Network Virtualization

Future Internet Communications Technologies

Prof. Dr. Panagiotis Papadimitriou

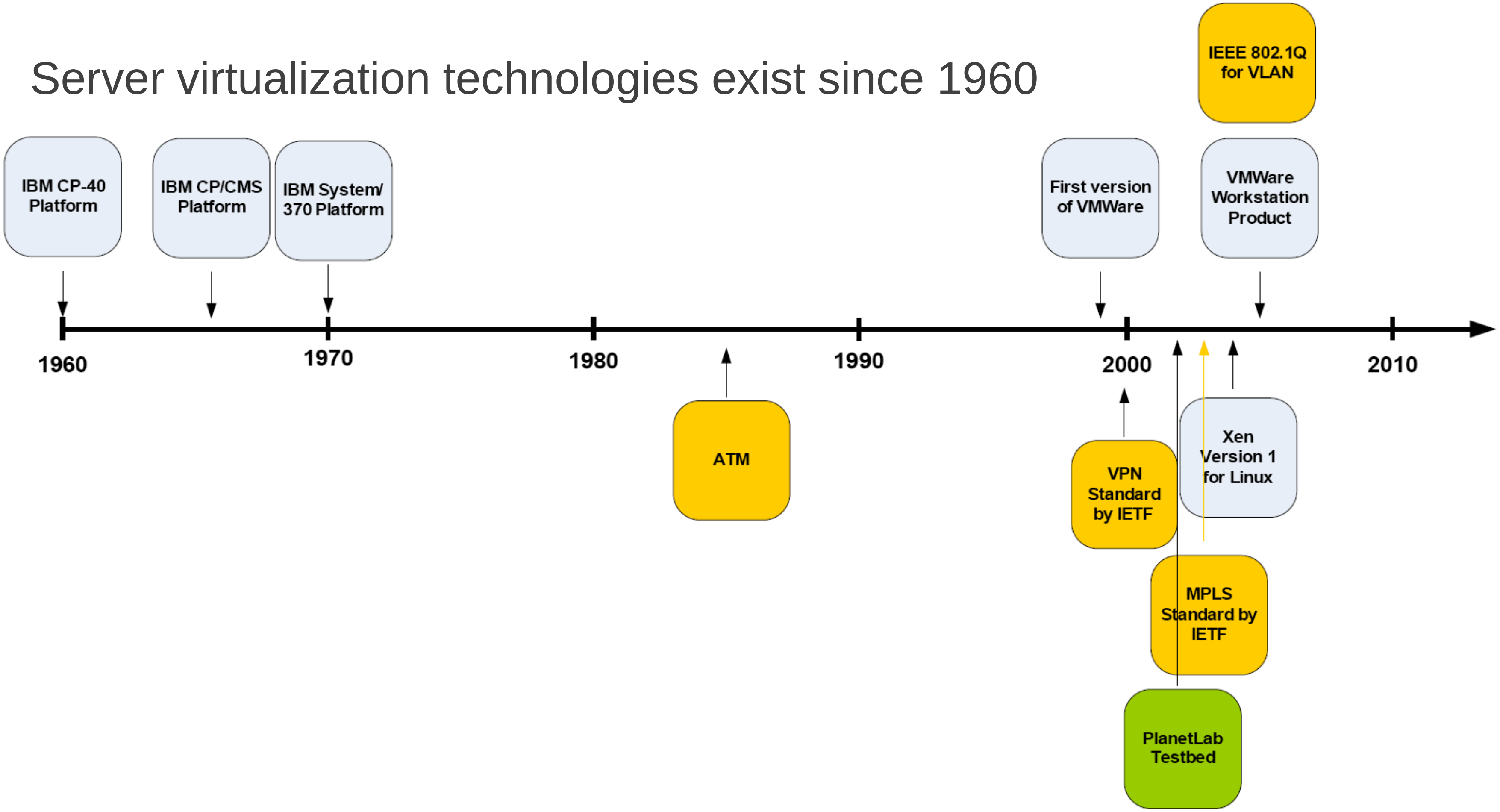
Outline

- Introduction to Network Virtualization
- Network Virtualization Technologies
- Server Virtualization
- Link Virtualization
- Network Interface Virtualization
- Router Virtualization
- Virtual Network Embedding
- Virtual Network Embedding Across Multiple Substrate Networks
- Virtual Link Setup
- Software-Defined Network Virtualization

