CMPE 209
Network Function Virtualization

NFV II

Dr. Younghee Park

SAN JOSÉ STATE
UNIVERSITY

Network functions/middleboxes


Firewall


Caching Proxy


Intrusion Prevention

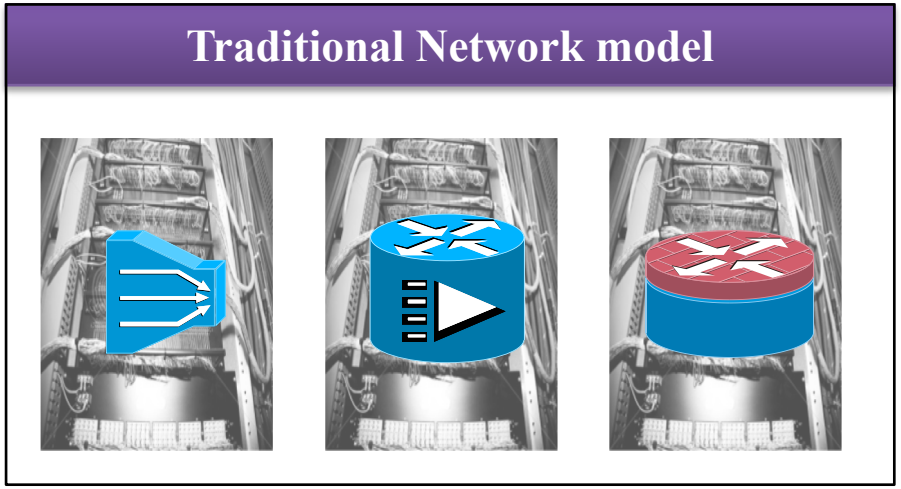

Traffic scrubber


Load balancer


SSL Gateway


WAN optimizer

...



- Network functionalities are **based on specific HW&SW**
- **One physical node per role**

Problem Statement

- Complex carrier networks
  - with a large variety of proprietary nodes and hardware appliances.
- Launching new services is difficult and take long
  - Space and power to accommodate
  - requires just another variety of boxes, which needs to be integrated.
- Operation is expensive
  - Rapidly reach end of life
  - due to existing procure-design, integrate-deploy cycle.

Network Function Virtualization (NFV)

- A mean to make the network more flexible and simple by minimizing dependence on HW constraints

•Traditional Network Model:
•APPLIANCE APPROACH


DPI


BRAS


GGSN/SGSN


PE Router

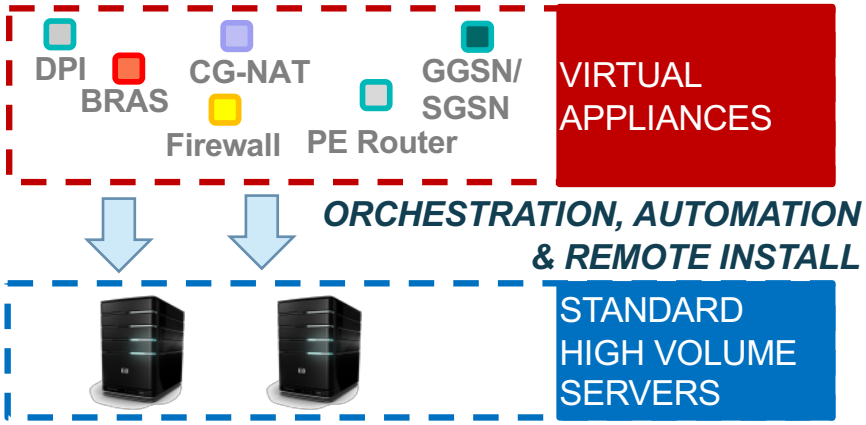

Firewall


CG-NAT


Session Border Controller

- Network Functions are **based on specific HW&SW**
- **One physical node per role**

•Virtualised Network Model:
•VIRTUAL APPLIANCE APPROACH



- Network Functions are **SW-based over well-known HW**
- **Multiple roles** over **same HW**

Network Functions Virtualization

- Network Functions Virtualization is **about implementing network functions in software** - that today run on proprietary hardware - leveraging (high volume) standard servers and IT virtualization
- Supports multi-versioning and multi-tenancy of network functions, which allows use of a single physical platform for different applications, users and tenants
- Allows use of a single physical platform for different applications, users and tenants
- Facilitates innovation towards new network functions and services that are only practical in a pure software network environment

Benefits & Promises of NFV

- Reduced equipment costs (CAPEX)
  - through consolidating equipment and economies of scale of IT industry.
- Increased speed of time to market
  - by minimising the typical network operator cycle of innovation.
- Availability of network appliance multi-version and multi-tenancy,
  - allows a single platform for different applications, users and tenants.
- Enables a variety of eco-systems and encourages openness.
- Encouraging innovation to bring new services and generate new revenue streams.

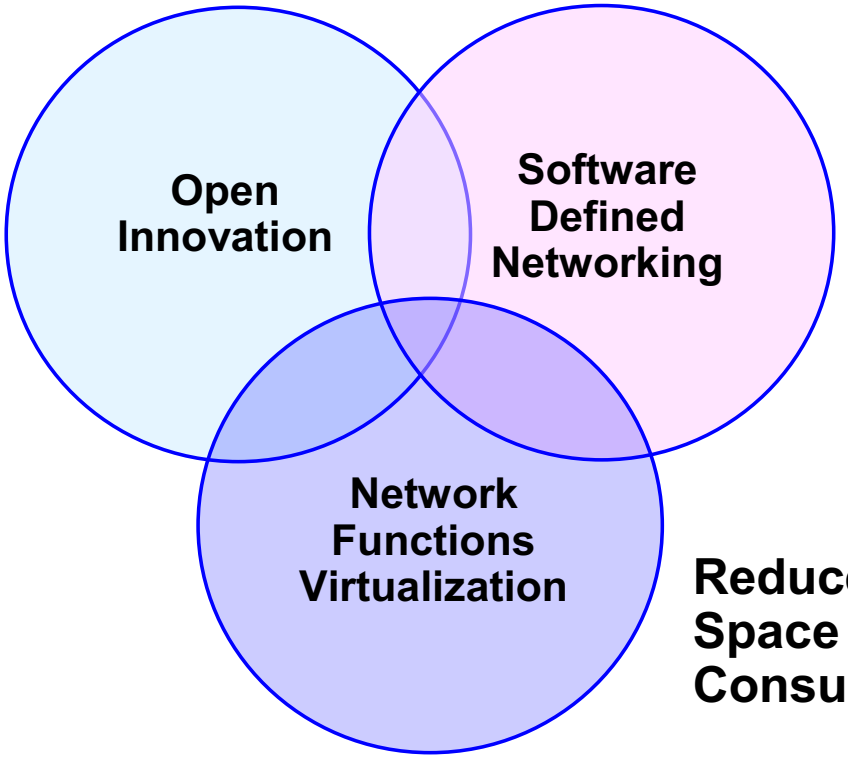
So, why we need/want NFV(/SDN)?

1. Virtualization: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. Orchestration: Manage thousands of devices
3. Programmable: Should be able to change behavior on the fly.
4. Dynamic Scaling: Should be able to change size, quantity
5. Automation
6. Visibility: Monitor resources, connectivity
7. Performance: Optimize network device utilization
8. Multi-tenancy
9. Service Integration
10. Openness: Full choice of modular plug-ins

NFV and SDN

- NFV and SDN are highly complementary
- Both topics are mutually beneficial but not dependent on each other

Creates competitive supply of innovative applications by third parties



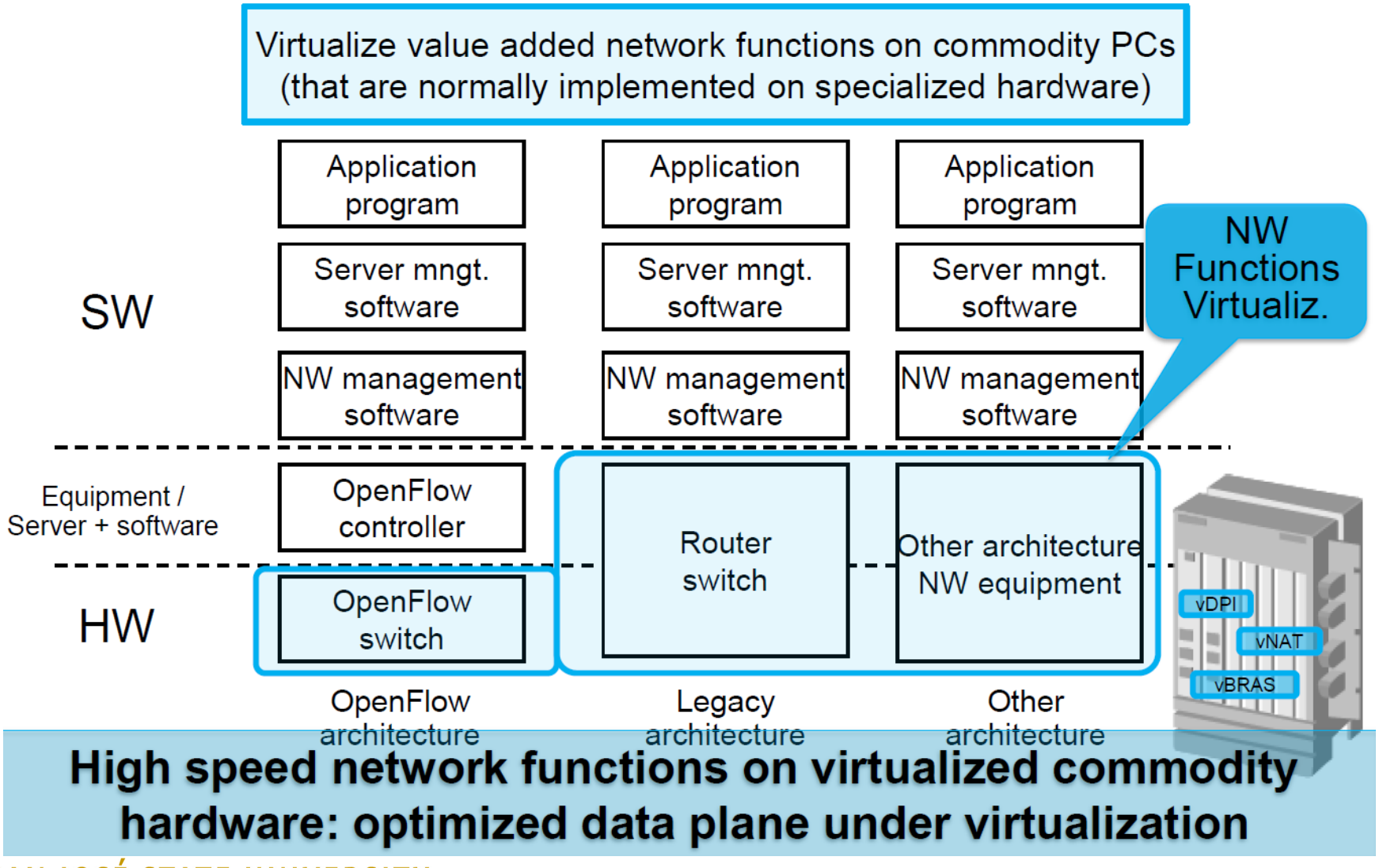
Creates network abstractions to enable faster innovation

