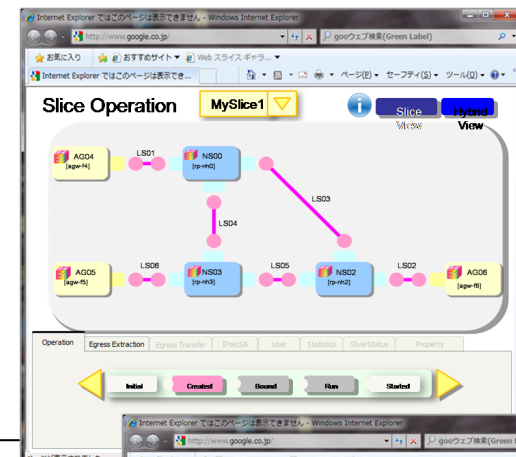
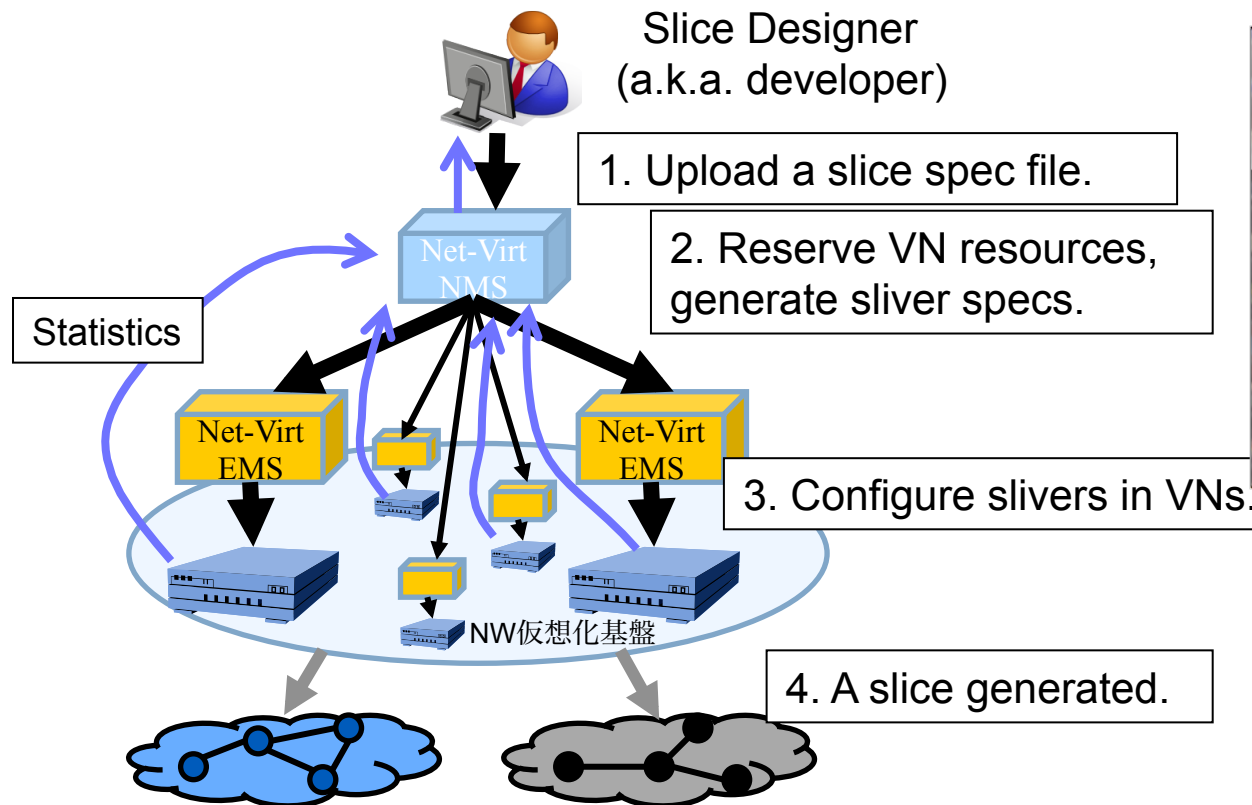


Simplified Picture of Prototype



Management/Control for VNode Platform

Get requests from *developers* and send control messages to multiple *VNodes* to create *slices* dynamically



VNode Control Plane Demo

Login

User ID

Password

Benefit of VNode Infrastructure

- (A) Accommodate **Multiple Independent Networks**
=> **Meta Architecture**
- (B) Enable **In-Network Processing**
- (C) Create **App-Specific Networks**
- (D) Adapt **Computational Resources**
- (E) Enable **Non-IP E2E Communication**

