OpenFlow Virtualization Framework with Advanced Capabilities

Balázs Sonkoly, András Gulyás, Felicián Németh, János Czentye, Krisztián Kurucz,

Barnabás Novák, Gábor Vaszkun

Department of Telecommunications and Media Informatics

Budapest University of Technology and Economics

Email: {sonkoly,gulyas,nemethf}@tmit.bme.hu

vaszkun@gmail.com

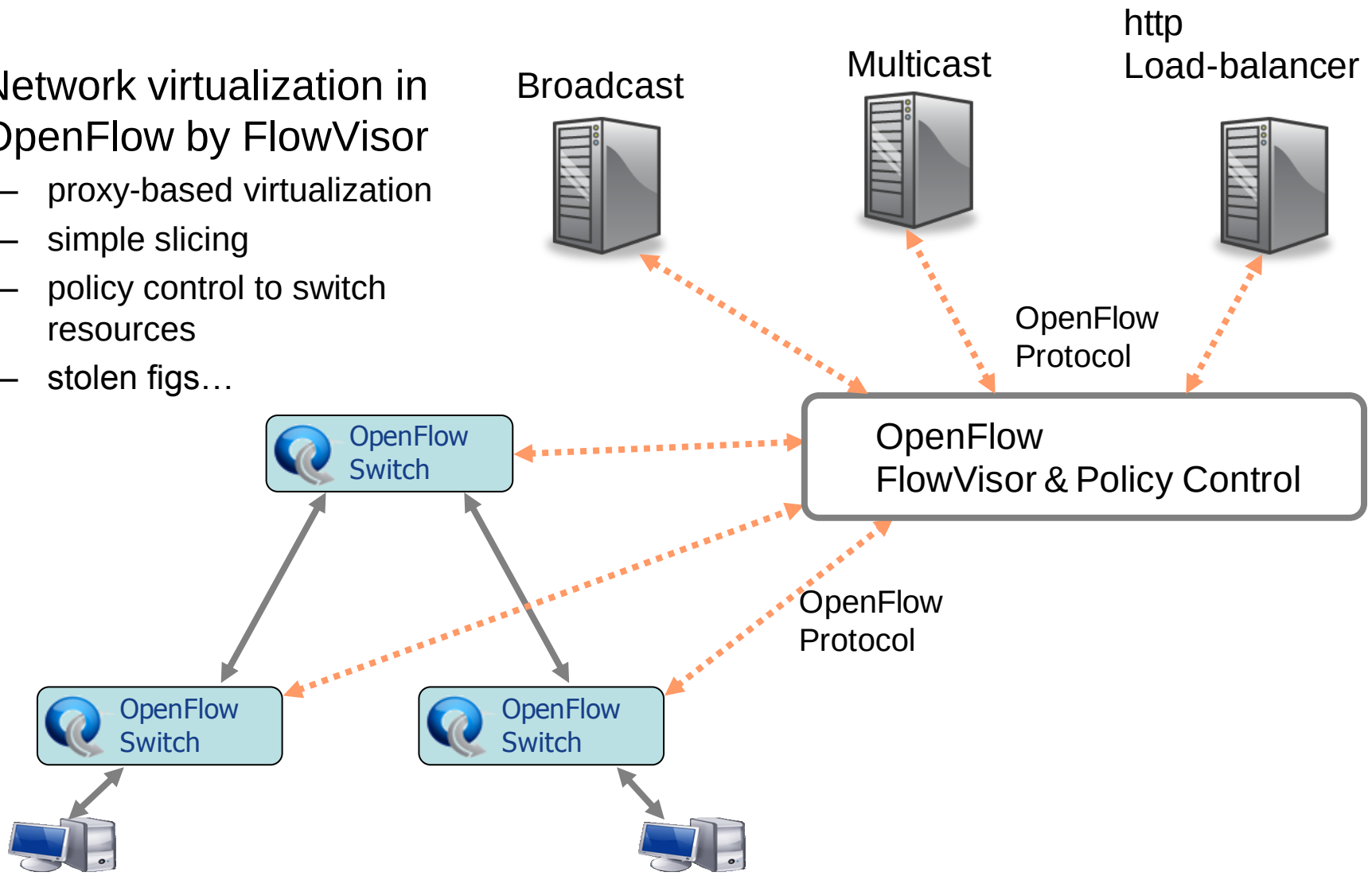
MŰEGYETEM 1782

HSNLab

www.hsnlab.hu

Background

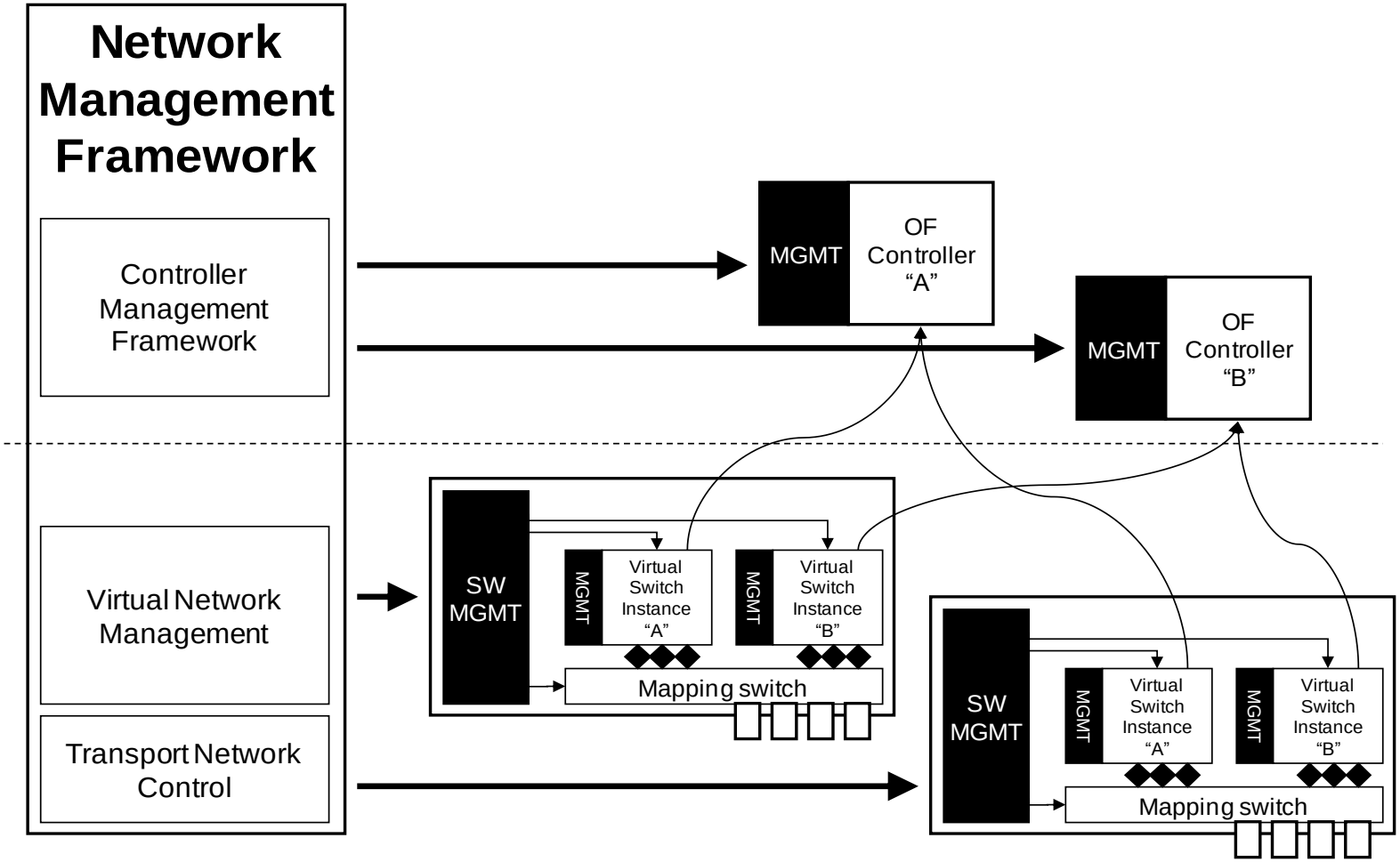
- Network virtualization in OpenFlow by FlowVisor
  - proxy-based virtualization
  - simple slicing
  - policy control to switch resources
  - stolen figs…