Reduces CAPEX, OPEX, Space & Power Consumption

NFV vs SDN

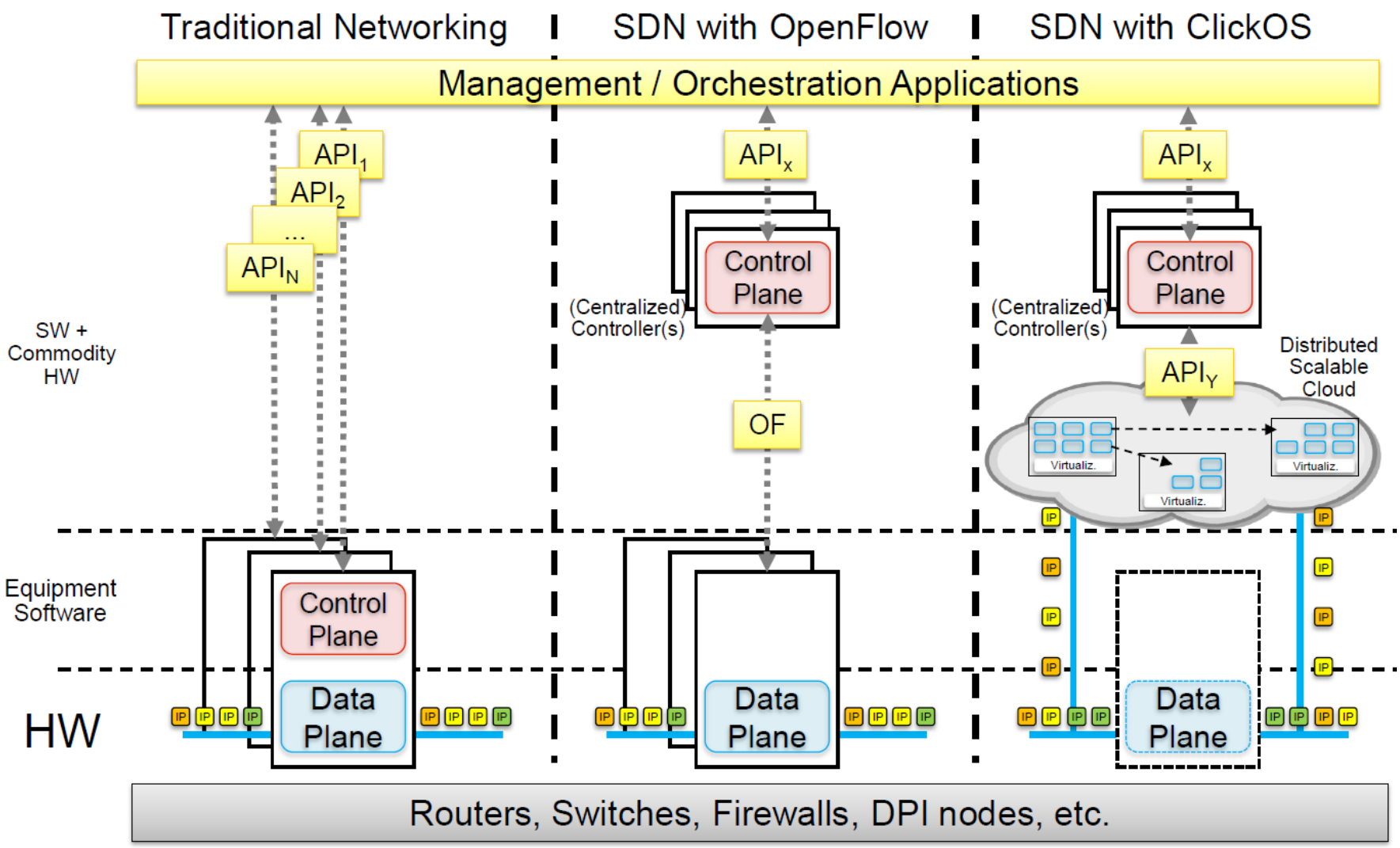
- NFV: re-definition of **network function equipment architecture**
- NFV was born to meet Service Provider (SP) needs:
  - Lower CAPEX by reducing/eliminating proprietary hardware
  - Consolidate multiple network functions onto industry standard platforms
- SDN: re-definition of **network switch/router architecture**
- SDN comes from the IT world:
  - Separate the data and control layers, while centralizing the control
  - Deliver the ability to program network behavior using well-defined interfaces