6 Slices Running on VNode Prototype

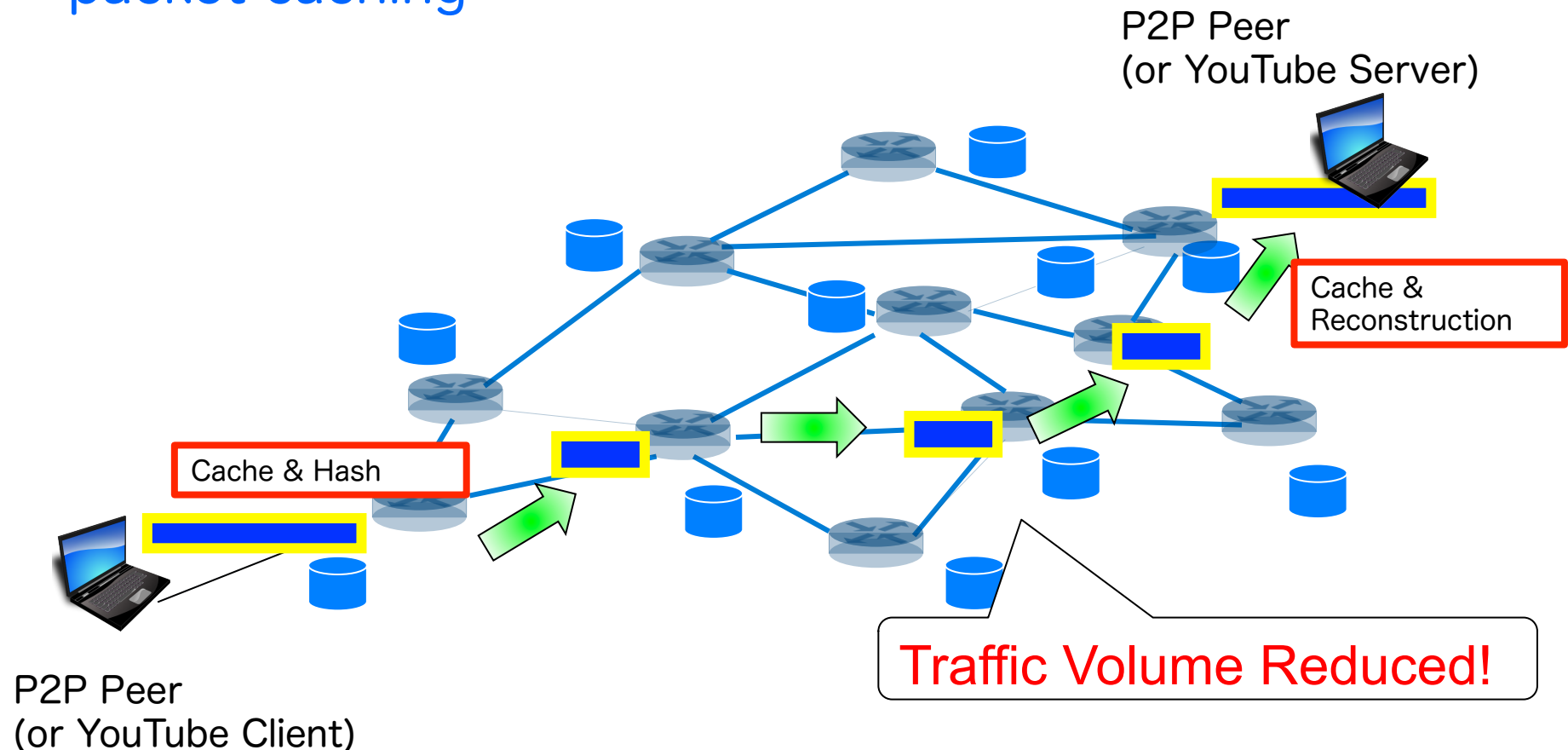
Which features of VNode Infra that each slice demonstrates

Group	NW Architecture	Meta Architecture	In Network Processing	App Specific Network	Adaptive Computing Resources	Non-IP (E2E) Architecture
UTokyo N I C T	Packet Cache	Slice 1	Packet Cache		SlowPath FastPath ※	
N T T	Flexcast	Slice 2	Flexible Multicast			
Fujitsu	Sensor Specific Networking	Slice 3	Sensor Data Processing	Sensor Specific Network		
N E C	Stream Computing	Slice 4	Stream Computing		SlowPath FastPath	Non-IP Ephemeral Multicast
Hitachi1	IP-Ethernet-Chimera (IPEC)	Slice 5	IPEC (L2/L3)			IPEC (L2/L3)
Hitachi2	Data Sync for Distributed Data Center	Slice 6		Data Sync Network QoS		

