"Smart Society Innovation through Mobile Internet!".

Future Internet – OpenFlow
Software Defined Network

Prof. Thomas B.YOON
tomayoon@ieee.org
Kyonggi University, Korea

Content

What is problem in current Internet? #1

- Internet has become part of the critical infrastructure of our businesses, homes and schools.
- This success has been both a blessing and a curse for networking researchers;
- Their work is more relevant, but their chance of making an impact is more remote.

What is problem in current Internet? #2

- The reduction in real-world impact of any given network innovation is because the enormous installed equipment and protocols,

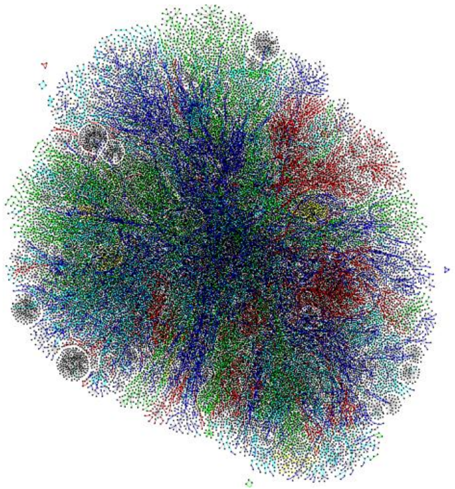
- the reluctance to experiment with real traffic, which have created an exceedingly high barrier to entry for new ideas.

What is problem in current Internet? #3

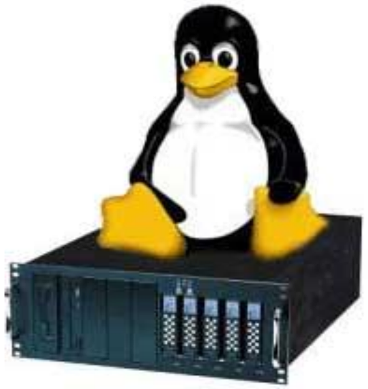
- Today, there is almost no practical way to experiment with new network protocols in sufficiently realistic settings to gain the confidence needed for their widespread deployment. .

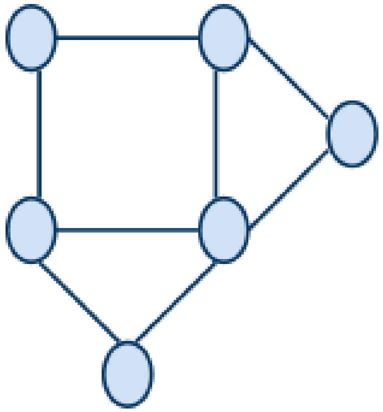
Why is Evaluation Hard?

Real Networks



Testbeds





What is problem in current Internet? #4

- The result is that most new ideas from the networking research community go untried and untested;
- hence the commonly held belief that the network infrastructure has "ossified".

Summary of Problems

Realistically evaluating new network services is hard

Because services require changes to switches and routers

- e.g.,
  - routing protocols
  - traffic monitoring services

Result: Many good ideas don't get deployed;
Many deployed services still have bugs

A Proposed Solution #1

- Having recognized the problem,

- the networking community is hard at work developing programmable networks,

- such as GENI a proposed nationwide research facility for experimenting with new network architectures and distributed processing systems.

A Proposed Solution #2

- These programmable networks call for programmable switches and routers
- That, using virtualization, can process packets for multiple isolated experimental networks simultaneously.

A Proposed Solution #3

- Thru GENI it is envisaged that a researcher will be allocated a slice of resources across the whole network, consisting of a portion of network links, packet processing elements (e.g. routers) and end-hosts;

- Researchers can program their slices to behave as they wish.

A Proposed Solution #4

- A slice could extend across the backbone, into access networks, into college campuses, industrial research labs, and include wiring closets, wireless networks, and sensor networks

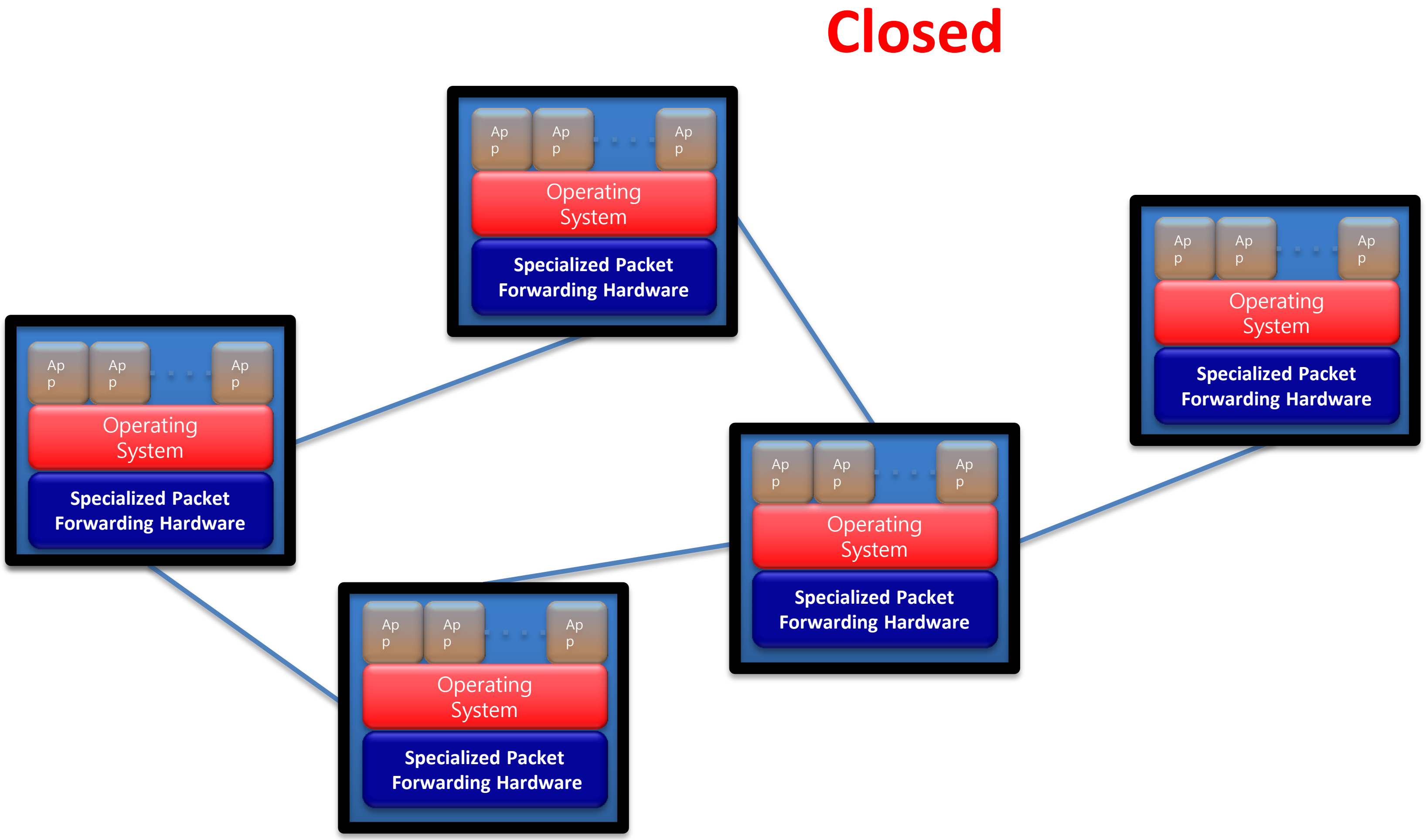
A Proposed Solution #5

- Virtualized programmable networks could lower the barrier to entry for new ideas, increasing the rate of innovation in the network infrastructure.

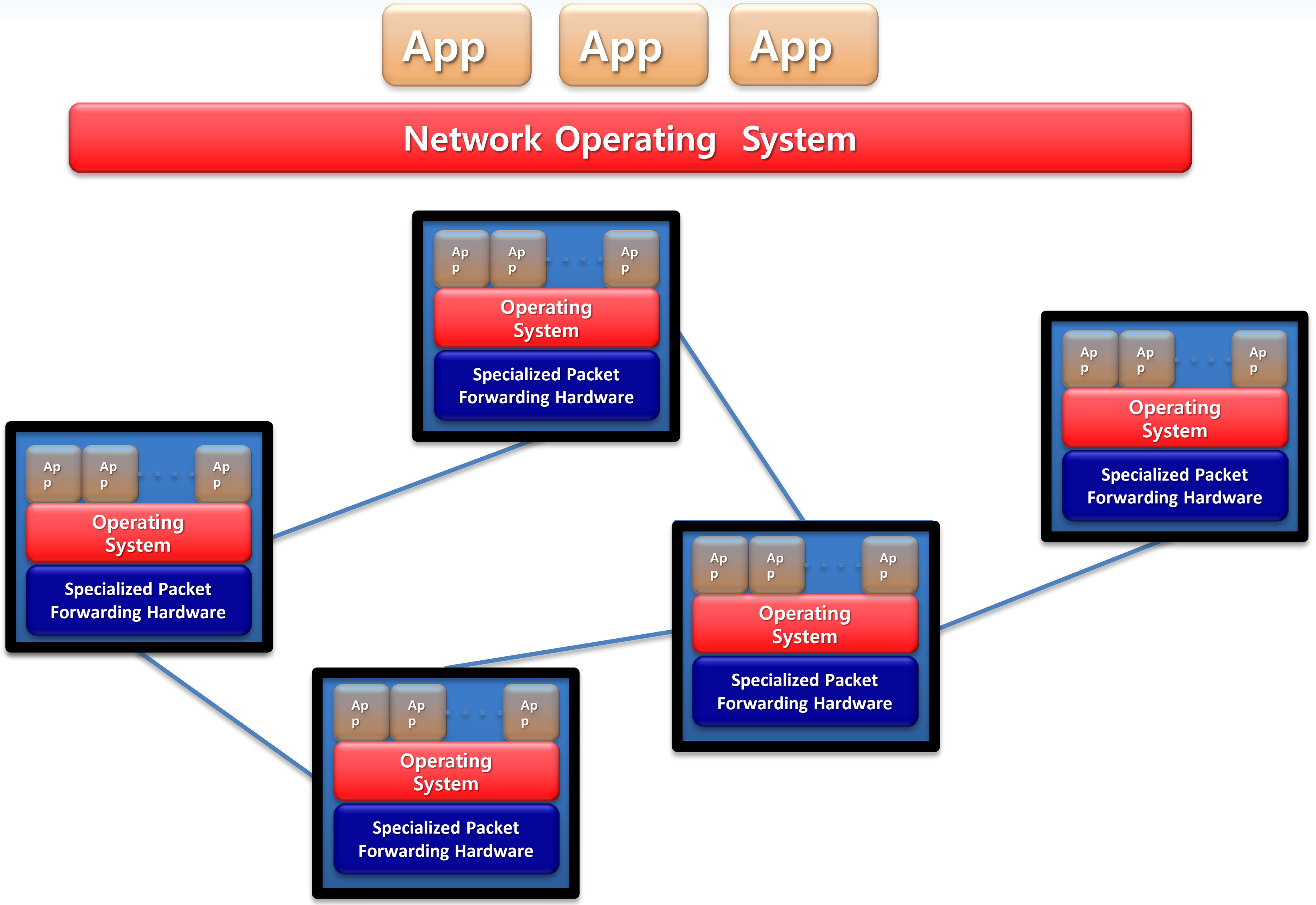
- But the plans for nationwide facilities are ambitious and costly, and it will take years for them to be deployed

Current Internet

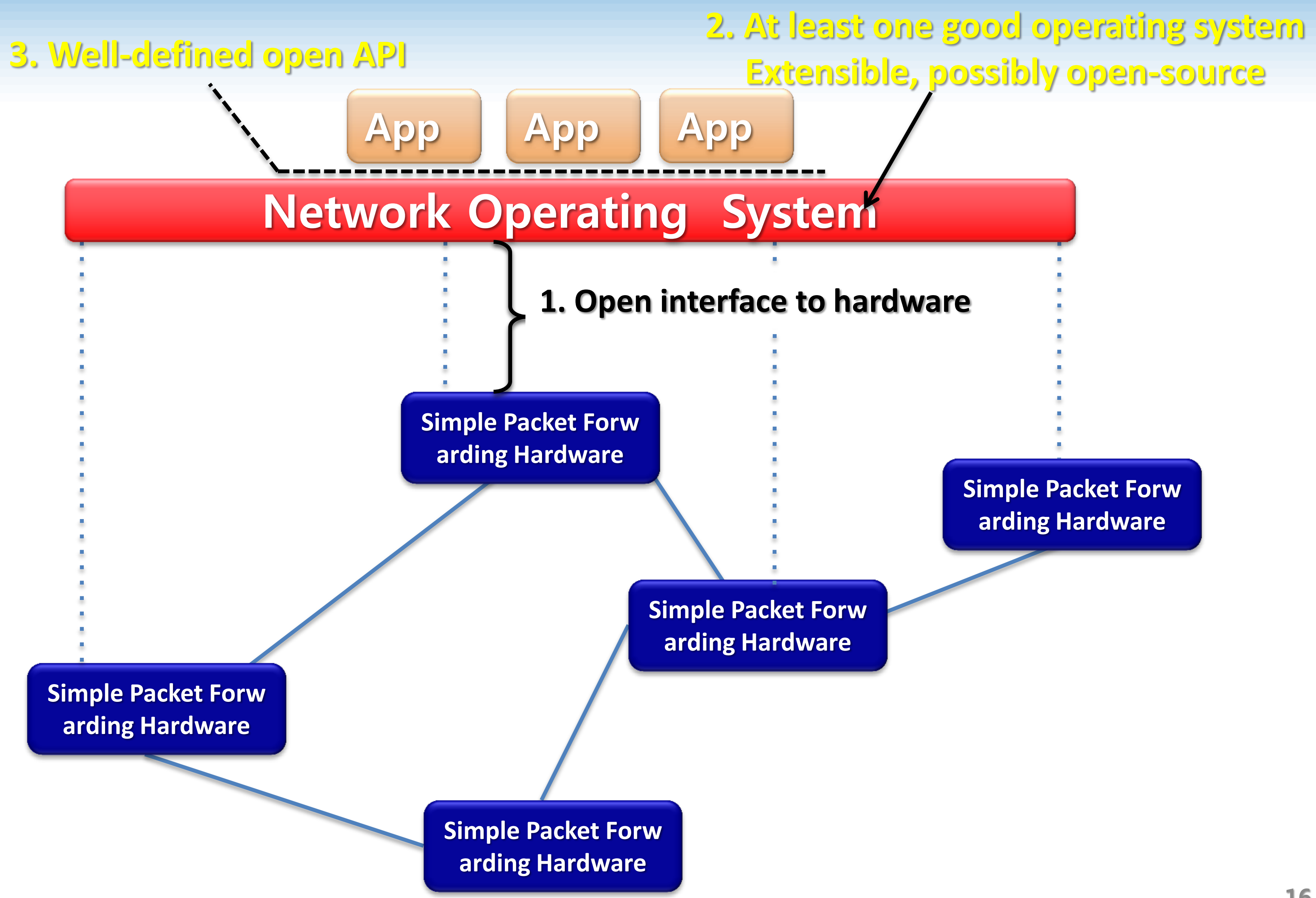
Closed to Innovations in the Infrastructure

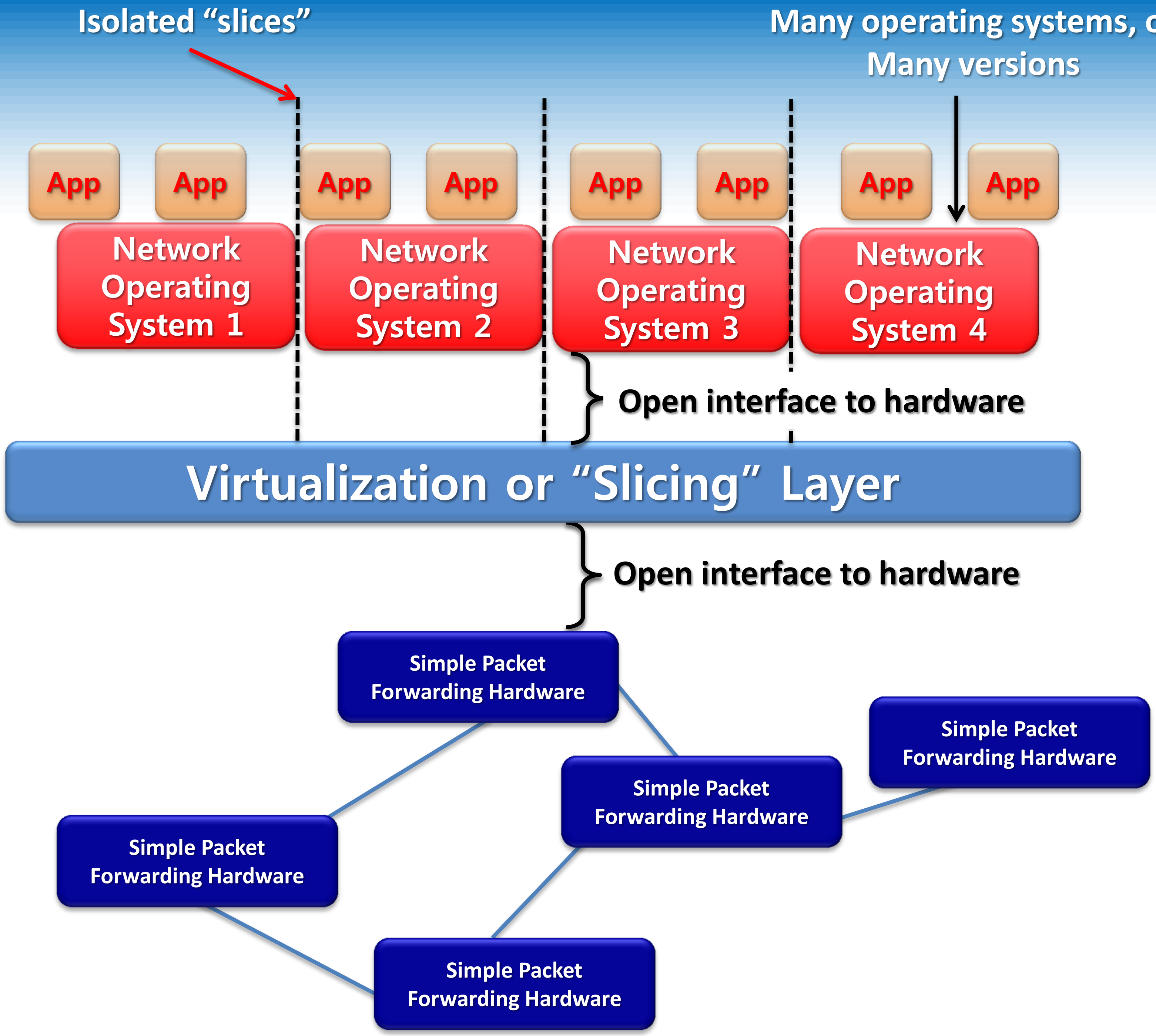


"Software Defined Networking" approach to open the Internet devices



“Software-defined Network”

