Flow Processing

Future Internet Communications Technologies

Prof. Dr. Panagiotis Papadimitriou

Outline

- Flow Processing
- Flow Processing on Commodity Hardware
- Programmable Switches
- Accelerated Software Routers
- Distributed Flow Processing
- In-Network Processing

Flow Processing

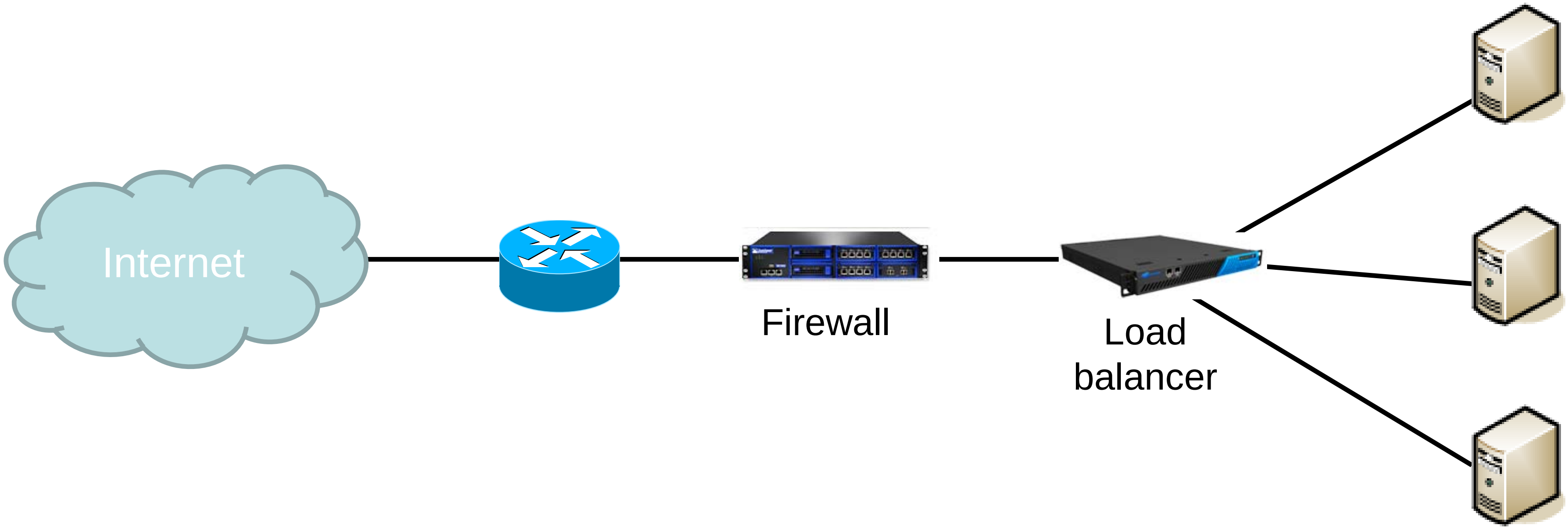
Flow Processing with Middleboxes

- The Internet infrastructure includes a large number of “appliances”, known as middleboxes, with flow processing functionalities at layers L3–L7:
  - Intrusion detection
  - Intrusion inspection
  - Encryption
  - Access control
  - Filtering
  - Measurement and logging
  - Application acceleration

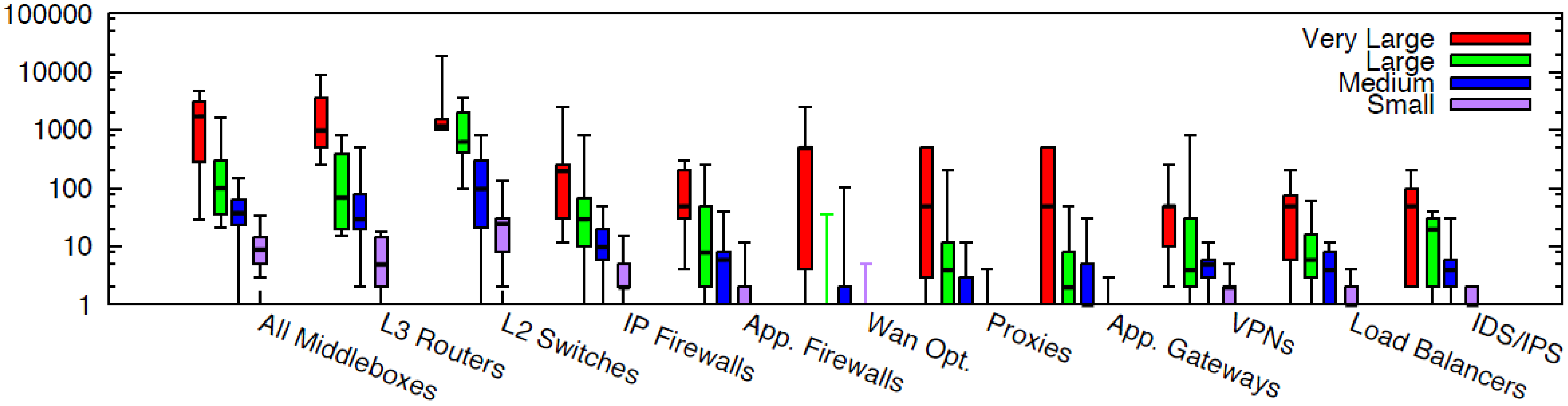
- Some routers also have packet processing capabilities (besides IPv4 forwarding) and can be used for flow processing

Middlebox Deployment Example

- Private network with 2 middleboxes:
  - Firewall: permits/filters flows according to security policy
  - Load balancer: balances traffic across servers



Statistics of Middlebox Deployments



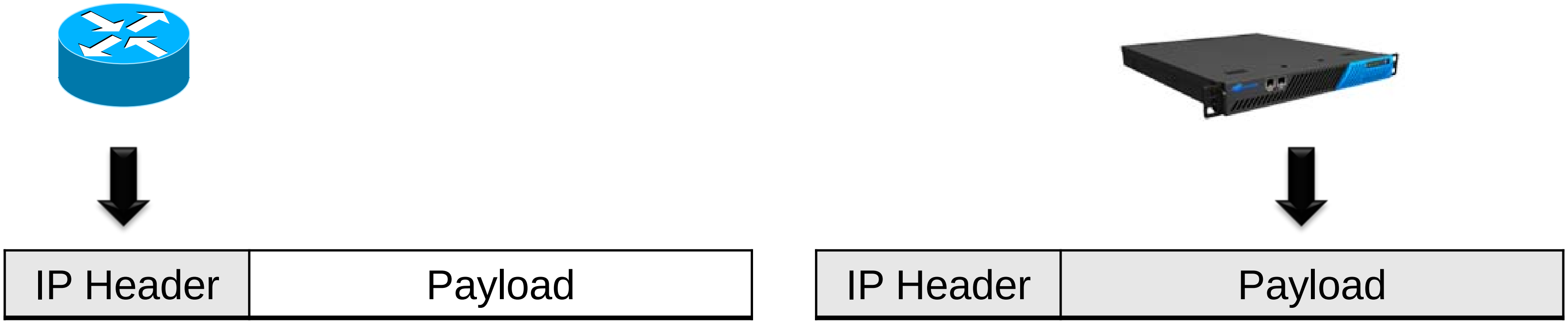
- Enterprise networks:
  - Small: < 1k hosts
  - Medium: 1k-10k hosts
  - Large: 10k-100k hosts
  - Very large: > 100k hosts

J. Sherry and S. Ratnasamy, “A Survey of Enterprise Middlebox Deployments”, 2012

Flow Processing

- Flow processing may require “deep-packet inspection” (DPI):
  - Many middleboxes examine the packet payload
- IP routers examine only the IP header of the packet

- DPI is computationally intensive:
  - encryption (AES)
  - intrusion detection

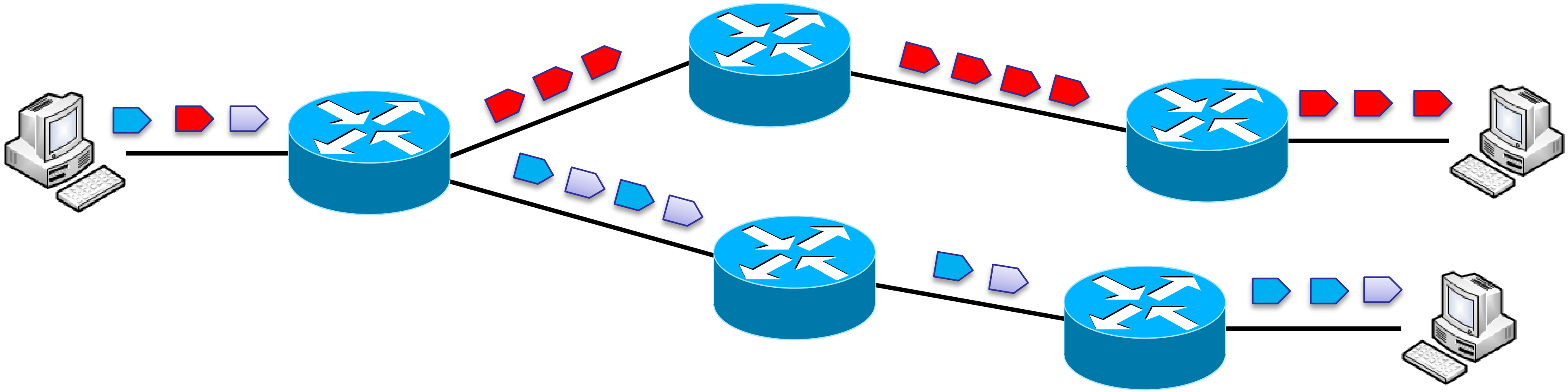


IP Header	Payload

IP Header	Payload

What is a Network Flow?

- How can a network flow be defined?
  - Naive definition: The sequence of packets from a source to a destination
  - However, multiple streams or connections can be established between a given pair of end-points
    - How can these streams/connections be distinguished as separate flows?

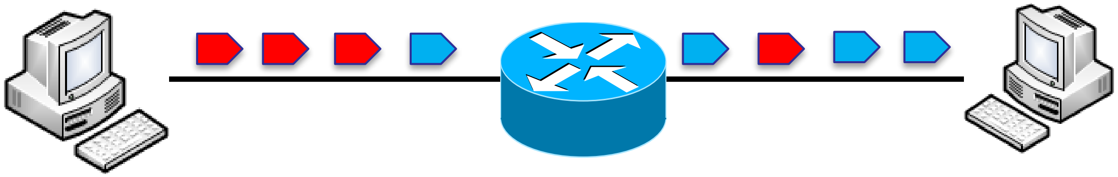


Defining Flows (1)

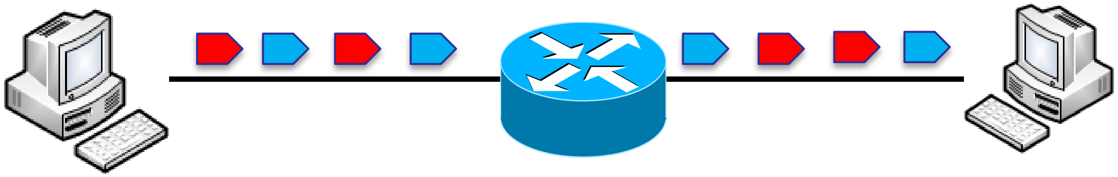
- A flow can be identified by combinations of the following:
  - Transport layer:
    - Source / destination port
    - Protocol (TCP or UDP)
  - Network layer:
    - Source / destination IP address

Defining Flows (2)

- Transport Protocol:
  - 🟥 UDP
  - 🟦 TCP

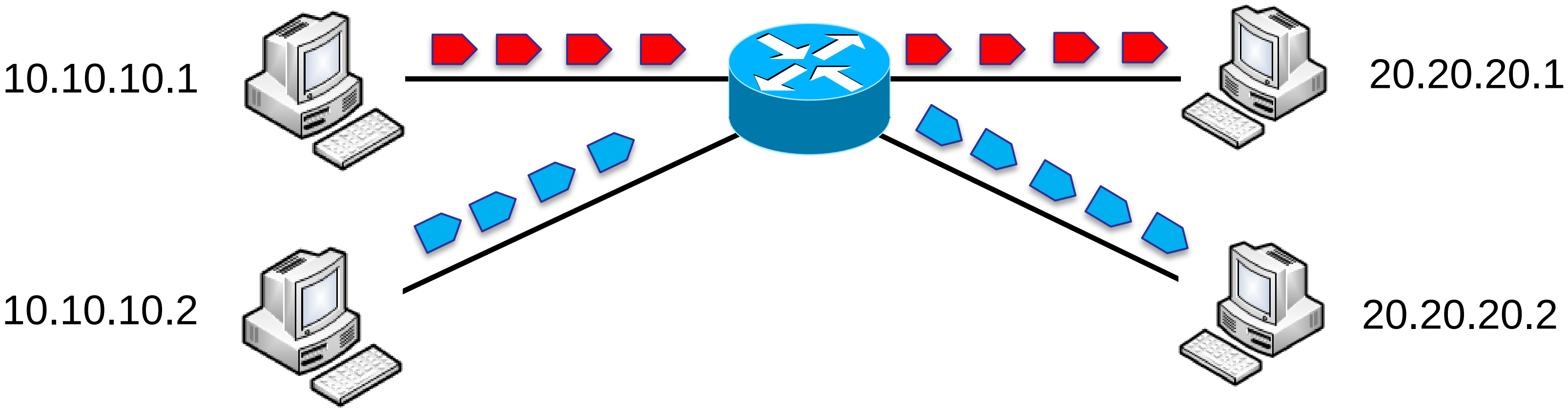


- Port Numbers:
  - 🟥 (SP: 8080, DP: 8100)
  - 🟦 (SP: 1010, DP: 7603)



Defining Flows (3)

- IP Addresses:
  - 🟥 (SA: 10.10.10.1, DA: 20.20.20.1)
  - 🟦 (SA: 10.10.10.2, DA: 20.20.20.2)



Defining Flows (4)

- In many cases, flows are identified based on the 5-tuple:
  - Source IP address
  - Destination IP address
  - Source Port
  - Destination Port
  - Protocol (TCP or UDP)

Flow Processing on Commodity Hardware

Limitations of Flow Processing with Middleboxes

- Middleboxes are built of special-purpose hardware and lack:
  - Programmability
  - Extensibility for future network services

- The deployment of a new network service or application might require some new functionality:
  - Existing middleboxes cannot be extended/upgraded to support additional processing operations
  - Additional middleboxes may have to be deployed in the network
  - The deployment cost of new middleboxes is substantial
  - This costly upgrade may discourage an ISP from offering new services, despite potential user demand

Flow Processing on Commodity Hardware (1)

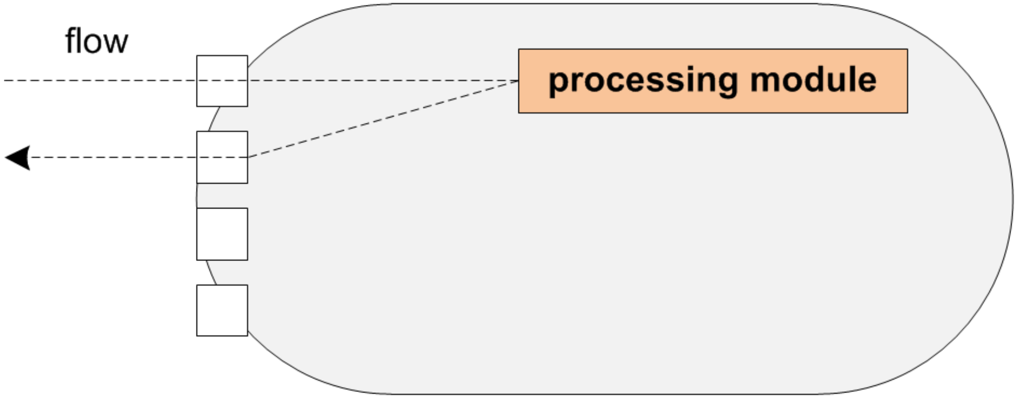
- Commodity servers can be used as a platform for flow processing

- Middleboxes built of commodity hardware are:
  - Extensible
  - Inexpensive

- Commodity servers can achieve high-performance with computational-/memory-intensive traffic workloads, exploiting:
  - Multi-core CPUs
  - Large caches
  - Faster interconnects (PCIe bus)
  - GPUs that provide a large number of (small) cores and higher memory bandwidth
    - Very efficient for parallelizable packet processing

Flow Processing on Commodity Hardware (2)

- The processing module provides a packet processing function, e.g.:
  - encryption
  - packet filtering
  - load balancing
  - intrusion detection
  - intrusion prevention



Flow Processing Software

- Click Modular Router
  - Plenty of available packet processing elements
  - Extensible
  - Multi-core

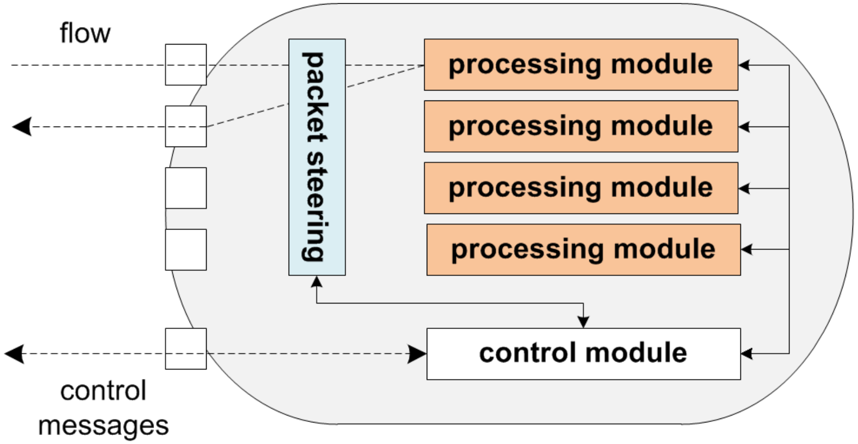
- Snort
  - Multi-mode packet analysis tool
    - 3 operational modes: Sniffer, packet logger and intrusion detection
  - Widely used for intrusion detection and prevention
  - Packet capture using the “libpcap” library
  - Logging and real-time alerting for traffic that matches given rules or patterns