Scope of NFV and OpenFlow/SDN



Source: NEC

Networking with SDN & NFV

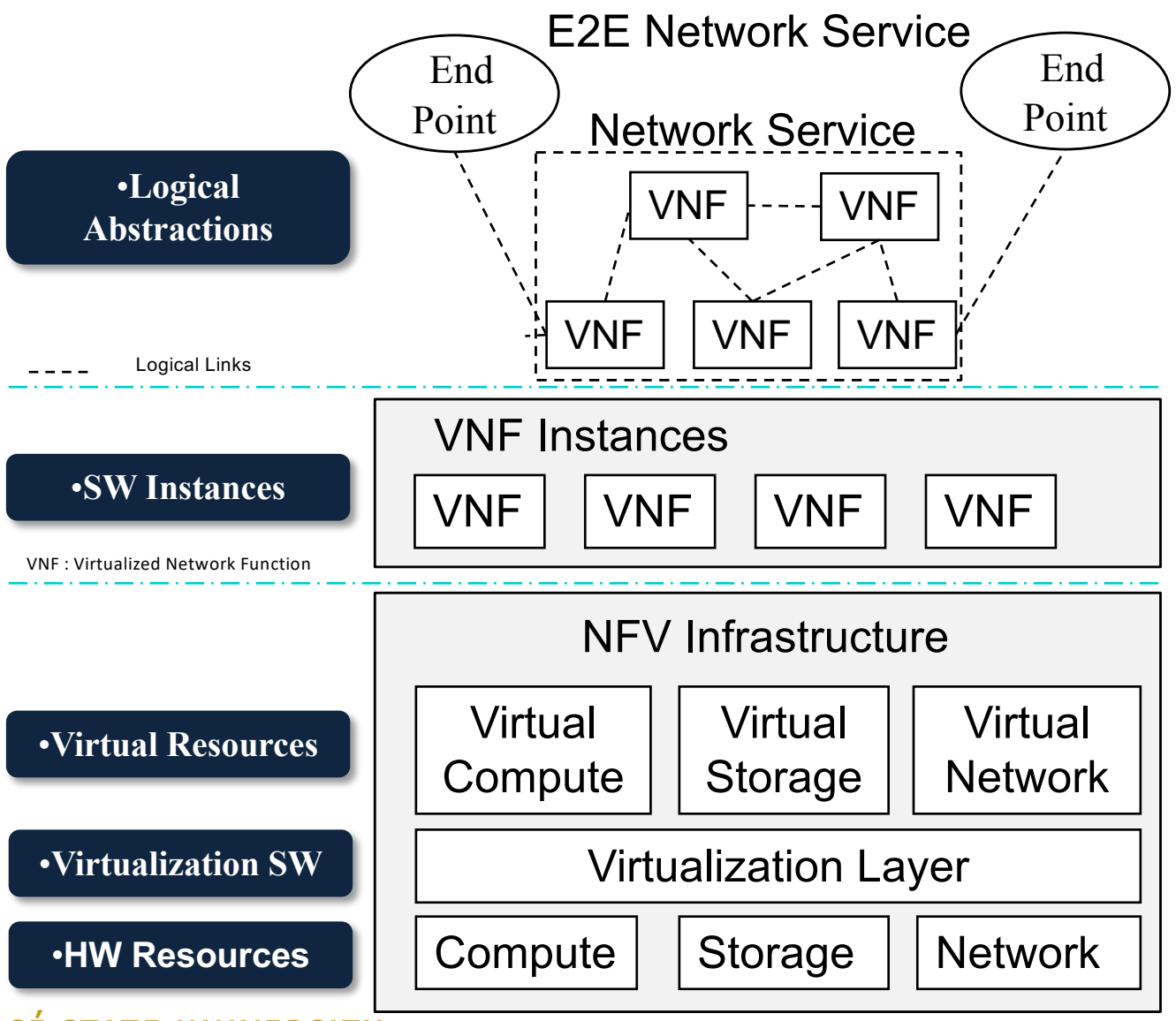


Some Use Case Examples

…not in any particular order

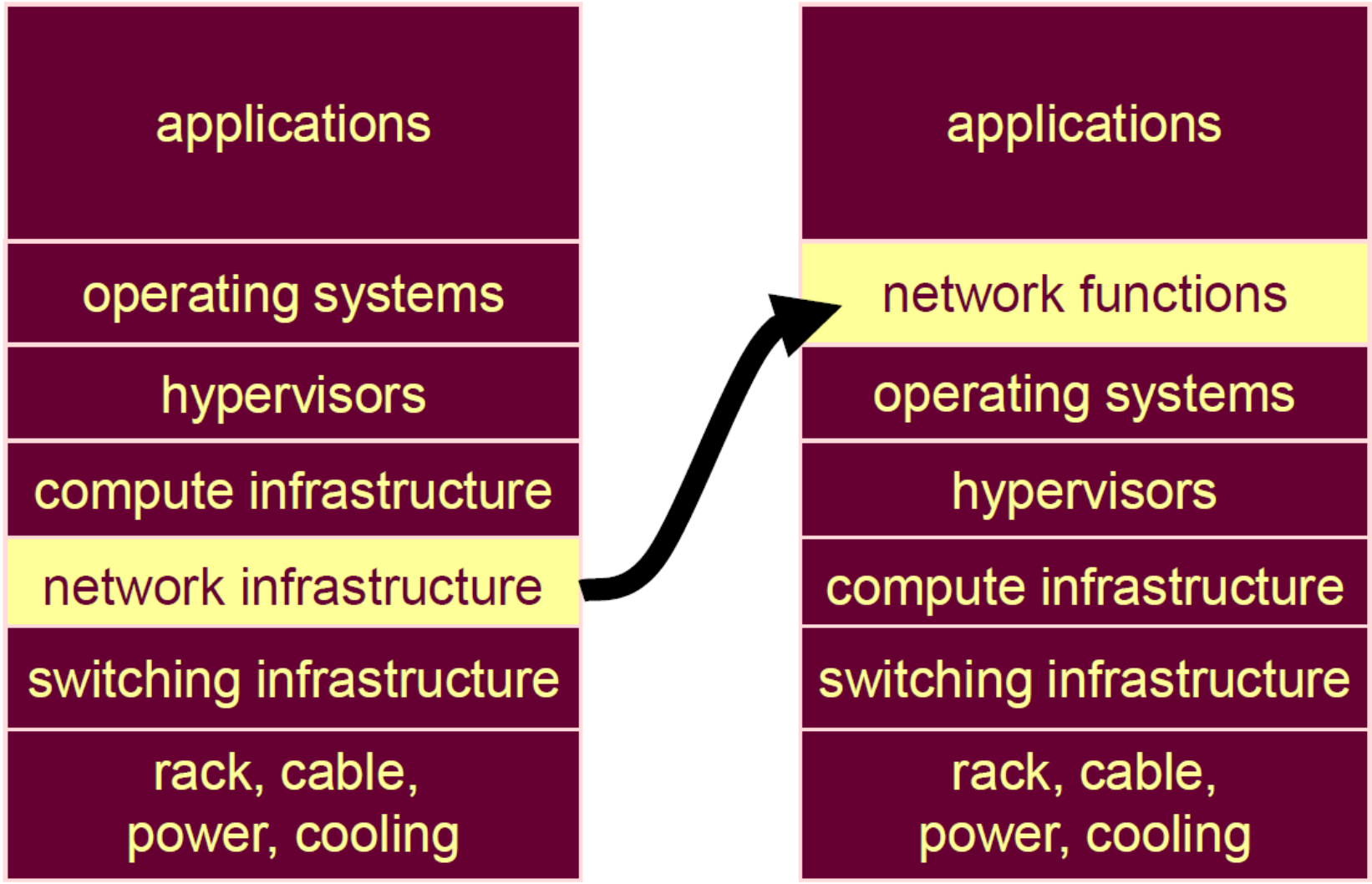
- Switching elements
- Mobile network nodes
- Home networks: Functions contained in home routers and set top boxes to create virtualised home environments.
- Tunnelling gateway elements: IPSec/SSL VPN gateways.
- Traffic analysis: DPI, QoE measurement.
- Service Assurance: SLA monitoring, Test and Diagnostics.
- Converged and network-wide functions: AAA servers, policy control and charging platforms.
- Application-level optimisation: CDNs, Cache Servers, Load Balancers, Application Accelerators.
- Security functions: Firewalls, virus scanners, intrusion detection systems, spam protection.

NFV Layers



Source: Adapted from D. Lopez Telefonica I+D, NFV

Rethinking relayering



A Few Challenges

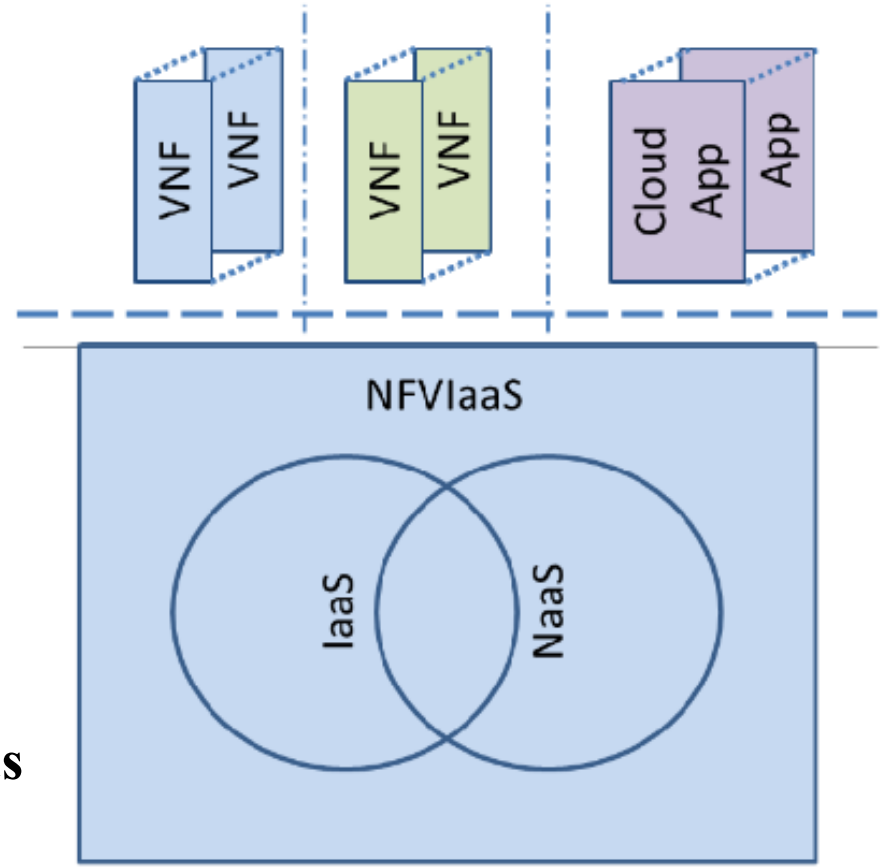
- Achieving high performance virtualised network appliances
  - portable between different HW vendors, and with different hypervisors.
- Co-existence with bespoke HW based network platforms
  - enabling efficient migration paths to fully virtualised network platforms.
- Management and orchestration of virtual network appliances
  - ensuring security from attack and misconfiguration.
- NFV will only scale if all of the functions can be automated.
- Appropriate level of resilience to HW and SW failures.
- Integrating multiple virtual appliances from different vendors.
  - Network operators need to be able to "mix & match" HW,
  - hypervisors and virtual appliances from different vendors,
  - without incurring significant integration costs and avoiding lock-in.
- NFV and SDN

NFV Infrastructure as a Service (NFVIaaS)

NFV Infrastructure :

- provide the capability or functionality of providing an environment in which Virtualized network functions (VNF) can execute

- **NFVIaaS** provides compute capabilities comparable to an **IaaS cloud computing service** as a run time execution environment **as well as support the dynamic network connectivity services** that may be considered as comparable to **NaaS**



NFV Summary

- Network functions virtualization (NFV) is an initiative to virtualize the network services that are now being carried out by dedicated hardware.
- The goal of NFV is to decouple network functions from dedicated hardware devices and allow network services that are now being carried out by routers, firewalls, load balancers and other dedicated hardware devices to be hosted on virtual machines (VMs).
- Once the network functions are under the control of a hypervisor, the services that once require dedicated hardware can be performed on standard x86 servers.