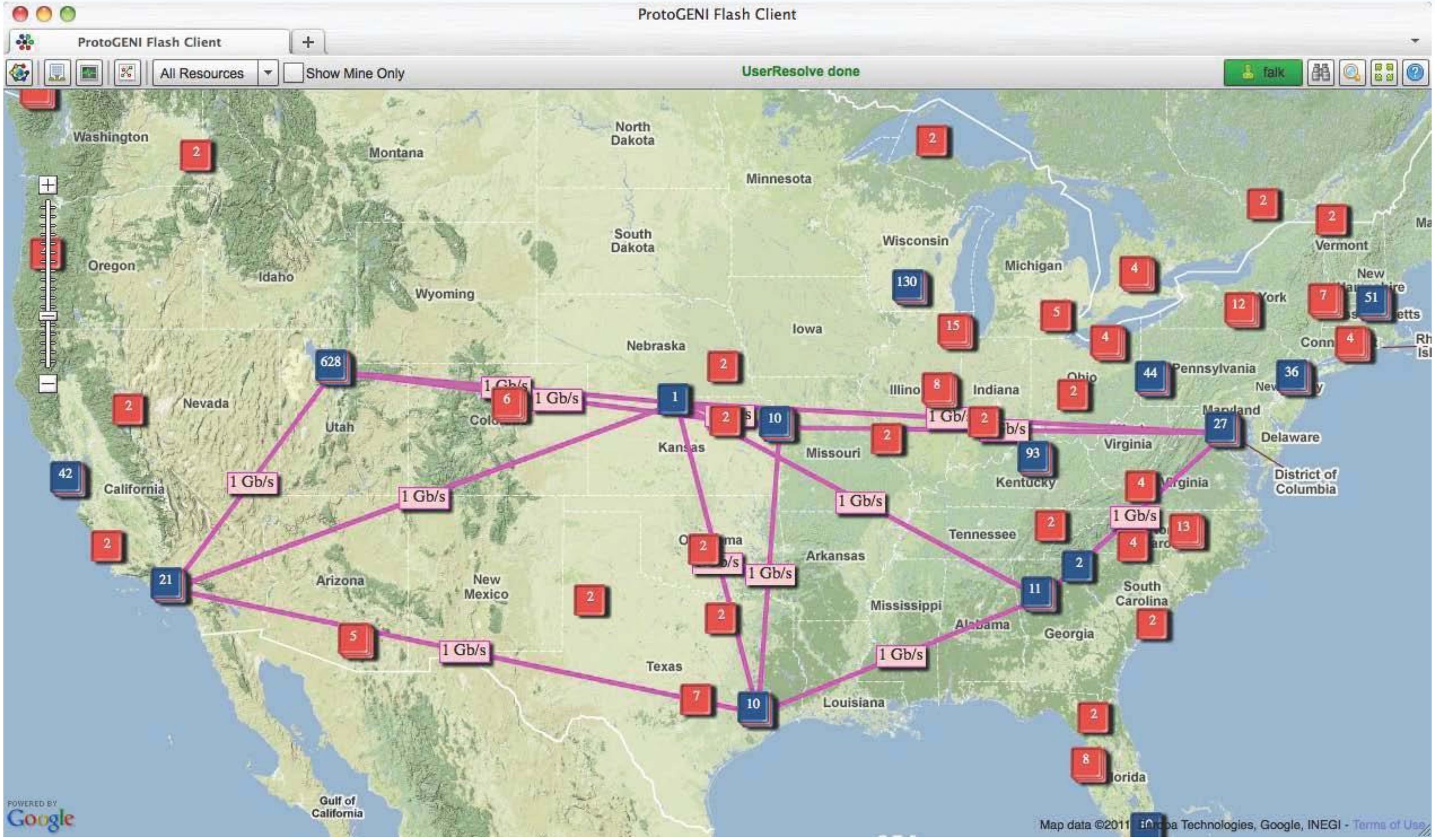

3. Well-defined open API
2. At least one good operating system
Extensible, possibly open-source
App
App
App
Network Operating System
1. Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware


Isolated “slices”
Many operating systems, or Many versions
App
App
App
App
App
App
App
App
Network Operating System 1
Network Operating System 2
Network Operating System 3
Network Operating System 4
Open interface to hardware
Virtualization or “Slicing” Layer
Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

A Proposed Solution #6

- The OpenFlow focuses on a shorter-term question closer to campus:
- As researchers, how can we run experiments in our campus networks?
- If we can figure out how we can start soon and extend the technique to other campuses to benefit the whole community.

Live map showing ProtoGENI and PlanetLab cluster resources



Challenge #1

- To meet this challenge, several questions need answering, including:

- In the early days, how will college network administrators get comfortable putting experimental equipment like switches, routers, access points, etc. into their network?

Challenge #2

- How will researchers control a portion of their local network in a way that does not disrupt others who depend on it?

Challenge #3

- Exactly what functionality is needed in network switches to enable experiments?
- Goal is to propose a new switch feature that can help extend programmability into the wiring closet of campuses.

What is reality? #1

- One approach - that we do not take - is to persuade commercial “name-brand” equipment vendors to provide an open, programmable, virtualized platform on their switches and routers

- so that researchers can deploy new protocols, while network administrators can take comfort that the equipment is well supported.

What is reality? #2

- This outcome is very unlikely in the short-term.
- Commercial switches and routers do not typically provide an open software platform,
- let alone provide a means to virtualize either their hardware or software.
- The practice of commercial networking is that the standardized external interfaces are narrow (i.e., just packet forwarding),
- and all of the switch’s internal flexibility is hidden.

What is reality? #3

- The internals differ from vendor to vendor, with no standard platform for researchers to experiment with new ideas.
- Further, network equipment vendors are understandably nervous about opening up interfaces inside their boxes:
- Because they have spent years deploying and tuning fragile distributed protocols and algorithms,
- they fear that new experiments will bring networks crashing down.

What is reality? #4

- And, of course, open platforms lower the barrier-to-entry for new competitors.

What is reality in primitive platforms? #1

- A few open software platforms already exist, but do not have the performance or port-density we need.
- The simplest example is a PC with several network interfaces and an operating system.
- All well-known operating systems support routing of packets between interfaces, and

What is reality in primitive platforms? #2

- open-source implementations of routing protocols exist (e.g., as part of the Linux distribution, or from XORP-open source Internet protocol routing software suite);

- The simplest example is a in most cases it is possible to modify the operating system to process packets in almost any manner (e.g., using Click modular router).

What is reality in primitive platforms? #3

- The problem, of course, is performance:
- A PC can neither support the number of ports needed for a college wiring closet a fan-out of 100+ ports is needed per box,
- nor the packet-processing performance; wiring closet switches process over 100Gbits/s of data whereas a typical PC struggles to exceed 1Gbit/s the gap between the two is widening.

What is reality in primitive platforms? #4

- Existing platforms with specialized hardware for line-rate processing are not quite suitable for college wiring closets either.

- For example, an ATCA-based virtualized programmable router called the Supercharged PlanetLab Platform is under development at Washington University

What is reality in current platforms? #5

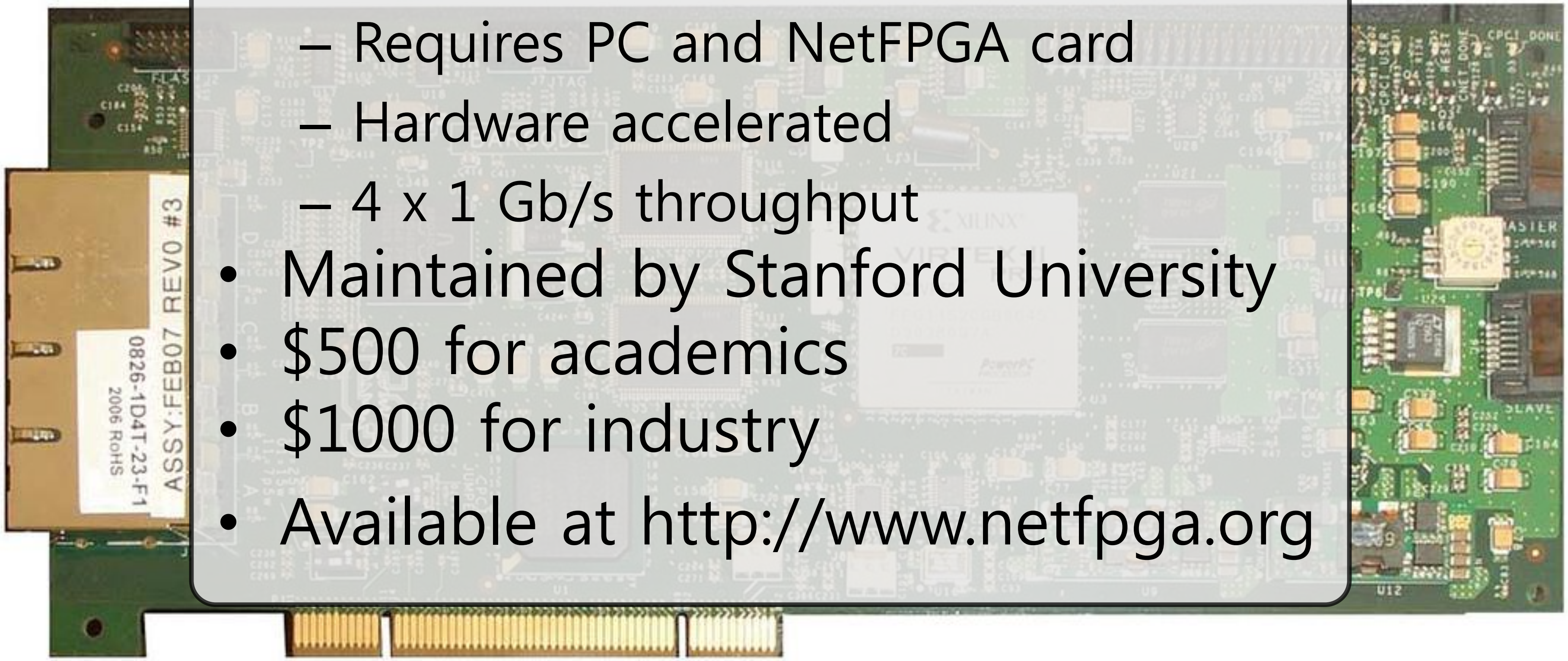
- Supercharged PlanetLab Platform can use network processors to process packets from many interfaces simultaneously at line-rate.

- This approach is promising in the long-term, but for the time being is targeted at large switching centers and is too expensive for widespread deployment in college wiring closets.

What is reality in primitive platforms? #6

- At the other extreme platform is NetFPGA targeted for use in teaching and research labs.
- NetFPGA is a low-cost PCI card with a user-programmable FPGA for processing packets, and 4-ports of Gigabit Ethernet.
- NetFPGA is limited to just four network interfaces — insufficient for use in a wiring closet.

NetFPGA



- NetFPGA-based implementation
  - Requires PC and NetFPGA card
  - Hardware accelerated
  - 4 x 1 Gb/s throughput
- Maintained by Stanford University
- $500 for academics
- $1000 for industry
- Available at http://www.netfpga.org

What is reality in primitive platforms? #7

- Thus, the commercial solutions are too closed and inflexible, and the research solutions either have insufficient performance or fan-out, or are too expensive.

- With their complete generality, It seems unlikely that the research solutions, can overcome their performance or cost limitations.

Summary:
Not a New Problem

- Build open programmable network hardware
  - NetFPGA, network processors
  - but: deployment is expensive, fan-out is small
- Build bigger software testbeds
  - VINI/PlanetLab, Emulab
  - but: performance is slower, realistic topologies?
- Convince users to try experimental services
  - personal incentive, SatelliteLab
  - but: getting *lots* of users is hard

Survey Open Systems

	Performance Fidelity	Scale	Real User Traffic?	Complexity	Open
Simulation	medium	medium	**no**	medium	yes
Emulation	medium	**low**	**no**	medium	yes
Software Switches	**poor**	**low**	yes	medium	yes
NetFPGA	high	**low**	yes	**high**	yes
Network Processors	high	medium	yes	**high**	yes
Vendor Switches	high	high	yes	low	**no**

None have all the desired attributes!

What is promising approach? Satisfying four goals

A more promising approach is to compromise on generality and to seek a degree of switch flexibility that is:

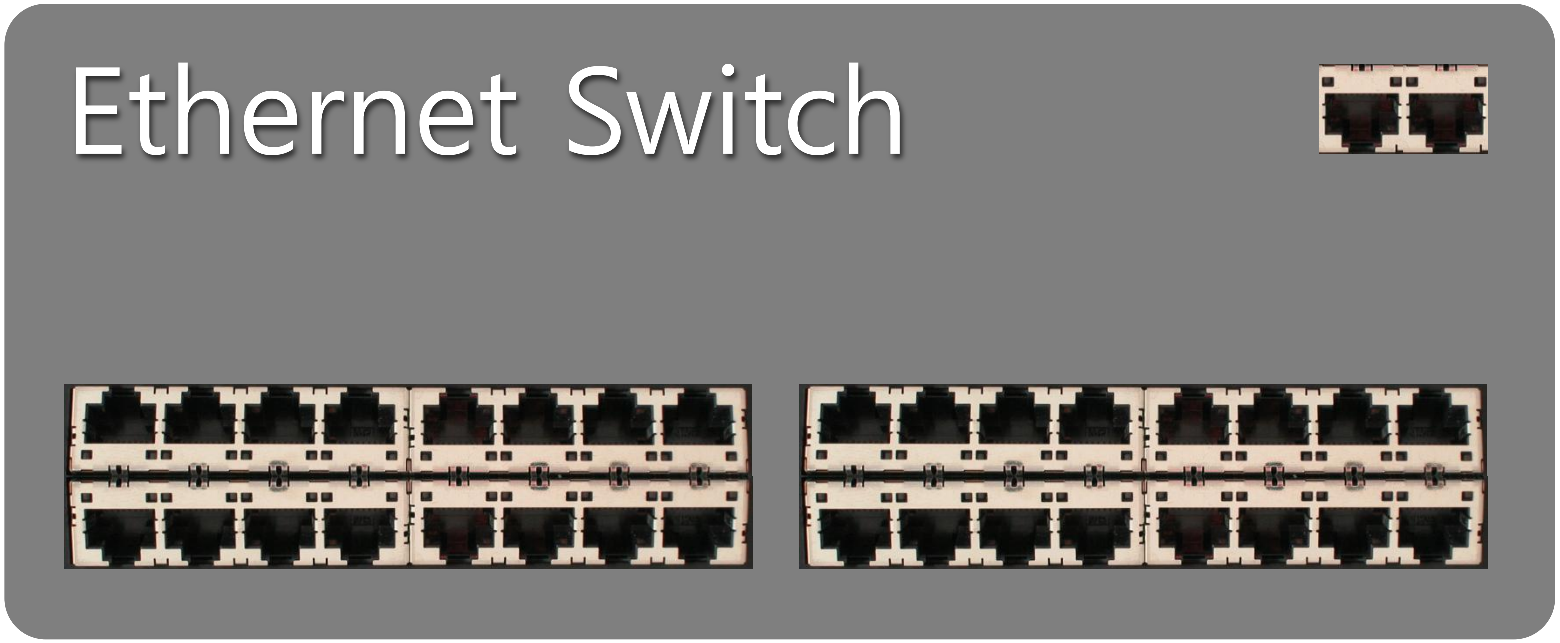
1. Amenable to high-performance and low-cost implementations.
2. Capable of supporting a broad range of research.
3. Assured to isolate experimental traffic from production traffic.
4. Consistent with vendors' need for closed platforms

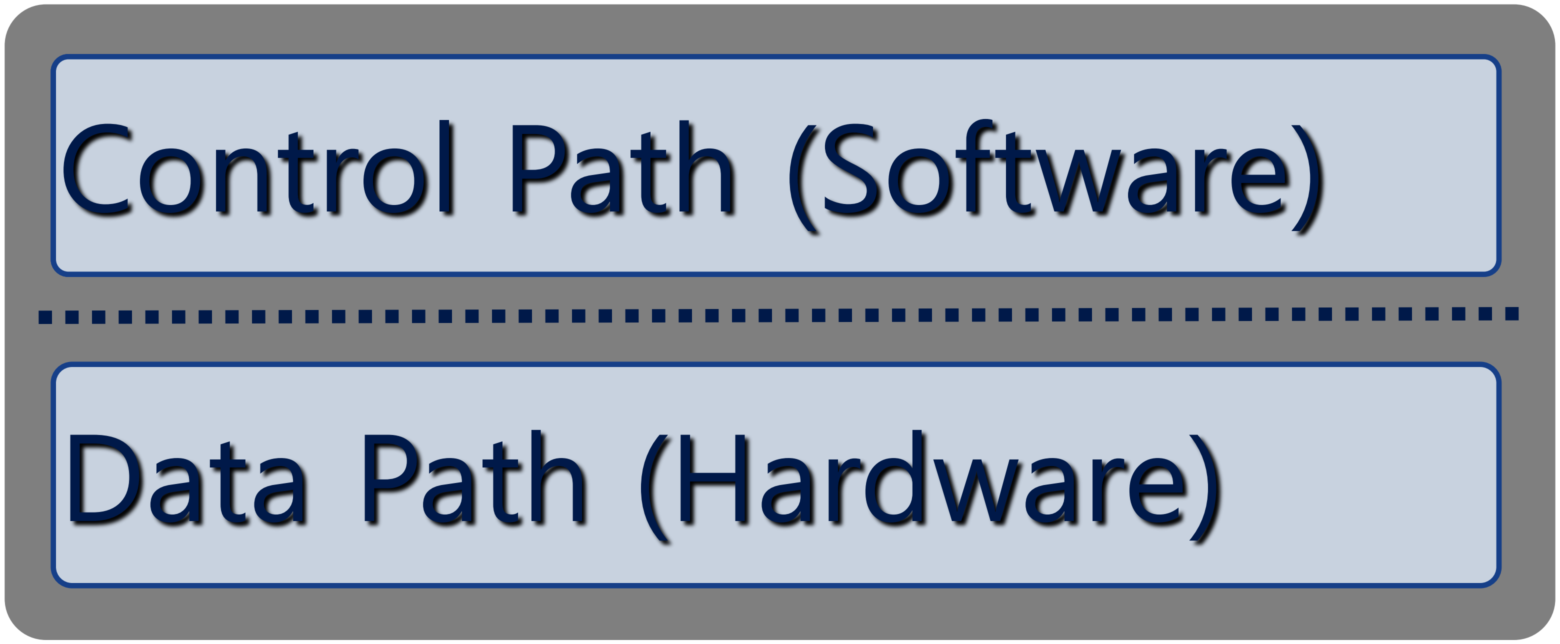
OpenFlow Switch--A specification that is an initial attempt to meet these four goals.