Motivation & Goals

- FlowVisor (proxy) based network virtualization
  - limits switch model & capabilities
  - depends on OpenFlow protocol & switches (currently v1.0)
  - FlowVisor should be modified if
    - modified matching
    - new OF messages
    - novel forwarding functions
- OpenFlow
  - OF-CONFIG out, but still has version questions
  - current tools: only running/configuring data plane & FlowVisor
- Goals: novel virtualization framework
  - add **heavy-weight virtualization** to OpenFlow
    - enhancing rapid prototyping & testing
    - maintain the physical network
    - provision virtual slices
  - support
    - **multiple** versions of **OF protocols** & **switches** (v1.0, v1.1)
    - switches with **novel forwarding** capabilities
    - different **controllers**
  - design OpenFlow management framework
    - **switch management**
    - **controller management**

Our framework



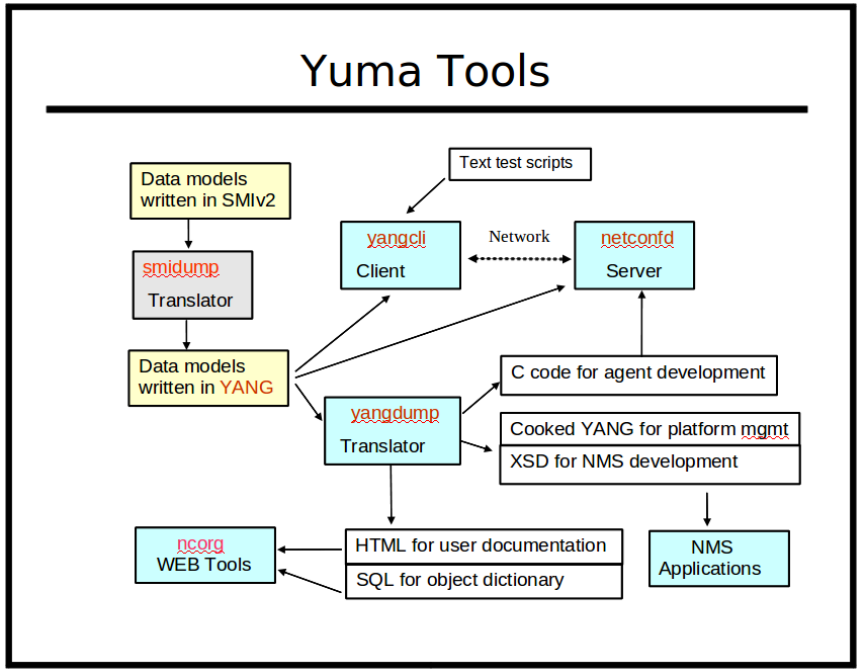
Development Environment

- Open-source NETCONF implementation
- Easy to use data modeling language – Yang

netconfd – server app

yangcli – client app

yangdump – code generator



Data model

```
container of−sys {
   list of−instance {
      leaf id { }
      container config {
         // b a s i c p a r a m e t e r s
         leaf status { }
         leaf dp−pid { }
         leaf of−pid { }
         leaf ofdatapath−path { }
         leaf ofprotocol−path { }
         leaf−list eth−if { }
         . . .
         / / c o n t r o l l e r p a r a m e t e r s
         leaf controller−address { }
         leaf controller−port { }
      }
      container datapath {
         / / OF s w i t c h p a r a m e t e r s
         leaf dp_desc { }
         . . .
      }
   }
}
```