Flow Processing on Commodity Hardware

- The abundance of CPU resources and server virtualization technologies can turn a commodity server into a multi-purpose flow processing platform:
  - Consolidation of multiple processing modules using virtualization
  - A control module is responsible for:
    - Managing processing modules (i.e., instantiation, configuration termination)
    - Resource monitoring (e.g. CPU load)

- Resource isolation is required for processing modules

- Admission control can be employed to reject flow processing requests when resources are no longer available



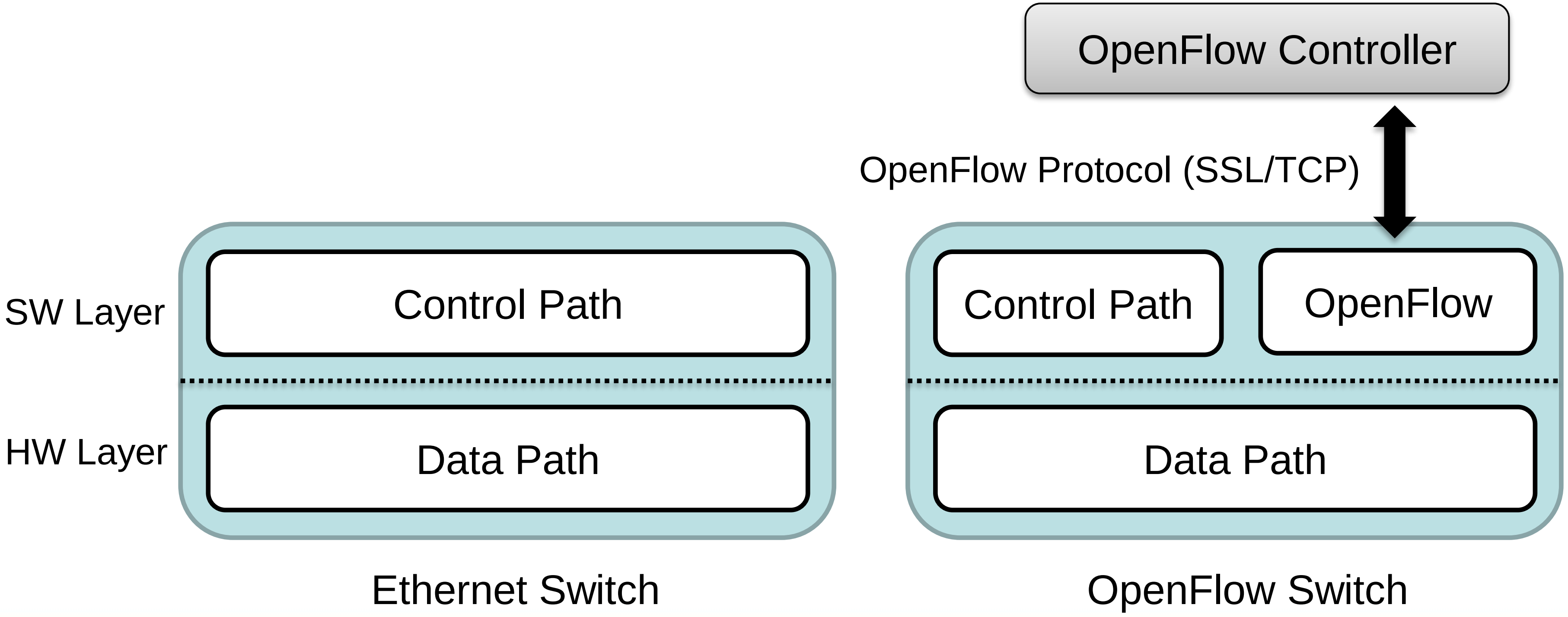
Programmable Switches

Programmable Switches

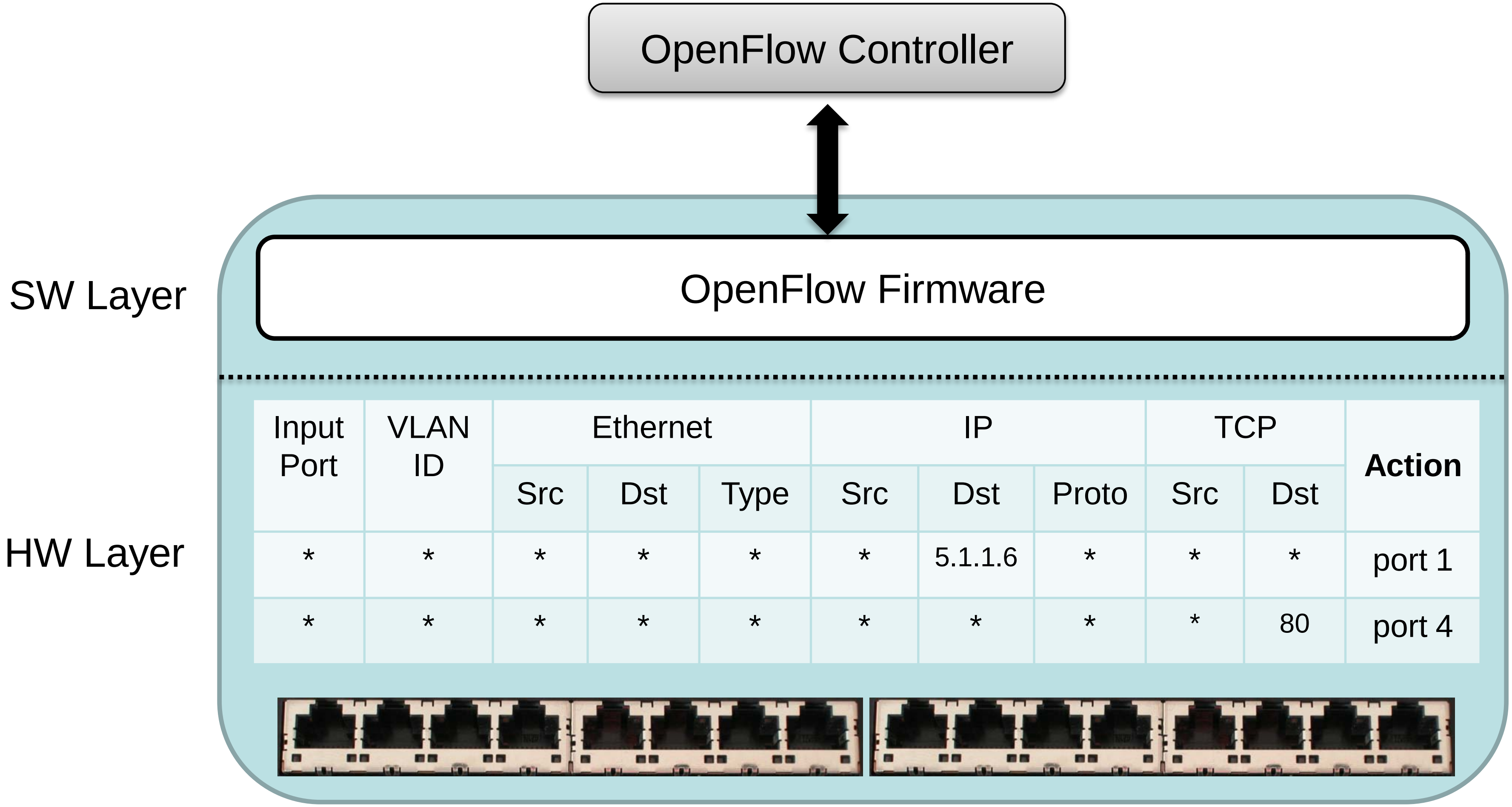
- Recent trends in switching hardware allow the modification of the switch control software:
  - Much flexibility within data-centers and enterprise networks
    - Easier to apply network configurations and policies (e.g., traffic redirection)
  - Innovation in smaller (e.g. campus) networks (e.g., OpenFlow):
    - Experimentation within the production network
    - Administrators can configure the switch to separate the production from the experimental flows
    - Users can control their own flows

OpenFlow

- Separation of control and data plane
  - OpenFlow exposes an API to control how packets are forwarded
- OpenFlow is already adopted by many vendors (e.g., HP, NEC)



Flow Table Abstraction



Input Port	VLAN ID	Ethernet			IP			TCP		**Action**
		Src	Dst	Type	Src	Dst	Proto	Src	Dst	
*	*	*	*	*	*	5.1.1.6	*	*	*	port 1
*	*	*	*	*	*	*	*	*	80	port 4

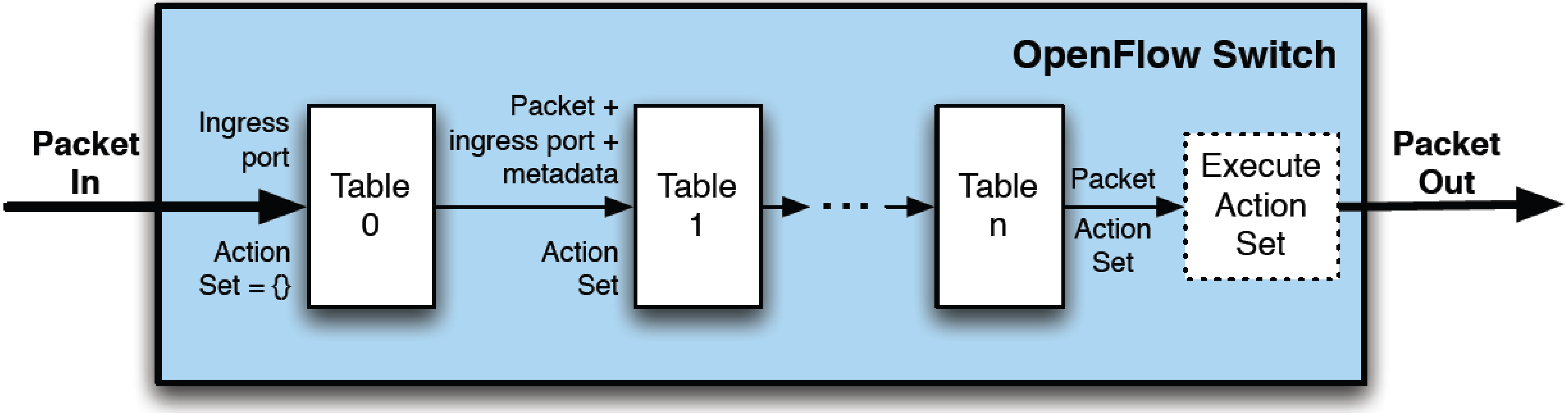
Flow Entry Operations and Supported Actions

- Flows entry operations:
  - Add
  - Modify
  - Remove

- Supported actions:
  - Forward to one or more output ports
  - Encapsulate and send to the controller (typical action for the 1st packet of a new flow)
  - Drop

Latest OpenFlow Specification (v1.2)

- New features:
  - Multiple flow tables
  - User-defined matching (masking)
  - MPLS
  - Time-to-Live (TTL)



Flow Control Examples

- Switching:

Input Port	VLAN ID	Ethernet			IP			TCP		**Action**
		Src	Dst	Type	Src	Dst	Proto	Src	Dst	
*	*	*	1D-6..	*	*	*	*	*	*	port 3

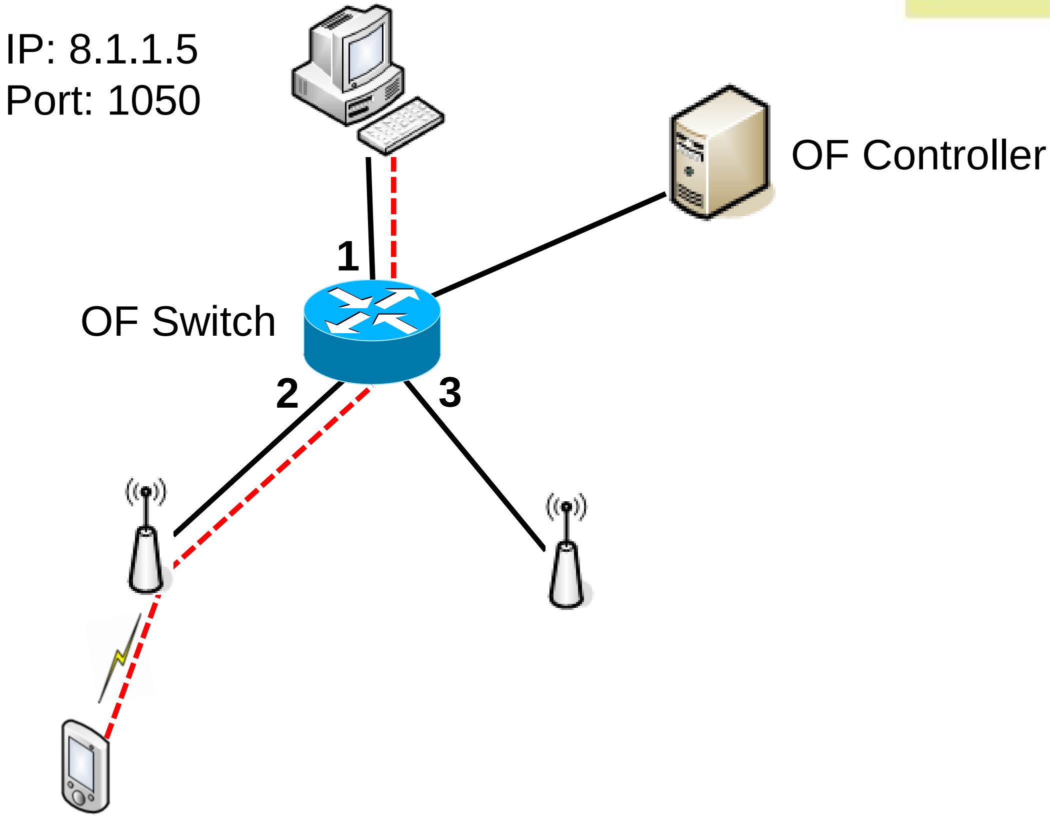
- IP Routing:

Input Port	VLAN ID	Ethernet			IP			TCP		**Action**
		Src	Dst	Type	Src	Dst	Proto	Src	Dst	
*	*	*	*	*	*	5.1.1.6	*	*	*	port 1

- Firewall:

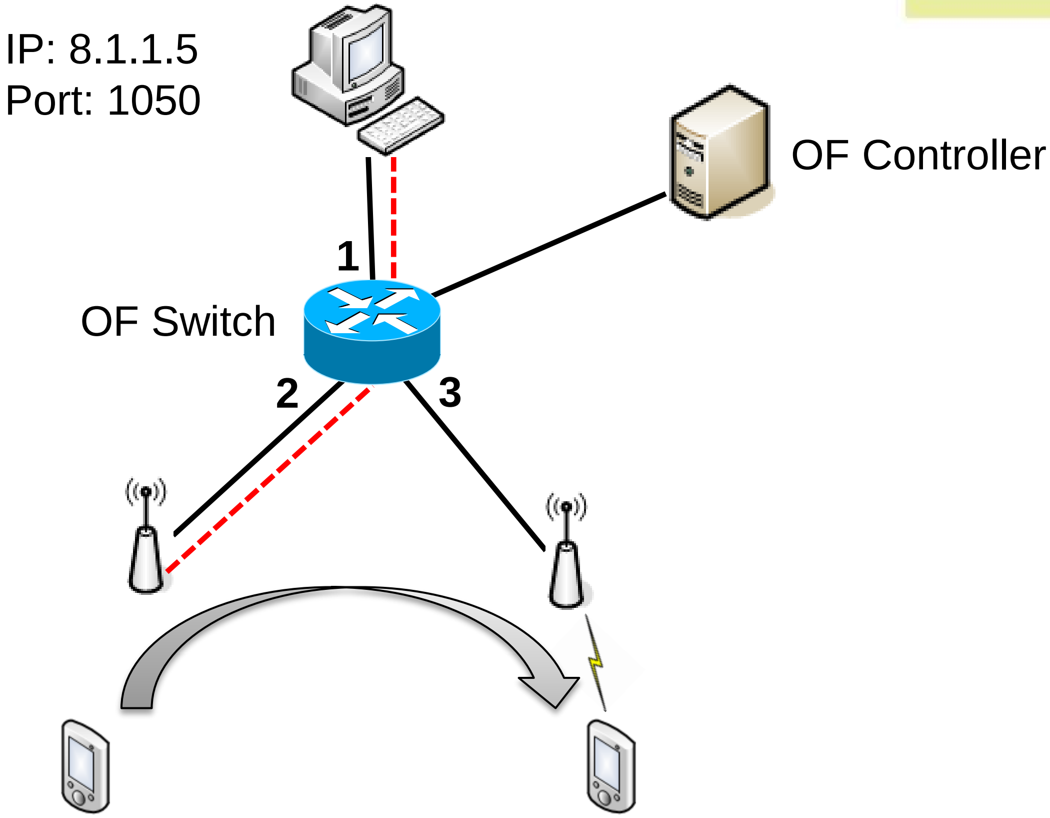
Input Port	VLAN ID	Ethernet			IP			TCP		**Action**
		Src	Dst	Type	Src	Dst	Proto	Src	Dst	
*	*	*	*	*	*	*	*	*	22	drop