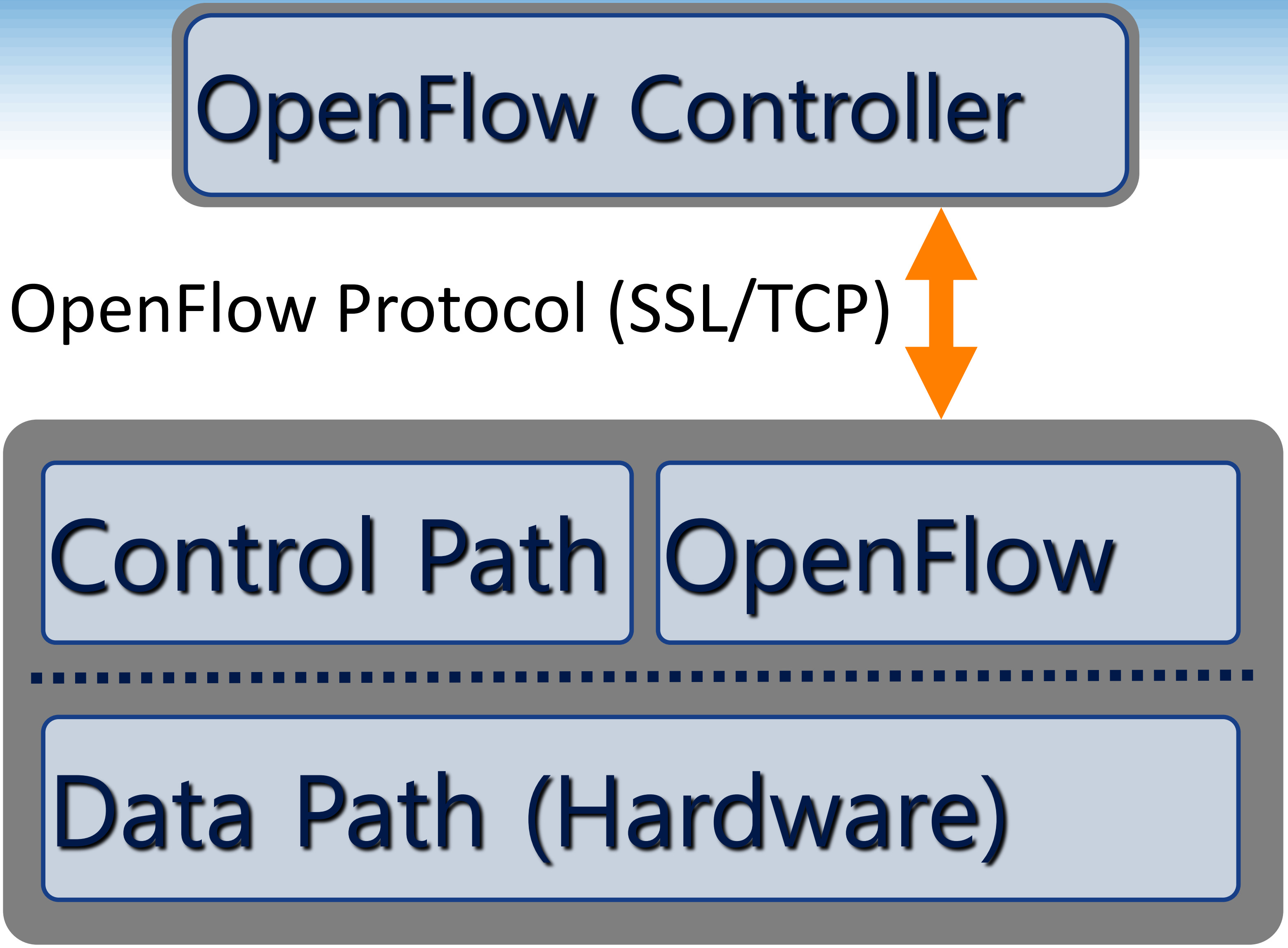
What is basic idea ?
OpenFlow Switch #1

- The basic idea is simple: we exploit the fact that most modern Ethernet switches and routers

- contain flow-tables typically built from TCAMs-(Ternary Content Addressable Memory) that run at line-rate to implement firewalls, NAT, QoS, and to collect statistics.


Ethernet Switch


Control Path (Software)
Data Path (Hardware)


OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

What is basic idea ?
OpenFlow Switch #2

- While each vendor’s flow-table is different,
- OpenFlow identified an interesting common set of functions that run in many switches and routers.
- OpenFlow exploits this common set of functions.
- OpenFlow provides an open protocol to program the flow-table in different switches and routers.

What is basic idea ?
OpenFlow Switch #3

- A network administrator can partition traffic into production and research flows.
- Researchers can control their own flows - by choosing the routes their packets follow and processing they receive.
- In this way, researchers can try new routing protocols, security models, addressing schemes, and even alternatives to IP.

What is basic idea ?
OpenFlow Switch #4

- On the same network, the real traffic is isolated and processed in the same way as today.

- The data path of an OpenFlow Switch consists of a Flow Table, and an action associated with each flow entry.

What is minimum requirement ? OpenFlow Switch #1

- The set of actions supported by an OpenFlow Switch is extensible, but below we describe a minimum requirement for all switches.
- For high-performance and low-cost
- the data-path must have a carefully prescribed degree of flexibility.
- This means forgoing the ability to specify arbitrary handling of each packet and seeking a more limited, but still useful, range of actions.
- Therefore, later in the tutorial, define a basic required set of actions for all OpenFlow switches

What is minimum requirement ? OpenFlow Switch #2

An OpenFlow Switch consists of **at least** three parts:

- 1) Flow Table, with an action associated with each flow entry, to tell the switch how to process the flow,
- 2) Secure Channel that connects the switch to a remote control process (called the controller), allowing commands and packets to be sent between a controller and the switch using

What is minimum requirement ? OpenFlow Switch #3

- 3) OpenFlow Protocol, which provides an open and standard way for a controller to communicate with a switch.
- By specifying a standard interface (the OpenFlow Protocol) through which entries in the Flow Table can be defined externally, the OpenFlow Switch avoids the need for researchers to program the switch.

What is OpenFlow Switch?

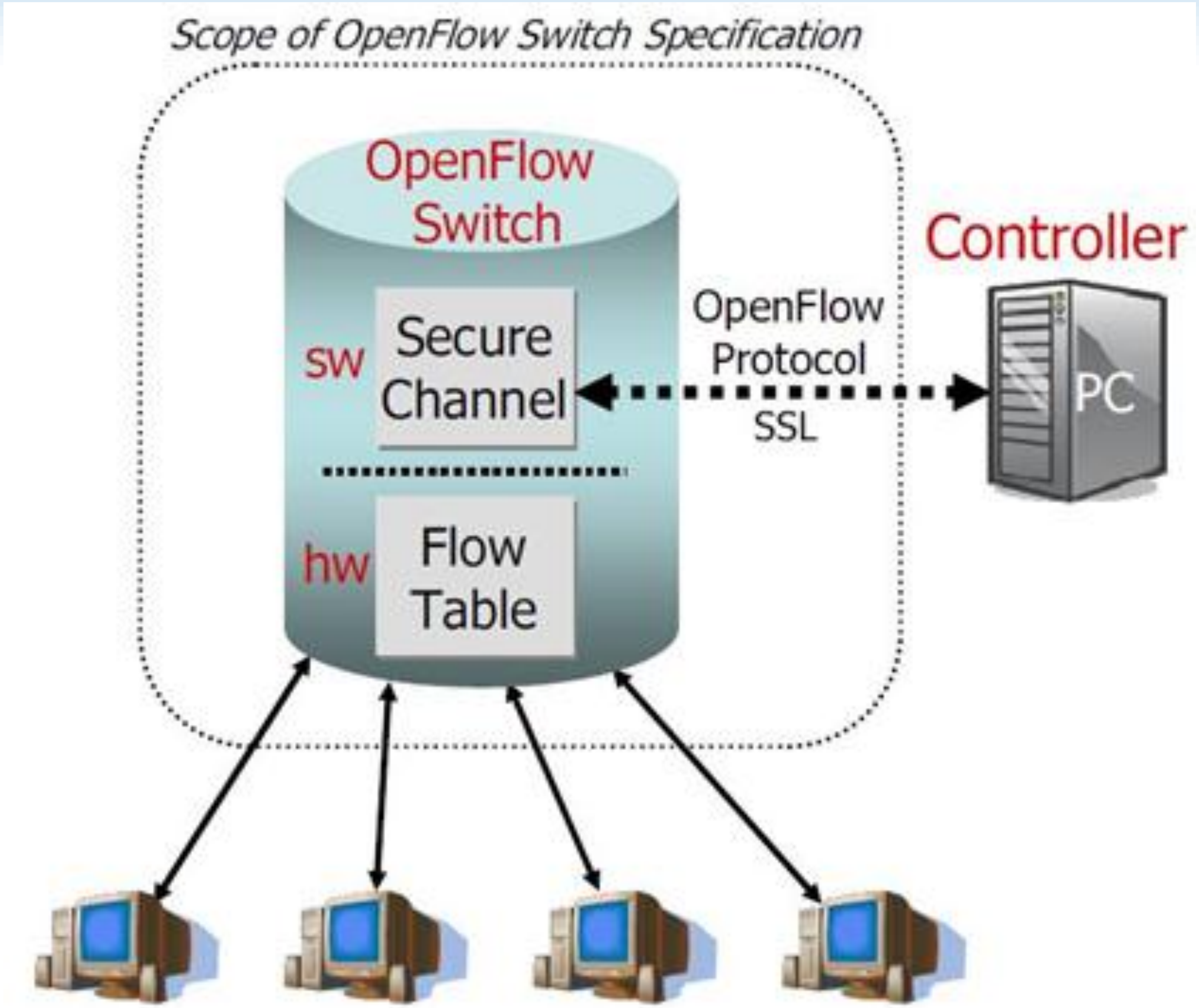
- It is useful to categorize switches into dedicated OpenFlow switches that do not support normal Layer 2 and Layer 3 processing,
- OpenFlow-enabled general purpose commercial Ethernet switches and routers, to which the Open-Flow Protocol and interfaces have been added as a new feature.

What is definition ?
OpenFlow switch #1

- Dedicated OpenFlow switches is a dumb datapath element that forwards packets between ports, as defined by a remote control process.
- Let see an example of an OpenFlow Switch.

OpenFlow Switch

The Flow Table is controlled by a remote controller via the Secure Channel



What is definition ? OpenFlow switch #2

- In this context, flows are broadly defined, and are limited only by the capabilities of the particular implementation of the Flow Table.
- For example, a flow could be a TCP connection, or all packets from a particular MAC address or IP address, or all packets with the same VLAN tag, or all packets from the same switch port.

What is definition ?
OpenFlow switch #3

- For experiments involving non-IPv4 packets,
- a flow could be defined as all packets matching a specific (but non-standard) header.
- Each flow-entry has a simple action associated with it

What is basic capability? OpenFlow switch #1

Three basic capabilities that all dedicated OpenFlow switches must support are:

1. Forward this flow's packets to a given port or ports.

- This allows packets to be routed through the network. In most switches this is expected to take place at line-rate

What is basic capability? OpenFlow switch #2

2. Encapsulate and forward this flow's packets to a controller.

- Packet is delivered to Secure Channel, where it is encapsulated and sent to a controller.
- Typically used for the first packet in a new flow, so a controller can decide if the flow should be added to the Flow Table.
- Some experiments, it could be used to forward all packets to a controller for processing.

What is basic capability? OpenFlow switch #3

3. Drop this flow's packets.

- It can be used for security, to curb denial of service attacks, to reduce spurious broadcast discovery traffic from end-hosts.

What is Flow-Table? #1

An entry in the Flow-Table has three fields:

1) Packet header that defines the flow,

2) Action, which defines how the packets should be processed, and

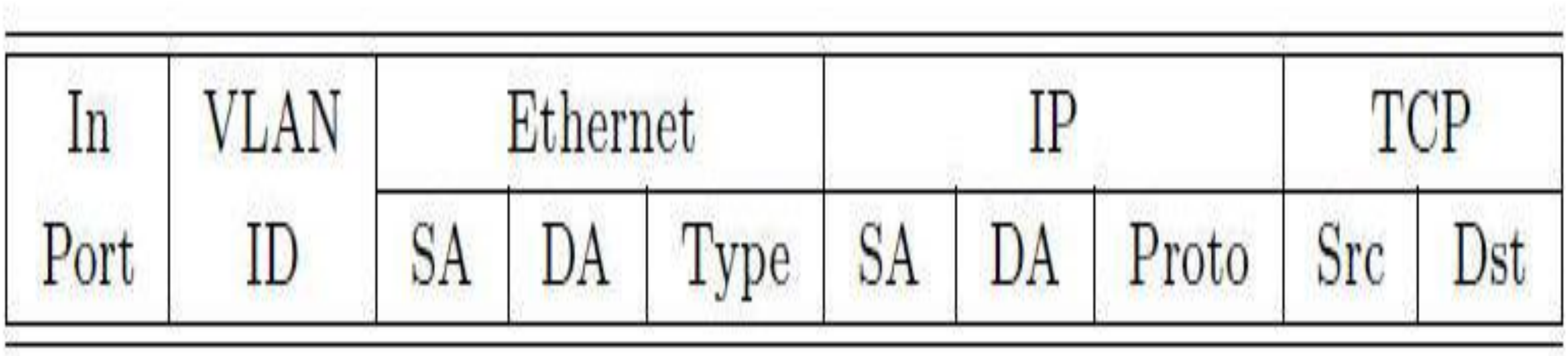
3) Statistics, which keep track of the number of packets and bytes for each flow, and the time since the last packet matched the flow (to help with the removal of inactive flows).

What is Flow-Table? #2

- In the first generation "Type 0" switches, the flow header is a 10-tuples.
- TCP flow could be specified by all ten fields, whereas an IP flow might not include the transport ports in its definition.
- Header field can be a wildcard to allow for aggregation of flows,
- Thus, flows in which only the VLAN ID is defined would apply to all traffic on a particular VLAN.

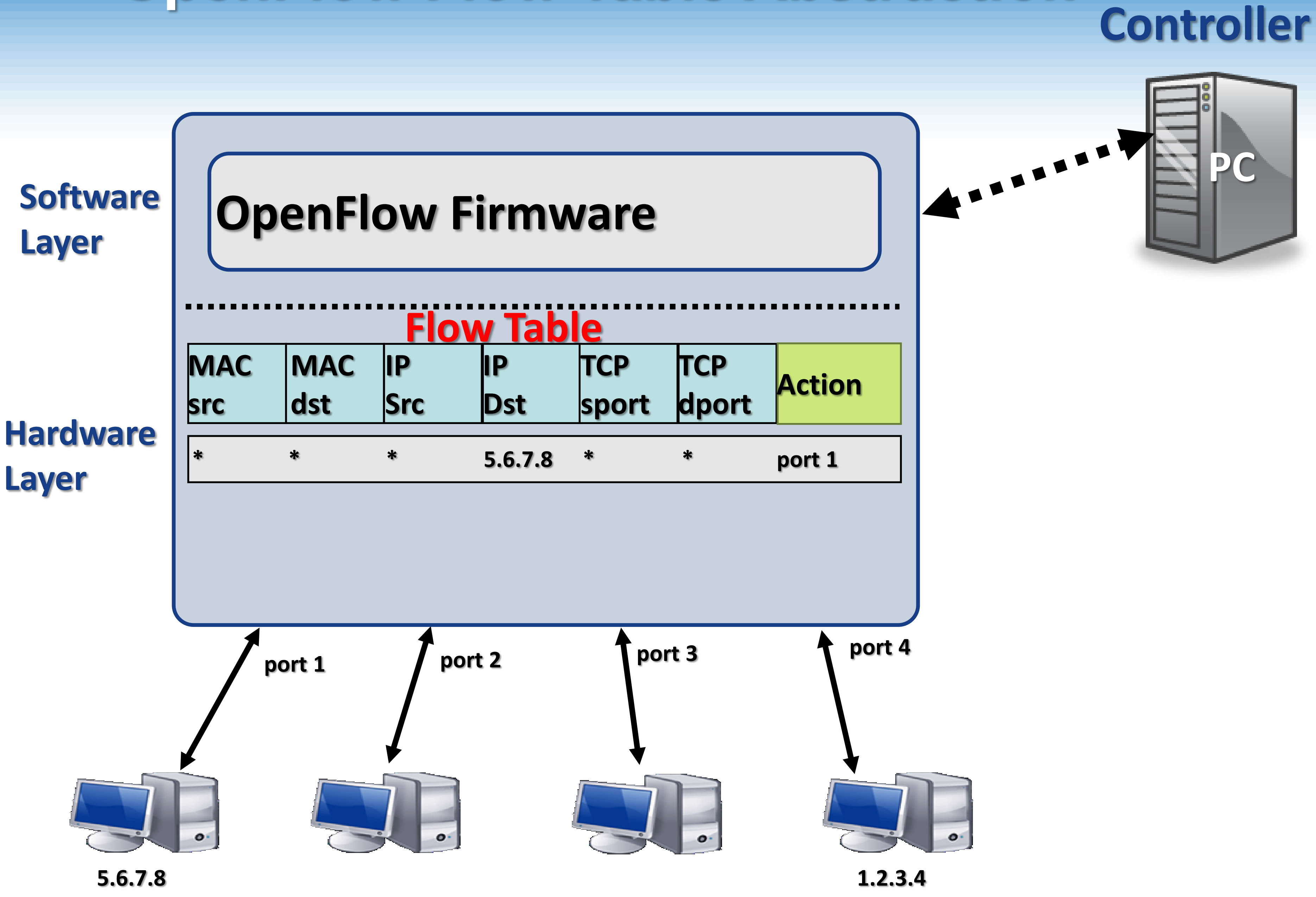
Header fields matched in "Type 0" OpenFlow switch

- The detailed requirements of an OpenFlow Switch are defined by
- the OpenFlow Switch Specification.