CLEMSON UNIVERSITY

ClickOS and the Art of Network function Virtualization

Joao Martins*, Mohamed Ahmed*, Costin Raiciu¨, Roberto Bifulco*, Vladimir Olteanu¨, Michio Honda*, Felipe Huici*

* NEC Labs Europe, Heidelberg, Germany

¨ University Politehnica of Bucharest

Shifting Middlebox Processing to Software

- Can share the same hardware across multiple users/tenants
- Reduced equipment/power costs through consolidation
- But can it be built using commodity hardware while still achieving high performance?
- ClickOS: tiny Xen-based virtual machine that runs Click

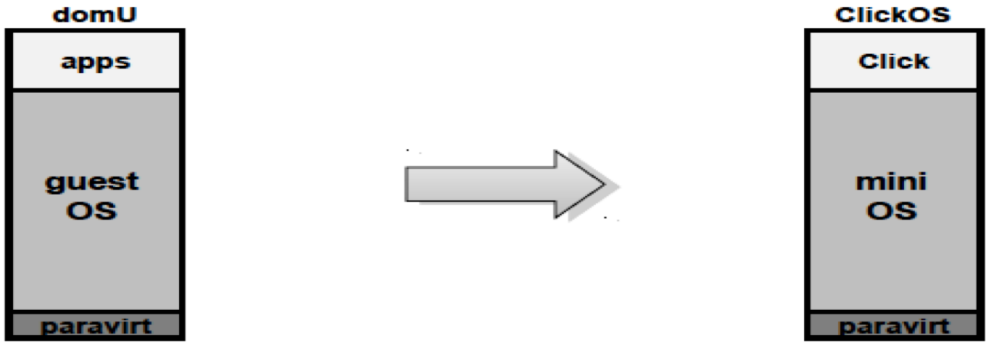
From Thought to Reality - Requirements

	ClickOS
Fast Instantiation	30 msec boot times
Small footprint	5MB when running
Isolation	provided by Xen
Performance	10Gb/s line rate* 45 µsec delay
Flexibility	provided by Click

ClickOS

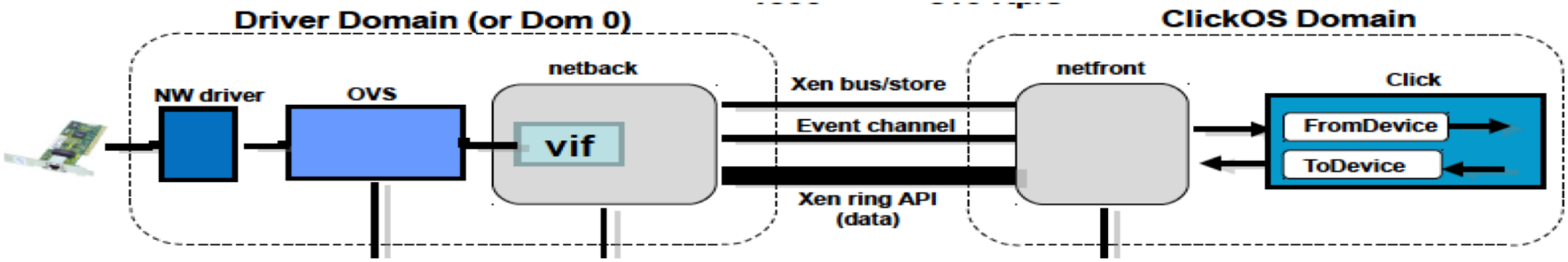
- As part of NFV Operating System family we introduce ClickOS, a high-performance, virtualized software middlebox platform.
- NEC has developed Click OS, a Xen-based open source platform.
- ClickOS virtual machines are small (5MB),
- Boot quickly (about 30 milliseconds),
- Add little delay in packet processing (45 microseconds)
- Over one hundred of them can be concurrently run while saturating a 10Gb pipe on a commodity server.
- We further implement a wide range of middleboxes including a firewall and show that ClickOS can handle packets in the millions per second.

ClickOS



Work consisted of:

- Build system to create ClickOS images (5 MB in size)
- Emulating a Click control plane over MiniOS/Xen
- Reducing boot times (roughly 30 milliseconds)
- Optimizations to the data plane (10 Gb/s for almost all pkt sizes)
- Implementation of a wide range of middleboxes



Internal architecture of ClickOS

- ClickOS is a minios based Virtual Operating System that runs on top of Virtual Platforms like Xen Hypervisor.It simulates a simple OS that is configured, started and destroyed dynamically; just by pressing few commands.
- ClickOS is really small and consumes very less resources. We just need to create a new xen domain using a xen configuration file, and then use cosmos to run a ClickOS element inside this new xen domain. These simple few steps allow us to simulate a OS /domain that acts as an alternative to a complete Network Function, namely firewall, load balancer etc.
- ClickOS is very powerful in simulating Network Function Virtualization (NFV) elements.
- ClickOS works with Open vSwitch, Linux bridge, and other soft-switches.

Click

- The Click language is used to describe Click router configurations. Two fundamental statements suffice to describe any router:
  - declarations introduce click elements
    - name::class(config);
  - connections define packet flow between them
    - Name1 [port1] -> [port2] name2;
- Click element: a set of predefined and customized actions executed on incoming packets to the ClickOS VM.

An Example

- Lab: (1) create a network element who virtual-interface is connected to the eth port of the VM, (2) assign an IP address to the network element, and (3) Ping the network element.

STEP 1:
Go to Super user mode: 'sudo -i'

Create the file icmp.xen using vi editor and save it. Create it in /home/ubuntu/cmpe210_clickos_setup

- **icmp.xen :**

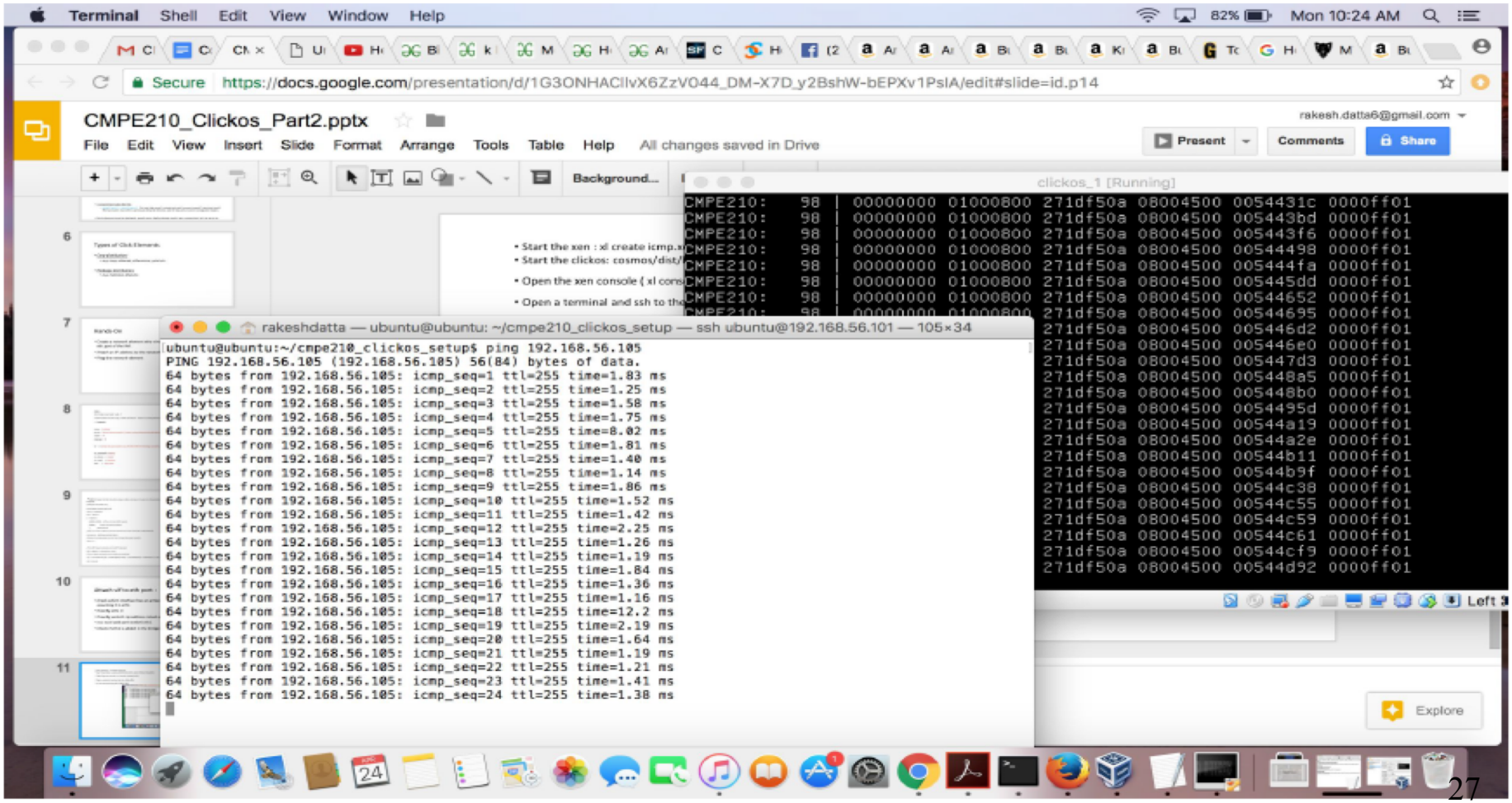
```
name   = 'clickos'
kernel = '/home/ubuntu/cmpe210_clickos_setup/clickos/minios/build/clickos_x86_64'
vcpus  = '1'
memory = '8'

vif    = ['script=vif-openvswitch,mac=00:00:00:00:01:00,bridge=xenbr0']

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'preserve'
click       = 'icmp.click'
```

An Example (icmp.click)

```
define($IP 192.168.56.105)
define($MAC 00:00:00:00:01:00)
source :: FromDevice
dest :: ToDevice
c :: Classifier(
   12/0806 20/0001,   //This is to match ARP requests
   12/0800,           //This is to match IP packets
   -);                //Dedault case
//This is to create an alias for the click element that creates and sends an ARP response
arpresponse :: ARPResponder($IP $MAC)
//Receive incoming packets and run them through the packer classifier
source -> c

//If an ARP request received, send an ARP reply back
c[0] -> ARPPrint -> arpresponse ->dest;
//If an IP packet received, send an ICMP echo reply back
c[1] -> CheckIPHeader(14) -> ICMPPingResponder() -> Print('CMPE210') -> EtherMirror() -> dest;
c[2] -> Discard;
```