Flow Rerouting with OpenFlow



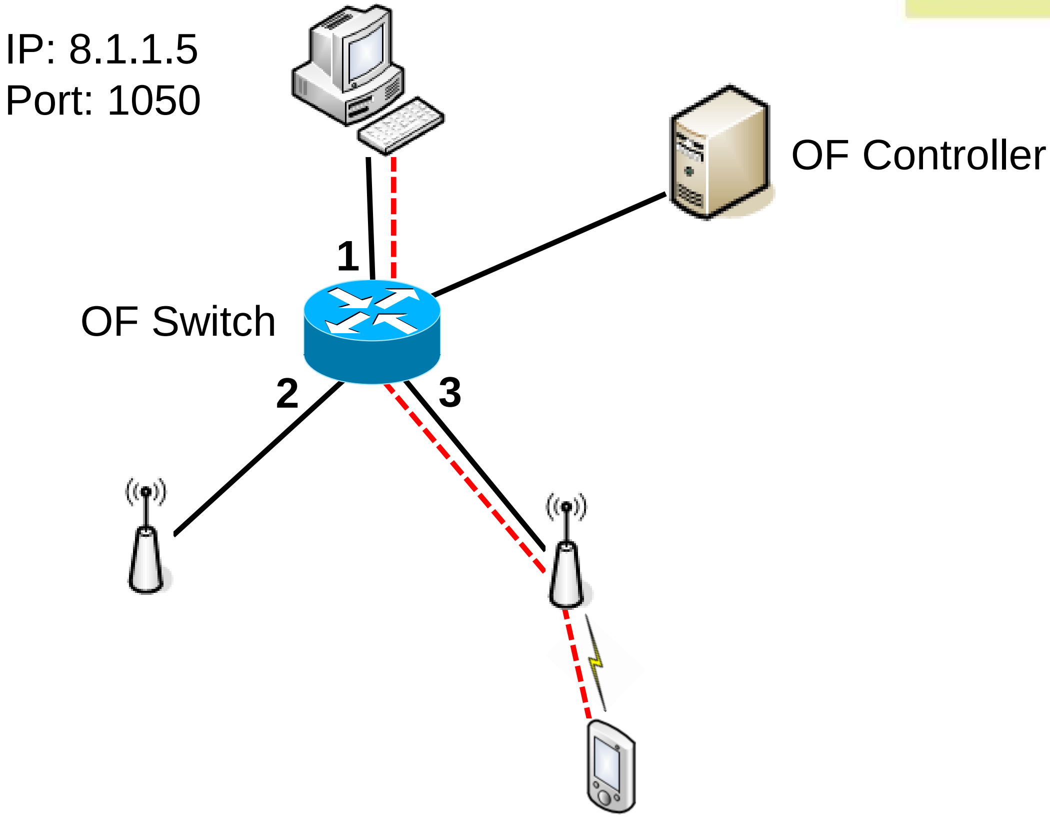
Input Port	VLAN ID	Ethernet			IP			TCP		**Action**
		Src	Dst	Type	Src	Dst	Proto	Src	Dst	
*	*	*	*	*	8.1.1.5	*	*	1050	*	port 2

Flow Rerouting with OpenFlow



Input Port	VLAN ID	Ethernet			IP			TCP		**Action**
		Src	Dst	Type	Src	Dst	Proto	Src	Dst	

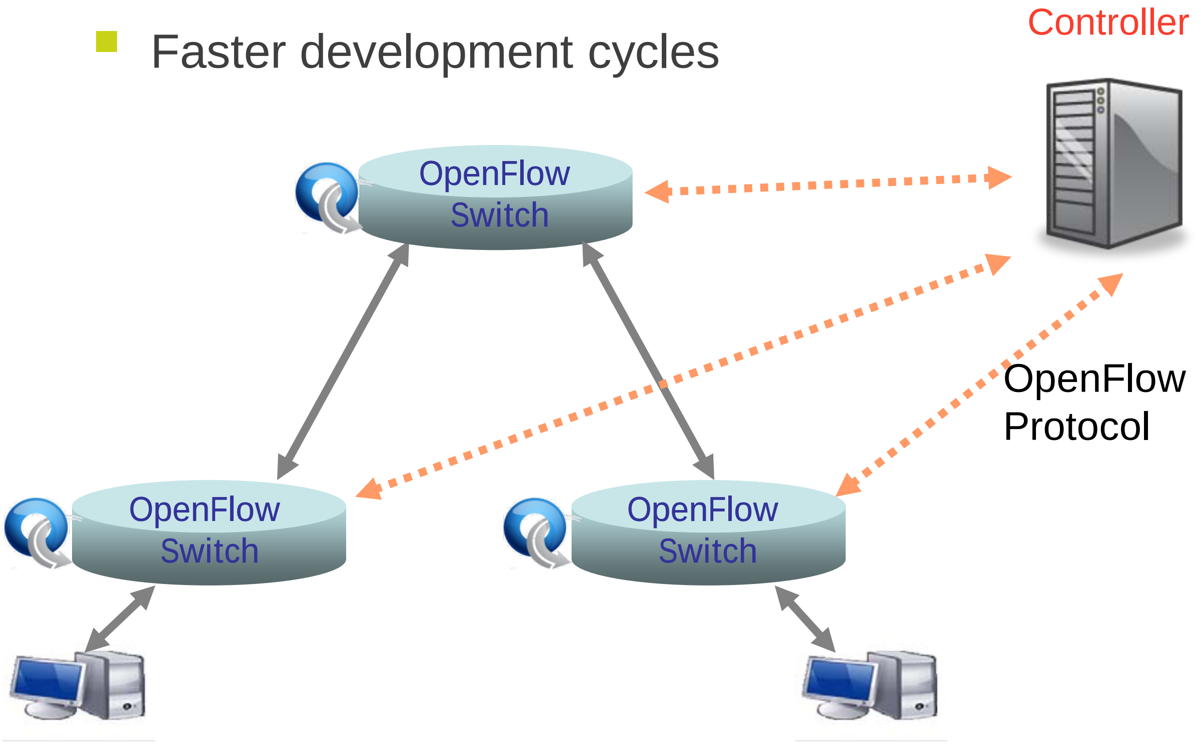
Flow Rerouting with OpenFlow



Input Port	VLAN ID	Ethernet			IP			TCP		**Action**
		Src	Dst	Type	Src	Dst	Proto	Src	Dst	
*	*	*	*	*	8.1.1.5	*	*	1050	*	port 3

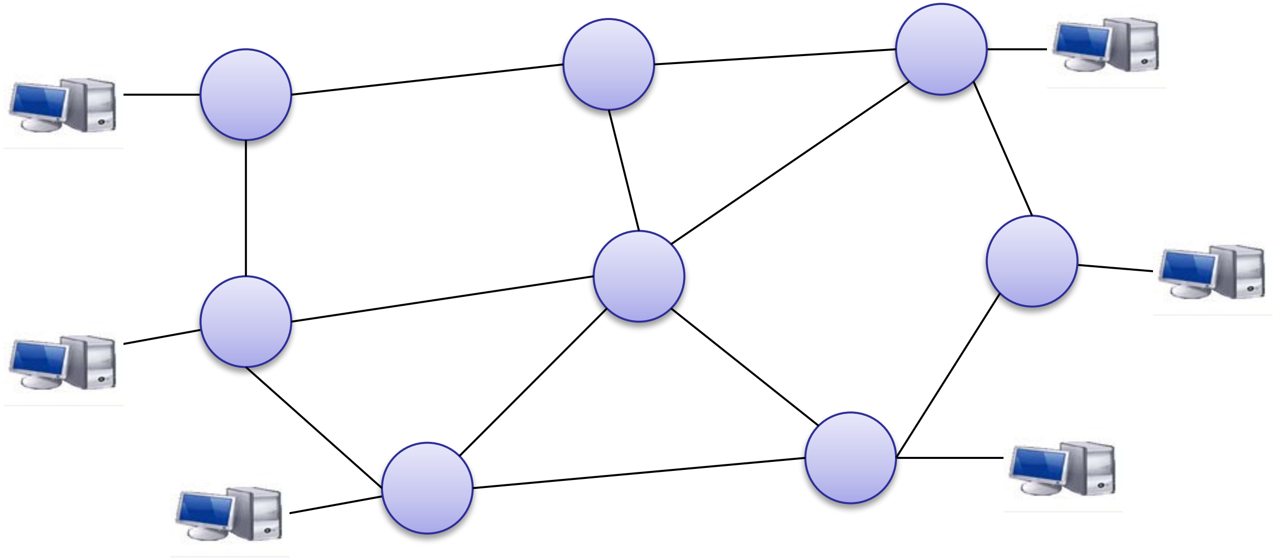
Software-Defined Network (SDN)

- Centralized control
- Network-wide visibility
- Control functions (e.g., routing, access control) in software
  - Easy deployment of updates
- Network abstraction
  - Faster development cycles



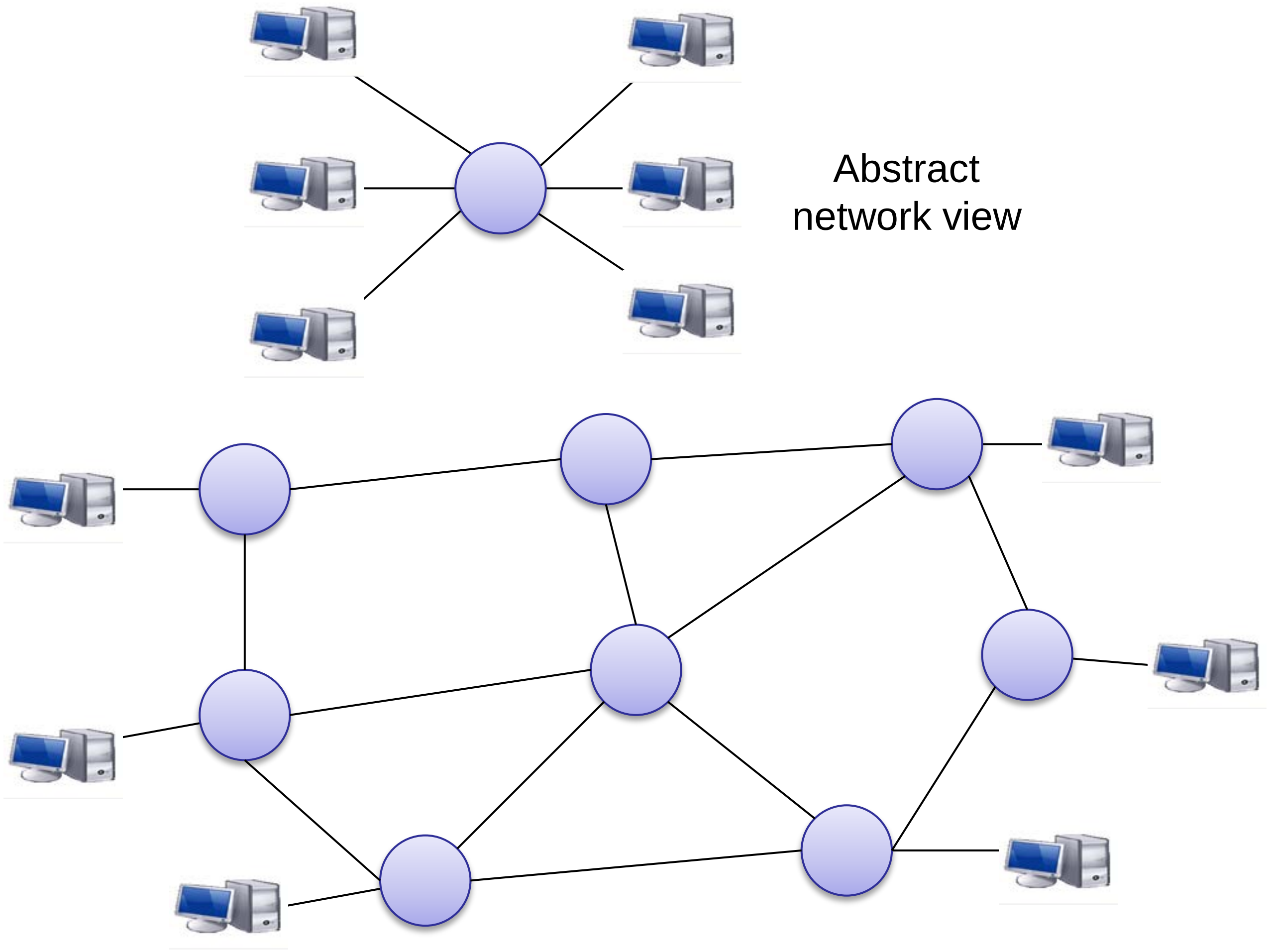
SDN Abstraction

- How can access control be easily configured?

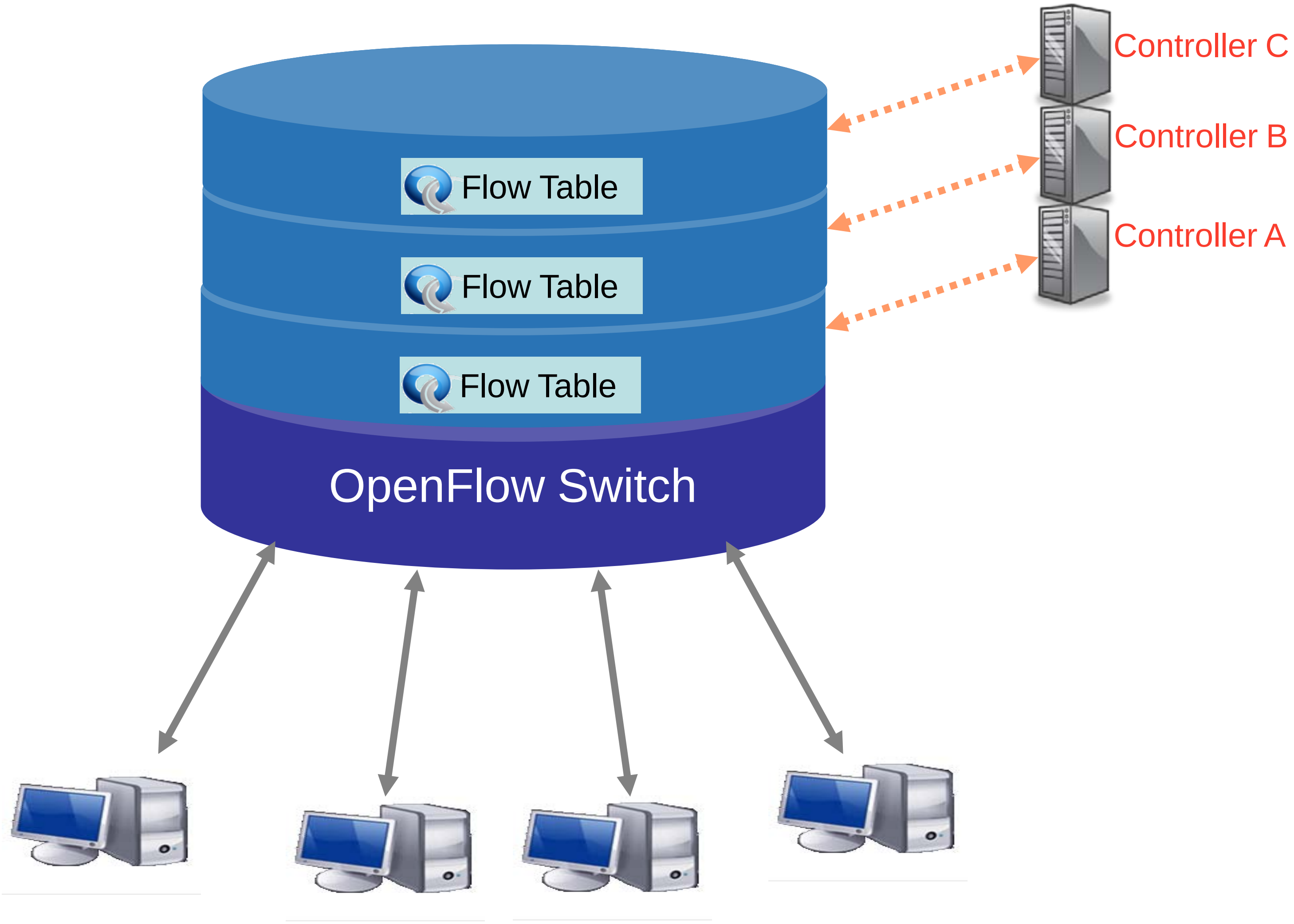


SDN Abstraction

- How can access control be easily configured?