In	VLAN	Ethernet			IP			TCP	
Port	ID	SA	DA	Type	SA	DA	Proto	Src	Dst

OpenFlow Flow Table Abstraction



MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
*	*	*	5.6.7.8	*	*	port 1

Network Slicing Architecture

- Network Slice is a collection of sliced switches / routers
  - Data plane is unmodified, thus Packets forwarded without performance penalty
  - Slicing by exploiting the existing Data Path ASIC
- Transparent Slicing Layer:
  - each slice believes it owns the data path
  - enforces isolation between slices
  - Rewrites / drops rules to adhere to slice police
  - forwards exceptions to correct slice(s)

Network Slicing Architecture

Existing Network Device

Switch / Router

Control Plane

- Computes forwarding rules
  - “128.8.128/16 --> port 6”
- Pushes rules down to data plane

General-purpose CPU

Rules

Control/Data Protocol

Excepts

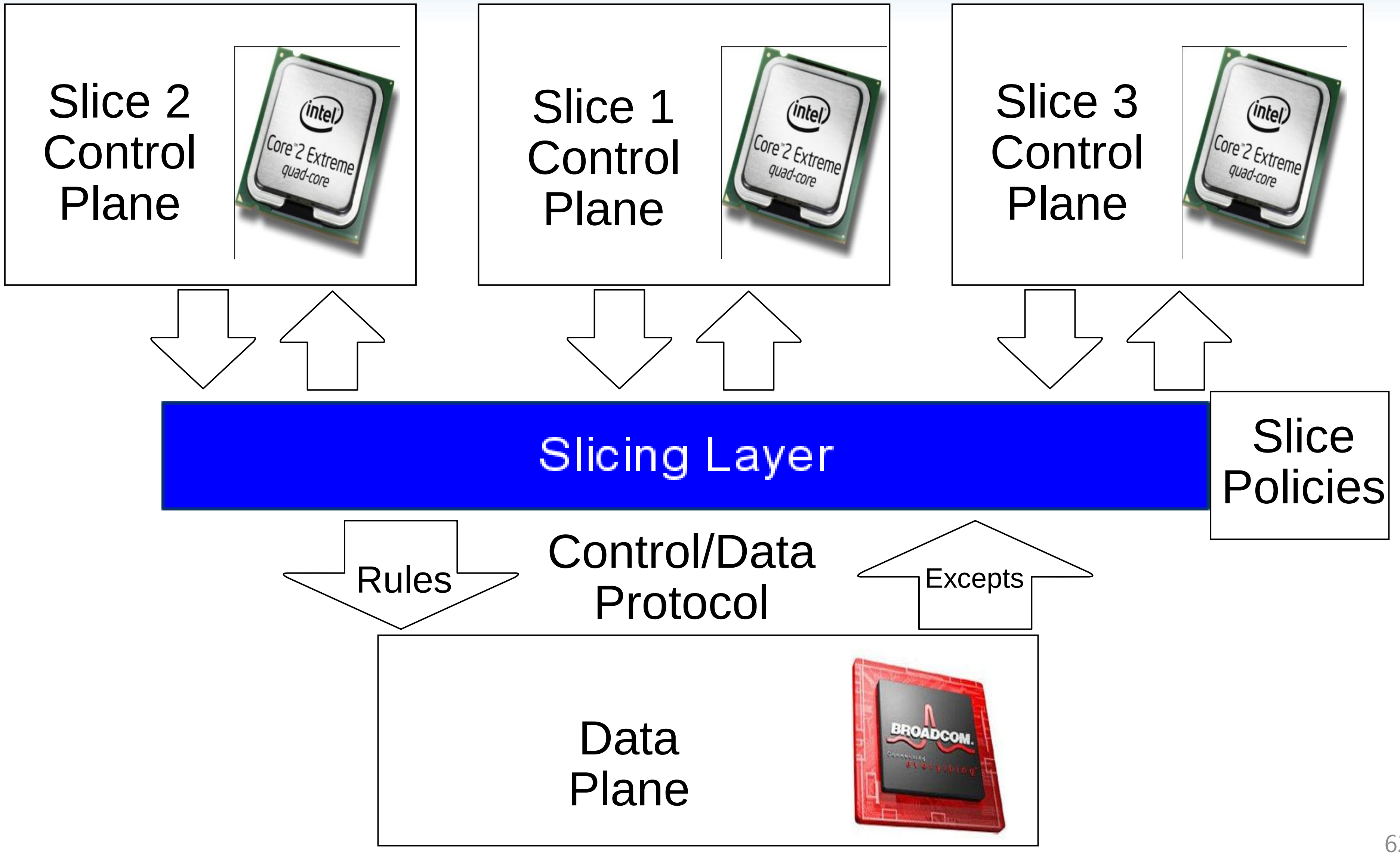
Data Plane

- Enforces forwarding rules
- Exceptions pushed back to control plane
  e.g., unmatched packets

Custom ASIC

Network Slicing Architecture

Add the Slicing Layer Between Planes



Network Slicing Architecture

Slicing Policies

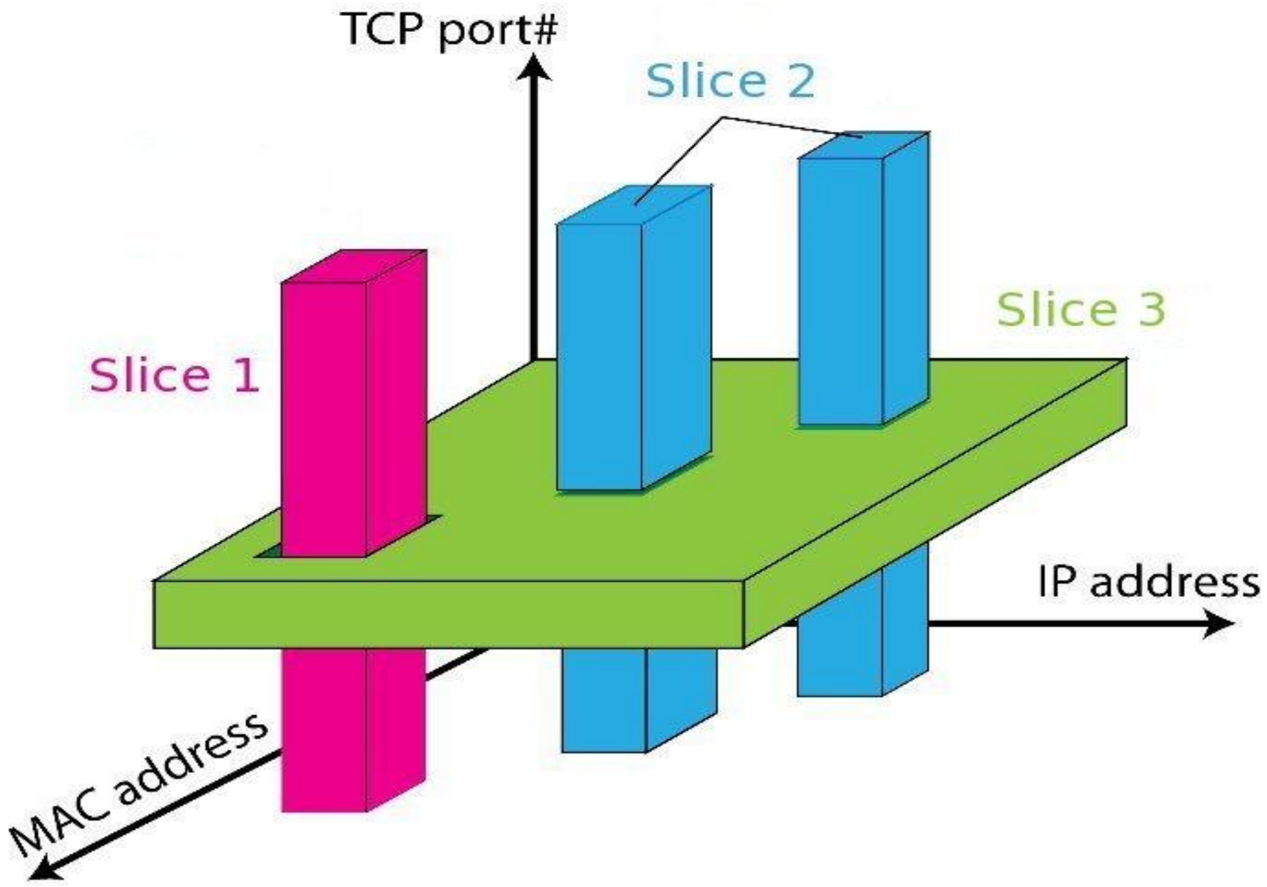
The policy specifies resource limits for each slice:

- Link bandwidth
- Maximum number of forwarding rules
- Topology
- Fraction of switch / router CPU

Network Slicing Architecture

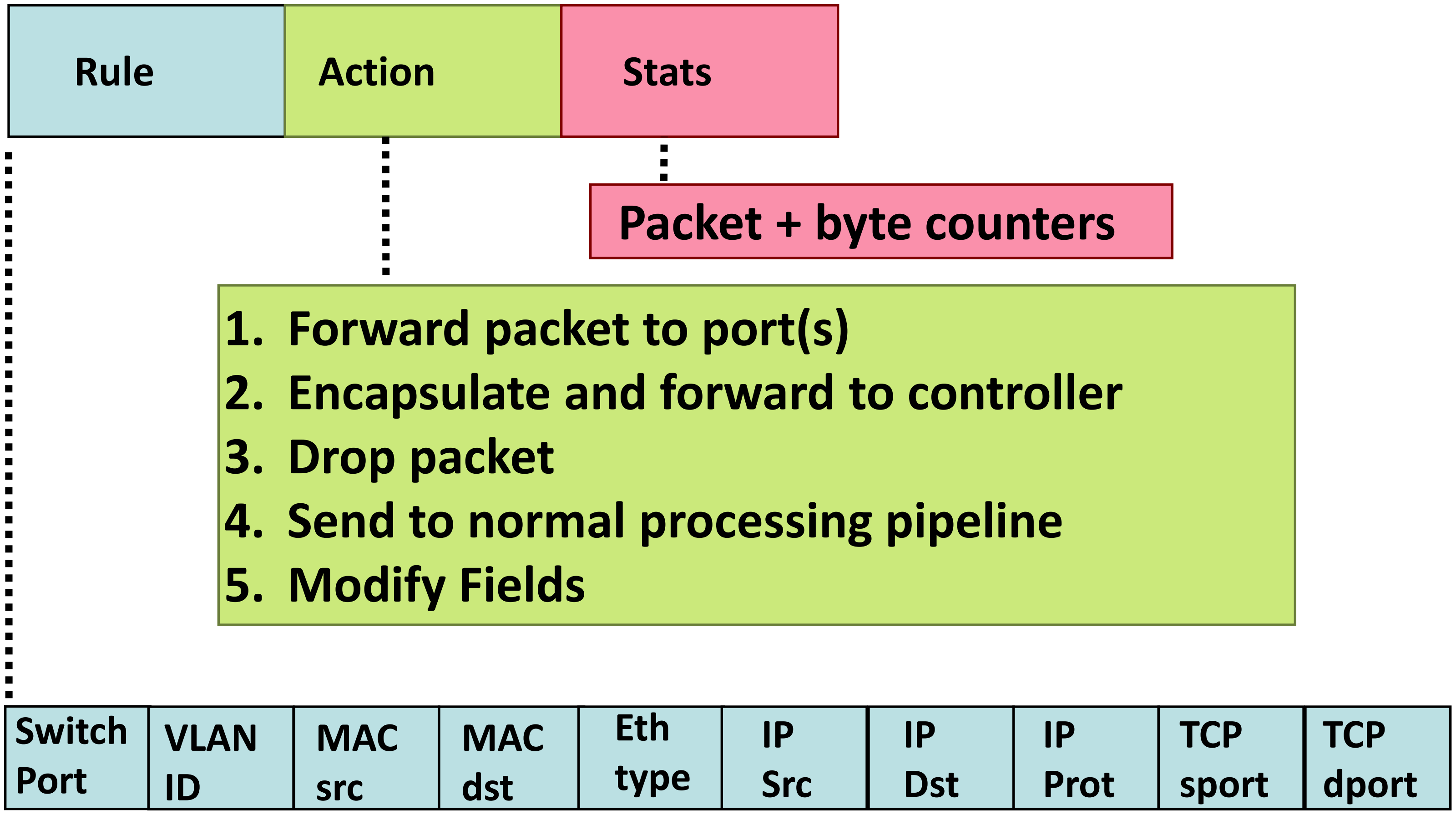
FlowSpace: Map Packets to Slices

Which packets does the slice control?



Network Slicing Architecture

Flow Table Entries



1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

+ mask what fields to match

Network Slicing Architecture

Examples: Flow Table Entries

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

Network Slicing Architecture
Examples: Flow Table Entries

Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

Network Slicing Architecture
What is OpenFlow-enabled switch? #1

Some commercial switches, routers and access points enhanced with the OpenFlow Feature by adding the Flow Table, Secure Channel and OpenFlow Protocol.

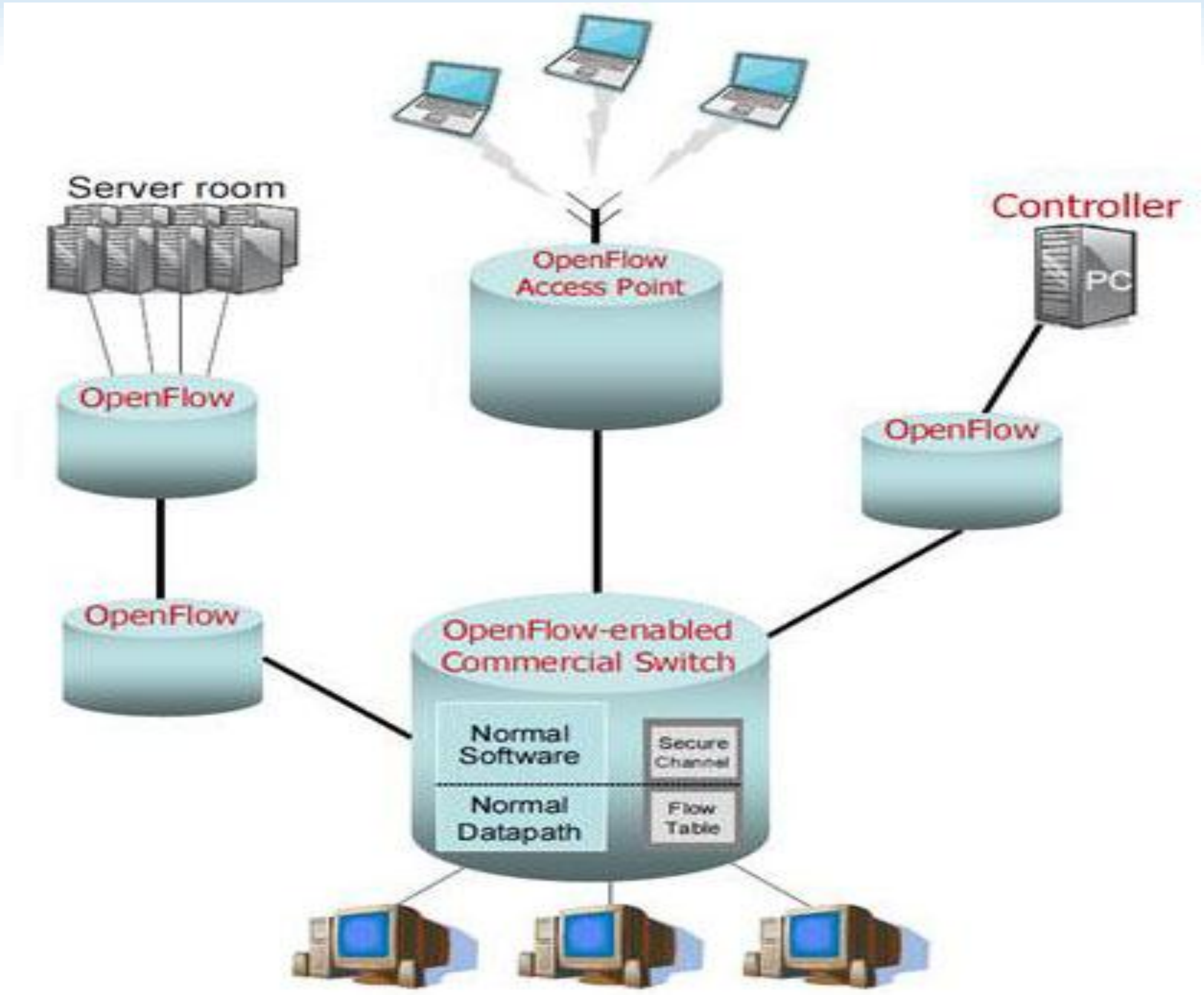
- Typically, the Flow Table will reuse existing hardware, such as a TCAM(Ternary Content Addressable Memory);
- the Secure Channel and Protocol will be ported to run on the switch's operating system.