An Example

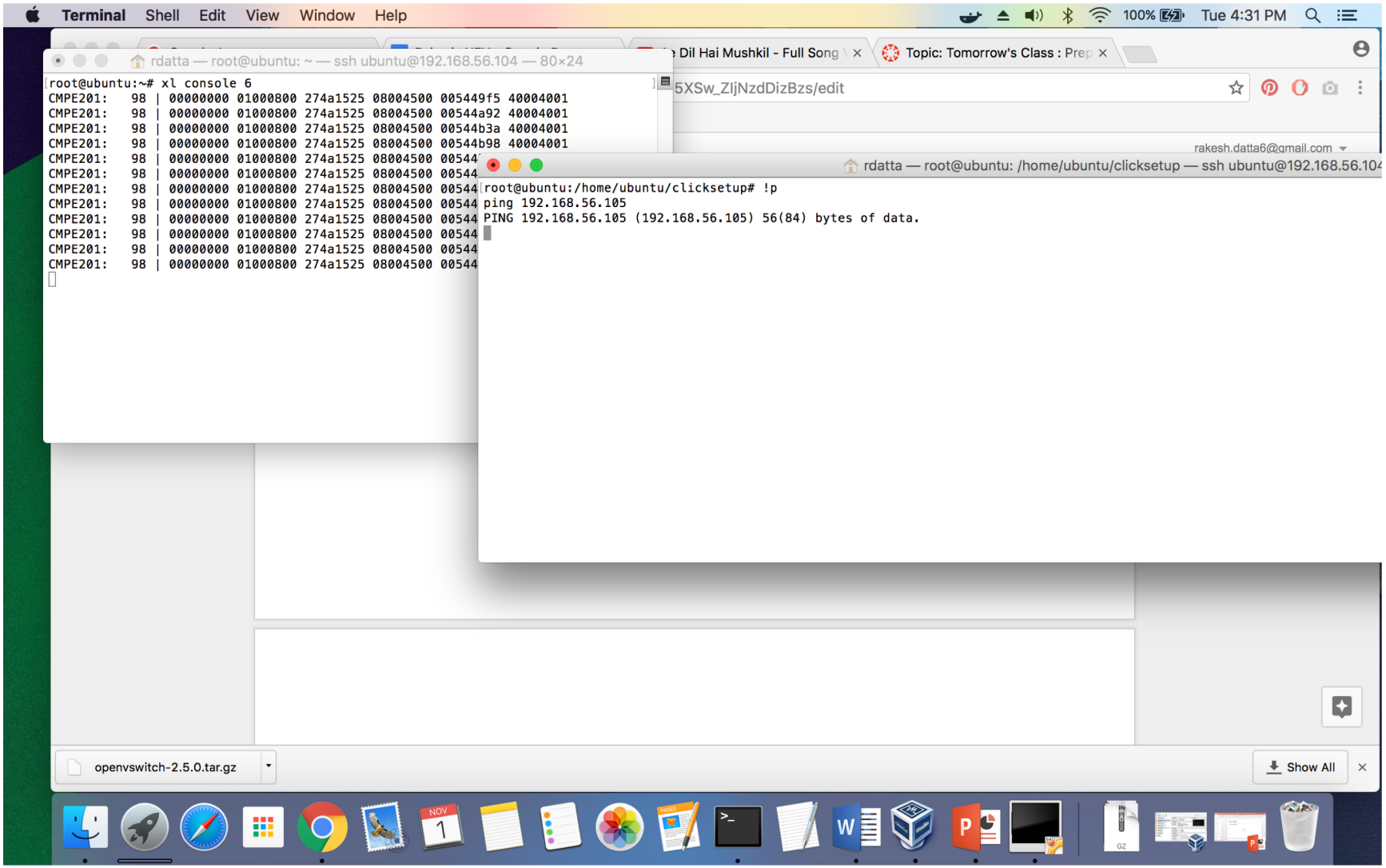
- Start the xen : xl create icmp.xen
- Start the clickos: cosmos/dist/bin/cosmos start clickos icmp.click
- Open the xen console (xl console <domain id>)
- Open a terminal and ssh to the clickos VM
- In the terminal 'ping 192.168.56.105'



- example2.click

```
// define the IP address of this network function.
// This IP must NOT be the same with your VM's IP.
    define($IP 192.168.56.105);

// This MAC must be identical with the one you defined
in the Xen configuratio file.

    define($MAC 00:00:00:00:01:00);

    source :: FromDevice;
    sink   :: ToDevice;

    source -> Print('CMPE201') -> EtherMirror -> sink;
```

Demo for example 2



Click Elements Directory Structure

```
root@ubuntu:/home/ubuntu/cmpe210_clickos_setup/clickos/elements# ls -al
total 104
drwx--x--x 26 264787 10000 4096 May 13  2015 .
drwx--x--x 20 264787 10000 4096 May 13  2015 ..
drwx--x--x  2 264787 10000 4096 May 13  2015 analysis
drwx--x--x  2 264787 10000 4096 May 13  2015 app
drwx--x--x  2 264787 10000 4096 May 13  2015 aqm
drwx--x--x  2 264787 10000 4096 May 13  2015 bsdmodule
drwx--x--x  2 264787 10000 4096 May 13  2015 ethernet
drwx--x--x  2 264787 10000 4096 May 13  2015 etherswitch
drwx--x--x  2 264787 10000 4096 May 13  2015 grid
drwx--x--x  2 264787 10000 4096 May 13  2015 icmp
drwx--x--x  2 264787 10000 4096 May 13  2015 ip
drwx--x--x  2 264787 10000 4096 May 13  2015 ip6
drwx--x--x  2 264787 10000 4096 May 13  2015 ipsec
drwx--x--x  2 264787 10000 4096 May 13  2015 json
drwx--x--x  2 264787 10000 4096 May 13  2015 linuxmodule
drwx--x--x  2 264787 10000 4096 May 13  2015 local
drwx--x--x  2 264787 10000 4096 May 13  2015 minios
drwx--x--x  2 264787 10000 4096 May 13  2015 ns
drwx--x--x  2 264787 10000 4096 May 13  2015 radio
drwx--x--x  2 264787 10000 4096 May 13  2015 simple
drwx--x--x  2 264787 10000 4096 May 13  2015 standard
drwx--x--x  2 264787 10000 4096 May 13  2015 tcpudp
drwx--x--x  2 264787 10000 4096 May 13  2015 test
drwx--x--x  2 264787 10000 4096 May 13  2015 threads
drwx--x--x  2 264787 10000 4096 May 13  2015 userlevel
drwx--x--x  4 264787 10000 4096 May 13  2015 wifi
root@ubuntu:/home/ubuntu/cmpe210_clickos_setup/clickos/elements#
```

Click Elements

- Click Elements documentation:

https://github.com/kohler/click/wiki/Elements

- **Basic Sources and Sinks -** Discard , TimedSink, TimedSource
- **Basic Classification and Selection –** Classifier, RoundRobinSwitch
- **TCP –** CheckTCPHeader ,TCPIPSend
- **ICMP -** ICMPError , CheckICMPHeader
- **IPv4 –** IPMirror, IPPrint, CheckIPHeader
- **ARP -** ARPFaker , ARPResponder
- **Ethernet -** EtherMirror , EnsureEther, EtherEncap

VNGuard: An NFV/SDN Combination Framework for Provisioning and Managing Virtual Firewalls

Juan Deng†, Hongxin Hu†, Hongda Li†, Zhizhong pan†,
Kuang-Ching Wang†, Gail-Joon Ahn‡, Jun Bi♯, Younghee Park§

† CLEMSON UNIVERSITY

‡ ASU ARIZONA STATE UNIVERSITY

♯ 清華大學 Tsinghua University

§ San José State UNIVERSITY

Agenda

- Motivation
- VNGuard Framework
  - High-Level Service-Oriented Policy Language
  - Optimal Virtual Firewall Placement
  - Dynamic Virtual Firewall Adaption
- Discussion and Future Work

Motivation

Tradition Hardware-based Firewall

- Fixed capacity with respect to the maximum traffic to process
- Fixed deployment location
- Long deployment time period: purchase, configuration, deployment

Motivation

Prevalence of Virtual Networks

- Fluid physical perimeter due to migrations of VMs and/or applications
- The ability of VMs and/or application to elastically scale
- Required new security features: small, customized, portable firewalls with short provision and deployment time period

Virtual Firewall meets the security needs of virtual networks.

VNGuard: Design Goals

High-level service oriented policy language

- User centric security needs
- Incapability to write low-level firewall rules that are associated with virtual network deployment
  - No access to information on virtual network deployment

Optimal virtual firewall placement

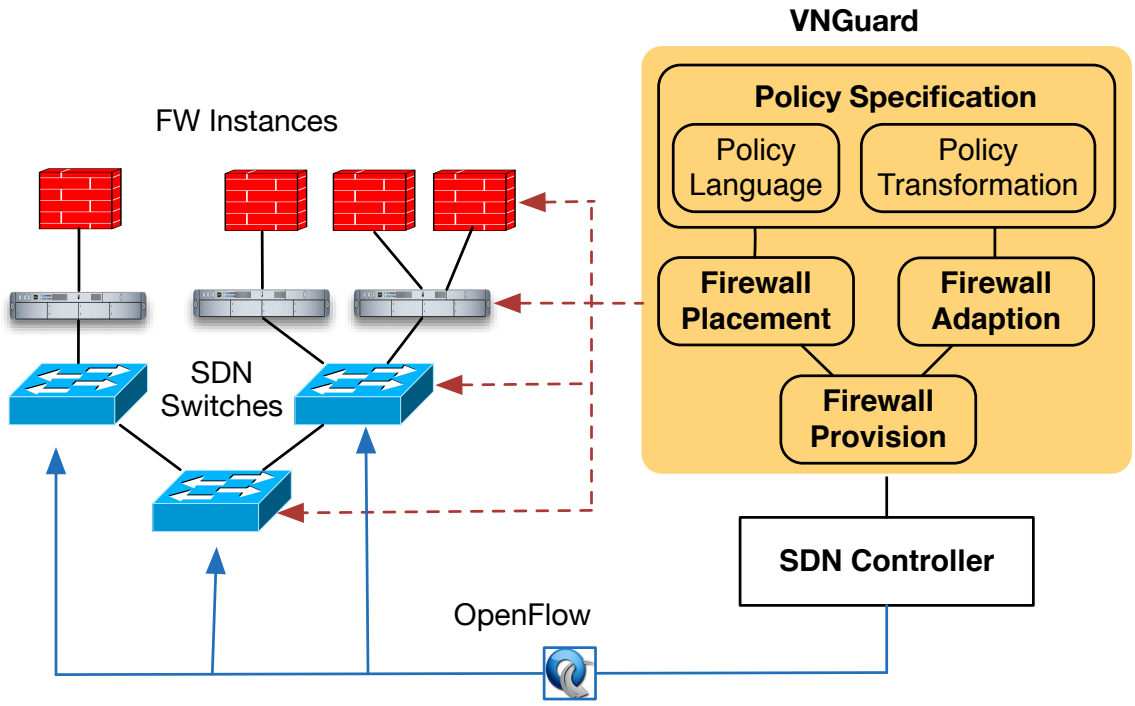
- Large number of firewall rules for a single user
- One large virtual firewall is impractical: performance consideration