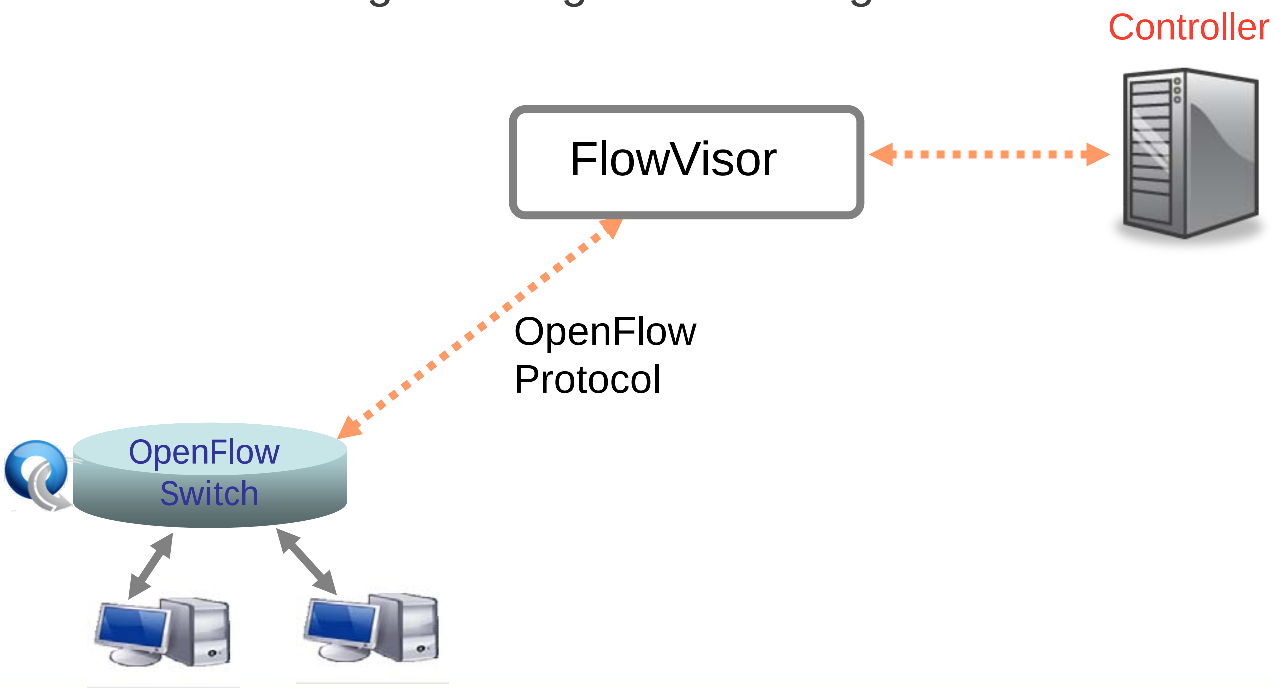
Slicing OpenFlow



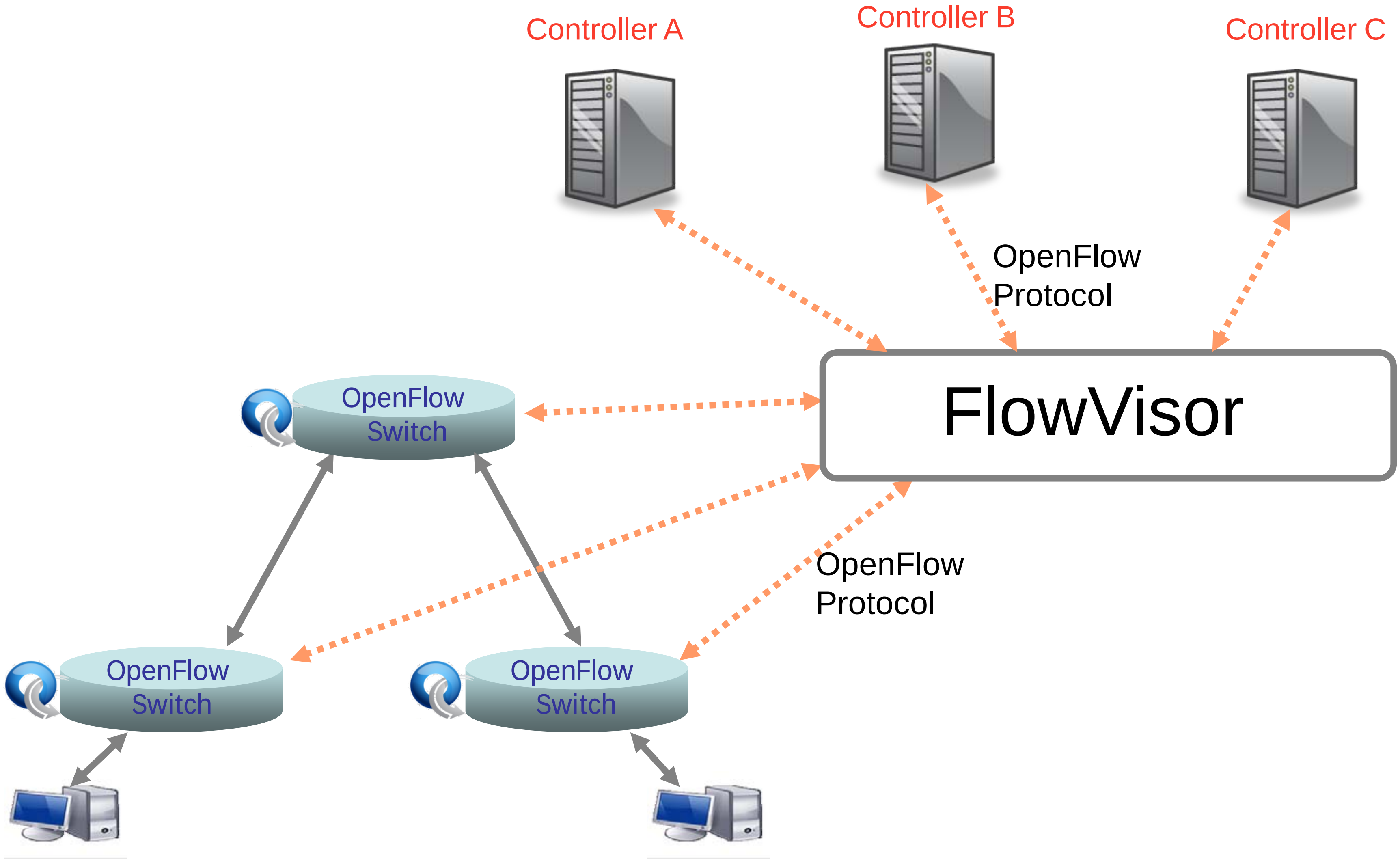
Slicing OpenFlow Networks with FlowVisor (1)

- FlowVisor is a versatile solution for slicing OpenFlow

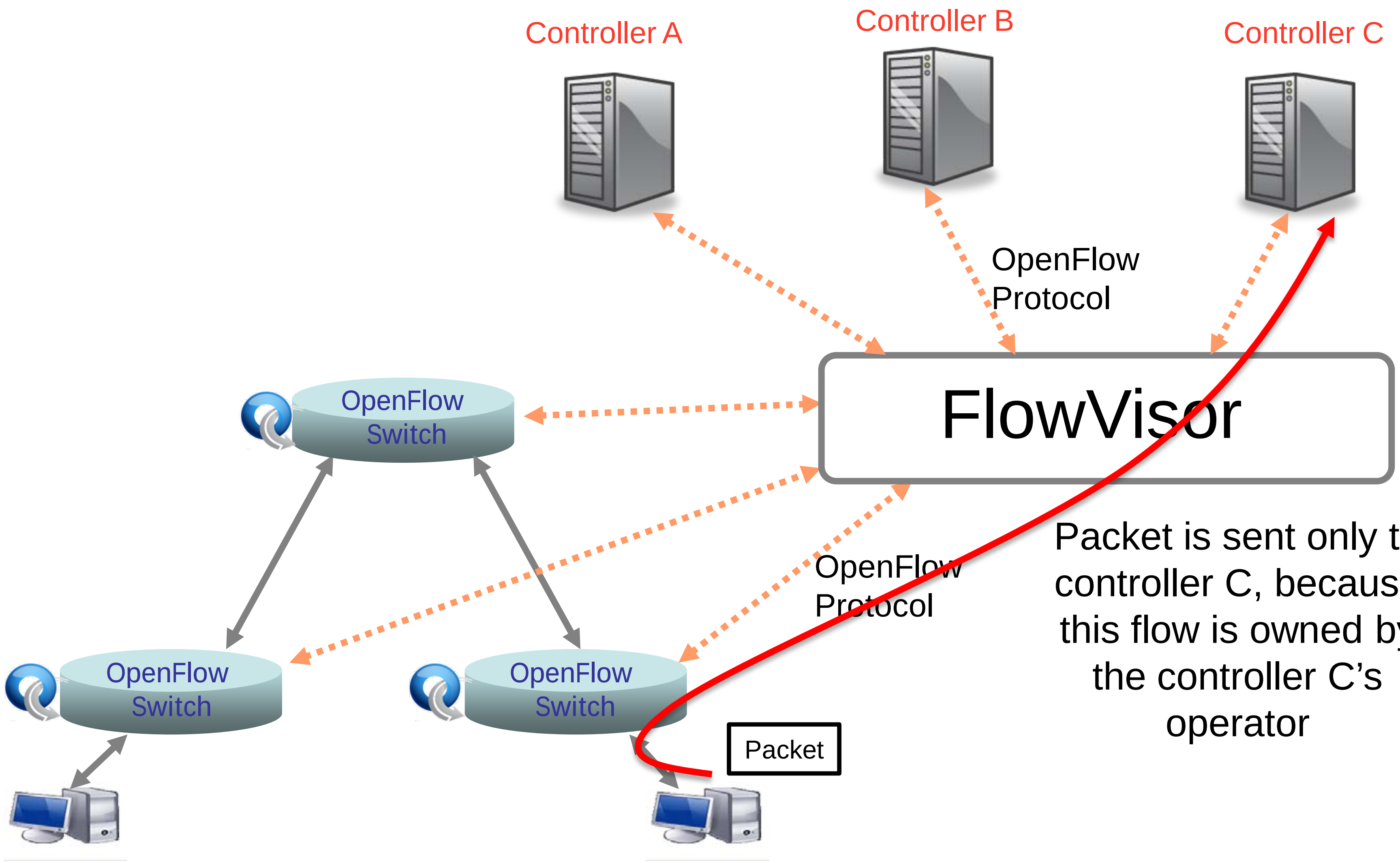
- FlowVisor acts as a proxy between the switch and the controller, offering:
  - Flow table isolation
  - Switch CPU isolation
  - Control message filtering and rewriting



Slicing OpenFlow Networks with FlowVisor (2)

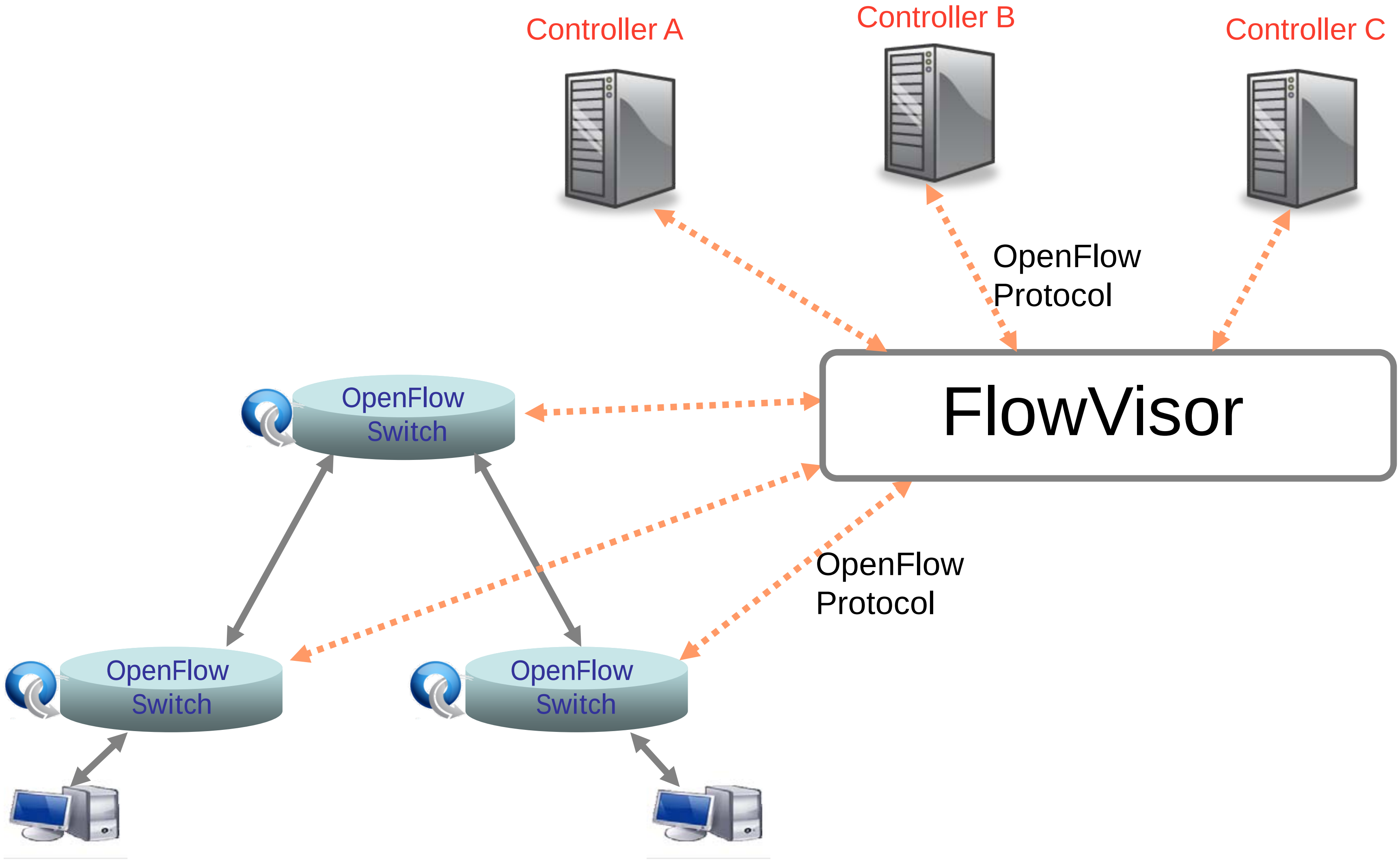


Slicing OpenFlow Networks with FlowVisor (2)

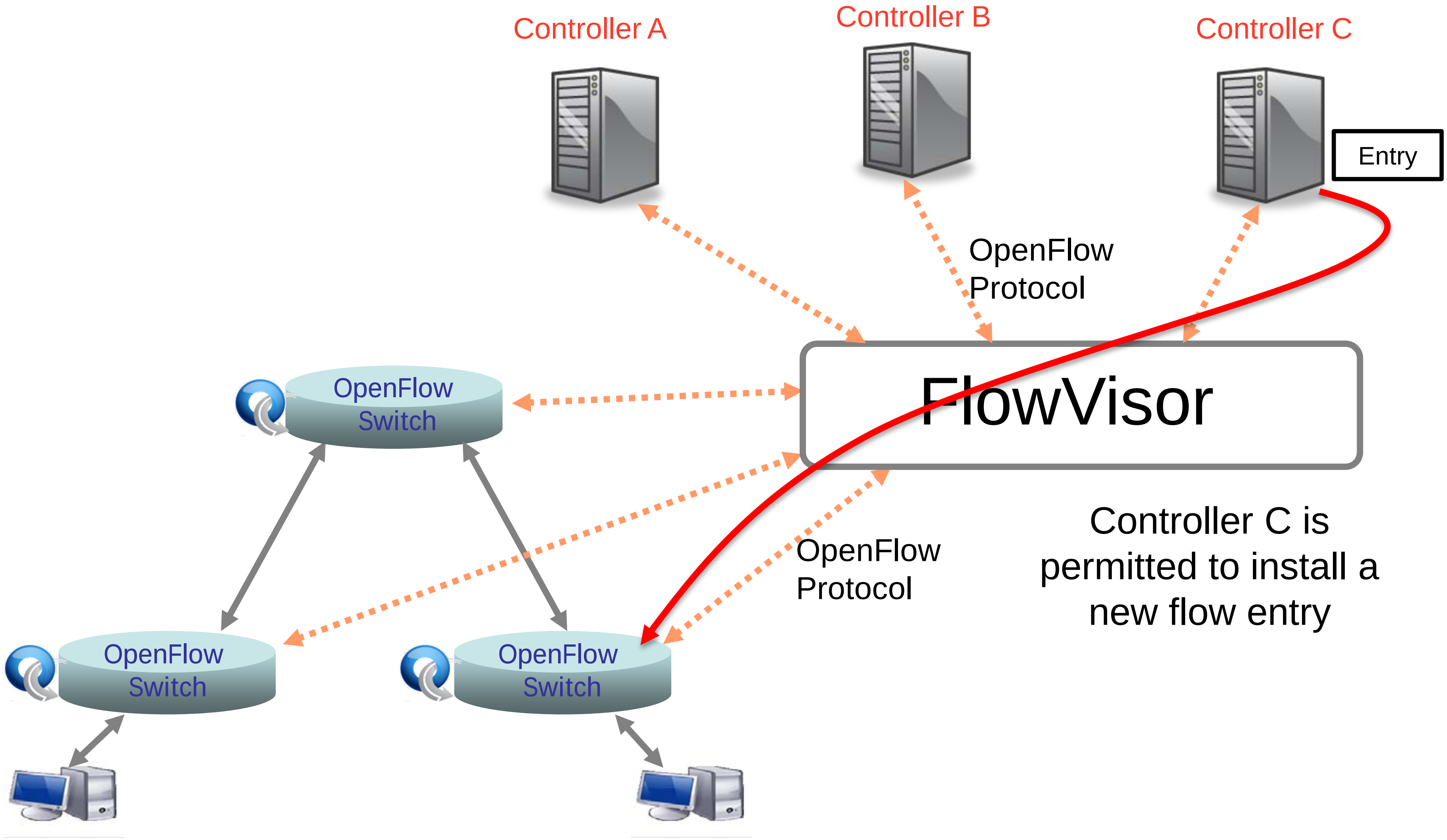


Packet is sent only to controller C, because this flow is owned by the controller C's operator

Slicing OpenFlow Networks with FlowVisor (2)



Slicing OpenFlow Networks with FlowVisor (2)



Controller C is permitted to install a new flow entry

Accelerated Software Routers

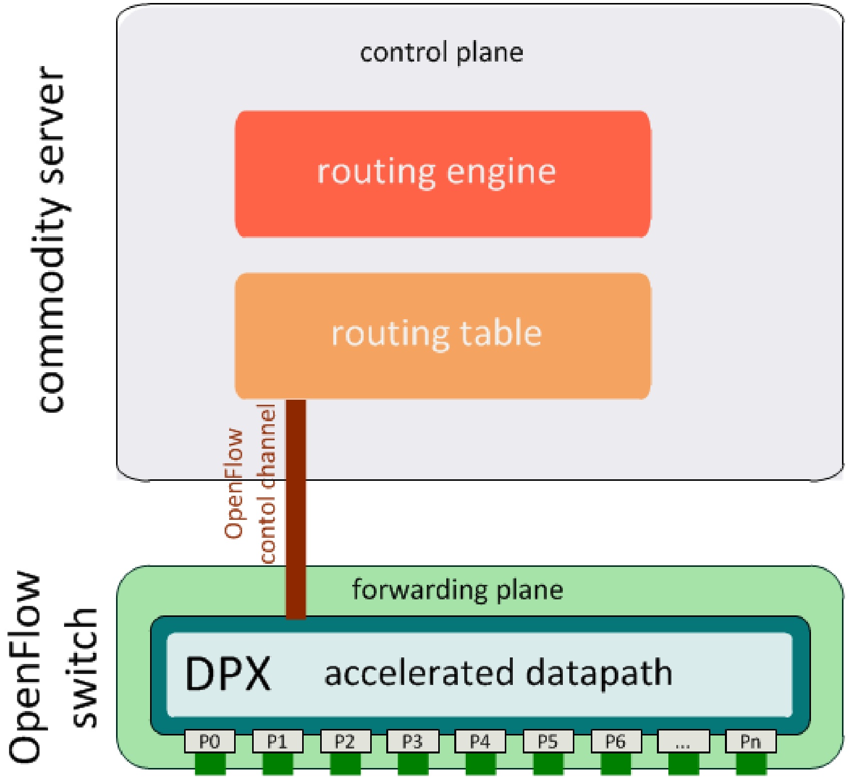
Towards High-Performance Programmable Routers