Network Slicing Architecture
What is OpenFlow-enabled switch? #2

- All the Flow Tables are managed by the controller
- OpenFlow Protocol allows a switch to be controlled by two or more controllers for increased performance or robustness.

Network Slicing Architecture

OpenFlow enabled commercial switches and routers



Network Slicing Architecture
How to isolate experimental traffic ?

- Goal is to enable experiments to take place in an existing real network alongside regular traffic and applications. to win the confidence of network administrators,
- Therefore, OpenFlow-enabled switches must isolate experimental traffic from real service traffic
- experimental traffic processed by the Flow Table, but real service traffic processed by the normal Layer 2 and Layer 3 pipeline of the switch
- .

Network Slicing Architecture
What is OpenFlow's 4th action?

- There are two ways to achieve this separation.
- One is to add a fourth action:
  4) Forward this flow's packets through the switch's normal processing pipeline.
- The other is to define separate sets of VLANs for experimental and real traffic.
- All OpenFlow-enabled switches support one approach or the other or both.

Network Slicing Architecture
What is "Type 1" switch?

We expect that many switches will support additional actions, for example

- Rewrite portions of the packet header, (e.g., for NAT, or to obfuscate addresses on intermediate links),
- Map packets to a priority class.
- Likewise, some Flow Tables will be able to match on arbitrary fields in the packet header, enabling experiments with new non-IP protocols.

As a particular set of features emerges, we will define a "Type 1" switch.

Network Slicing Architecture
What is Controller?

- A controller adds and removes flow-entries from the Flow Table on behalf of experiments.
- For example, a static controller might be a simple application running on a PC to statically establish flows to interconnect a set of test computers for the duration of an experiment.
- In this case, the flows resemble VLANs in current networks—providing a simple mechanism to isolate experimental traffic from the production network.
- Viewed this way, OpenFlow is a generalization of VLANs.

Network Slicing Architecture
What is Controller?

- Sophisticated controllers that dynamically add/remove flows as an experiment progresses.

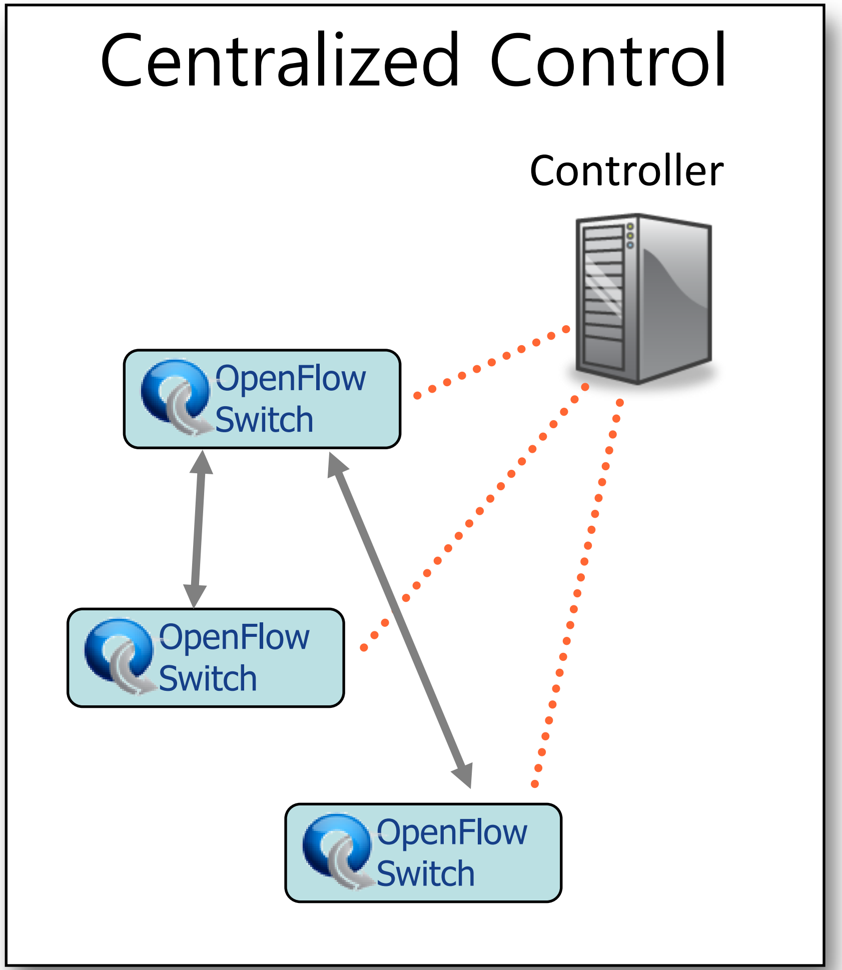
- By using the controller, a researcher might control the complete network of OpenFlow Switches and be free to decide how all flows are processed.

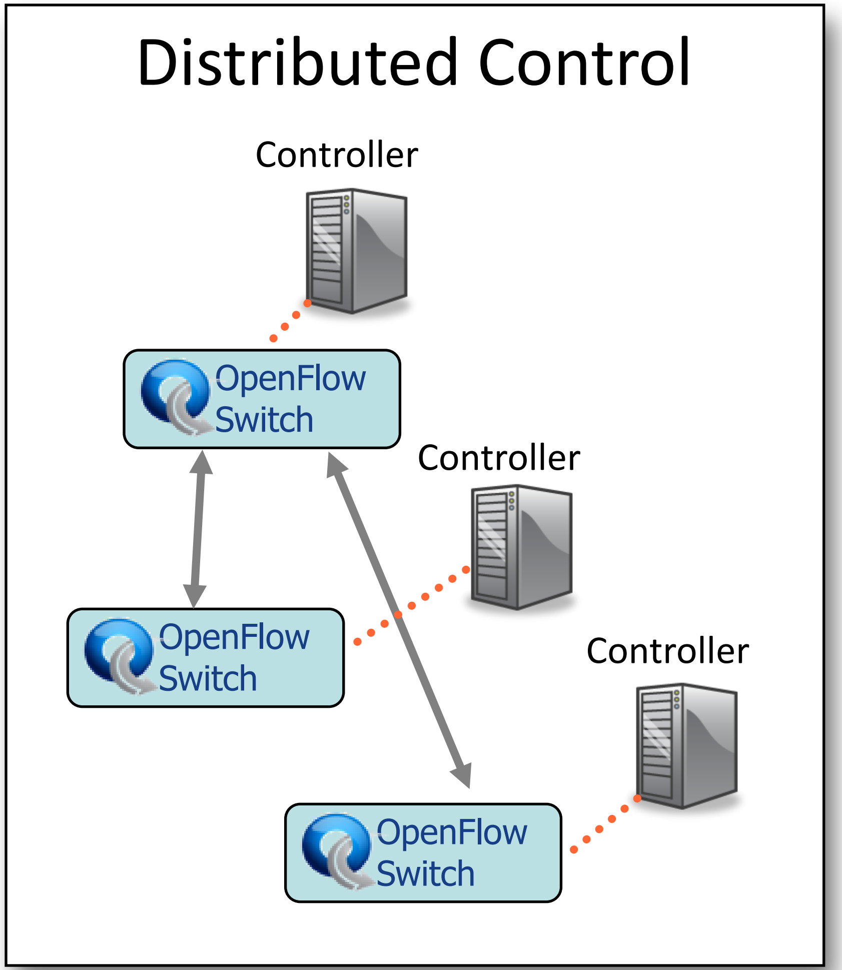
Network Slicing Architecture
What is Controller? #2

- Support multiple researchers, each with different accounts and permissions,
- enabling them to run multiple independent experiments on different sets of flows.
- Under the control of a particular researcher by a policy table running in a controller, Flows could be delivered to a researcher’s user-level control program
- In which it decides if a new flow-entry should be added to the network of switches.

Network Slicing Architecture

Centralized vs Distributed Control





Network Slicing Architecture
Flow Routing vs. Aggregation

Flow-Based

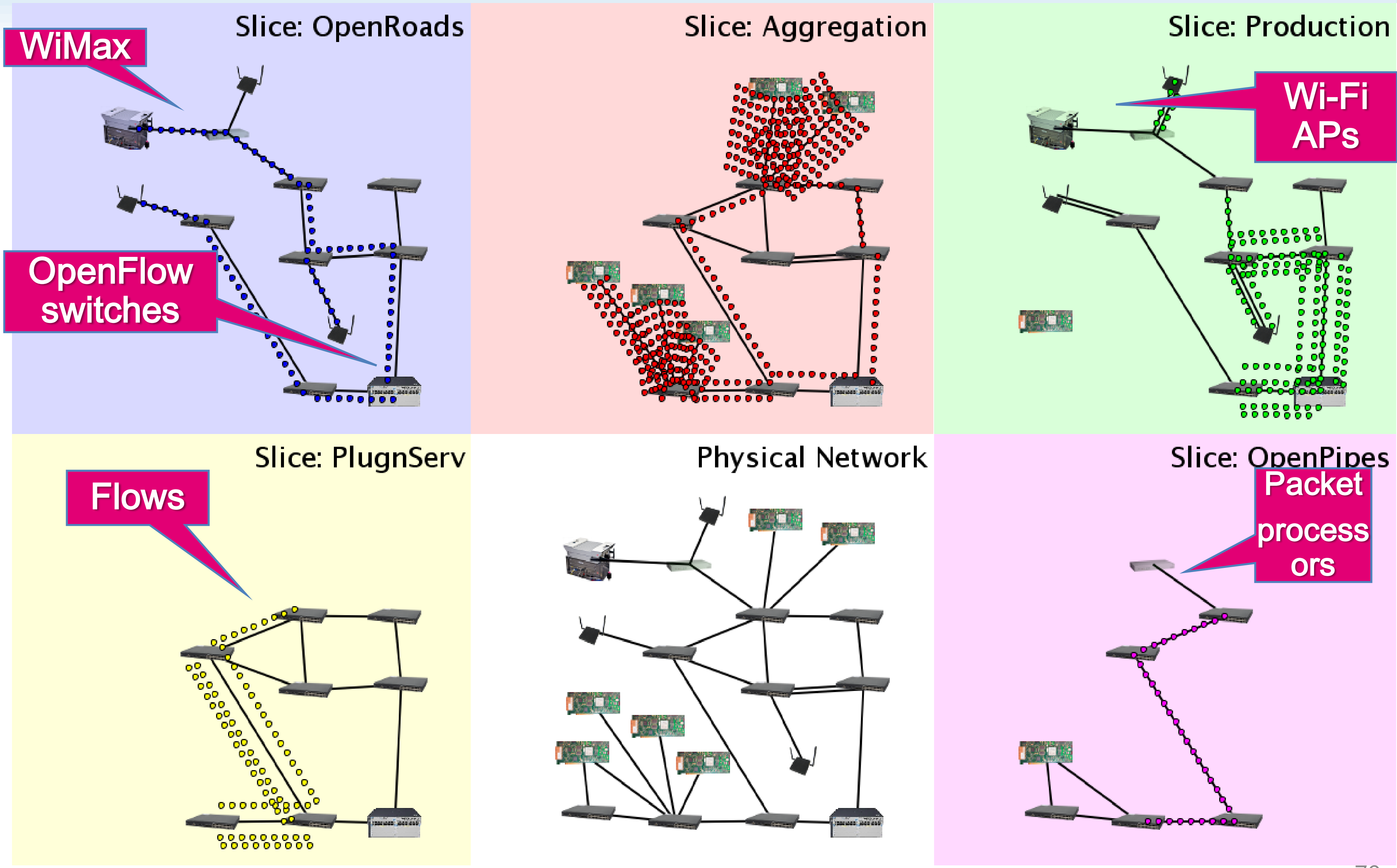
- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control,
- e.g. campus networks

Aggregated

- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows,
- e.g. backbone

Network Slicing Architecture

Stanford Infrastructure



Network Slicing Architecture
Reactive vs. Proactive

Reactive

- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggre gated (wildcard) rules

Network Slicing Architecture

Experiment Setup Overview

Step 1:
Write/Configure/Deploy
OpenFlow controller

- Each controller implements per-experiment custom forwarding logic
- Write your own or download pre-existing

Step 2:
Create Slice and
register experiment

- Configure per-experiment topology, queuing restricted to subset of real topology
- Specify desired user traffic: e.g., tcp.port=80

Step 3:
Control the traffic of
users that opt-in to
your experiment

- Users opt-in via the Opt-In Manager website
- Reserving a compute node makes the experi menter a user on the network

Network Slicing Architecture
Experiment :Amy-OSPF #1

- As a simple example of how an OpenFlow Switch might be used imagine that Amy (a researcher) invented Amy-OSPF as a new routing protocol to replace OSPF(Open Shortest Path First) IP routing protocol.
- She wants to try her protocol in a network of OpenFlow Switches, without changing any end-host software.

Network Slicing Architecture
Experiment :Amy-OSPF #2

- Amy-OSPF will run in a controller;
- each time a new application flow starts Amy-OSPF to pick a route through a series of OpenFlow Switches, and adds a flow- entry in each switch along the path.
- In her experiment, Amy decides to use Amy-OSPF for the traffic entering the OpenFlow network from her own desktop PC
- so she doesn’t disrupt the network for others.

Network Slicing Architecture

Experiment :Amy-OSPF #3

- To do this, she defines one flow to be all the traffic entering the Open-Flow switch through the switch port her PC is connected to,
- and adds a flow-entry with the action “Encapsulate and forward all packets to a controller”.
- When her packets reach a controller, her new protocol chooses a route and adds a new flow-entry for the application flow to every switch along the chosen path.
- When subsequent packets arrive at a switch, they are processed quickly and at line-rate by the Flow Table.

Network Slicing Architecture
Questions of OpenFlow Switch ?

- There are legitimate questions to ask about the performance, reliability and scalability of a controller that dynamically adds and removes flows as an experiment progresses:
- Can such a centralized controller be fast enough to process new flows and program the Flow Switches?
- What happens when a controller fails?

Network Slicing Architecture

Performance of Ethane prototype #1

- To some extent these questions were addressed in the context of the Ethane prototype,
- which used simple flow switches and a central controller.
- Preliminary results suggested that an Ethane controller based on a low-cost desktop PC could process over 10,000 new flows per second.
- enough for a large scale college campus.

Network Slicing Architecture
Performance of Ethane prototype #2

- Of course, the rate at which new flows can be processed will depend on the complexity of the processing required by the researcher’s experiment.
- But it gives us confidence that meaningful experiments can be run.
- Scalability and redundancy are possible by making a controller (and the experiments) stateless, allowing simple load-balancing over multiple separate devices.

Network Slicing Architecture
Two additional properties of OpenFolw #1

- Chances are, Amy is testing her new protocol in a network used by lots of other people. We therefore want the network to have two additional properties:

- 1. Packets belonging to users other than Amy should be routed using a standard and tested routing protocol running in the switch or router from a "name-brand" vendor.
- 2. Amy should only be able to add flow entries for her traffic, or for any traffic her network administrator has allowed her to control.

Network Slicing Architecture

Two additional properties of OpenFolw #2

- Property 1 is achieved by OpenFlow-enabled switches.
- In Amy’s experiment, the default action for all packets that don’t come from Amy’s PC could be to forward them through the normal processing pipeline.
- Amy’s own packets could be forwarded directly to the outgoing port, without being processed by the normal pipeline.

Network Slicing Architecture

Two additional properties of OpenFolw #3

- Property 2 depends on the controller.
- The controller should be seen as a platform that enables researchers to implement various experiments,
- Property 2 can be achieved with the appropriate use of permissions or other ways to limit the powers of individual researchers to control flow entries.

Network Slicing Architecture

Two additional properties of OpenFolw #4

- The exact nature of these permission-like mechanisms will depend on how the controller is implemented. – Controller is almighty!
- We expect that a variety of controllers will emerge.
- As an example of a concrete realization of a controller, some of the authors are working on a controller called NOX as a follow-on to the Ethane work.
- A quite different controller might emerge by extending the GENI management software to OpenFlow networks.

Network Slicing Architecture
More Experiments of OpenFlow

- As with any experimental platform, the set of experiments will exceed those we can think of up-front
- most experiments in OpenFlow networks are yet to be thought of.
- Here, for illustration, we offer some examples of how OpenFlow-enabled networks could be used to experiment with new network applications and architectures.

Network Slicing Architecture
Network Management and Access Control #1

- We'll use Ethane as our first example as it was the research that inspired OpenFlow.
- In fact, an OpenFlow Switch can be thought of as a generalization of Ethane's datapath switch.
- Ethane used a specific implementation of a controller, suited for network management and control, that manages the admittance and routing of flows

Network Slicing Architecture
Network Management and Access Control #2

- The basic idea of Ethane is to allow network managers to define a network-wide policy in the central controller, which is enforced directly by making admission control decisions for each new flow.
- A controller checks a new flow against a set of rules, such as “Guests can communicate using HTTP, but only via a web proxy” or “VoIP phones are not allowed to communicate with laptops.”

Network Slicing Architecture

Network Management and Access Control #3

- A controller associates packets with their senders by managing all the bindings between names and addresses
- it essentially takes over DNS, DHCP(Dynamic Host Configuration Protocol), and authenticates all users when they join, keeping track of which switch port or access point they are connected to.
- One could envisage an extension to Ethane in which a policy dictates that particular flows are sent to a user's process in a controller,
- hence allowing researcher-specific processing to be performed in the network.

Network Slicing Architecture VLANs

- OpenFlow can easily provide users with their own isolated network, just as VLANs do.
- The simplest approach is to statically declare a set of flows which specify the ports accessible by traffic on a given VLAN ID.
- Traffic identified as coming from a single user
- for example, originating from specific switch ports or MAC addresses is tagged by the switches via an action with the appropriate VLAN ID.
- A more dynamic approach might use a controller to manage authentication of users and use the knowledge of the users' locations for tagging traffic at runtime.