Dynamic virtual firewall adaption

- Migration of VMs and/or applications in a virtual network
- Addition/Deletion of VMs and/or applications in a virtual network
- Changes of user security requirements

VNGuard Design

A comprehensive framework for effectively provisioning and managing virtual firewalls, leveraging both NFV and SDN techniques



Four major components

VNGuard is implemented as an SDN application

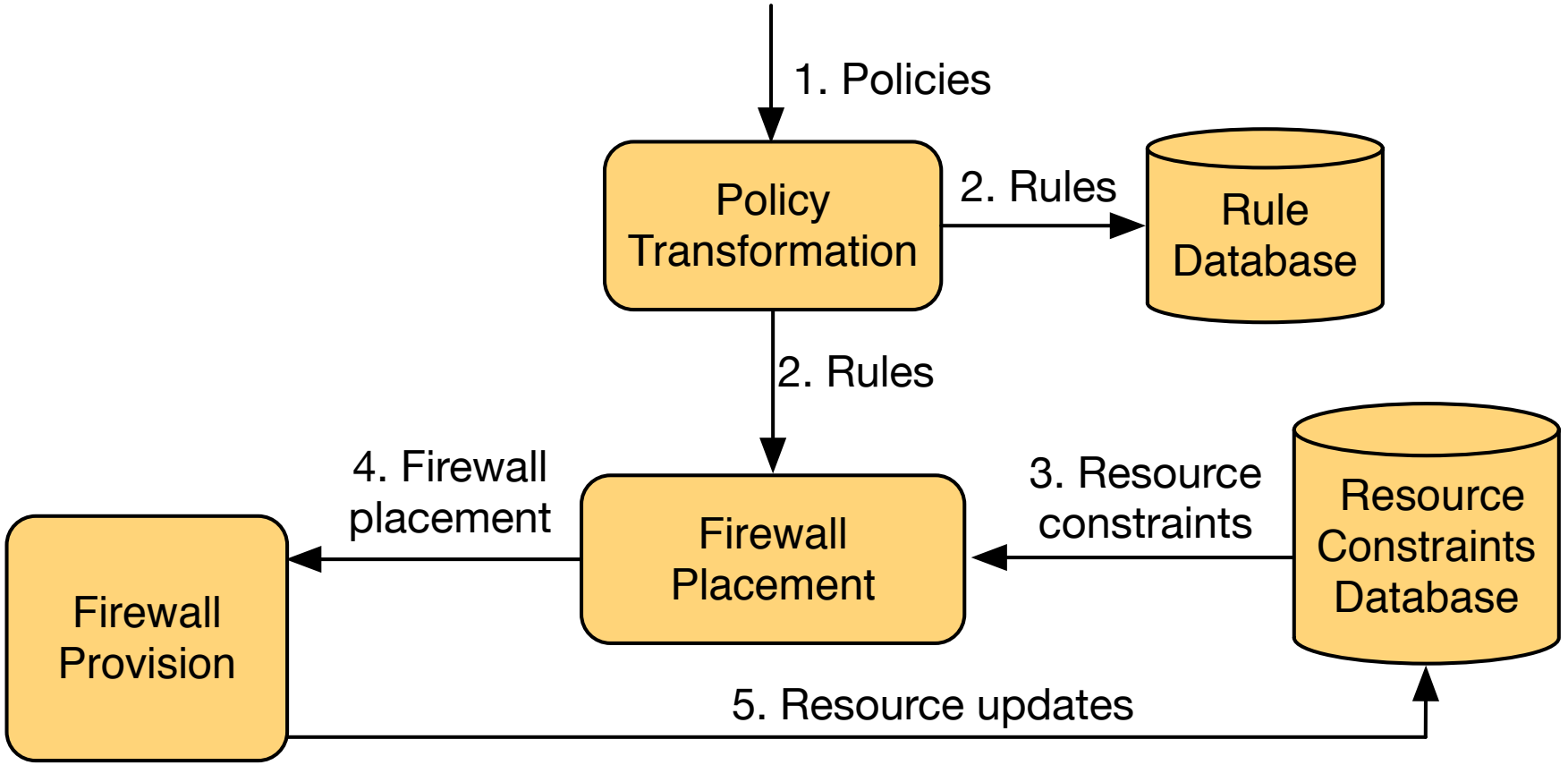
High-Level Service-Oriented Policy Language

Enable users to specify their security requirements without prior knowledge on the low level information of virtual firewall deployment

$P = \{p_1, p_2, ...\}$	***P*** **is a is a finite set of ports that services use.**
$T = \{tcp, udp, ...\}$	***T*** **is a finite set of protocols that services use.**
$V = \{v_1, v_2, ...\}$	***V*** **is a finite set of VNs that a user has.**
$s ::= \langle \bar{V}, p, t \rangle$	**A service *s* has three members with being a subset of a user's VNs that run *s* (that is,** $\bar{V} \subseteq V$ **),** $p \in P$ **being the port number used by *s*, and** $t \in T$ **being the protocol used by *s*.**
$a ::= \langle accept \mid deny \mid delete \rangle$	**An action *a* is one of *accept*, *deny* and *delete*. *delete* is used in the virtual firewall adaption.**
$o ::= \langle IP \mid domain\_name \mid VN\_name, p \rangle$	**An object *o* has two members with the first being either an IP address, a domain name, or the name of a VN, and the second element being a port number.**
$policy ::= \langle s, a, o \rangle$	**Policy defines the access right of object *o* on service *s***

- ***ex) http service s = < v1, 80, tcp>***
- **Support of global policies, group policies, and local polices**
- **Transformation of high-level policy to low-level firewall rules: Police Transformation component in VNGuard.**

Optimal Virtual Firewall Placement



Optimal Virtual Firewall Placement

Resource limits:

NFV service provider limits the number of virtual firewall instances

NFV service provider assigns limited resource to each virtual firewall instance

Integer Programing Model

Condition 1: the number of firewall rules placed on each instance cannot exceed its capacity

Condition 2: a firewall rule is placed on only one instance

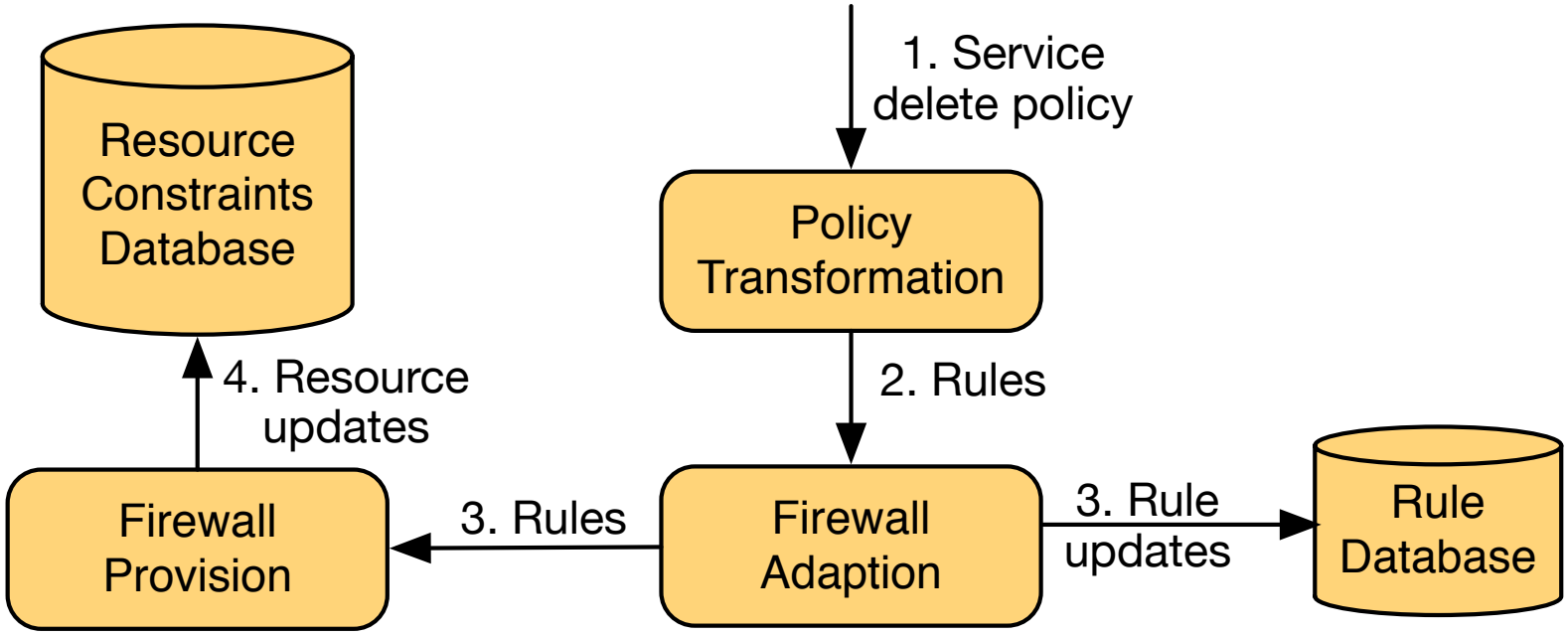
Dynamic Virtual Firewall Adaption

- Dynamic nature of virtual firewall
  - Type I: A user adds or deletes services in a virtual network
  - Type II: A user updates security policies
  - Type III: VM and/or service migration and scaling out/in