Introduction to Network Virtualization

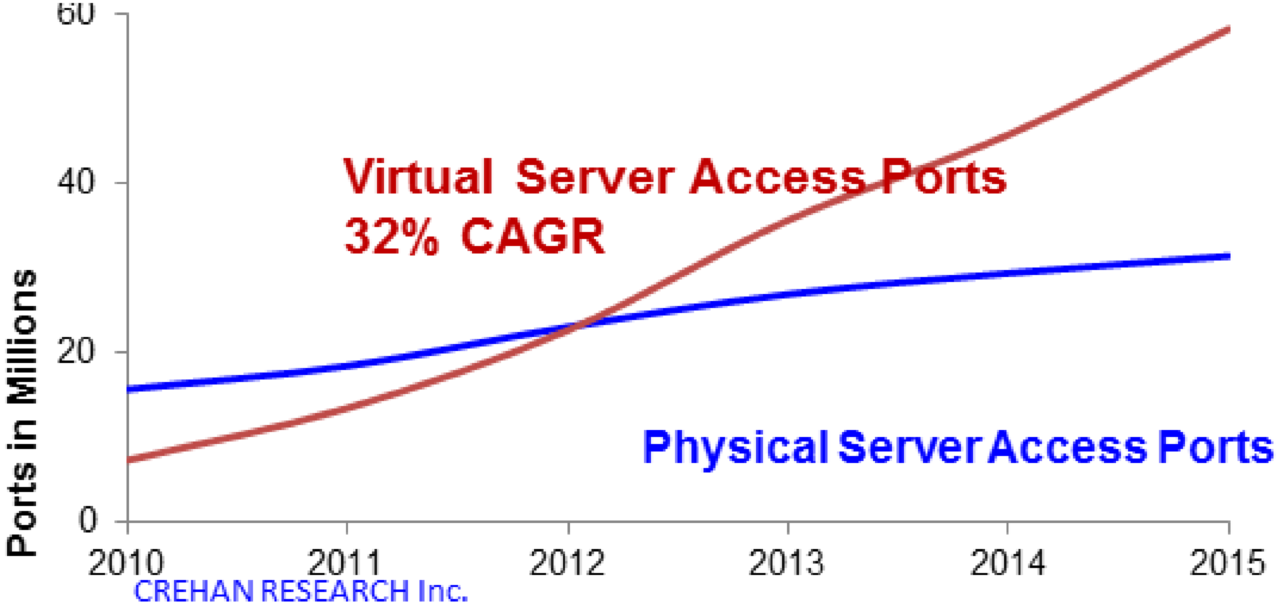
Virtualization

- Virtualization is not a new concept (e.g., virtual memory)
- Server virtualization technologies exist since 1960