Network Slicing Architecture
Mobile wireless VoIP clients

- For this example consider an experiment of a new call- handoff mechanism for WiFi-enabled phones.
- In the experiment VOIP clients establish a new connection over the OpenFlow- enabled network.
- A controller is implemented to track the location of clients, re-routing connections by reprogramming the Flow Tables
- as users move through the network, allowing seamless handoff from one access point to another.

Network Slicing Architecture
non-IP network #1

- So far, our examples have assumed an IP network, but OpenFlow doesn’t require packets to be of any one format
- so long as the Flow Table is able to match on the packet header.
- This would allow experiments using new naming, addressing and routing schemes.
- There are several ways an OpenFlow-enabled switch can support non-IP traffic.

Network Slicing Architecture

non-IP network #2

- For example, flows could be identified using their Ethernet header (MAC src and dst addresses), a new EtherType value, or at the IP level, by a new IP Version number.
- More generally, we hope that future switches will allow a controller to create a generic mask
- (offset + value + mask), allowing packets to be processed in a researcher-specified way

Network Slicing Architecture
Processing packets rather than flows #1

- The examples above are for experiments involving flows —where a controller makes decisions when the flow starts.
- There are, of course, interesting experiments to be performed that require every packet to be processed.
- For example, an intrusion detection system that inspects every packet,
- an explicit congestion control mechanism,
- when modifying the contents of packets, such as when converting packets from one protocol format to another.

Network Slicing Architecture
Processing packets rather than flows #2

- There are two basic ways to process packets in an OpenFlow-enabled network.

- First, and simplest, is to force all of a flow’s packets to pass through a controller.
- To do this, a controller doesn’t add a new flow entry into the Flow Switch — it just allows the switch to default to forwarding every packet to a controller.
- This has the advantage of flexibility, at the cost of performance.
- It might provide a useful way to test the functionality of a new protocol, but is unlikely to be of much interest for deployment in a large scale network.

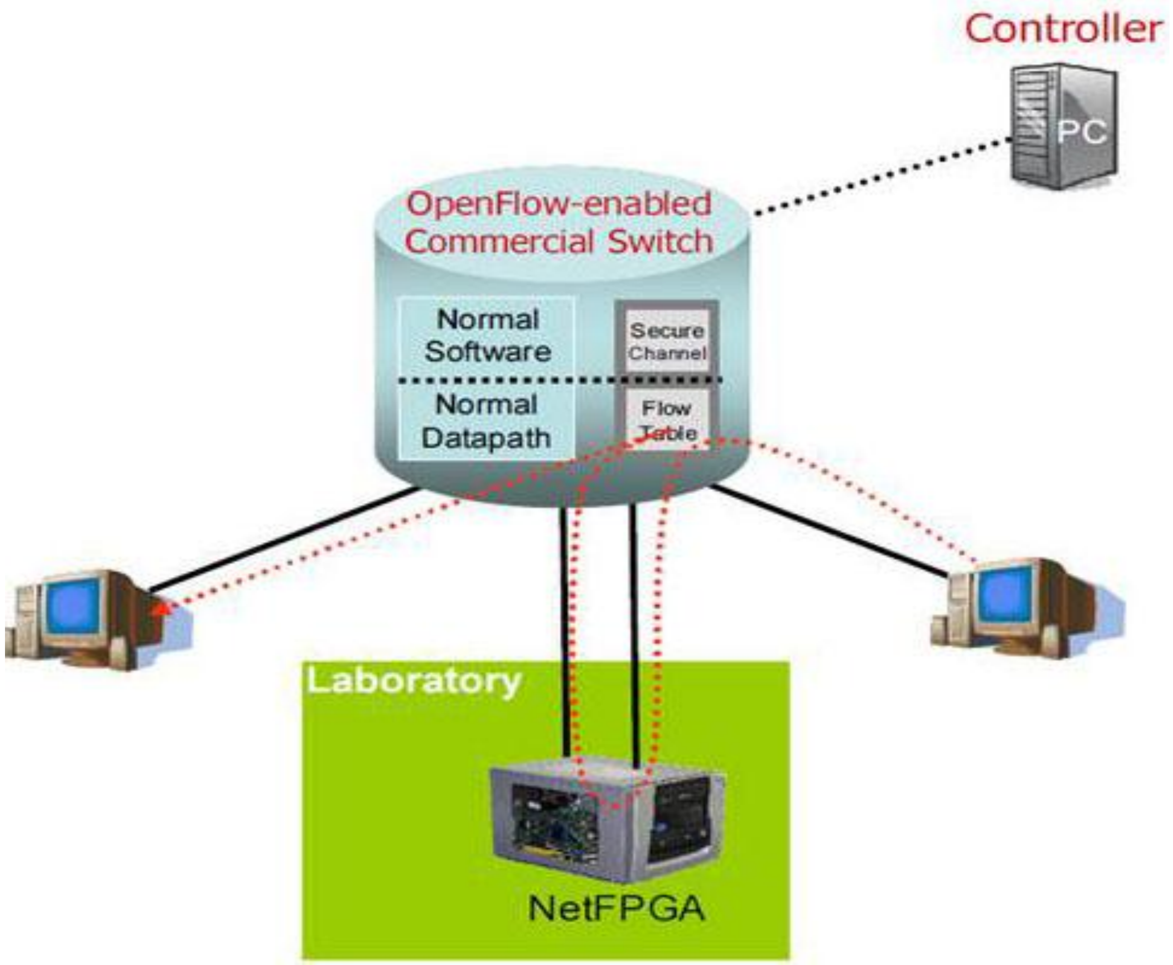
Network Slicing Architecture
Processing packets rather than flows #1

- The second way to process packets is to route them to a programmable switch that does packet processing — for example, a NetFPGA-based programmable router.
- The advantage is that the packets can be processed at line-rate in a user-definable way;
- OpenFlow-enabled switch operates essentially as a patch-panel to allow the packets to reach the NetFPGA.
- In some cases, the NetFPGA board, a PCI board that plugs into a Linux PC, might be placed in the wiring closet alongside the OpenFlow-enabled switch, or more likely in a laboratory.

Network Slicing Architecture

Processing packets rather than flows #1

through an external line-rate packet-processing device, such as a programmable NetFPGA router



Network Slicing Architecture
VLAN Based Partitioning

Partition Flows based on Ports and VLAN Tags

- Traffic entering system (e.g. from end hosts) is tagged
- VLAN tags consistent throughout substrate

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport
*	*	*	*	1,2,3	*	*	*	*	*
	*	*	*	4,5,6	*	*	*	*	*
*	*	*	*	7,8,9	*	*	*	*	*

Network Slicing Architecture
New CDN - Turbo Coral ++

- Basic Idea: Build a CDN where you control the entire network
  - All traffic to or from Coral IP space controlled by Experimenter
  - All other traffic controlled by default routing
  - Topology is entire network
  - End hosts are automatically added (no opt-in)

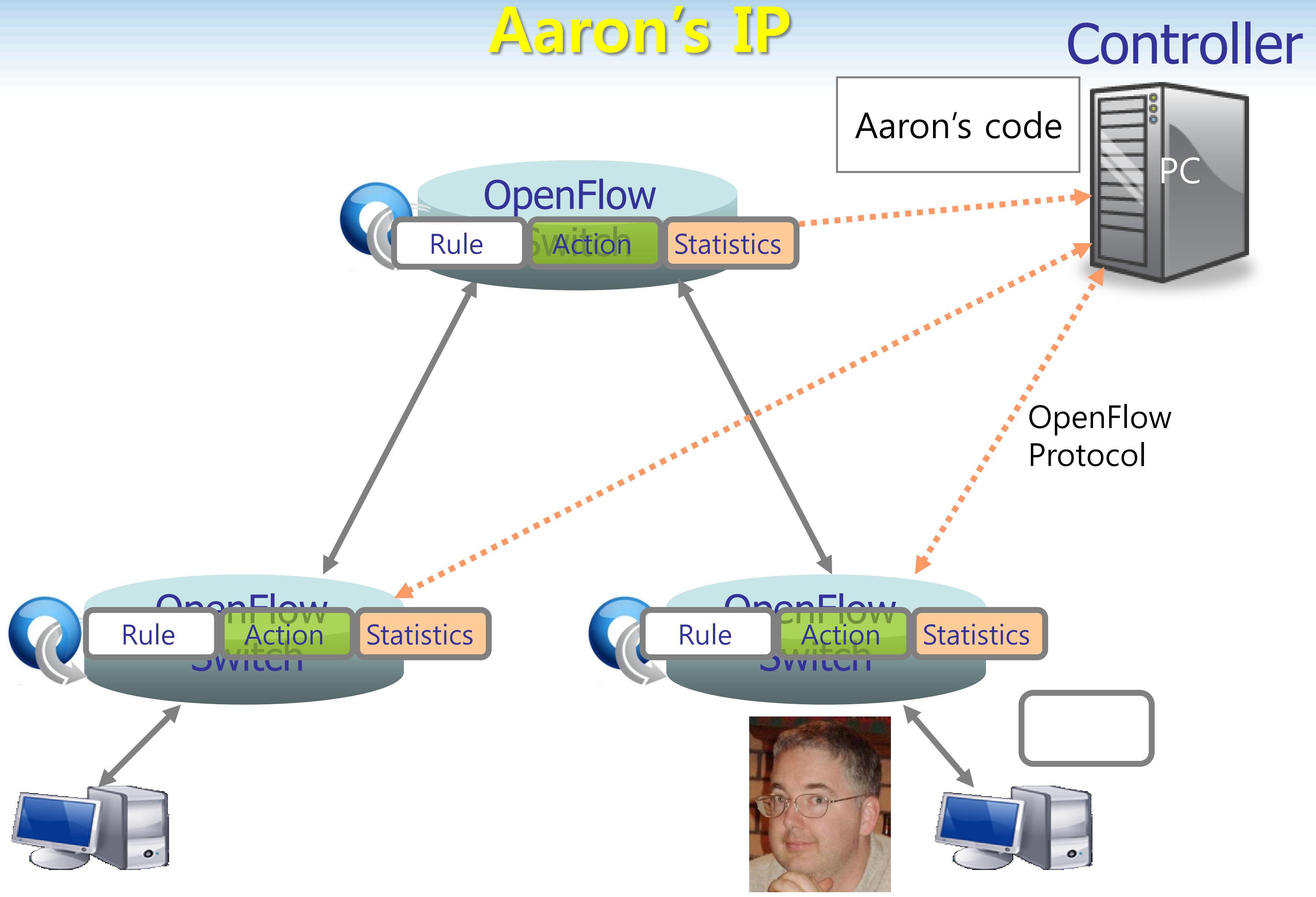
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport
*	*	*	*	*	84.65.*	*	*	*	*
*	*	*	*	*	*	84.65.*	*	*	*
*	*	*	*	*	*	*	*	*	*

Network Slicing Architecture Aaron's IP

- A new layer 3 protocol
- Replaces IPv4, IPv6, …
- Defined by a new Ether Type

	Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport
	*	*	*	AaIP	*	*	*	*	*	*
	*	*	*	!AaIP	*	*	*	*	*	*

Network Slicing Architecture



OpenFlow Consortium #1

- The OpenFlow Consortium aims to popularize OpenFlow and maintain the OpenFlow Switch Specification.
- The Consortium is a group of researchers and network administrators at universities and colleges who believe their research mission will be enhanced if OpenFlow-enabled switches are installed in their network.
- Membership is open and free for anyone at a school, college, university, or government agency worldwide.

OpenFlow Consortium #2

- The OpenFlow Consortium welcomes individual members who are not employed by companies that manufacture or sell Ethernet switches, routers or wireless access points, because we want to keep the consortium free of vendor influence.

- To join, send email to join@OpenFlowSwitch.org.

- The Consortium web-site http://www.OpenFlowSwitch.org contains the OpenFlow Switch Specification, a list of consortium members, and reference implementations of OpenFlow switches.

OpenFlow - Licensing Model

- The OpenFlow Switch Specification is free for all commercial and non-commercial use.
- The exact wording is on the web-site.
- Commercial switches and routers claiming to be "OpenFlow-enabled" must conform to the requirements of an OpenFlow Type 0 Switch,
- as defined in the OpenFlow Switch Specification.
- OpenFlow is a trademark of Stanford University, and will be protected on behalf of the Consortium

Market Opportunity #1

- There is an interesting market opportunity for network equipment vendors to sell OpenFlow-enabled switches to the research community.
- Every building in thousands of colleges and universities contains wiring closets with Ethernet switches and routers, and with wireless access points spread across campus.
- Several switch and router manufacturers who are adding the OpenFlow feature to their products by implementing a Flow Table in existing hardware; i.e. no hardware change is needed.
- The switches run the Secure Channel software on their existing processor

Market Opportunity #2

- Network equipment vendors to be very open to the idea of adding the OpenFlow feature.
- Most vendors would like to support the research community without having to expose the internal workings of their products.
- Large scale OpenFlow at Stanford University.
- Eventually, all traffic runs over the OpenFlow network, with production traffic and experimental traffic being isolated on different VLANs under the control of network administrators.

Market Opportunity #3

- Researchers can control their own traffic, and be able to add/remove flow-entries.
- Many different OpenFlow Switches developed by the research community.
- OpenFlow Website contains "Type 0" reference designs for several different platforms: Linux (software), OpenWRT (software, for access points), and NetFPGA (hardware, 4-ports of 1GE).
- As more reference designs are created by the community we will post them.
- The forum encourage developers to test their switches against the reference designs.
- All reference implementations of OpenFlow switches posted on the web site will be open-source and free for commercial and non-commercial use.

OpenFlow Vendor Hardware

Prototype ⟷ Product

Core Router

Cisco Catalyst 6k (prototype)

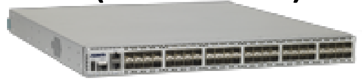
Juniper MX-series (prototype)

HP ProCurve 5400 and others

Enterprise Campus Data Center

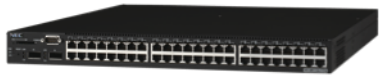
Cisco Catalyst 3750 (prototype)

Arista 7100 series (Q4 2010)



Pronto

NEC IP8800



Circuit Switch

Ciena CoreDirector

Wireless

WiMAX (NEC)

more to follow…

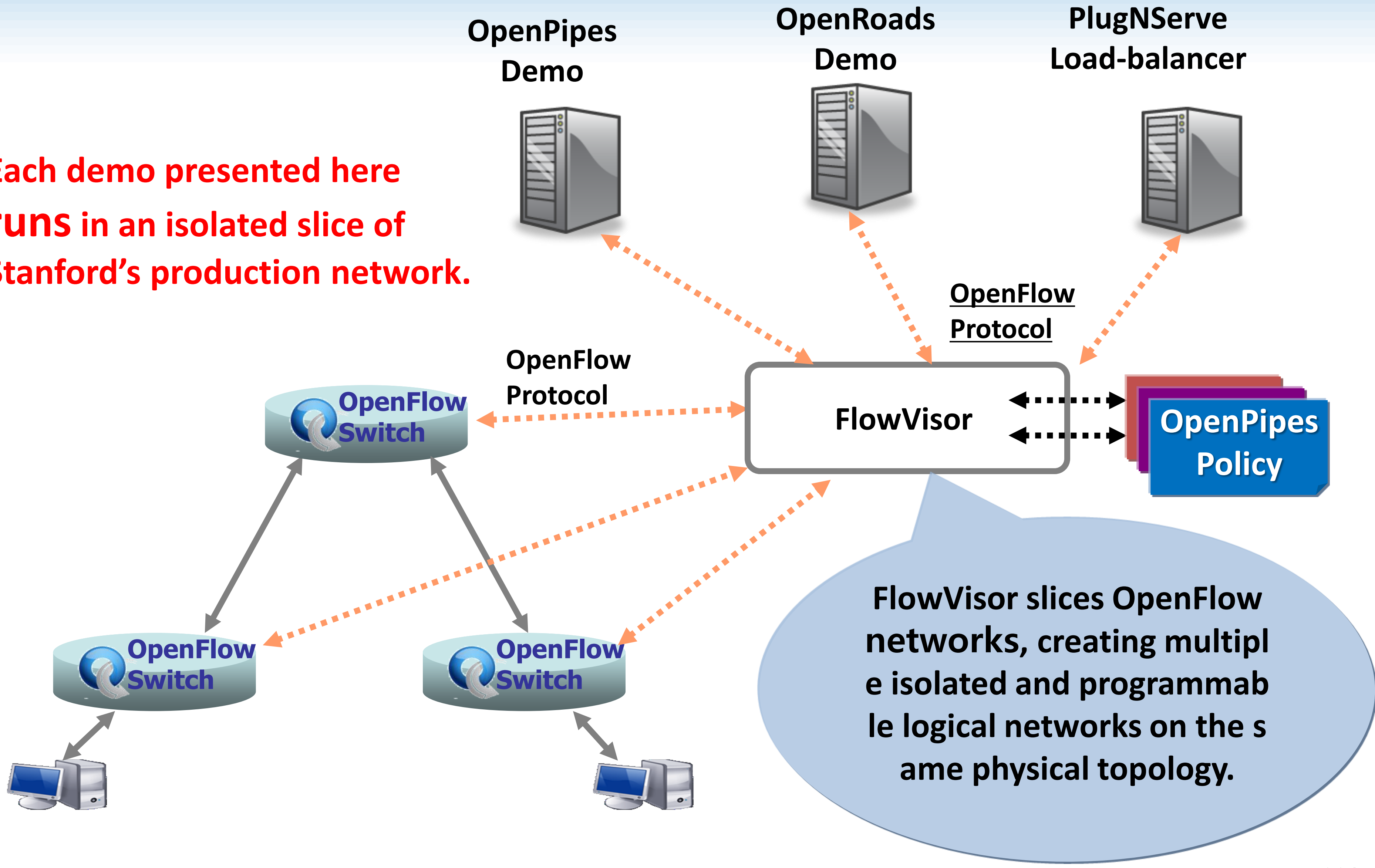
OpenFlow Demonstration Overview

Topic	Demo
Network Virtualization	FlowVisor
Hardware Prototyping	OpenPipes
Load Balancing	PlugNServe
Energy Savings	ElasticTree
Mobility	MobileVMs
Traffic Engineering	Aggregation
Wireless Video	OpenRoads