Slice 1: Cache Oriented NW Architecture

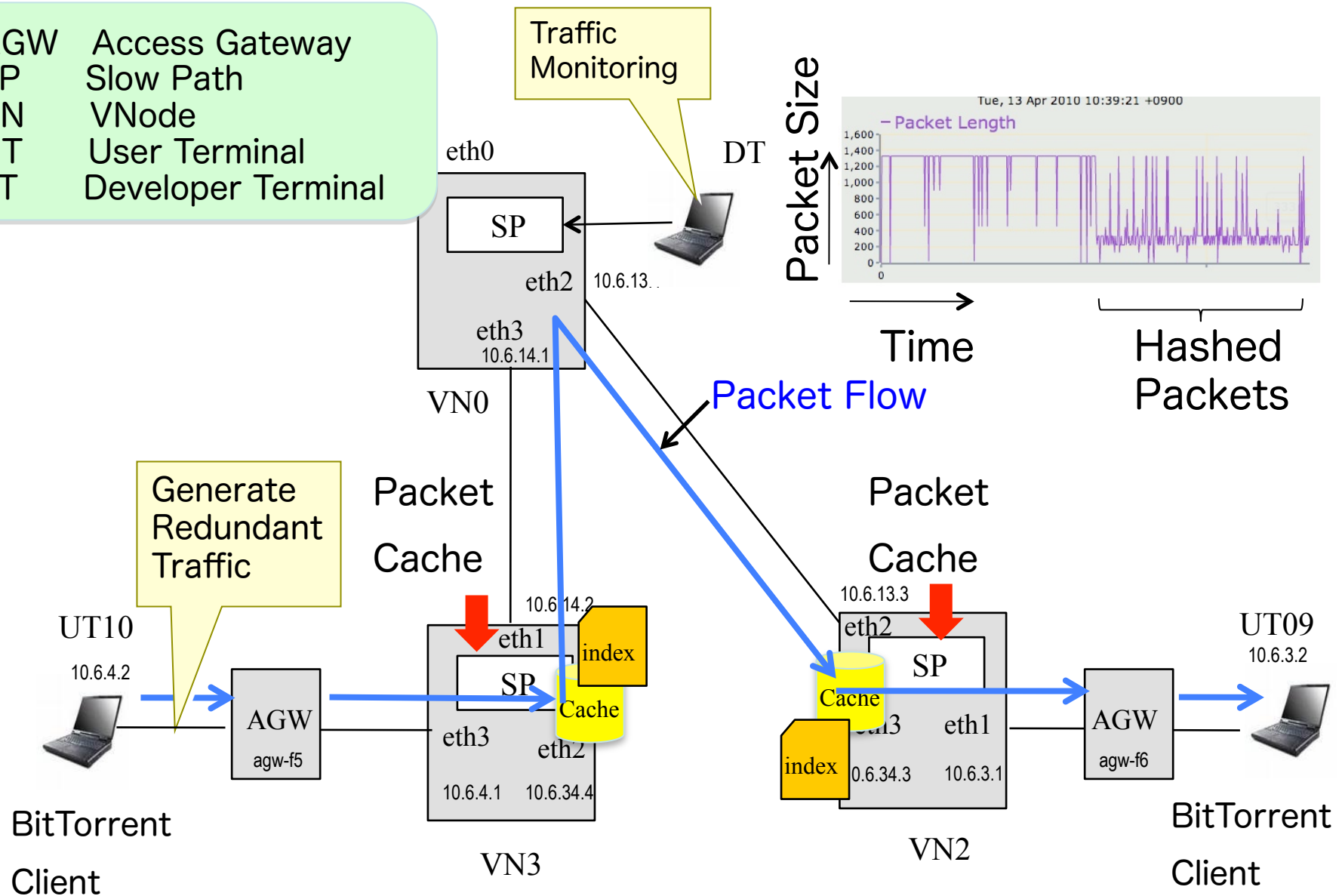
Goal : Reduce redundant traffic in P2P and Cloud Streaming

- Utilize **programmability** and **storage** in VNode
- Confine P2P and YouTube Traffic to a slice and perform **packet caching**



P2P Packet Caching

AGW Access Gateway
 SP Slow Path
 VN VNode
 UT User Terminal
 DT Developer Terminal



CONA Demo

The image shows a network traffic analysis tool interface. The top part displays a graph titled "Traffic@rp-nh0 NICT_Slive_006" for "Thu, 03 Jun 2010 09:09:11 +0900". The graph plots "Packet Length (bytes)" on the y-axis (0 to 1,600) against time on the x-axis. A single data point is visible at approximately 1,000 bytes. Below the graph, a status bar indicates "localhost からデータを転送しています...".