- Requirements for high-performance programmable software routers:
  - Control plane:
    - Extensibility
  - Forwarding plane:
    - Performance
    - Programmability
    - High port density
- Software routers on commodity servers:
  - ✓ Programmability
  - ✓ Respectable packet forwarding performance
  - ✗ Limited port density
  - ✗ Insufficient packet forwarding rates for the Internet core (≥ 40 Gbps)

Accelerating Software Routers (1)

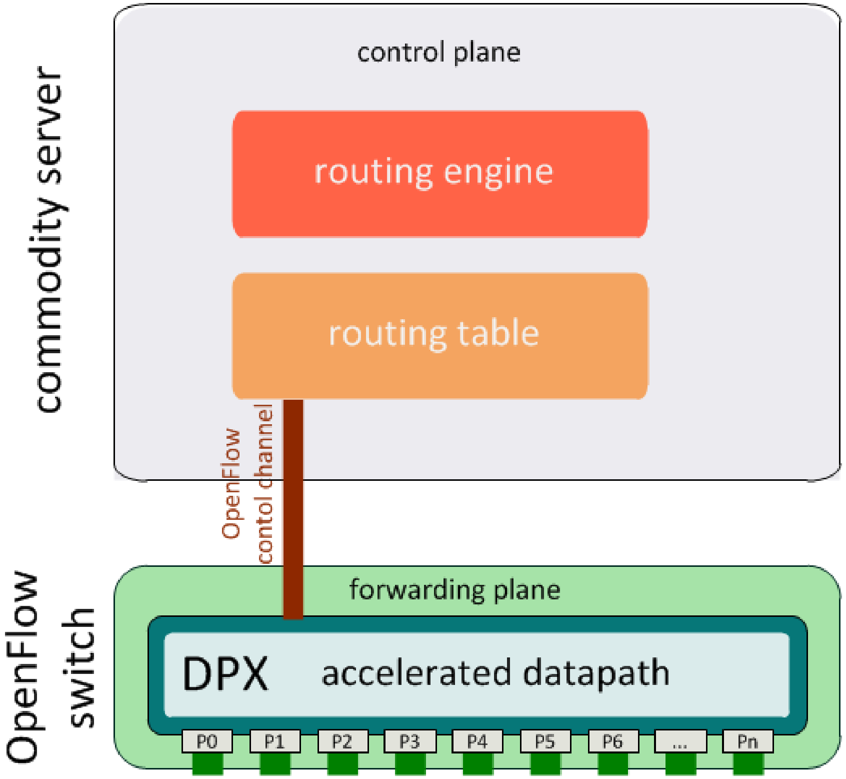
- A commodity server does not satisfy the requirements for the forwarding plane

- How about using an OpenFlow switch for packet forwarding:
  - OpenFlow offers:
    - Programmability
    - High port density
    - Packet forwarding at line rates



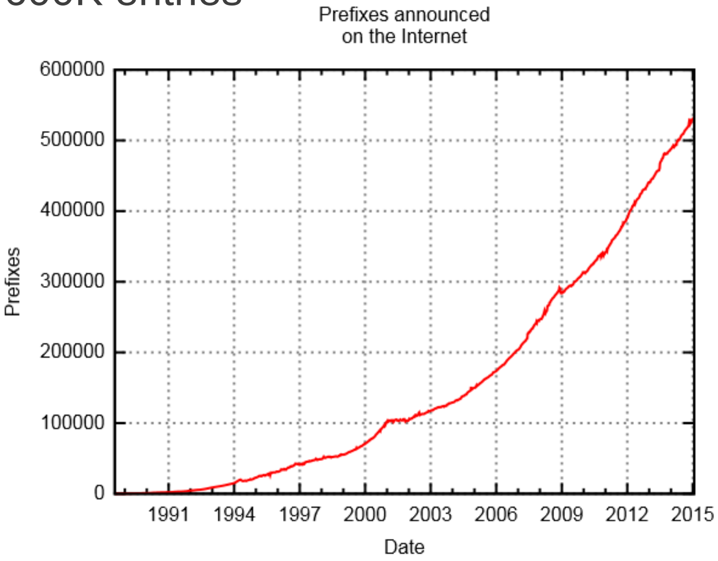
Accelerating Software Routers (2)

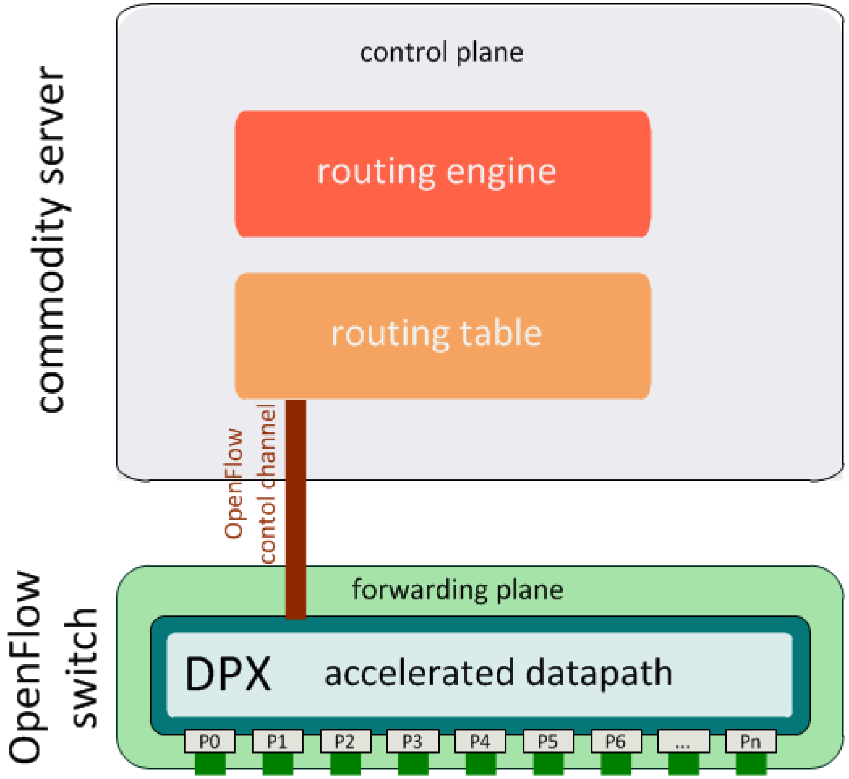
- Main idea:
  - Packet forwarding in the OpenFlow switch
    - Forwarding table stored in the switch flow table (i.e., as flow entries)
  - Control plane (routing protocols and routing table) hosted on a commodity server
    - Routing table is copied to the switch flow table
    - Routing updates received and processed by the control plane trigger the corresponding switch flow table updates



Limitations of Accelerated Software Routers

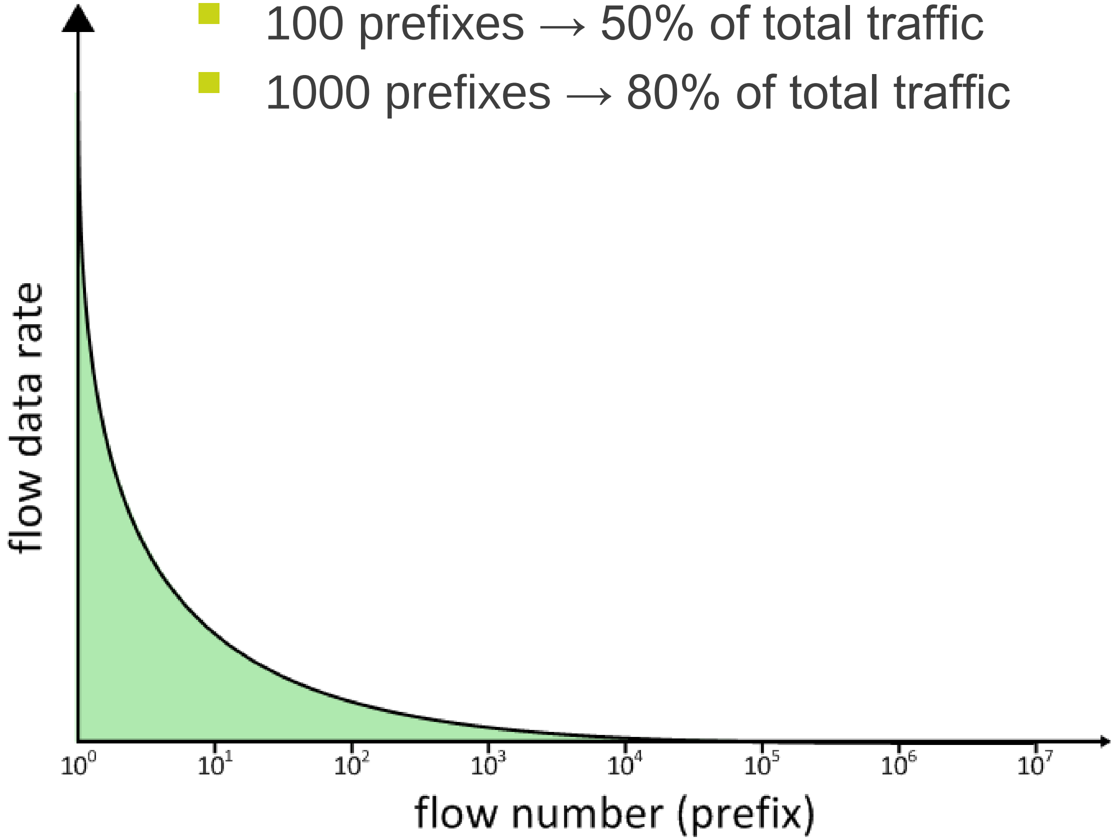
- OpenFlow switch flow table has small size (a few Mbytes)
  - can store only a few thousands of flow entries
- This limitation seems to make such a platform infeasible:
  - A full BGP routing table includes nearly 600K entries





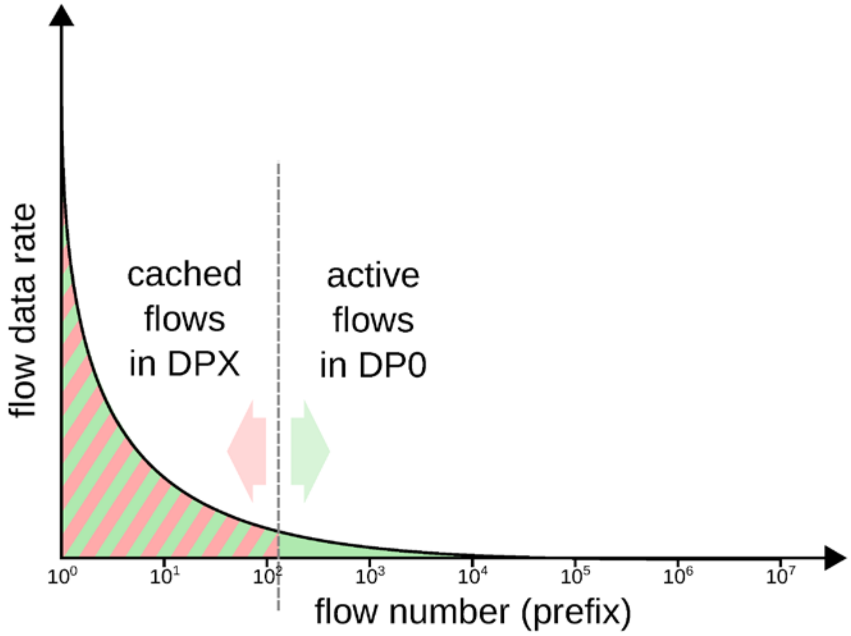
Flow Distribution in the Internet

- Flow distribution in the Internet:
  - A small subset of flows carries most of Internet traffic
    - Statistics from a residential ISP:
      - 100 prefixes → 50% of total traffic
      - 1000 prefixes → 80% of total traffic



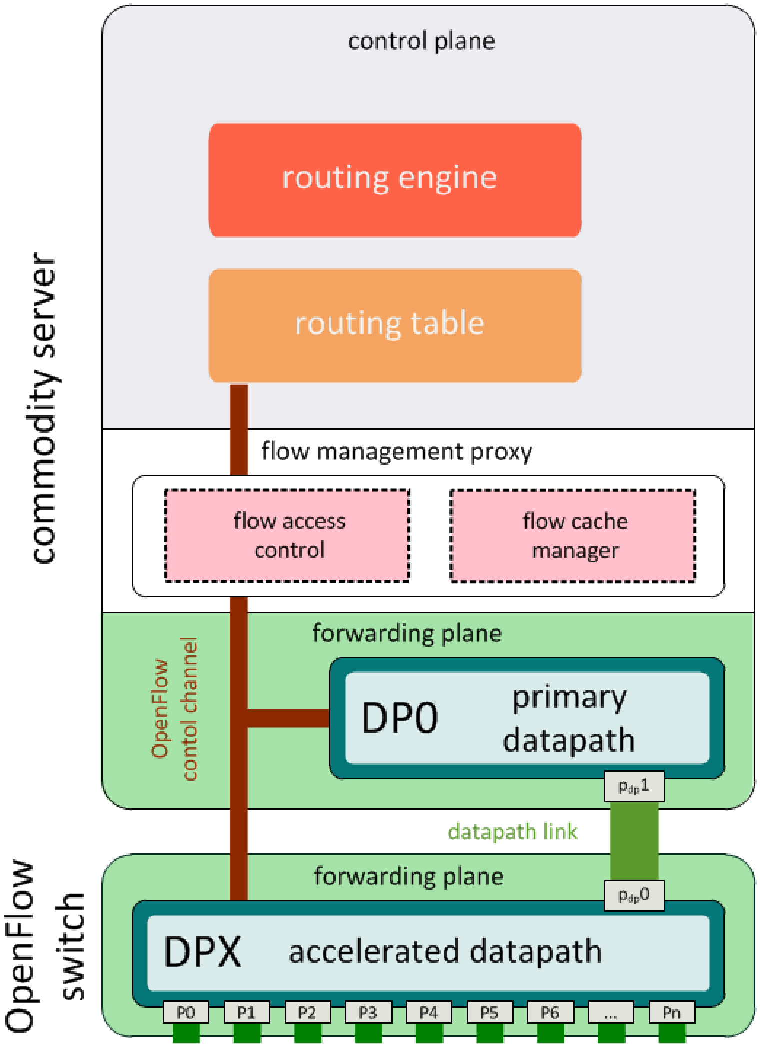
Towards an Accelerated Software Router

- Leverage on Internet flow distribution:
  - Dual-datapath approach:
    - Primary datapath (DP0) on a commodity server with forwarding entries for all flows
    - Accelerated datapath (DPX) on OpenFlow switch with forwarding entries for the subset of high-volume flows

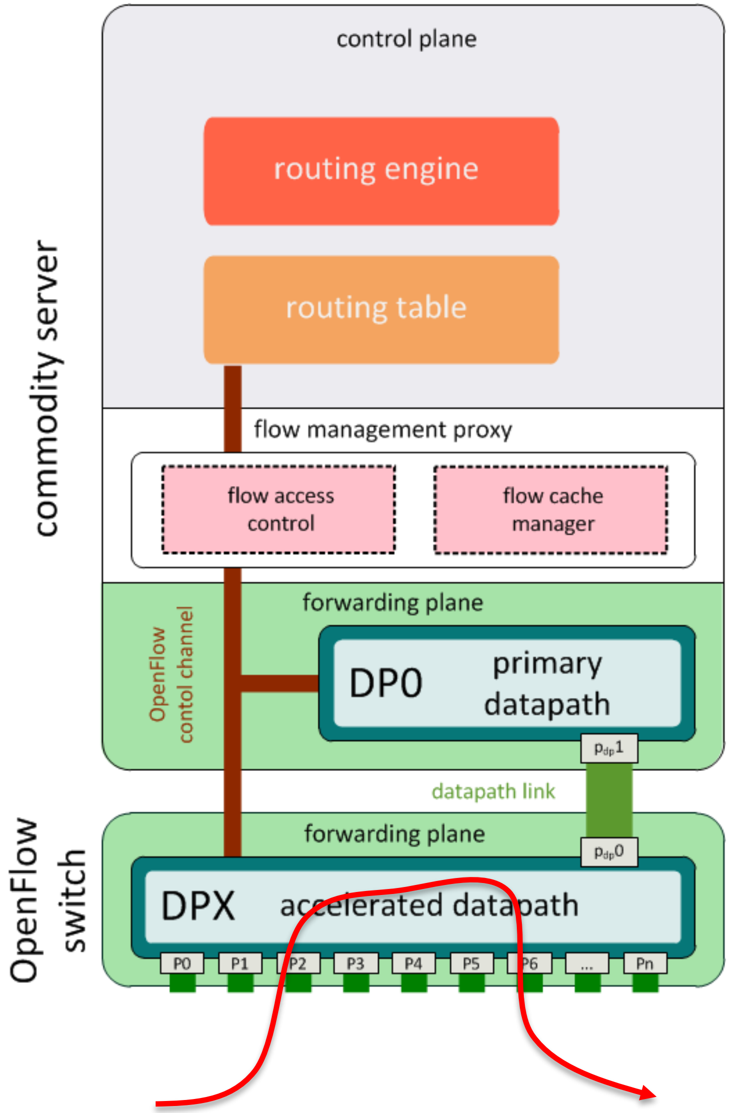


Accelerated Software Router

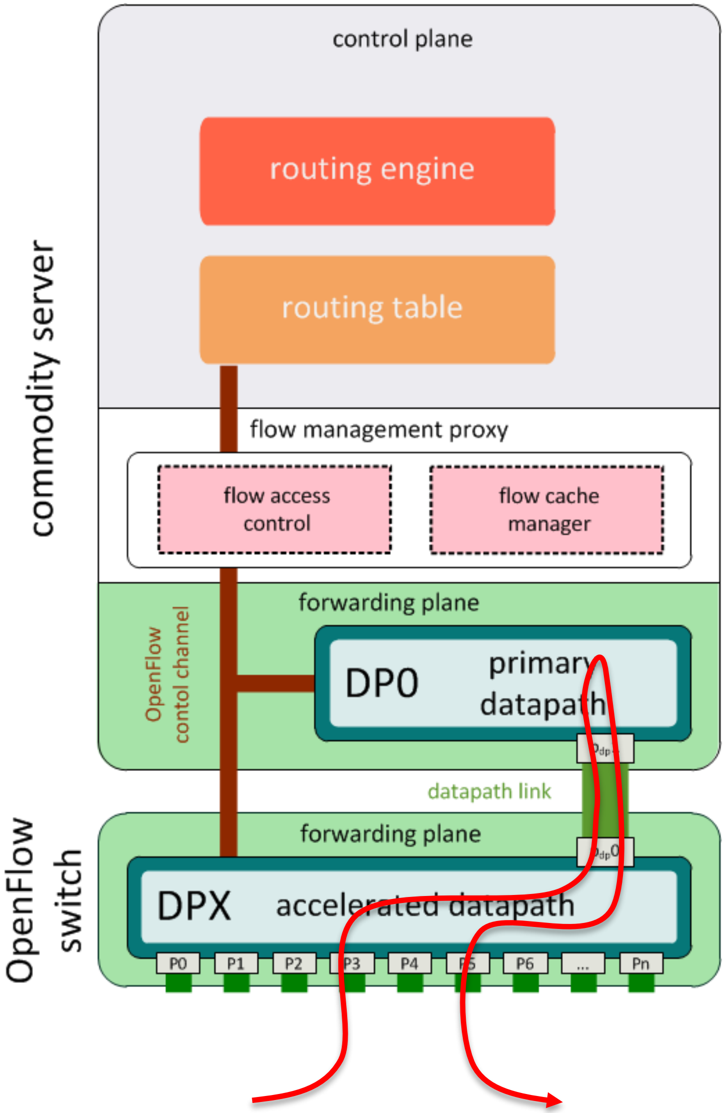
- Components:
  - Forwarding plane composed of primary and accelerated datapath
  - Control plane
  - Flow management proxy:
    - Transparent layer between the forwarding and control plane
    - Selection of flows that will be cached in the DPX
      - Caching mechanisms, e.g., LRU, LFU