Network Access Virtualization

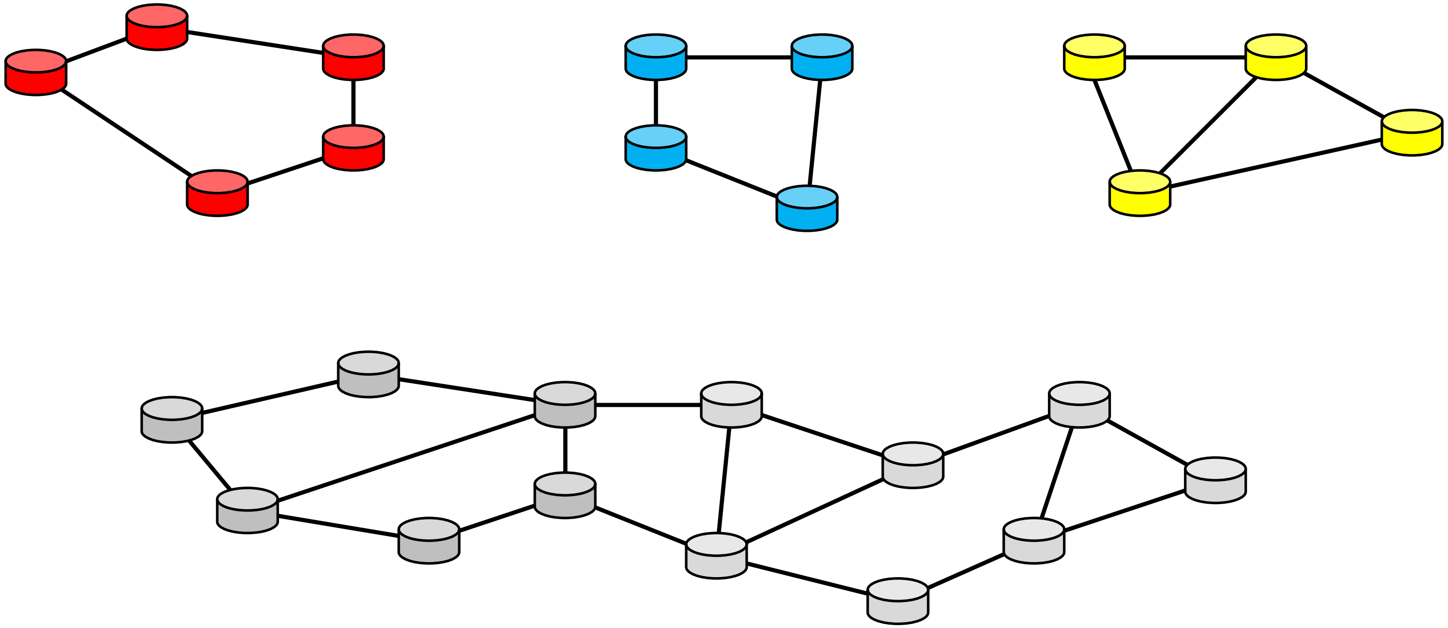
- The number of virtual access ports increases by 32%



CREHAN RESEARCH Inc.

Network Virtualization

- Multiple virtual networks can coexist on top of a shared physical (substrate) network.
- A virtual network is composed of virtual nodes and links.



Network Virtualization Benefits

- Resource sharing (e.g., CPU, bandwidth)
- Independent network management
- Abstraction
- Flexible network management
    - Elasticity
    - Fault tolerance
    - Easier maintenance of physical equipment
- Excellent platform for experimentation
    - Isolating experimental from production traffic
- Innovation in architecture design
    - Concurrent deployment of multiple network architectures

Virtual Networks vs. Overlays

- Virtual networks are conceptually similar to overlays:
  - Virtual networks provide a topology composed of virtual nodes and links on top of one or multiple physical networks

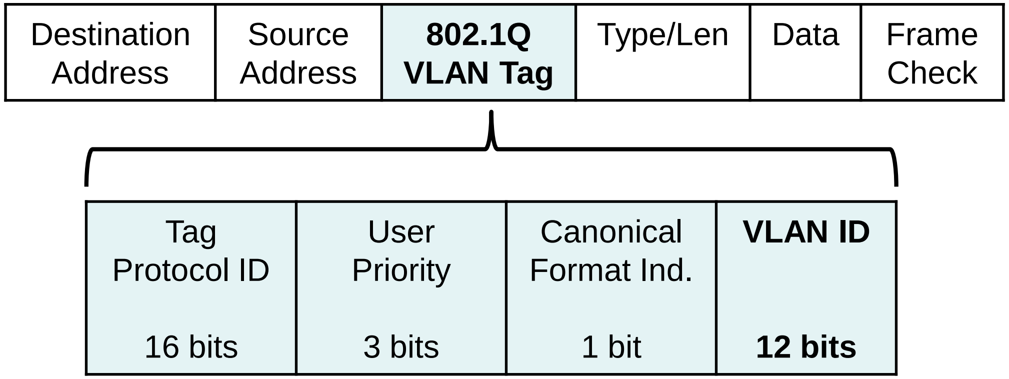
- Overlays:
  - introduce mechanisms above the transport layer to control the routing across the overlay nodes (e.g., multicast)
  - are typically deployed at the network edge
  - do not lead to any fundamental changes in the physical infrastructure

- Virtual networks can provide higher control on routing and packet delivery by utilizing virtual routers with customized protocol stacks and enhanced programmability

Network Virtualization Technologies

Virtual Local Area Networks (VLANs)