The bottom part of the image shows a terminal window with system logs. The logs list various processes and their states:

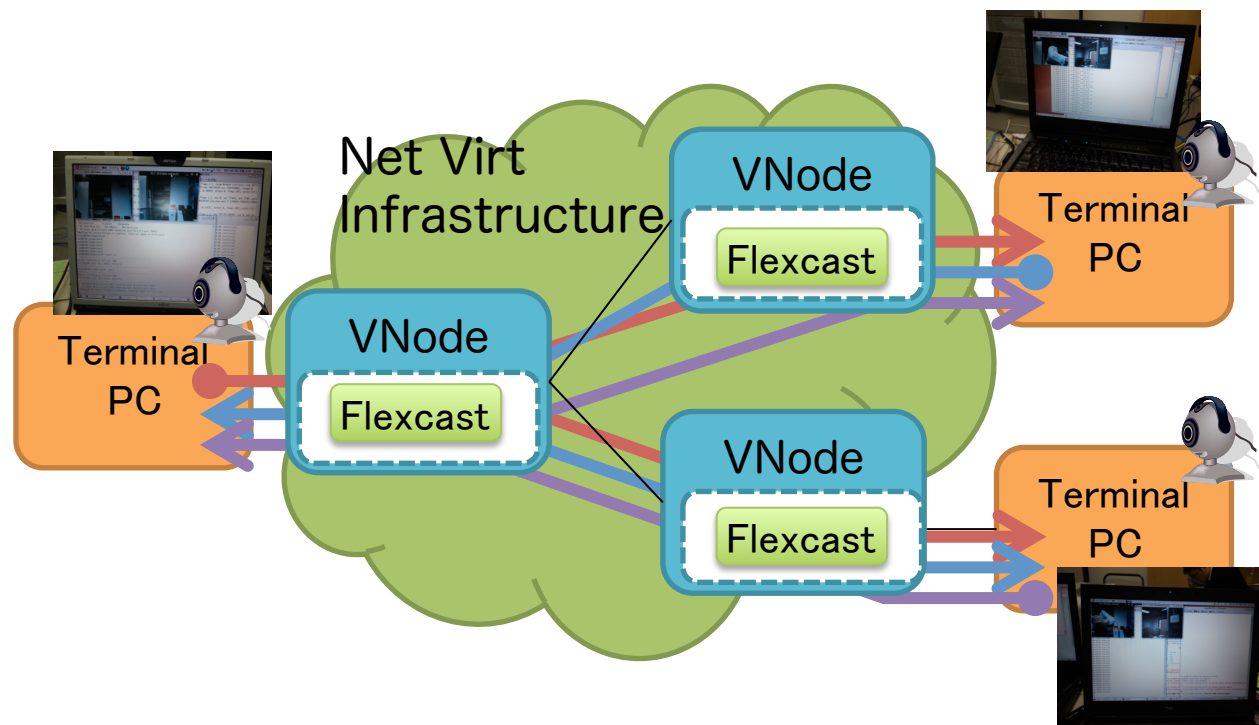
```
837 ? Ss 0:00 dhclient3 -e IF_METRIC=100 -pf /var/run/dhclien
888 ? Ss 0:00 /usr/sbin/sshd
952 ? Ss 0:00 sshd: root@notty
1048 ? S 13:48 ./flexcast_splitter -c ./flexcast_splitter.conf
7841 ? Ss 0:00 sshd: root@pts/1
7896 pts/1 Ss 0:00 -bash
8831 pts/1 Sl 12:34 ./ro_comd
8853 pts/1 R+ 0:00 ps ax
root@NICT-Slice-006-NS03-SP00:~/nict3/ingress# kill 8831
root@NICT-Slice-006-NS03-SP00:~/nict3/ingress# rmdir hcache_drv
[1]+ Done ./ro_comd > /dev/null
root@NICT-Slice-006-NS03-SP00:~/nict3/ingress# insmod hcache_drv/hcache_dr
v.ko
root@NICT-Slice-006-NS03-SP00:~/nict3/ingress# ./ro_comd > /dev/null &
```