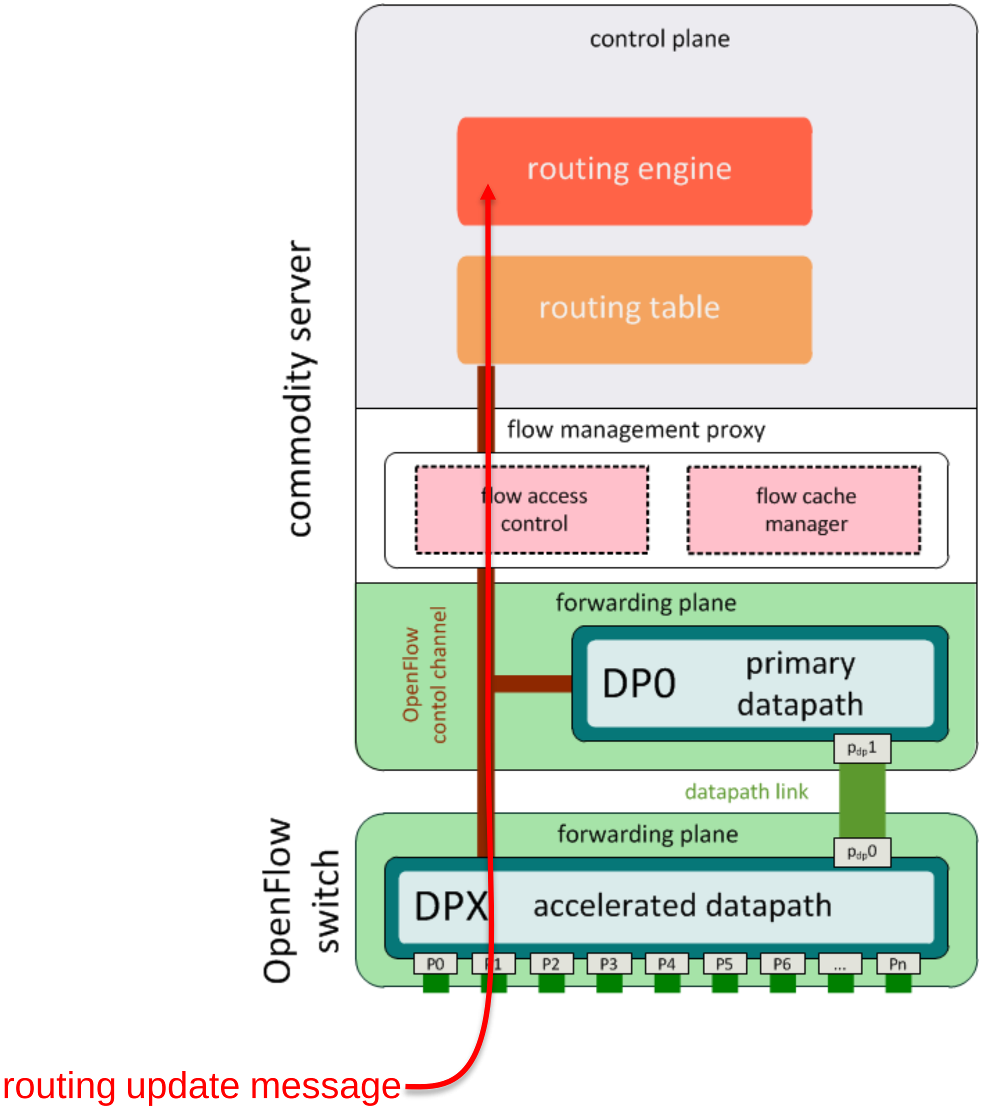
Forwarding at DPX



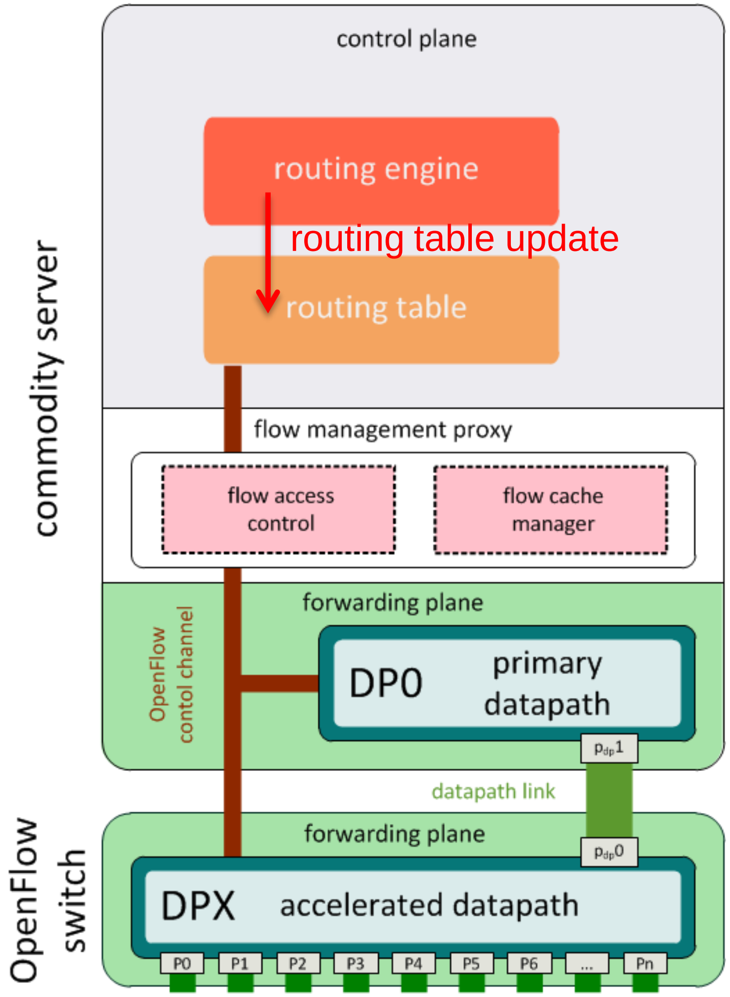
Forwarding at DP0



Handling Routing Updates

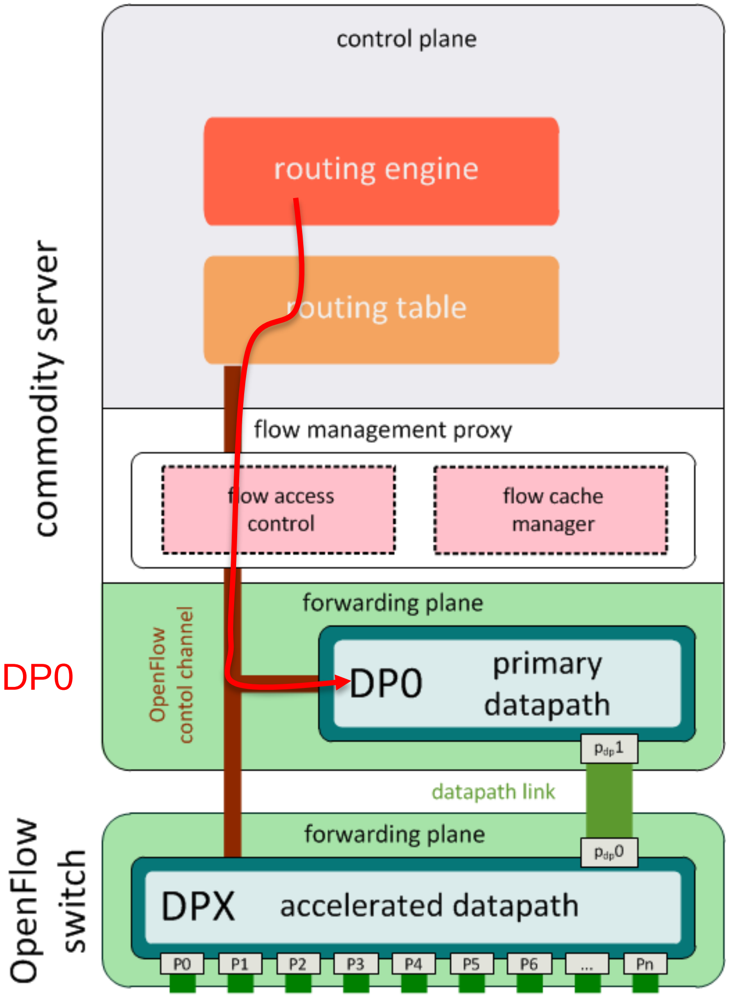


Handling Routing Updates

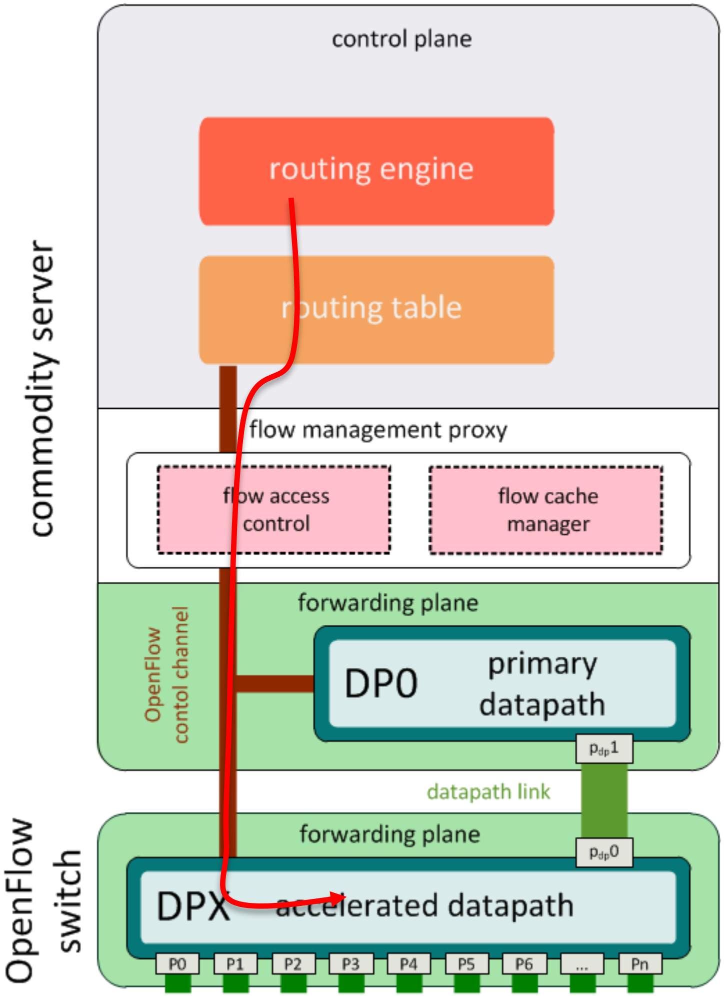


Handling Routing Updates

forwarding table update in DP0



Handling Routing Updates



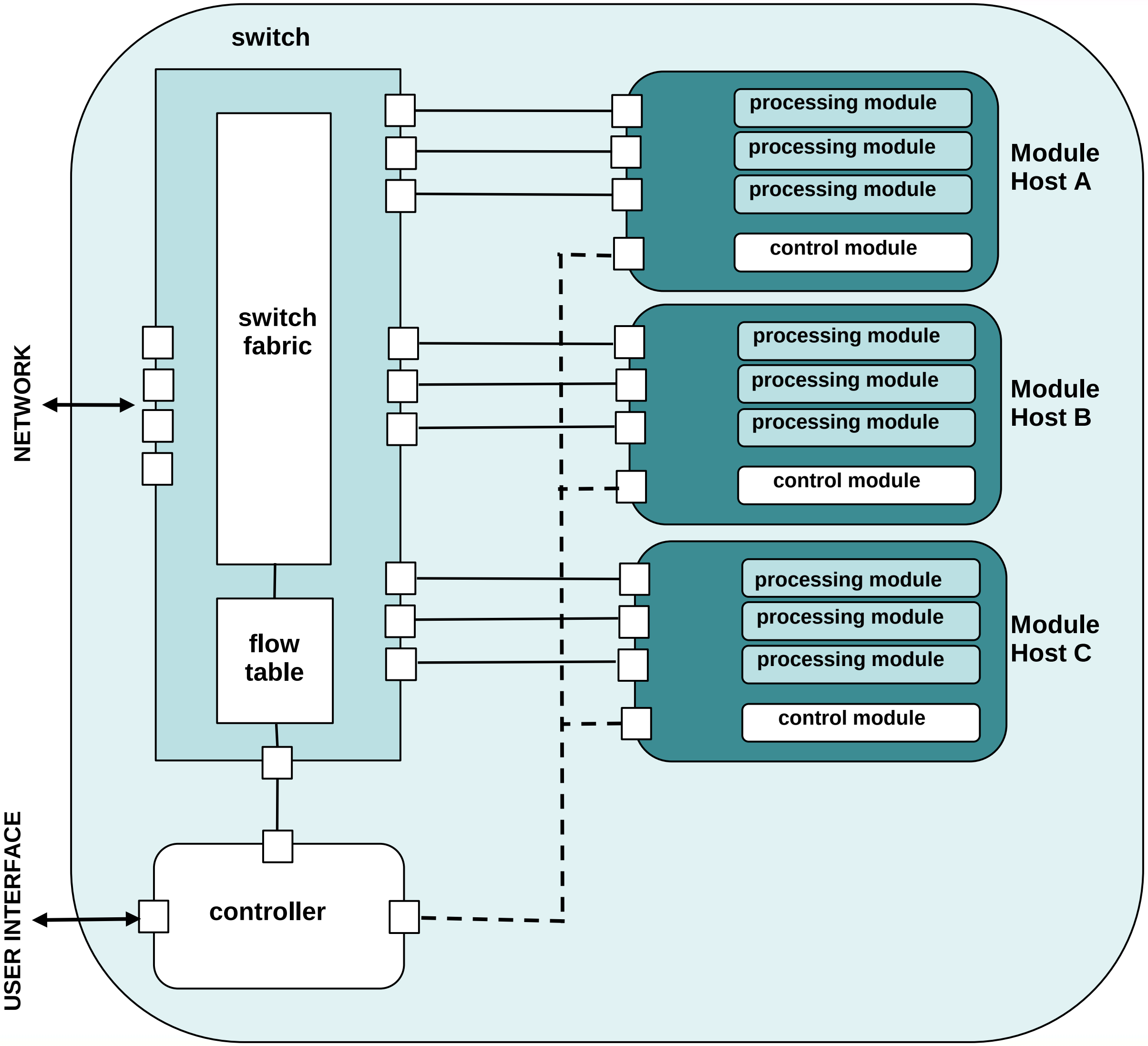
if the routing entry corresponds to an elephant flow, the entry is cached in DPX

Distributed Flow Processing

Distributed Flow Processing

- Building blocks:
  - Programmable switch (e.g., OpenFlow)
  - Commodity servers
  - Virtualization
  - Flow processing SW (e.g. Click)
  - Control SW (e.g., NOX)