Dynamic Virtual Firewall Adaption

Type I: A user adds or deletes services in a virtual network

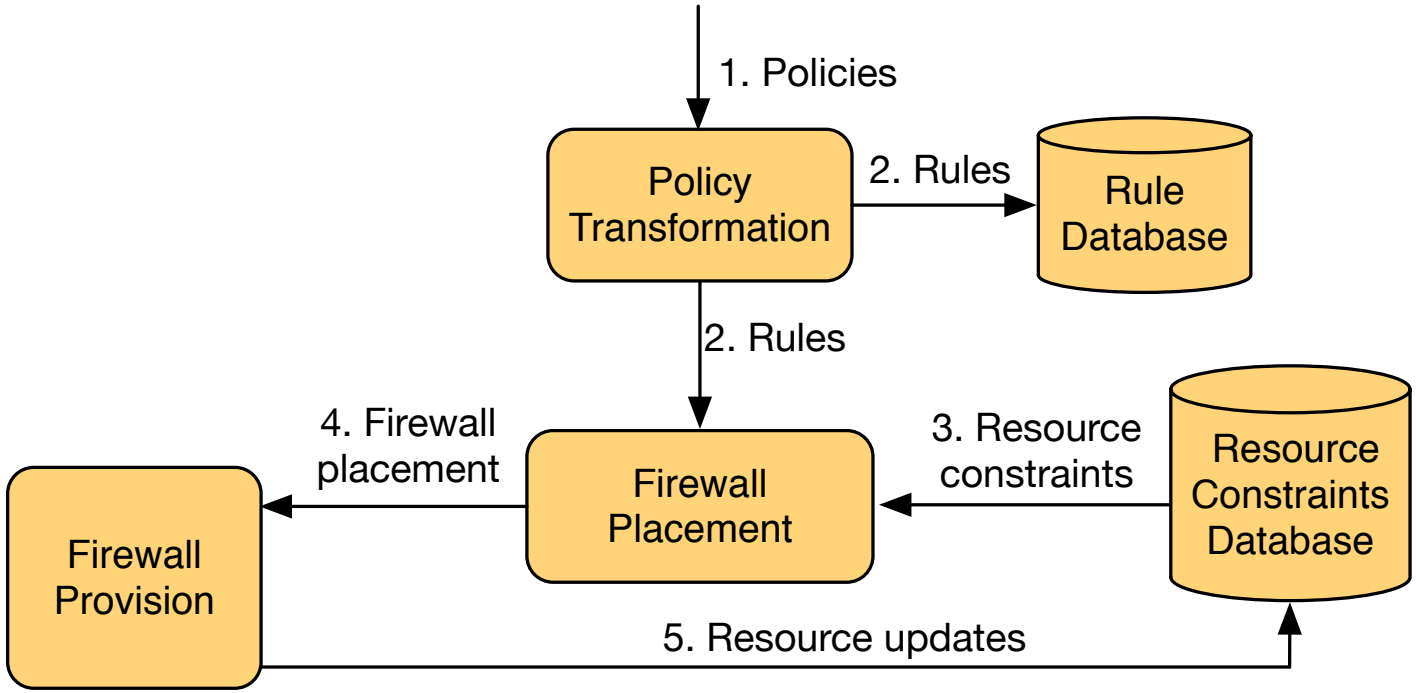
Virtual Firewall Adaption to **Service Deletion**



Dynamic Virtual Firewall Adaption

Type I: A user adds or deletes services in a virtual network

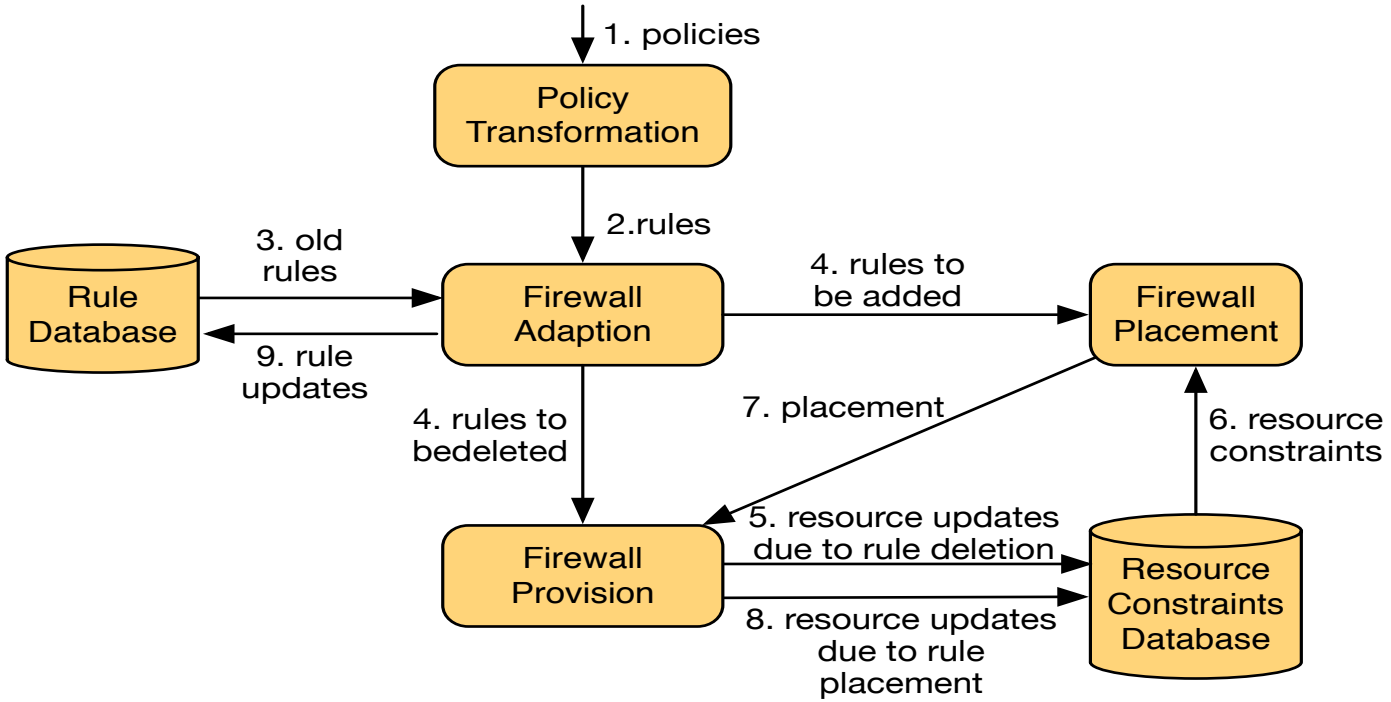
Virtual Firewall Adaption to **Service Addition**



Dynamic Virtual Firewall Adaption

Type II: add or delete security policies for a virtual network

The order of rule addition and deletion is critical: security considerations



Dynamic Virtual Firewall Adaption

Type III: VM and/or service migration and scaling out/in.

Virtual firewall rule migration
Virtual firewall scaling out/in

Implementation

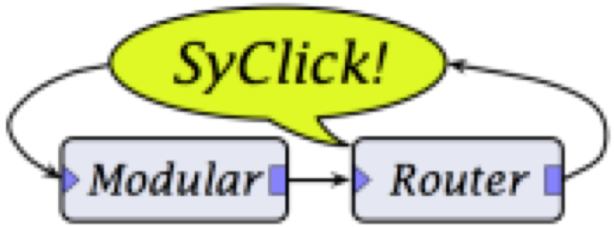
We have implemented the core components of VNGuard on top of ClickOS

- A Xen-based software platform optimized for fast provision of virtual network functions at large scale
- A virtual network function instance can be as small as 5MB and booted within 30ms

Implementation

Modification of ClickOS

- Features missing in ClickOS: does not allow firewall rules on a firewall instance to be updated without rebooting the instance.
- Three new Click elements to support virtual firewall **adaption**