Another terminal window shows a list of processes:

```
837 ? Ss 0:00 dhclient3 -e IF_METRIC=100 -pf /var/run/dhclien
888 ? Ss 0:00 /usr/sbin/sshd
952 ? Ss 0:00 sshd: root@notty
1048 ? S 14:34 ./flexcast_splitter -c ./flexcast_splitter.conf
18269 ? Ss 0:00 sshd: root@pts/1
18322 pts/1 Ss 0:00 -bash
18767 ? Ss 0:00 sshd: root@pts/0
18820 pts/0 Ss 0:00 -bash
18831 pts/0 S+ 0:00 ssh root@10.6.3.2
18869 pts/1 Sl 13:07 ./ro_comd
19944 pts/1 R+ 0:00 ps ax
root@NICT-Slice-006-NS02-SP00:~/nict2/egress# kill 18869
root@NICT-Slice-006-NS02-SP00:~/nict2/egress# rmdir hcache_drv
[1]+ Done ./ro_comd > /dev/null
root@NICT-Slice-006-NS02-SP00:~/nict2/egress# insmod hcache_drv/hcache_dr
v.ko
root@NICT-Slice-006-NS02-SP00:~/nict2/egress# ./ro_comd > /dev/null &
```

Slice 2: Ephemeral Streaming Protocol

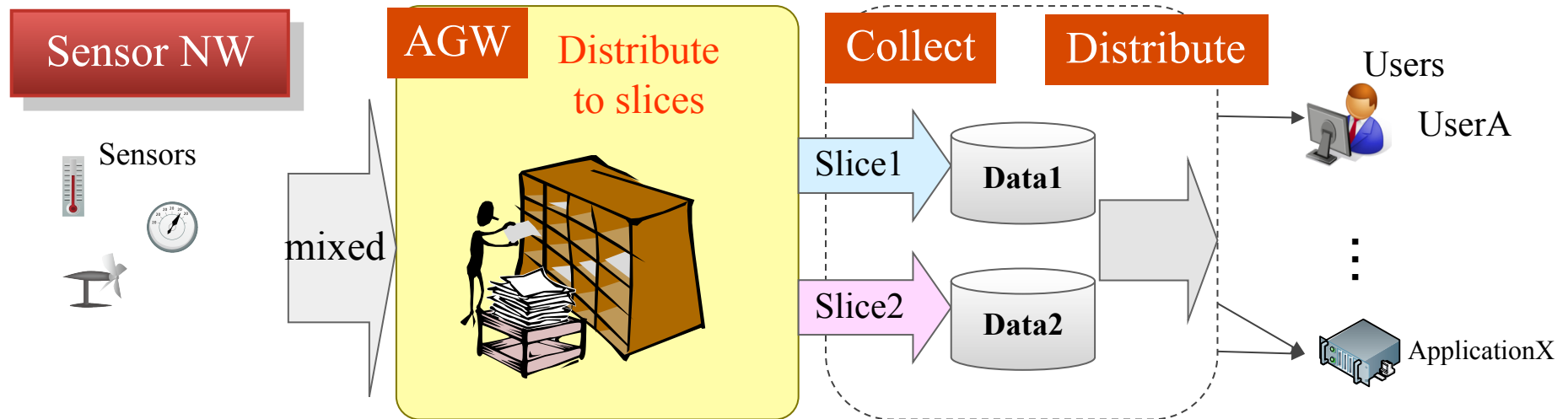
- Enabling *ephemeral multicast streaming* of videos
- Instantly deploy and shutdown *traffic splitters in NW*
- Flexcast: Non-IP protocol (planned)



Slice 3: Sensor Data In-Network Processing

Sensor Specific Network In A Slice

- Implemented NW service demo system to gather various sensor data, and distribute it after processing



- Issues
 - Massive sensor data --> Scalability
 - Add new type of sensors --> Increase development cost
 - Effect of network virtualization
 - Offload application processing into network
 - Hide sensor specific network protocols
- Users / Applications can concentrate on service processing.**