Configuration example:

```
of−instance 2 {
  id 2
  config {
    desc  ’1.0 of switch 1’
    status running
    dp−pid 1505
    of−pid 1552
    ofdatapath−path /usr/bin/of10/ofdatapath
    ofprotocol−path /usr/bin/of10/ofprotocol
    dp-of-socket /tmp/dp2.sock
    eth−if eth1
    eth−if eth2
    controller−address 192.168.213.230
    controller−port 6635
    controller-type tcp
    …
  }
}
```

QoS extension

```
rpc set−queue {
        input {
                leaf instance { }
                leaf q_id { }
                leaf bandwidth { }
                leaf port { }
} }
rpc get −queue {
        input { leaf instance { } }
        output {
                list queues {
                leaf port { }
                leaf q_id { }
                leaf bandwidth { } }
} }
rpc delete−queue {
        input {
                leaf instance { }
                leaf port { }
                leaf q_id { }
} }
```

- RPC-s to handle basic QoS queue creation and removal
- OF switches up to version 1.1 only supports min-rate
- Controller app capable to use the defined queues is under development

Integration with NMS

- Open-source network management system
- Easy to use surface
- Integration with Yuma's client program, **yangcli**

Active connections

Name	IP address	Username	Supported Modules	Disconnect
TP-Link Device2	192.168.213.32	root	Show	Disconnect
TP-Link Device1	192.168.213.31	root	Show	Disconnect
NOX Controllers	192.168.213.230	openflow	Show	Disconnect
TP-Link Device4	192.168.213.34	root	Show	Disconnect
TP-Link Device3	192.168.213.33	root	Show	Disconnect

Create OpenFlow Switch Instance

Id: (Unique identifier of configured OFswitch-instance.) Version: 1.0

OFdatapath:

OFprotocol:

Dpctl-path:

Interfaces: (Min 2 interface is required: eth[, eth, ...])

DP-OF socket: (Must be unique too!)

Connection type: tcp (Type of the connection between switch and controller.)

Controller address:

Port number:

Create OFswitch instance

Virtual ethernet pair

Number of veth:

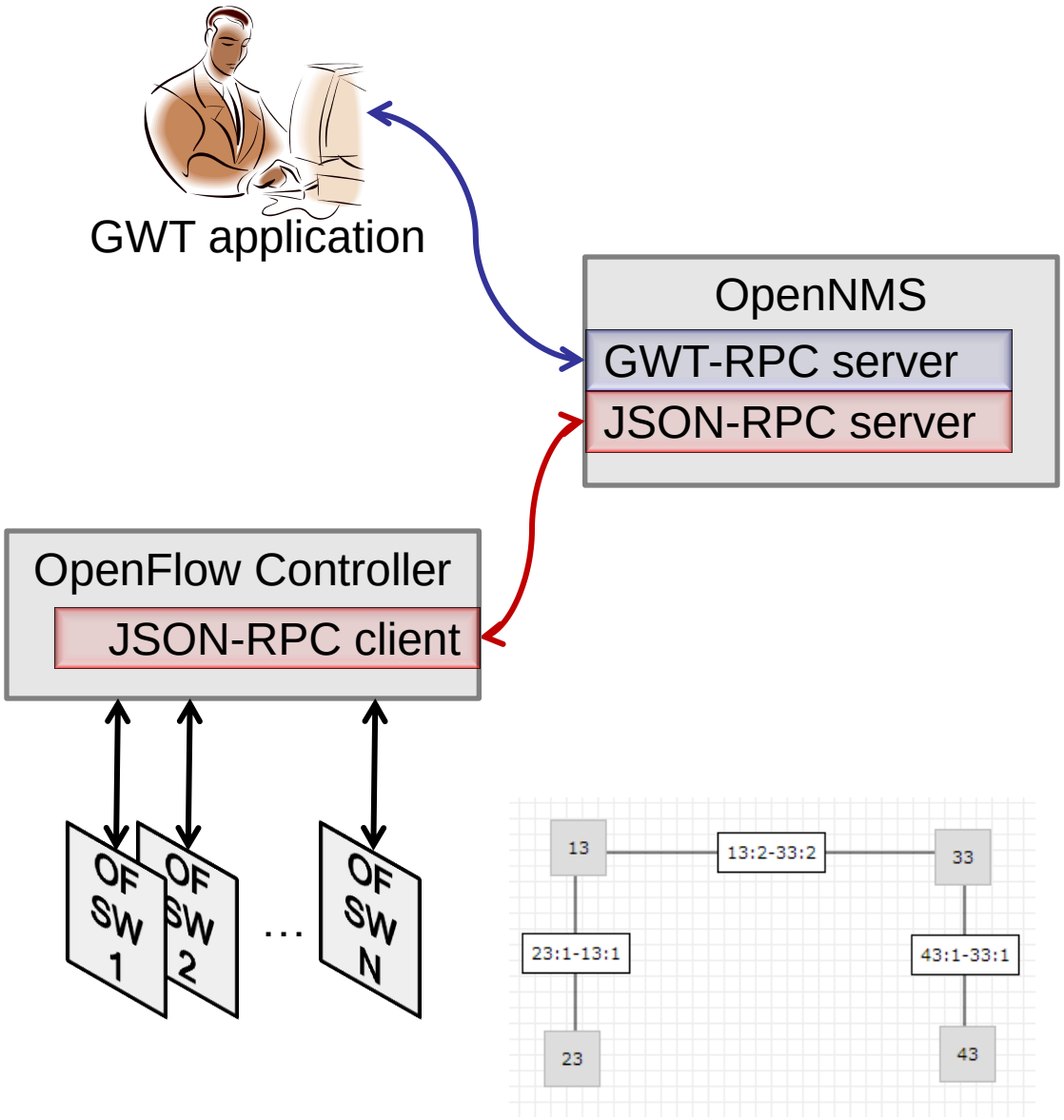
1 pair

Create

OpenFlow Switches

ID	Version	Netconf comp.	Interfaces	Controller	Port	Status	Delete	Start/Stop
2	1.0	false	veth1, veth3, veth5, veth7	192.168.213.230	6635	running	Delete	Stop
3	1.0	false	veth11, veth13, veth15, veth9	192.168.213.230	6634	stopped	Delete	Start
1	1.1	false	eth0.1, eth0.2, eth0.3, eth0.4, veth0, veth10, veth12, veth14, veth2, veth4, veth6, veth8	0.0.0.0	6633	running	Delete	Stop