Network Slicing Architecture Demo

FlowVisor : Creates Virtual Networks

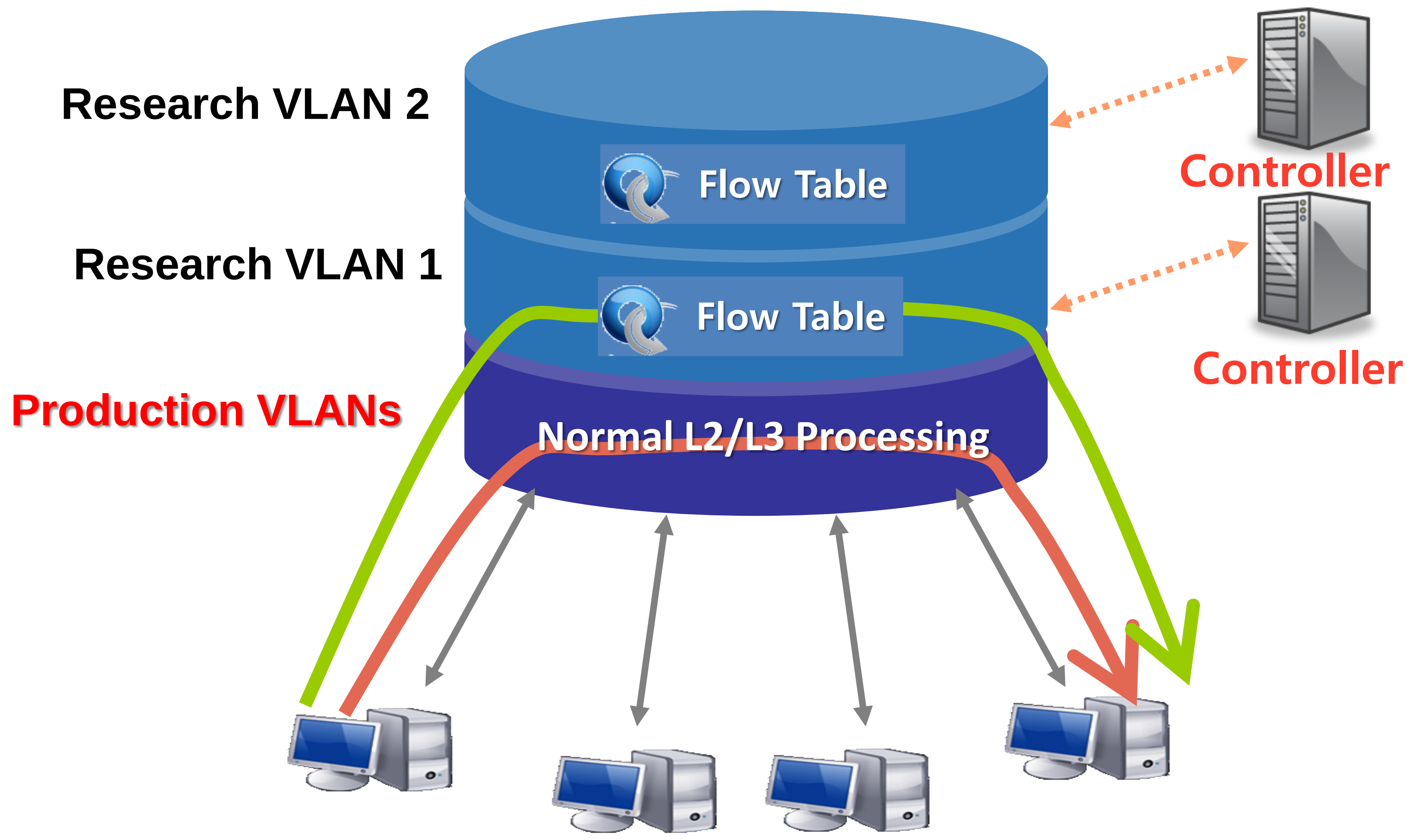
Each demo presented here **runs** in an isolated slice of Stanford's production network.



FlowVisor slices OpenFlow networks, creating multipl e isolated and programmab le logical networks on the s ame physical topology.

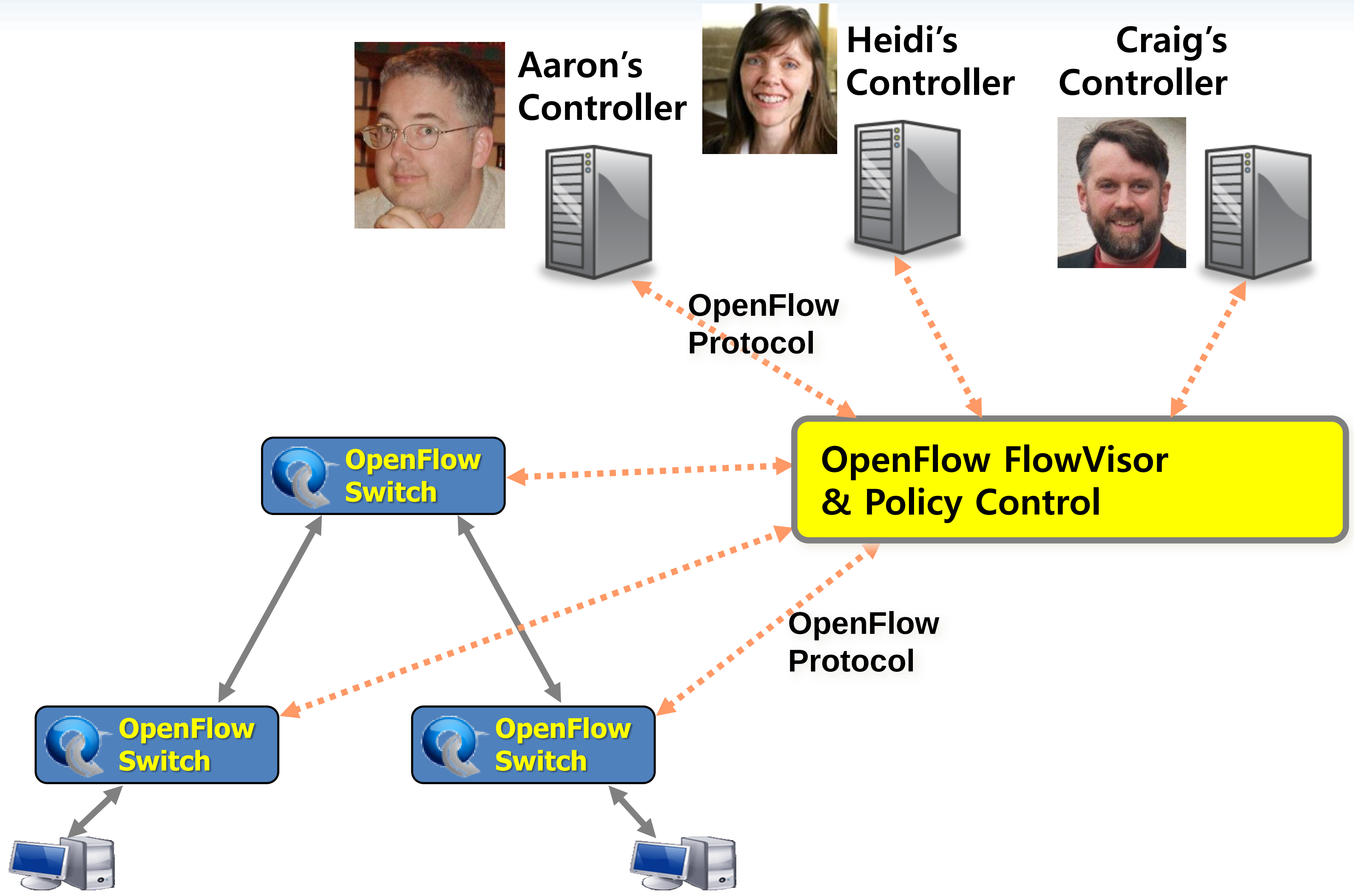
Network Slicing Architecture Demo

FlowVisor : Creates Virtual Networks



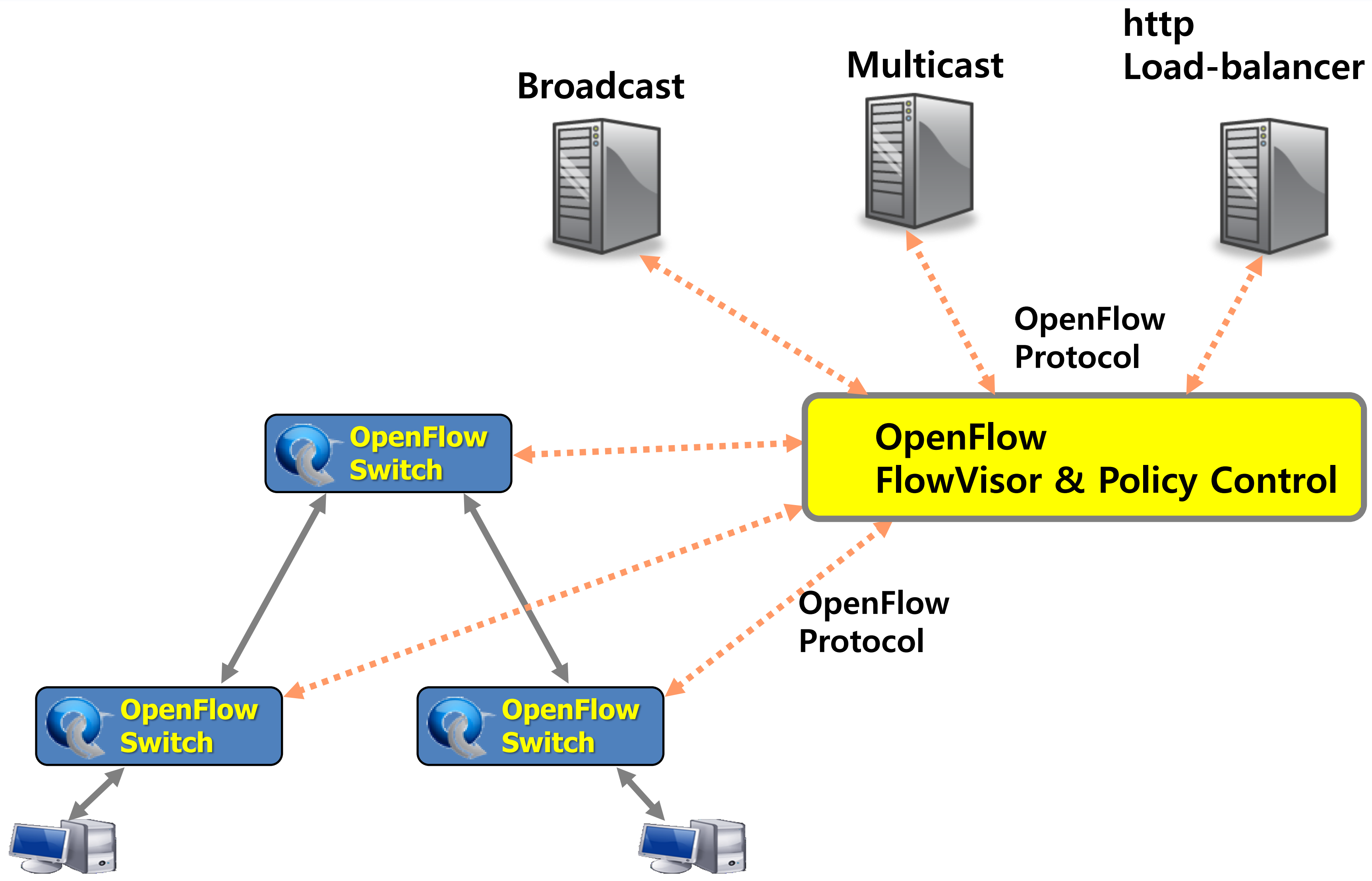
Network Slicing Architecture Demo

FlowVisor: Creates Virtual Networks



Network Slicing Architecture Demo

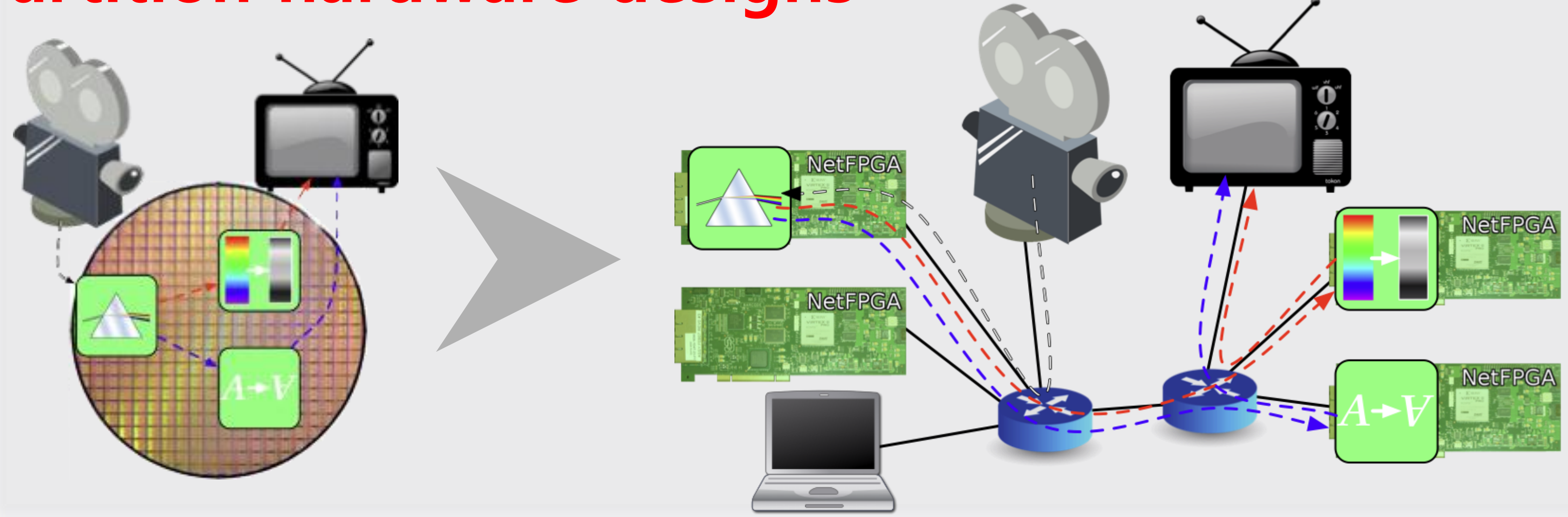
FlowVisor：Creates Virtual Networks, Separation not only by VLANs, but any L1-L4 pattern



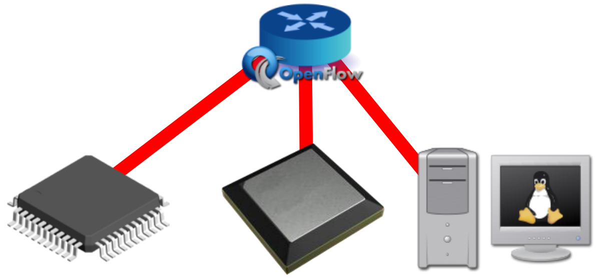
Network Slicing Architecture Demo

OpenPipes : Plumbing with OpenFlow to build hardware systems

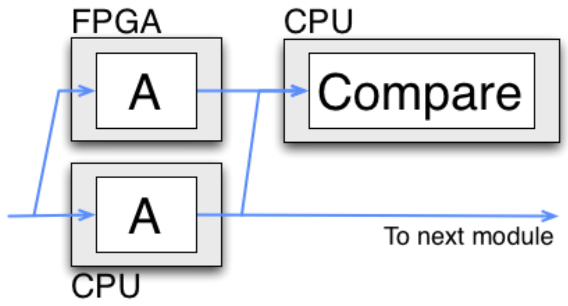
Partition hardware designs



Mix resources



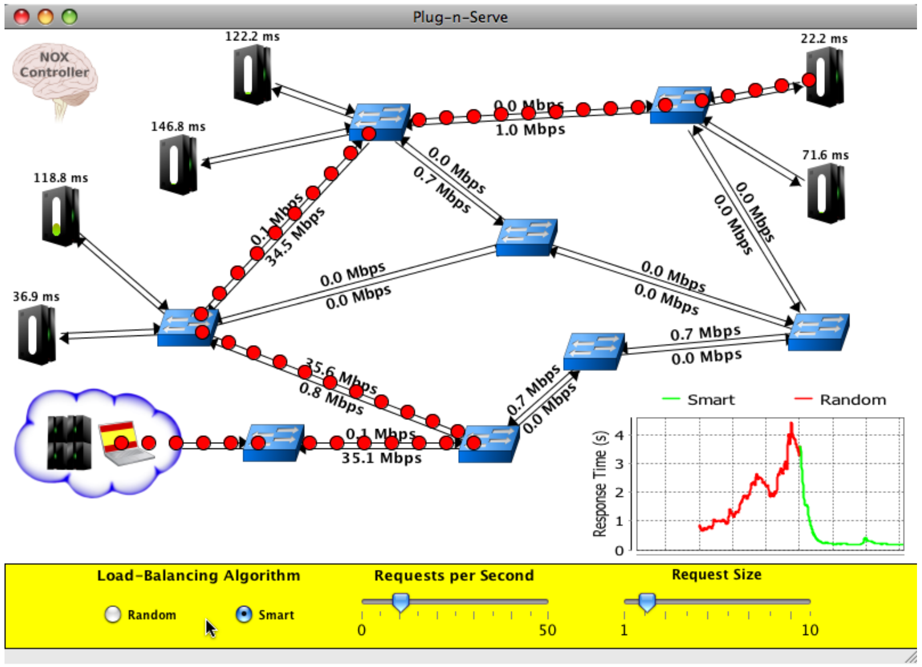
Test



Network Slicing Architecture Demo

Plug-n-Serve: Load-Balancing Web Traffic

Goal: Load-balancing requests in unstructured networks



Load-Balancing Algorithm: Random [✓ Smart]