Slice 4: Stream Computing Platform

Chaining Computational Resources Along E2E Data Paths

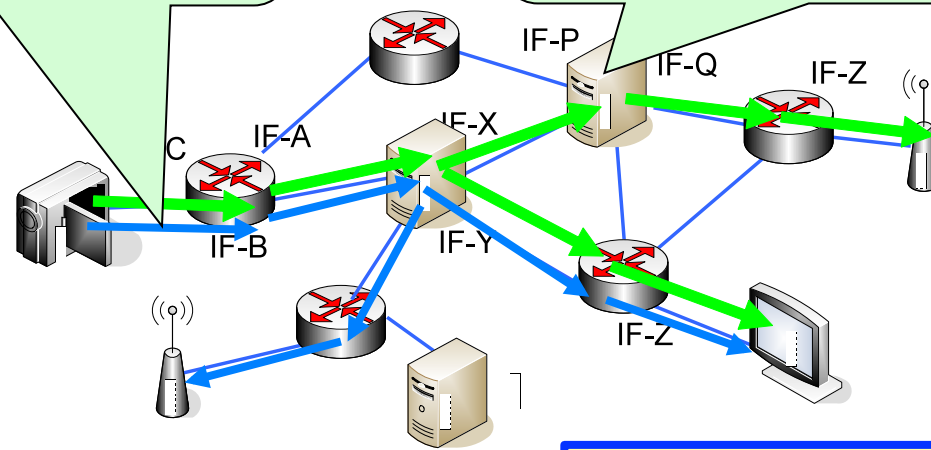
Key technologies for Stream Computing Platform

Ephemeral Multicast

Efficient method for short time and frequent Multicast

In-Network Processing

Support realtime on-the-fly processing for various data.



On VNode, we can

- Implement a original routing protocol, (Source Routing based on output INF)
- Use our own packet format (Not Ethernet or IP packet format)

On VNode, we can

- Use computing resource in network,
- Select appropriate computing resource according to the processing requirements
 - ✓ Slow-Path (VM)
 - ✓ Fast-Path (Communication Processor)

Stream Computing Platform (Demo)



Slice 5: L2/L3 Consolidation (IPEC)

IPEC* — Non-IP Protocol that Operates on the Virtualization Nodes

* IP-Ether-Chimera

• Features of IPEC

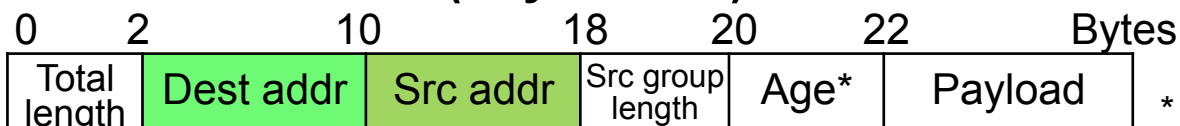
- IPEC has a *group-learning function* that is more scalable than Ethernet learning.
 - When a VNode learns a terminal (PC) in group G, it can switch packets to all the terminals in G.
 - IPEC networks are scalable because of the group-learning function — good for wide-area network.
- **Group motion is efficient in IPEC.**
 - If terminals of a group moves to the same place, each VNode learns only once.
- IPEC works well in networks with loops.
 - In contrast to Ethernet, the learning and forwarding work well even when the data path has loops.

• The address format of IPEC

- An address consists of an ID and a Group.
 - **ID: the identifier of each terminal** — similar to MAC addresses in Ethernet networks.
 - **Group: the identifier of each user group** — used for structured address or location indicator