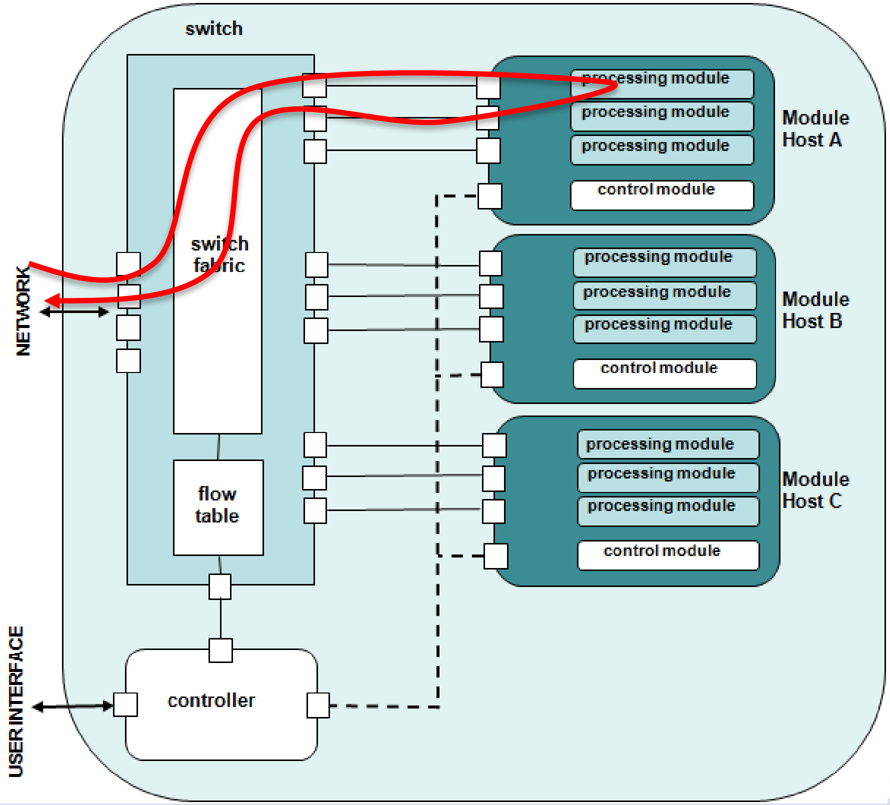
- Properties:
  - Flexibility
  - Scalability
  - Fault tolerance
  - Low cost



Distributed Flow Processing Scenarios

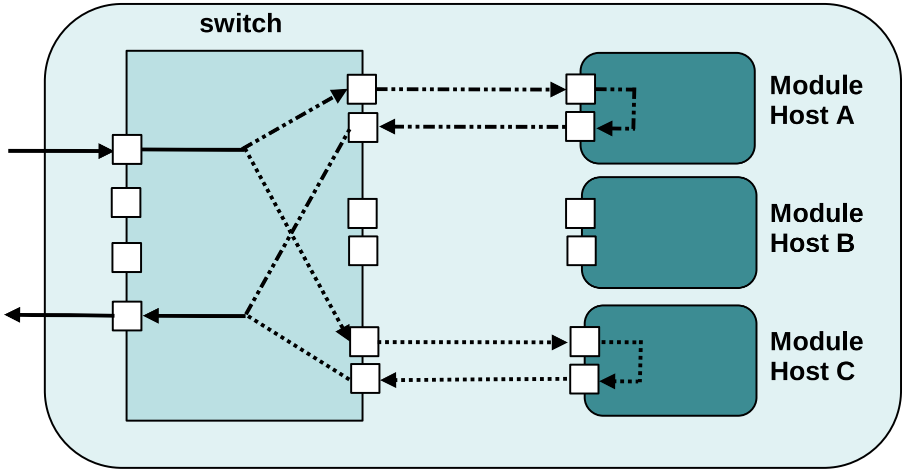
- Typical flow processing scenario:
  - The switch forwards a flow to a module hosting a suitable processing element
  - Upon processing, the flow is sent back to the switch and then is forwarded onto the network

- Other flow processing scenarios:
  - Parallel
  - Serial
  - Traffic splitting (offloading)
  - Inclusion of third-party hardware



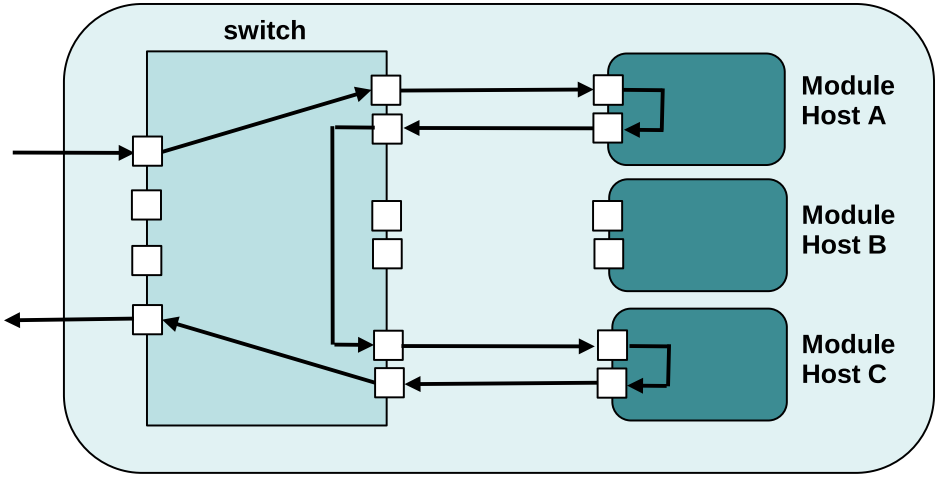
Parallel Flow Processing

- Different flows are forwarded to different hosts (with the same processing module)
- Traffic load balancing, e.g. using ECMP (Equal Cost Multi-Path) algorithm
- Parallel processing of a single flow might cause packet reordering



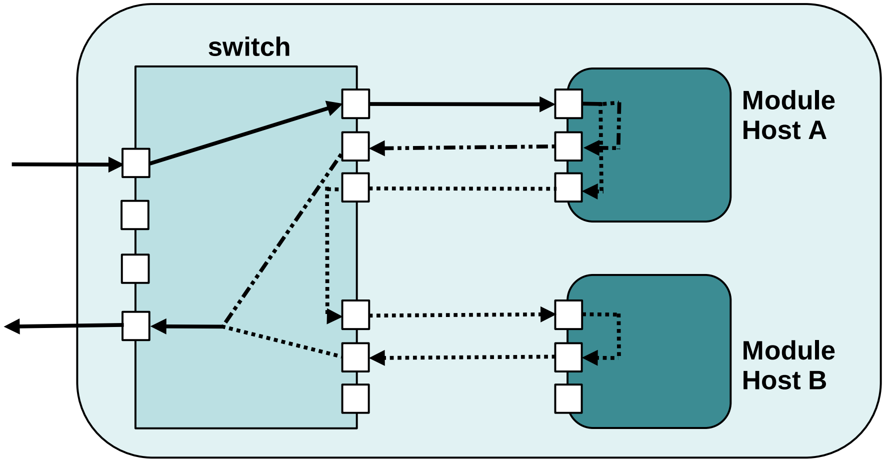
Serial Flow Processing

- A flow is processed by more than one modules sequentially
- Each module typically carries out a different flow processing operation
- Suitable for applications that require different types of flow processing in a given order:
  - e.g., VPN
    - encryption at host A
    - encapsulation at host C



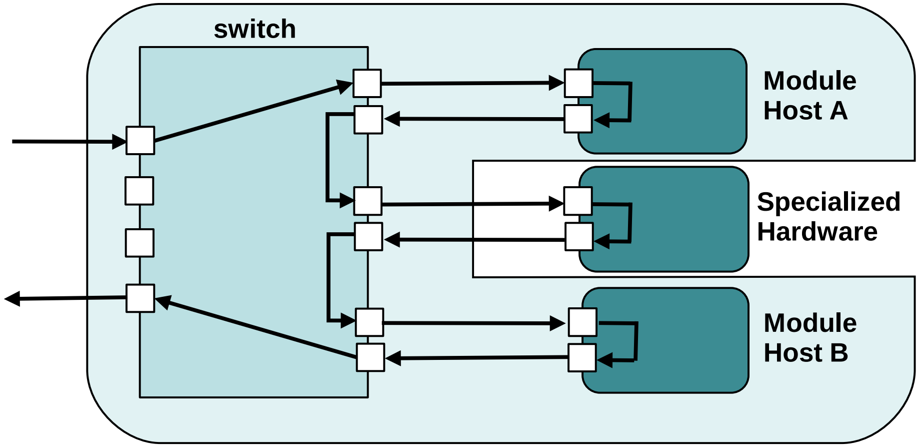
Traffic Splitting

- A processing module can be used to split a subset of flows and forward to another module for further processing
- Intrusion detection:
  - Inspection of a flow aggregate at host A
  - Suspicious flows are forwarded to host B for in-depth intrusion detection
  - Remaining flows are sent back to the switch which forwards them to the network



Third-party Hardware

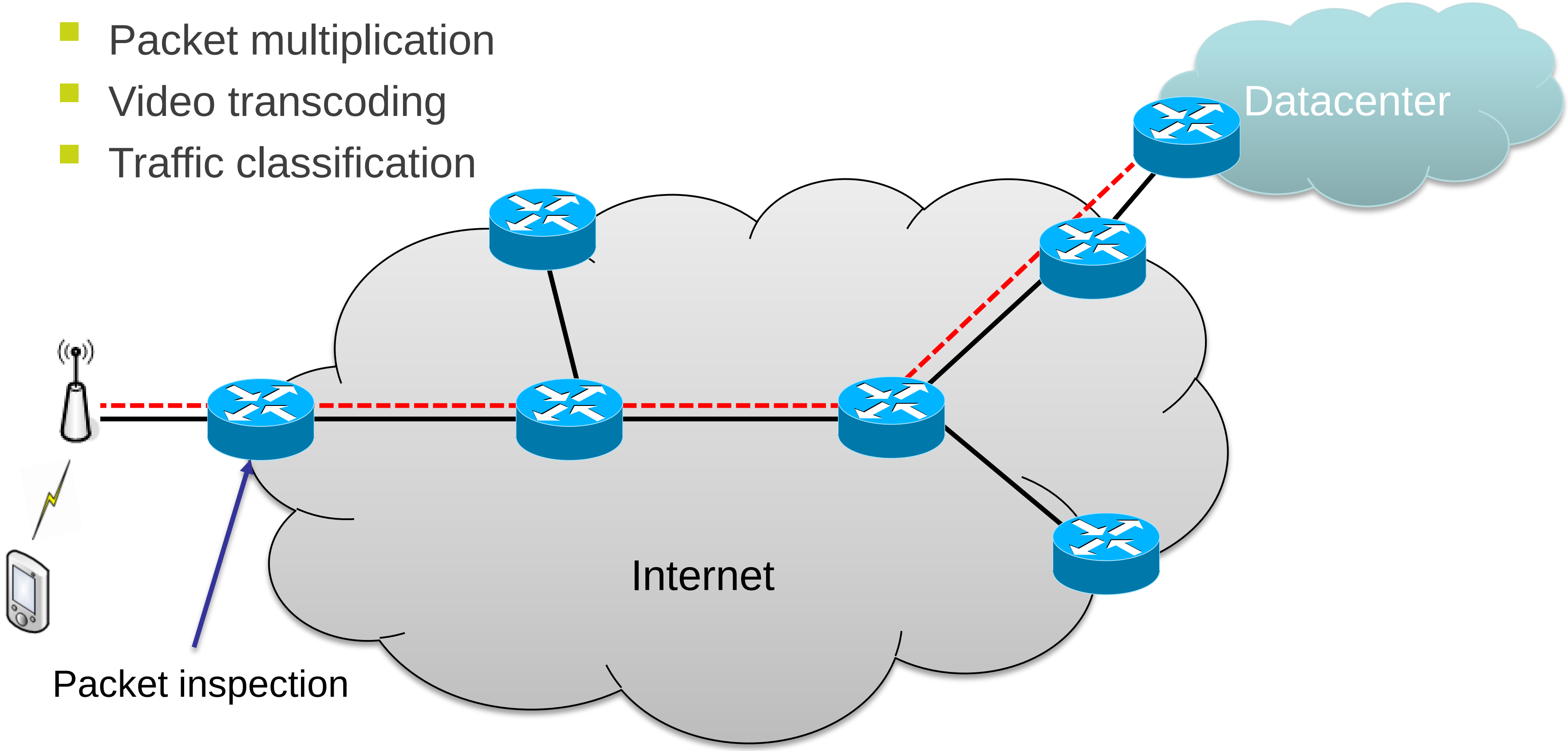
- Third-party specialized hardware (middleboxes) can be integrated in the platform and used for specific flow processing operations:
  - Operators might want to use available middleboxes
  - Software for some flow processing operations might be unavailable or unstable
  - CPU-intensive processing operations may have to be performed on specialized hardware to achieve line rates



In-Network Processing

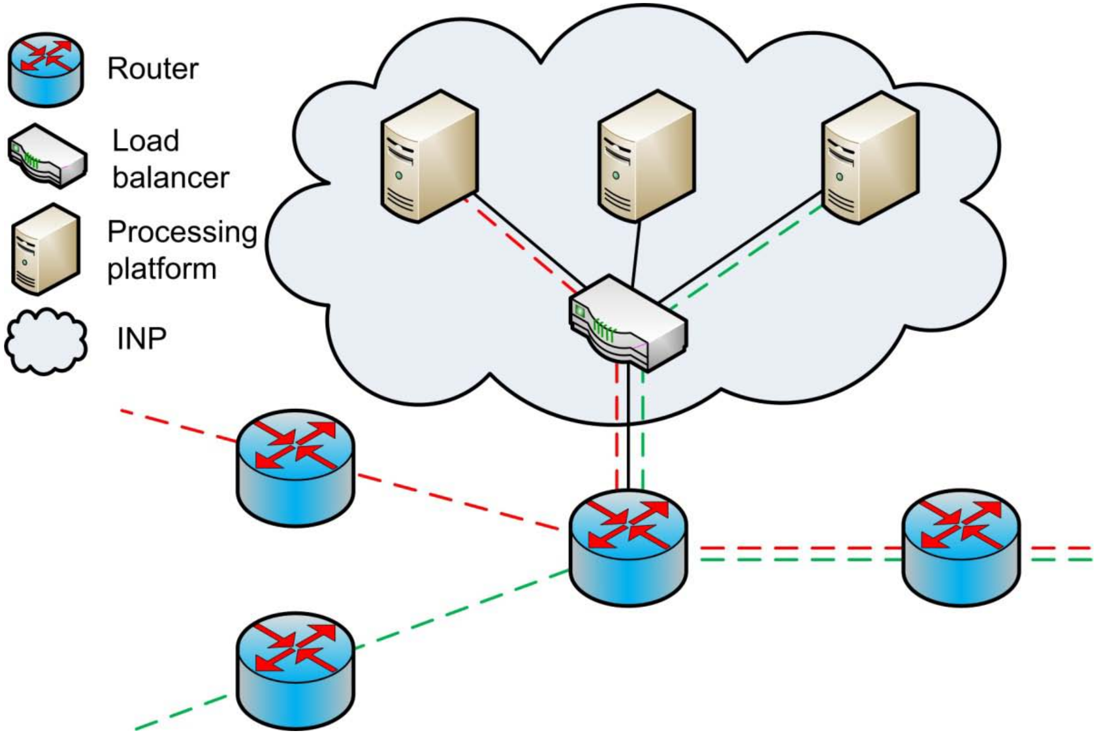
In-Network Processing

- Flow processing incarnated in the network:
  - Packet inspection and filtering
  - Intrusion detection
  - Packet multiplication
  - Video transcoding
  - Traffic classification



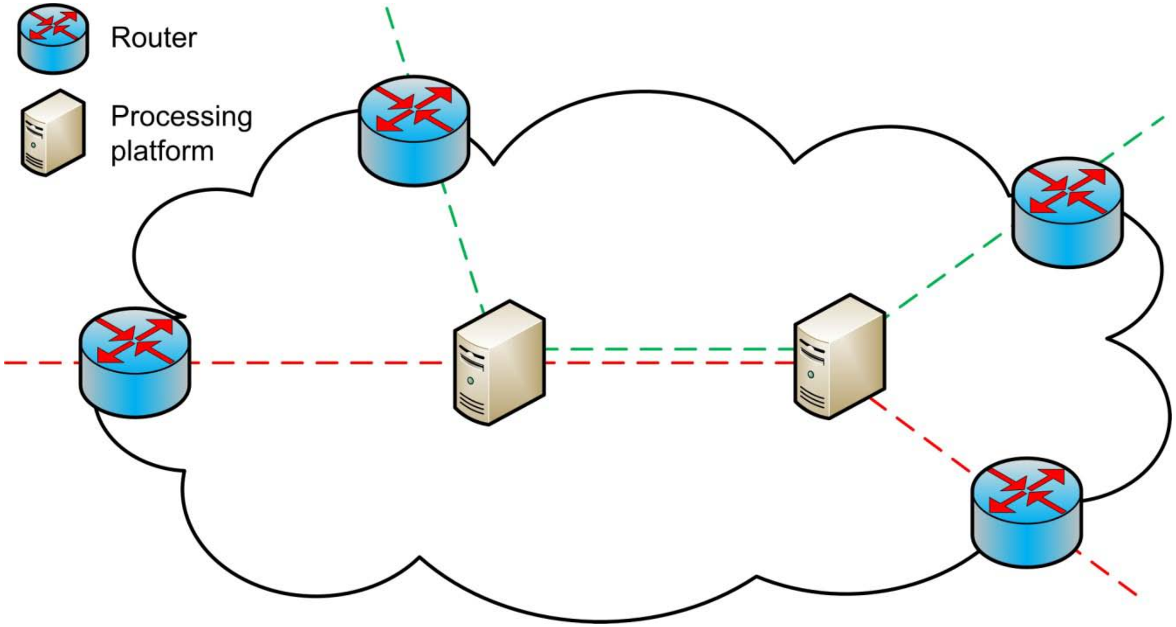
Flow Processing off the Traffic Path

- Traffic redirection is required (e.g., OpenFlow)
  - Only the traffic that needs processing is redirected to the platforms
    - Fewer network devices along the traffic path
  - More bandwidth needed along the paths used for traffic redirection