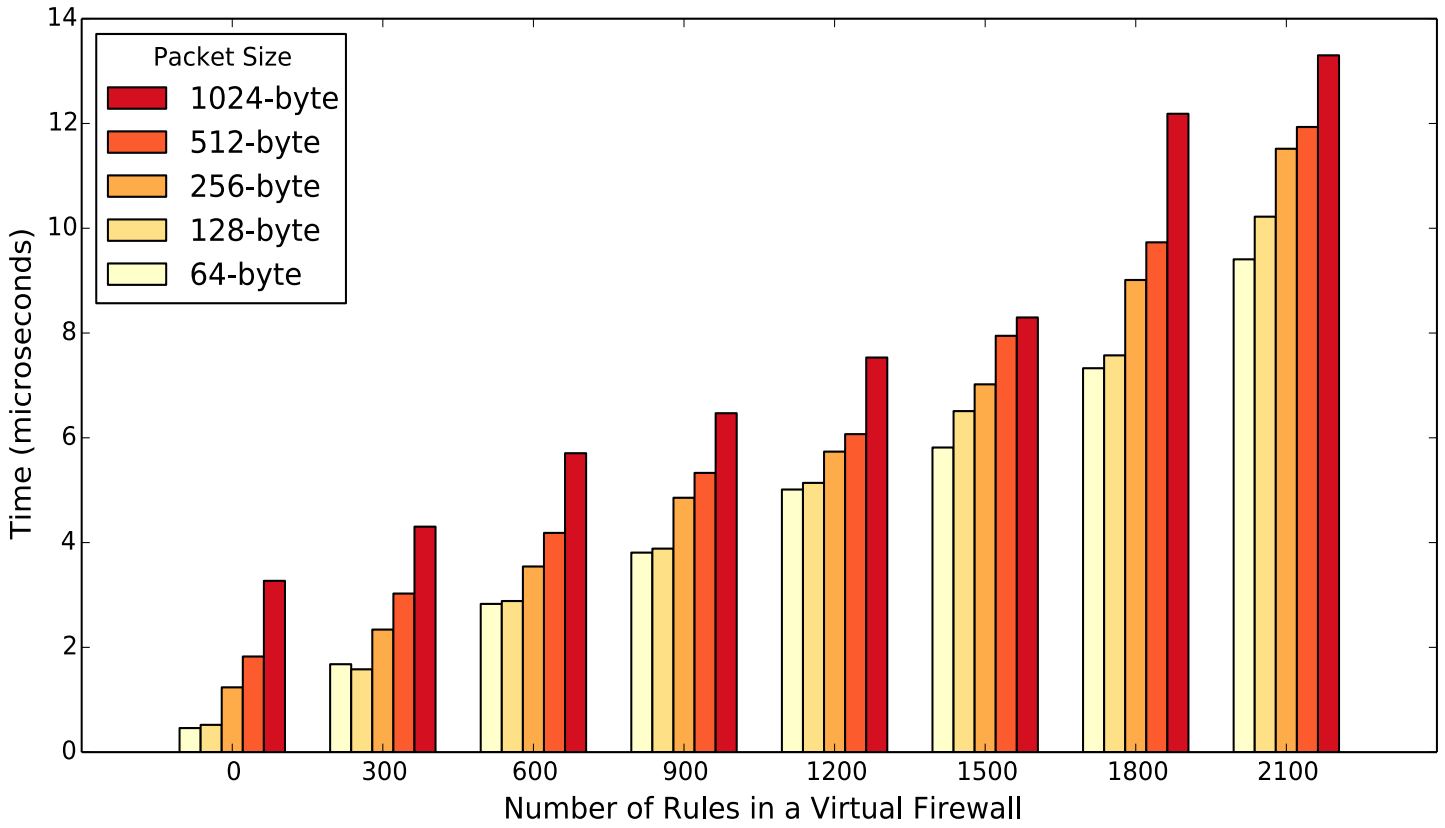


Evaluation

- Goals
  - The packet processing performance of a virtual firewall provisioned by VNGuard
  - The performance of virtual firewall Placement
  - The performance of virtual firewall adaption
- Platform
  - CloudLab
  - ClickOS

Evaluation

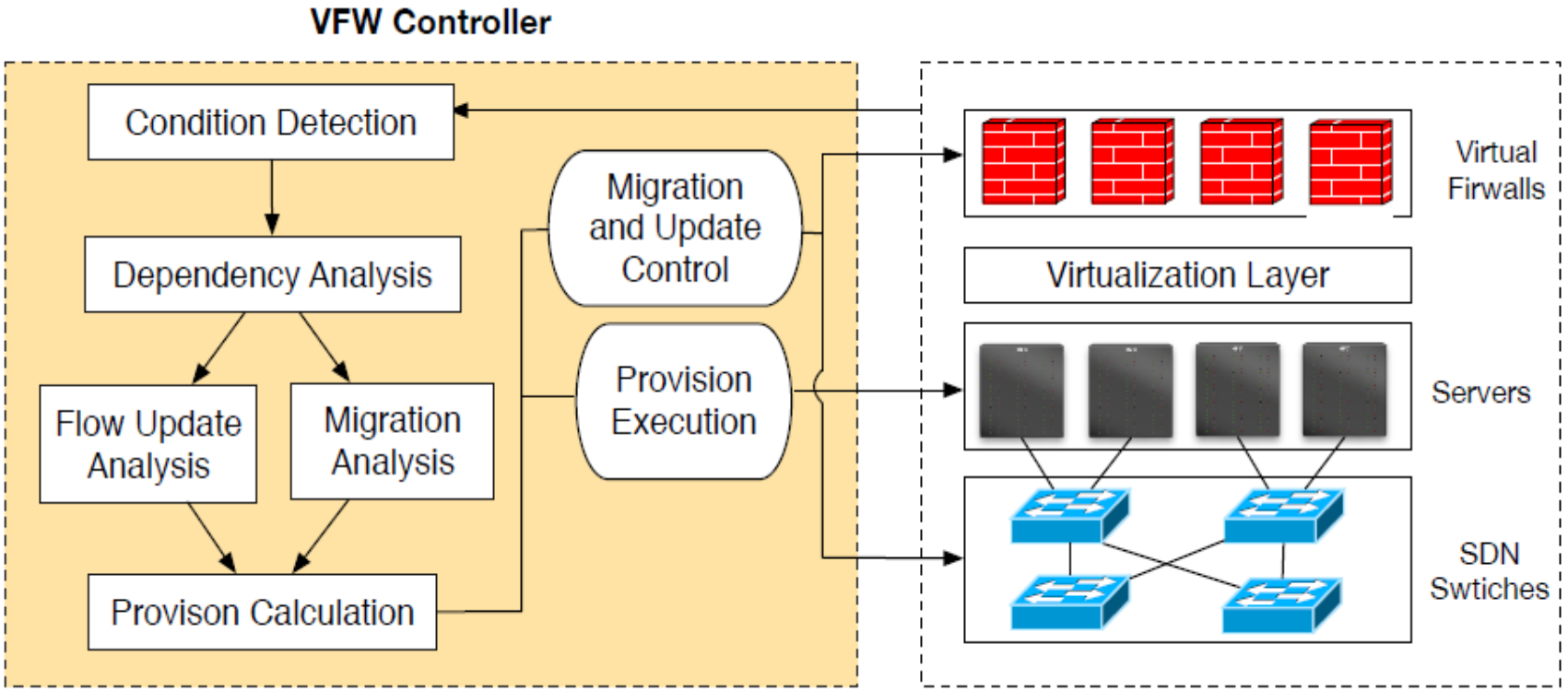
Experiment 1: The packet processing performance of a virtual firewall - Fix the incoming traffic rate: 90Mbps



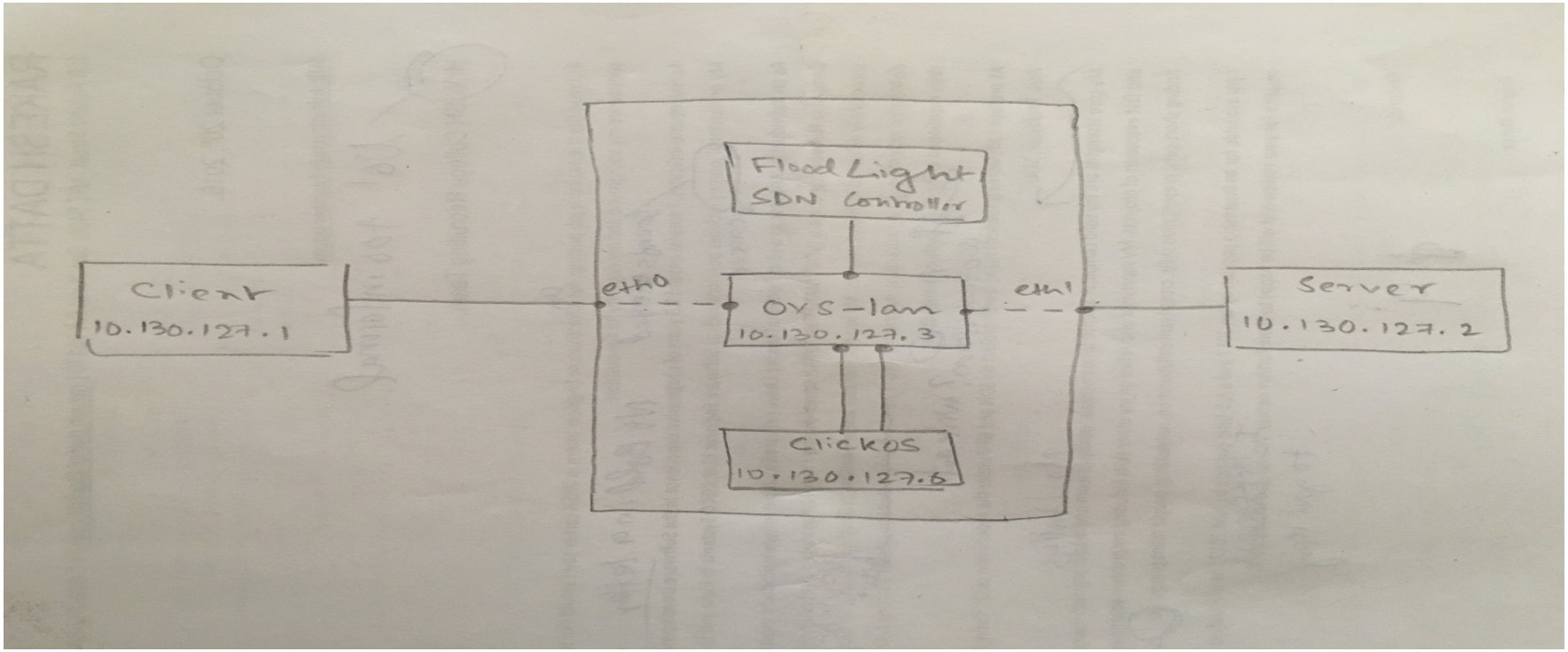
Discussion

Integer Programming approach for firewall rule placement

Virtual firewall scaling in/out – NFV Controller



Topology


Flood Light
SDN Controller
Client
10.130.127.1
eth0
OVS-lan
10.130.127.3
eth1
Server
10.130.127.2
ClickOS
10.130.127.6