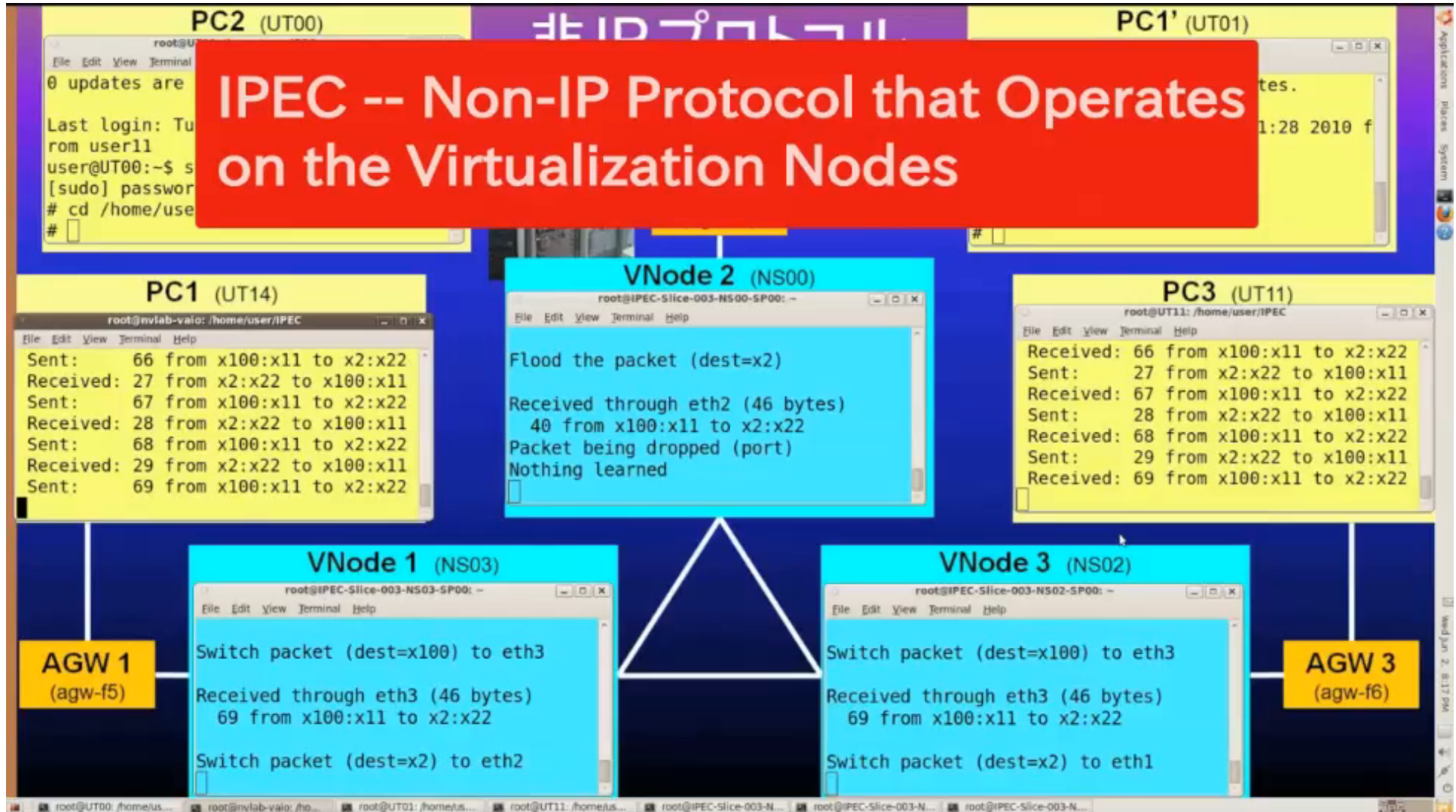


• Packet format of IPEC (any format)



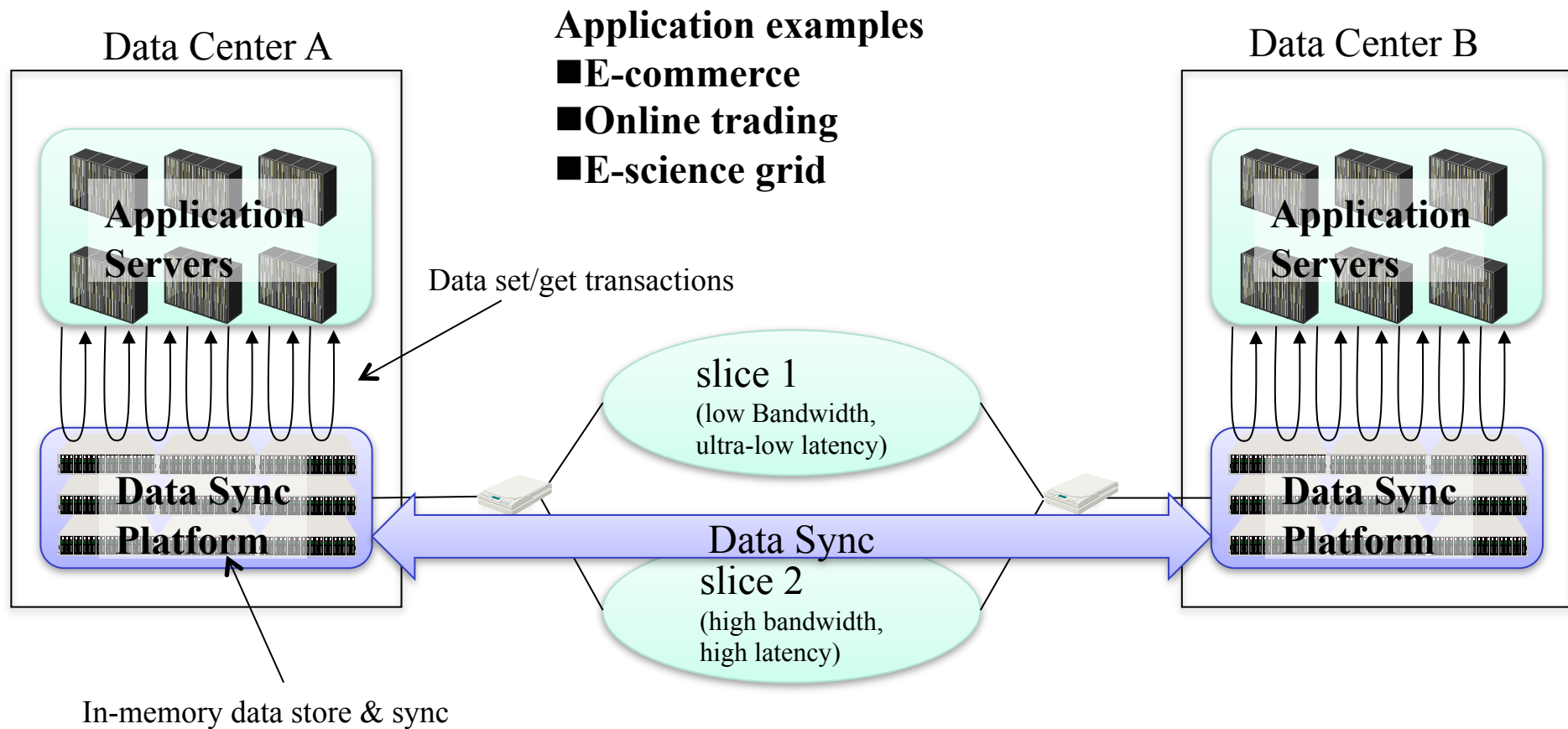
* Age is used to avoid looping.