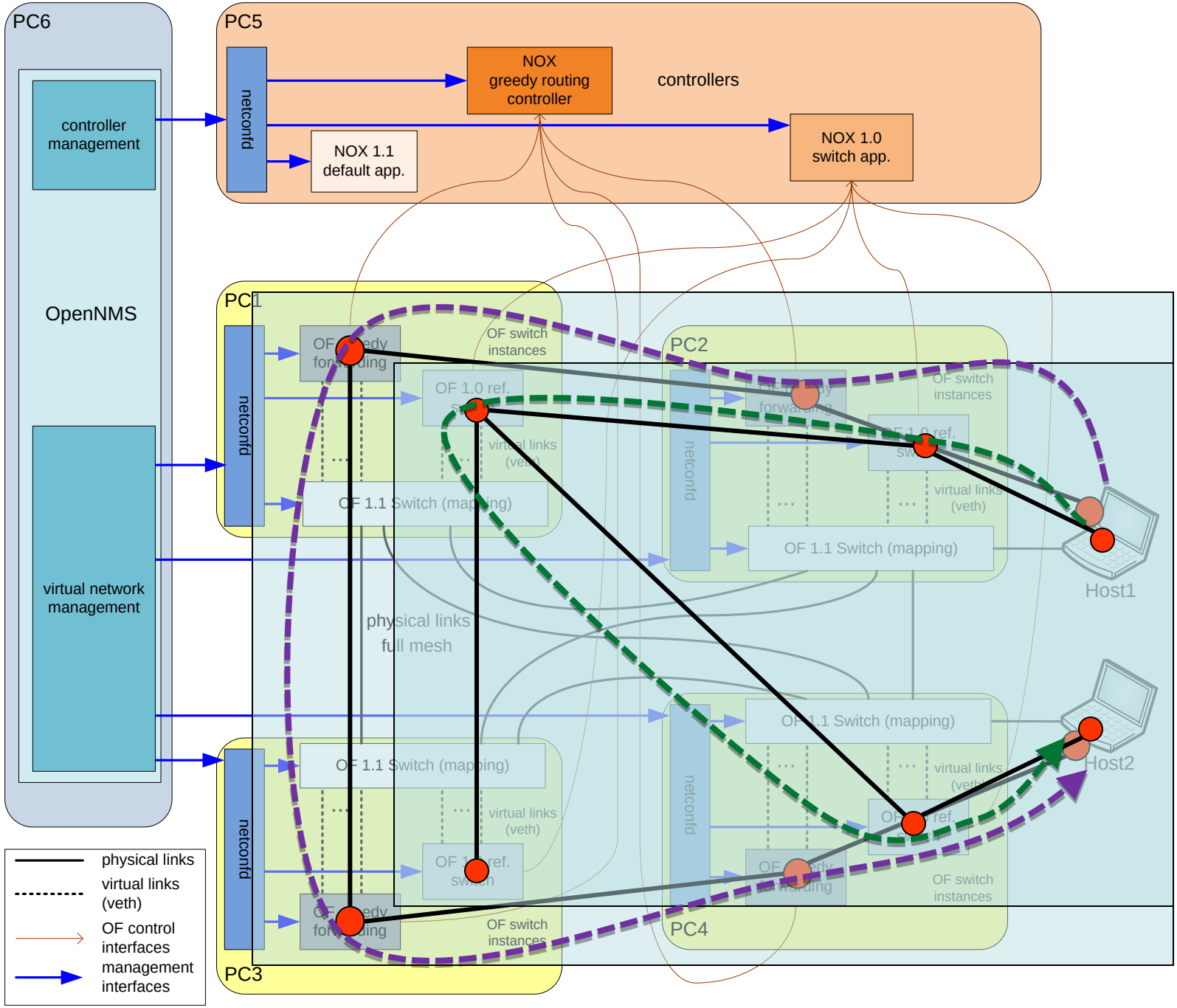
Topology discovery module



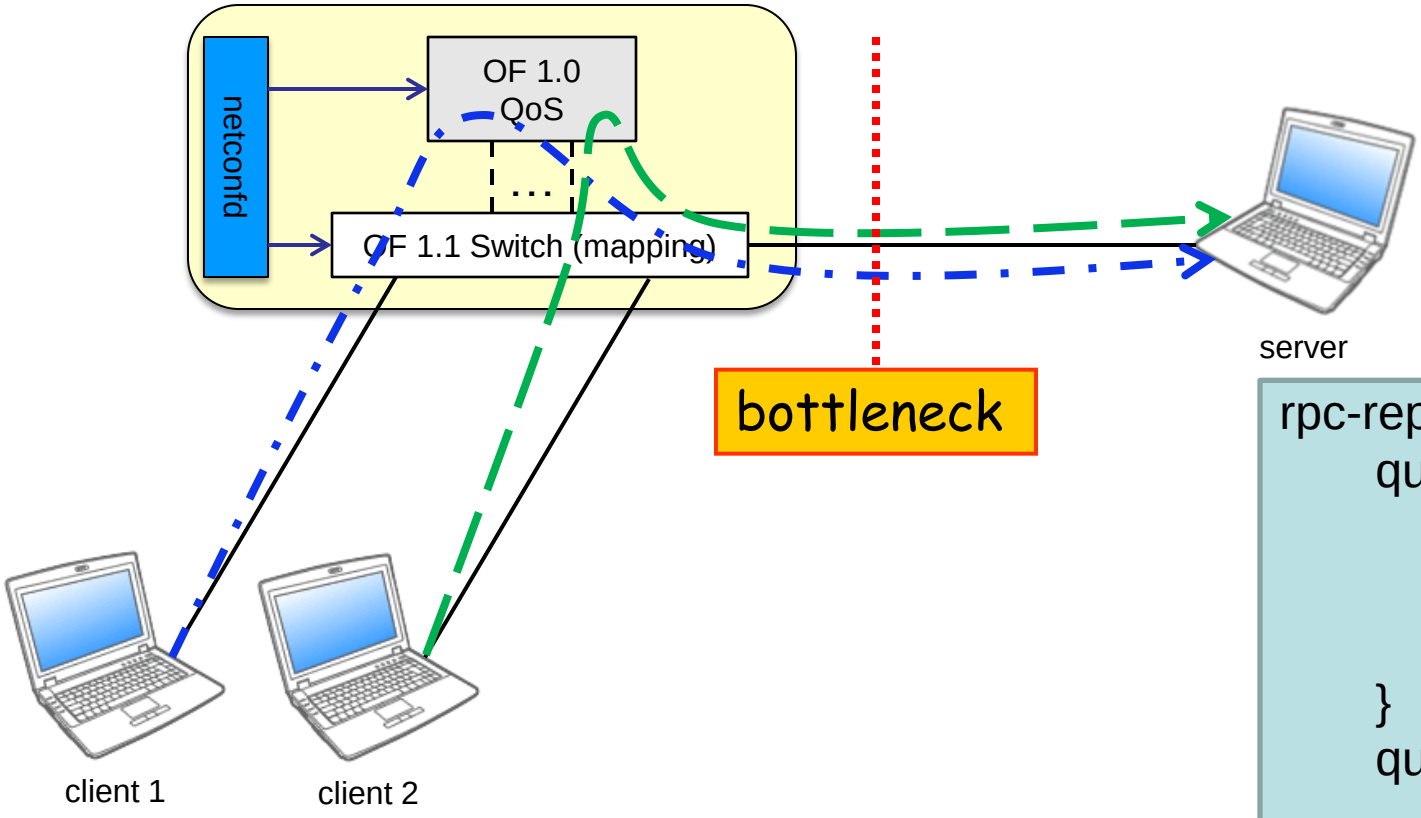
- Integrated with OpenNMS
- OpenNMS and OF controller communicates through HTTP
- Topology viewer uses Google Web Toolkit
- Discovery module based on the NOX controllers discovery app

OpenWrt implementation

- Framework is compilable on SOHO routers
  - Device needs an open operating system, such as OpenWrt
  - Cross-compiling needed
- Software based OF Ref. Switch's performance still an issue
  - Running multiple switch instaces can degrade a Gbits/s interface's performance to 50 Mbits/s or even further


PC6
controller management
OpenNMS
virtual network management
PC5
netconfd
NOX greedy routing controller
controllers
NOX 1.1 default app.
NOX 1.0 switch app.
PC1
PC2
PC3
PC4
OF switch instances
OF 1.0 ref. switch
OF 1.1 Switch (mapping)
virtual links (veth)
physical links full mesh
Host1
Host2
physical links
virtual links (veth)
OF control interfaces
management interfaces

QoS demo traffic




```
rpc-reply {
        queues 3 1 {
                port 3
                q_id 1
                bandwidth 6
        }
        queues 3 2 {
                port 3
                q_id 2
                bandwidth 4
        }
}
```

virtual links
10 Mbits/s physical links
6 Mbits/s min rate TCP traffic
4 Mbits/s min rate UDP traffic

Conclusions & Future works

- We defined a virtualization framework which could replace the FlowVisor structure
- Extension of the framework with QoS
- Visualization of each virtual/physical topology
- Ongoing/Future works:
  - NOX controller applications:
    - Dynamic use of QoS queues
    - Dynamic mapping switch configuration
  - OF-CONFIG 1.0 and 1.1 implementation for software based reference switches with the help of Yuma-tools