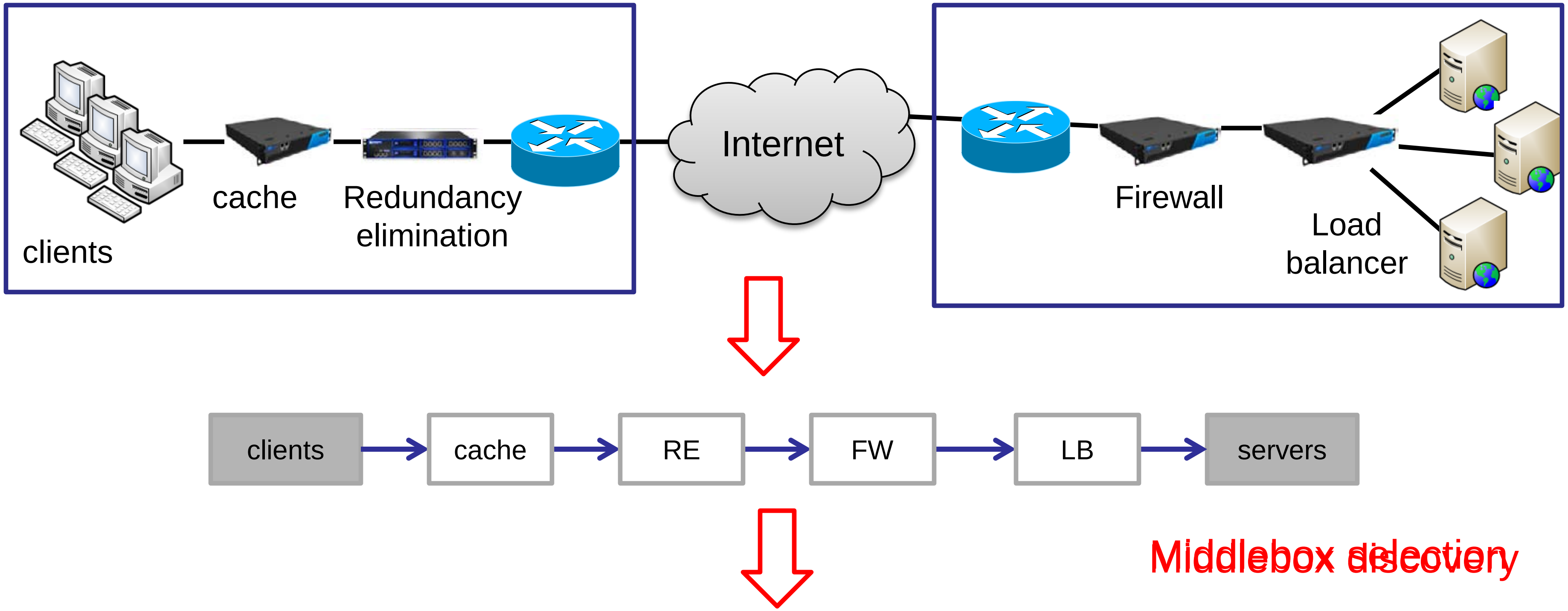


Flow Processing along the Traffic Path

- Where each flow should be processed?
  - Processing load should be distributed across the processing platforms along the traffic path
  - Each processing platform should be aware of the flows assigned to it
    - Encoding platform IDs into flows

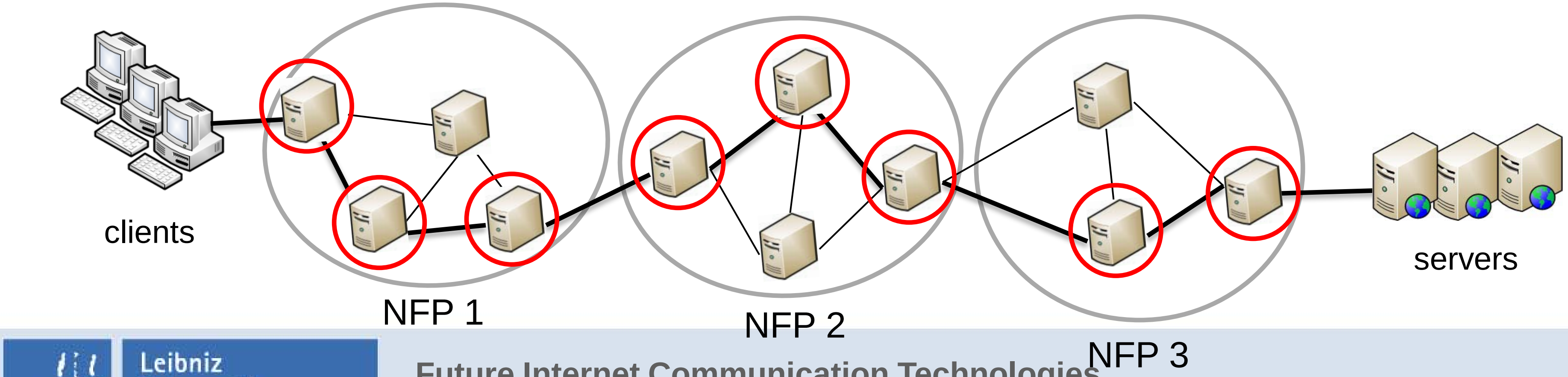


On-Path Flow Processing Across Multiple Providers



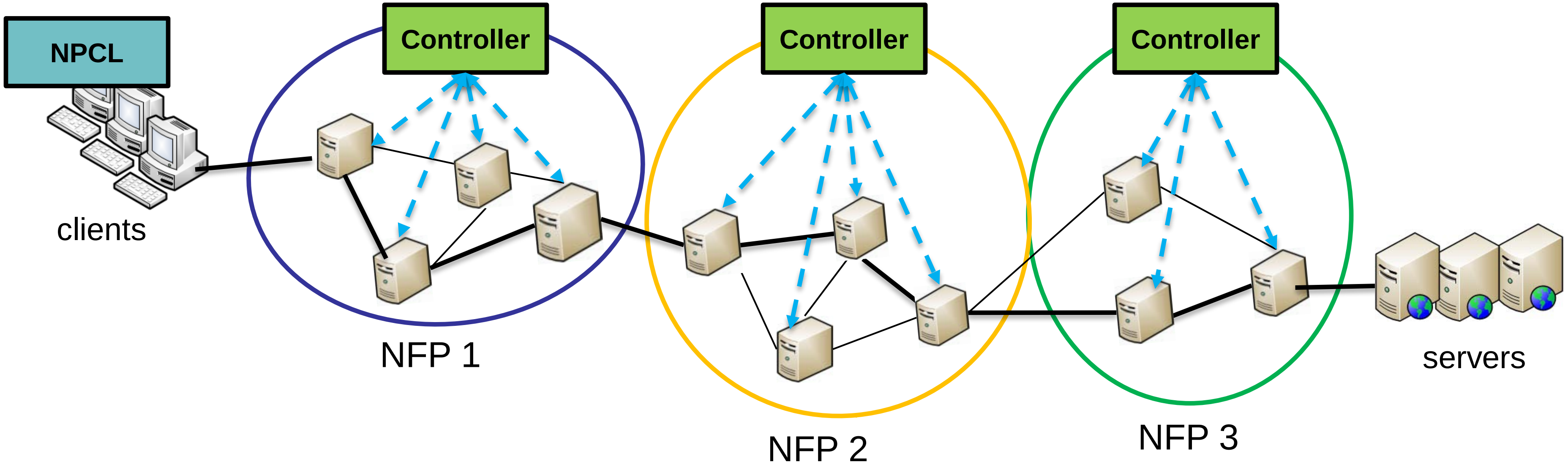
Middlebox selection

Middlebox discovery

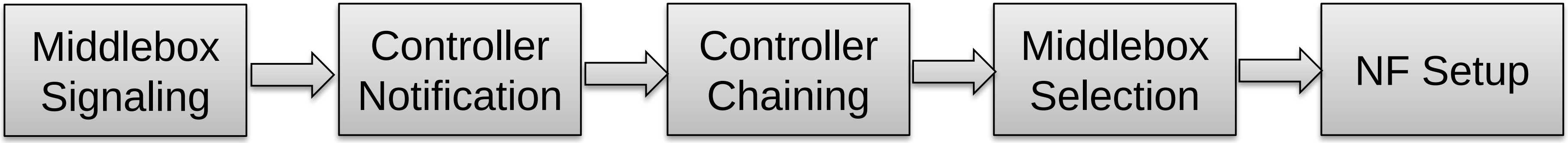


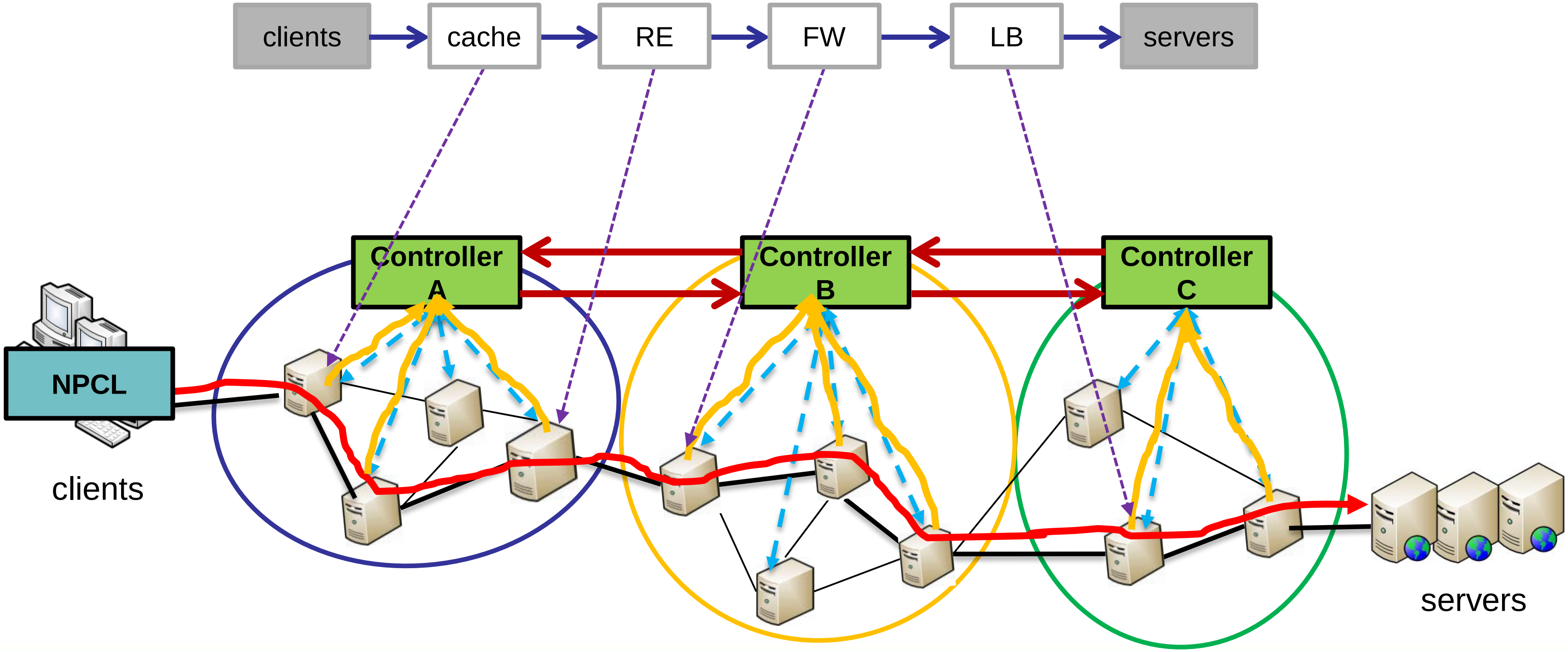
MIDAS Components

- Main components:
  - Consolidated middlebox (CoMB)
  - Centralized CoMB controller in each NFP
  - Network processing client (NPCL)

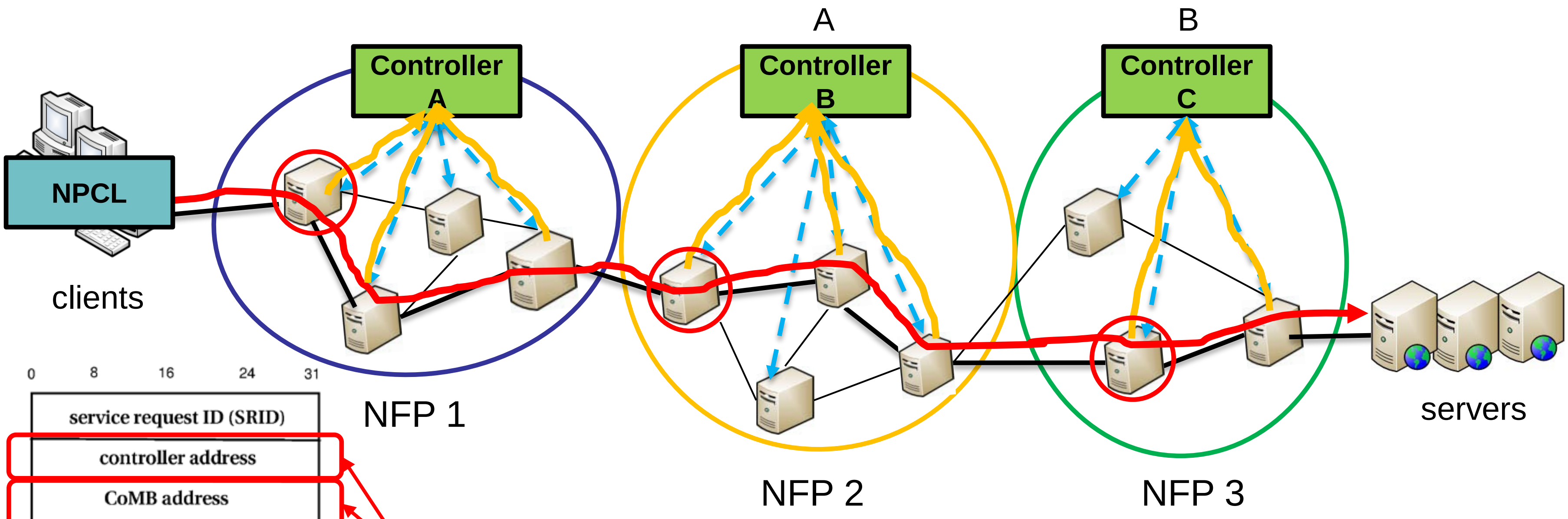


MIDAS Overview





Middlebox Signaling



0 8 16 24 31

	service request ID (SRID)	
	controller address	
	CoMB address	
	processing type identification	
	proximity	
	bandwidth	
flow specification	source IP	
	destination IP	
	source port	destination port
	protocol	

REQUEST message

Extracted by each ingress CoMB and inserted into the DISCOVERY message

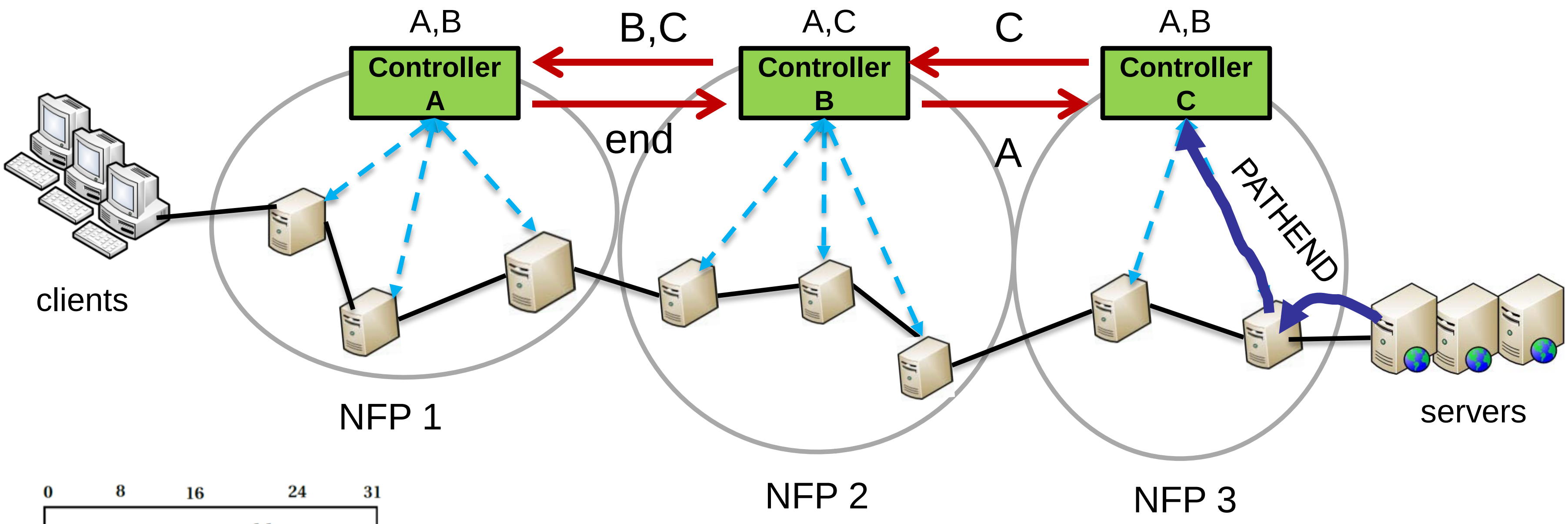
Each CoMB inserts its IP address

0 8 16 24 31

CoMB ID
CoMB utilization
REQUEST message

DISCOVERY message

Controller Chaining



0 8 16 24 31

requester address
service request ID (SRID)
controller address
controller address
⋮
controller address

CONTROLLER message

References

- B. Carpenter and S. Brim, **Middleboxes: Taxonomy and Issues,** RFC 3234, 2002
- J. Sherry, et al., **Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service**, ACM SIGCOMM 2012
- A. Greenhalgh, et al., **Flow Processing and the Rise of Commodity Network Hardware**, ACM Communication Review, 2009
- N. McKeown, et al., **OpenFlow: Enabling Innovation in Campus Networks**, 2008
- B. Nunes, et al., **A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks**
- R. Sherwoord, et al., **Can the Production Network Be the Testbed?**, USENIX OSDI 2010
- N. Sarrar, et al., **Leveraging Zipf's Law for Traffic Offloading**, ACM Communication Review, 2012
- A. Abujoda and P. Papadimitriou, **MIDAS: Middlebox Discovery and Selection for On-Path Flow Processing**, IEEE COMSNETS 2015