L2/L3 Consolidation (IPEC) Demo



Slice 6: Data Sync for Distributed Data Centers

- Enables data sync between distributed data centers.
- Performs efficient data sync using various types of slices.
- Adaptive traffic control for dynamically changing sync demands. (demo)



Data Sync for Distributed Data Centers (Demo)

Priority:
App A > App B > App C

— : App A (Slice 1)
— : App B (Slice 2)
— : App C (Slice 2)

No. of transactions
(trans/sec)

set/get transaction load



Bandwidth(Mbps)

data sync traffic



VNode Deployment on JGN2Plus/JGN-X

Scheduled in the end of August 2010 (4 to 6 VNodes)



Future Directions

- ⊕ Optical Path Integration
- ⊕ Cloud Computing & Networking
- ⊕ More Applications
- ⊕ CoreLab Deployment and Integration
 - ⊕ NetFPGA (Switch Blade) Integration
 - ⊕ Multicore Network Processor Integration
- ⊕ CoreLab & VNode Federation
- ⊕ Federation (SFA)
 - ⊕ CoreLab and PlanetLab
 - ⊕ VNode and GENI (Cluster X)
- ⊕ Network Operation and Business Model

Credits

✦ NICT

- ✦ Ping Du, Maoke Chen, **Yuji Nishida**, Haruki Denpo, Ryota Ozaki, **Kiyohide Nakauchi**, et. al. (Hakusan)
- ✦ **Jin Tanaka**, **Hideki Otsuki** (JGN2Plus)

✦ NTT

- ✦ Atsushi Takahara, **Noriyuki Takahashi**, Yukio Tsukishima, **Takeshi Yamamoto** et. al.

✦ NEC

- ✦ Motoo Nishihara, Takeo Hayashi, Akihiro Motoki, Junichi Higuchi, et. al.

✦ Hitachi

- ✦ Shinji Nishimura, **Daisuke Matsubara**, Yasushi Kanada, Toshiaki Tarui, et. al.

✦ Alaxala

- ✦ Yoshiyuki, Kurosaki, Makoto Kitani, et. al.

✦ Fujitsu

- ✦ Tomohiro Ishihara, Kenichi Abiru, **Kazumine Matoba**, et. al.

People marked in red are around this room...

Conclusion

- ❊ Network Virtualization is one of the key technologies for proceeding further in defining NwGN architecture(s)
- ❊ We are turning this vision into reality..

Net-Virtualization Research Lab Contact:

nakao@iii.u-tokyo.ac.jp

info@nvlab.org

<http://www.nvlab.org>



THE UNIVERSITY OF TOKYO

NICT

National Institute of
Information and
Communications
Technology



PLANETLAB Japan 

An open platform for developing, deploying, and accessing planetary-scale services