Requests per Second: 0 – 50

Request Size: 1 – 10

What we are showing

- OpenFlow-based distributed load-balancer
- Smart load-balancing based on network and server load
- Allows incremental deployment of additional resources

OpenFlow means…

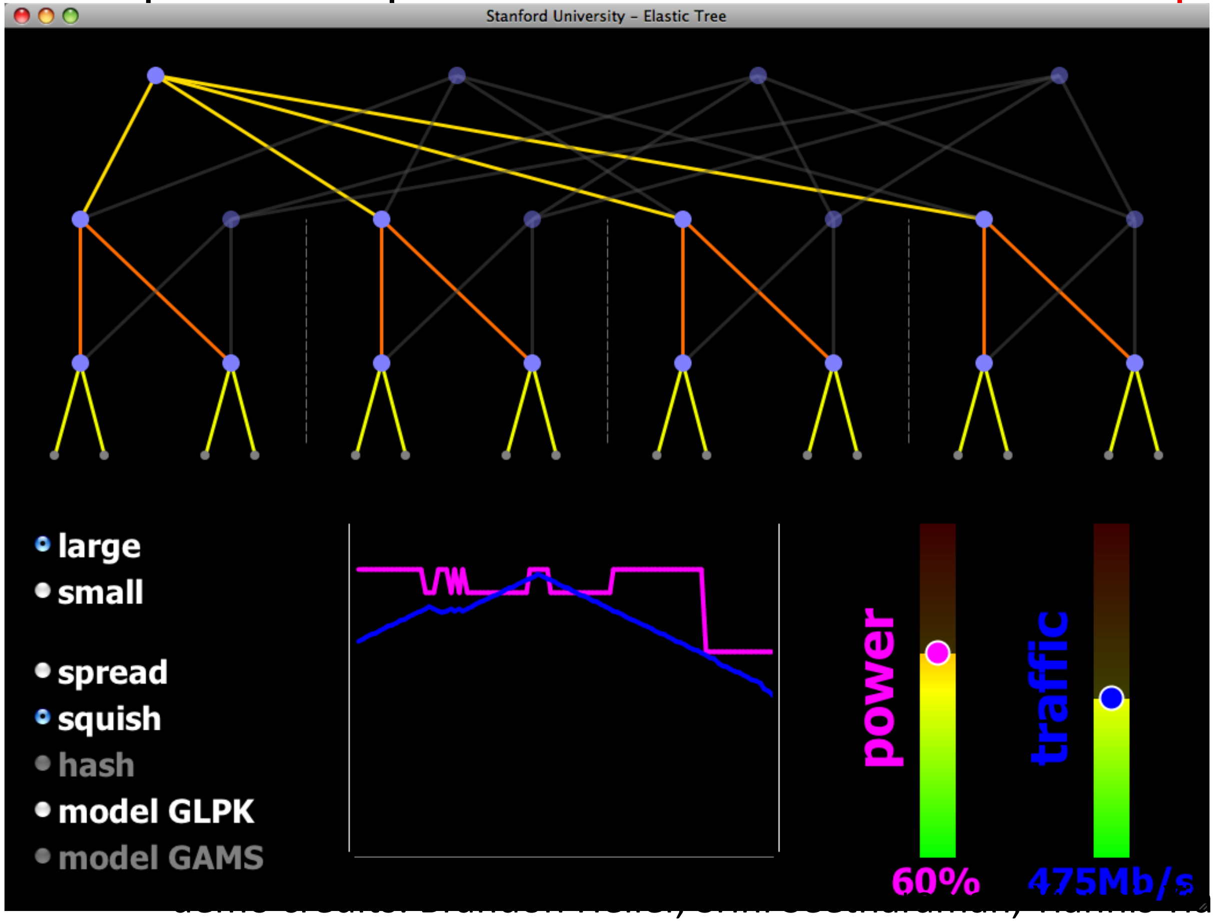
- Complete control over traffic within the network
- Visibility into network conditions
- Ability to use existing commodity hardware

This demo runs on top of the FlowVisor, sharing the same physical network with other experiments and production traffic.

Network Slicing Architecture Demo

ElasticTree : Reducing Energy in Data Center Networks

- Shuts off links and switches to reduce data center power
- Choice of optimizers to balance power, fault tolerance, and BW
- OpenFlow provides network routes and port statistics



Demo:

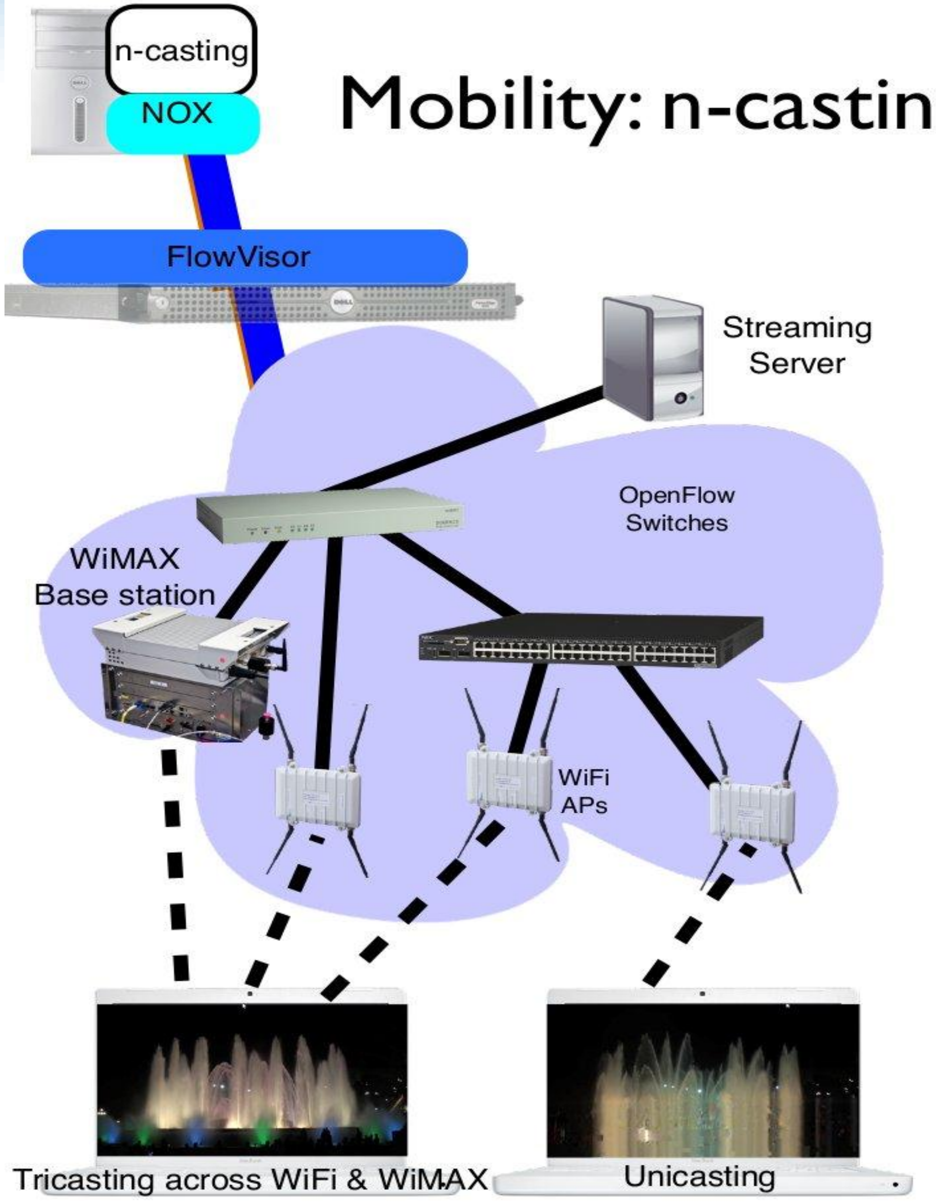
- Hardware-based 16-node Fat Tree
- Your choice of traffic pattern, bandwidth, optimization strategy
- Graph shows live power and latency variation

akoumis, David Underhill

Network Slicing Architecture Demo

MobileVMs : Dynamic Flow Handover

Mobility: n-casting with OpenFlow



- Demonstrate what flexibility of routing enables in mobile networks
- Show how technology agnostic handover can be easily achieved
- Customized network services for applications, devices and technologies
- Simplify control and services
- Unified control for wireline and wireless networking equipments

- Demonstration: n-casting
  - Reroute flows between WiFi and WiMAX without additional logic
  - n-casting provided over for video streaming where application handles duplication well
  - coded in 227 lines of C/C++

Network Slicing Architecture Demo

Scope

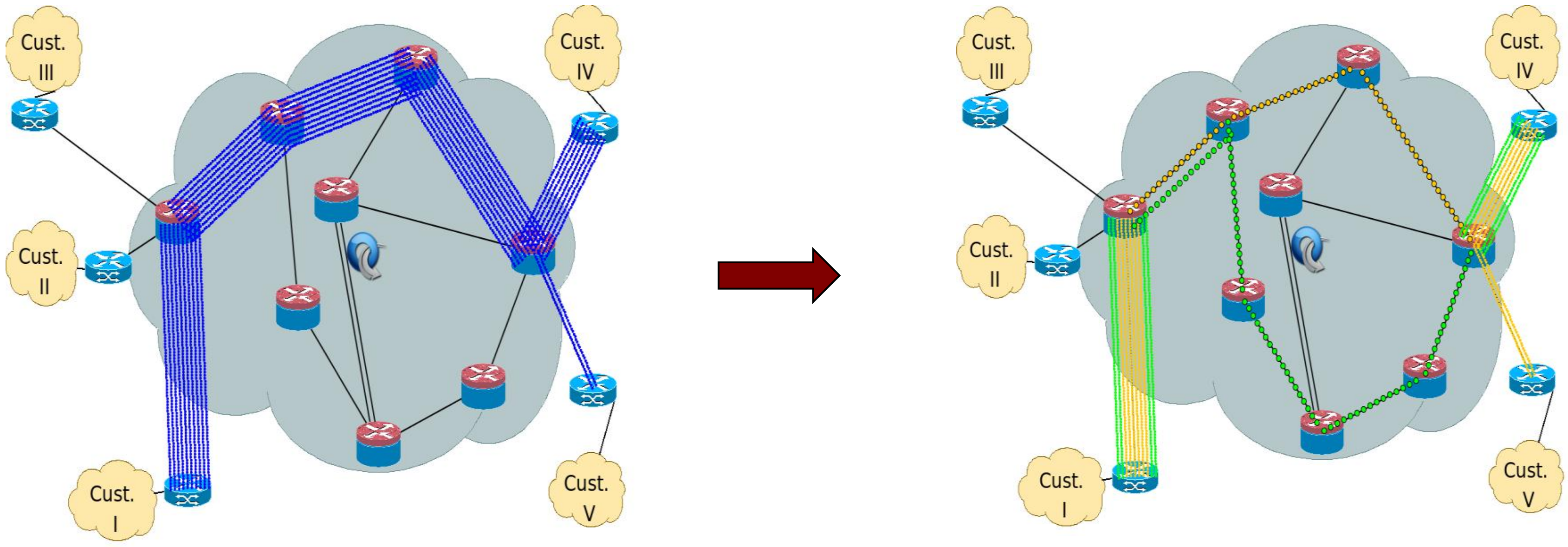
Aggregation : Dynamic Flow Aggregation

- Different Networks want different flow granularity (ISP, Backbone,…)
- Switch resources are limited (flow entries, memory)
- Network management is hard
- Current Solutions : MPLS, IP aggregation

How OpenFlow Helps?

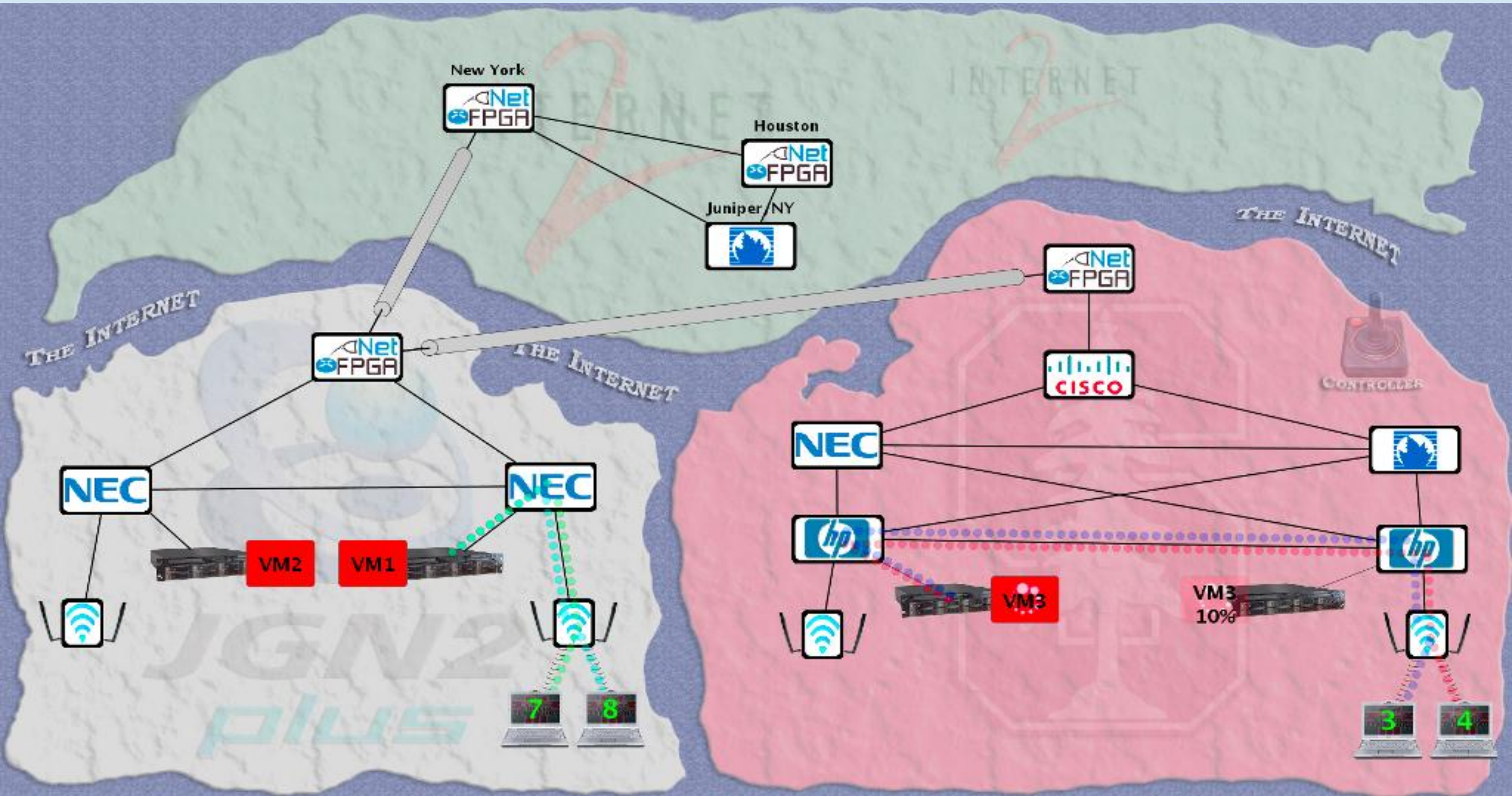
- **Dynamically define flow granularity by wildcarding arbitrary header fields**
- Granularity is on the switch flow entries, no packet rewrite or encapsulation
- Create meaningful bundles and manage them using your own software (reroute, monitor)

Higher Flexibility, Better Control, Easier Management, Experimentation



Network Slicing Architecture Demo

OpenRoad : Intercontinental VM Migration



Moved a VM from Stanford to Korea, Japan without changing its IP.

VM hosted a video game server with active network connections.

Summary:
What is OpenFlow? #1

- OpenFlow is a platform for researchers to run their experimental protocols in the networks they use every day.
- OpenFlow is based on an Ethernet switch, with an internal flow-table, and a standardized interface to add and remove flow entries.

Summary : What is OpenFlow? #2

- The goal is to encourage networking vendors to add OpenFlow platform to their switch products for deployment in college campus backbones and wiring closets.

Summary : What is OpenFlow? #3

- OpenFlow is a pragmatic compromise:
- It allows researchers to run experiments on heterogeneous switches in a uniform way at full-line-speed and with high port-density;
- Vendors do not need to expose the internal workings of their switches.

Summary :
What is OpenFlow? #4

- Allowing researchers to evaluate their ideas in real-world traffic settings,
- OpenFlow could serve as a useful campus component in proposed large-scale testbeds like GENI.
- Stanford University are running OpenFlow networks, using commercial Ethernet switches and routers.

Conclusion

- OpenFlow is a pragmatic compromise that allows researchers to run experiments on heterogeneous switches and routers in a uniform way, without the need for vendors to expose the internal workings of their products, or researchers to write vendor-specific control software.
- Successful in deploying OpenFlow networks in campuses, OpenFlow will gradually catch-on in other universities, increasing the number of networks that support experiments.
- New generation of control software emerges, allowing researchers to reuse controllers and experiments, and build on the work of others.
- OpenFlow networks at different universities interconnected by tunnels and overlay networks
- New Open-Flow networks running in the backbone networks that connect universities to each other.

Future Work

- Internet needs innovation. But we still don't know exactly what functions and features that the future Internet should include.
- We think, it may be not proper to build a concrete and fixed network for the future Internet now.
- We think, innovation ability is what the future Internet really needs now!
- OpenFlow's openness and standardization give Internet more powerful abilities to reform and innovate.
- More hardware resources in devices should be exposed and standardized

The 14th ICACT 2012(February 19 ~22, 2012 in Phoenix Park)

"Smart Society Innovation through Mobile Internet!".

Future Internet – OpenFlow
Software Defined Network

Prof. Thomas B.YOON
tomayoon@ieee.org
Kyonggi University, Korea