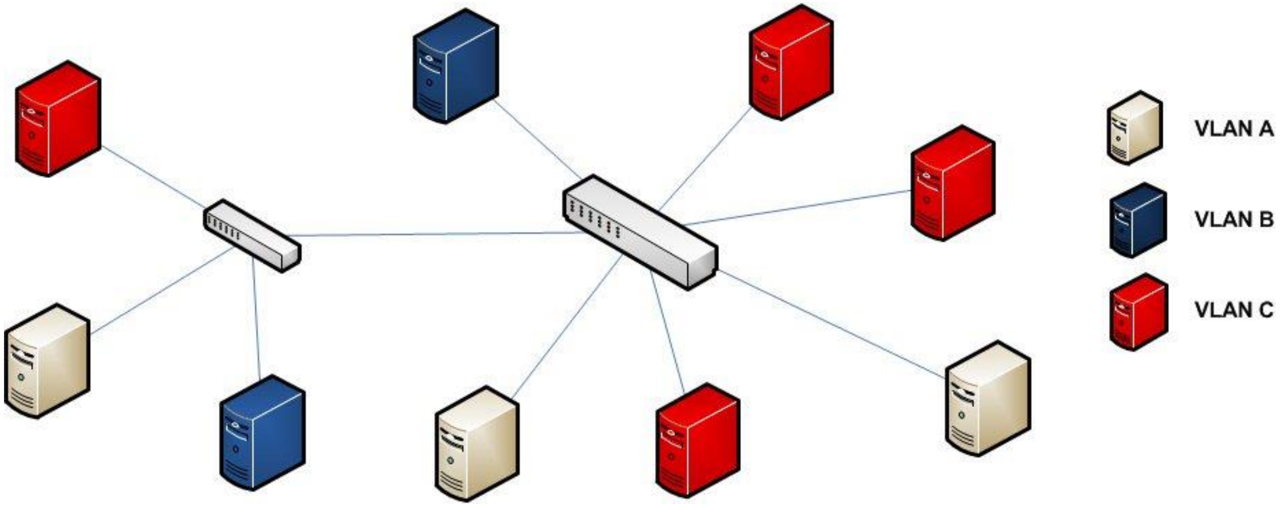
- VLANs isolate groups of hosts into separate broadcast domains using the IEEE 802.1Q standard
  - ✓ Widely available (supported by many switches)
  - ✓ Up to 4096 VLANs
  - ✗ Limited capability for WANs



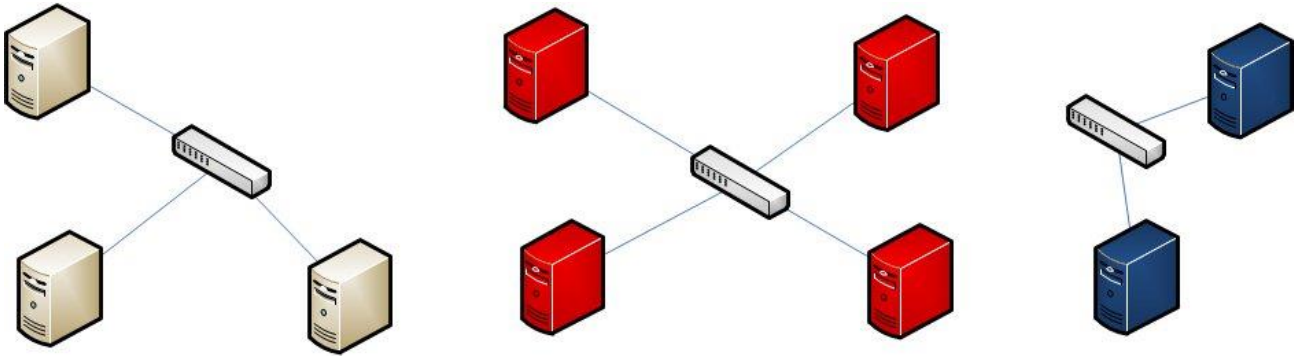
Destination Address	Source Address	**802.1Q VLAN Tag**	Type/Len	Data	Frame Check

Tag Protocol ID 16 bits	User Priority 3 bits	Canonical Format Ind. 1 bit	**VLAN ID** **12 bits**

VLAN Physical and Logical View



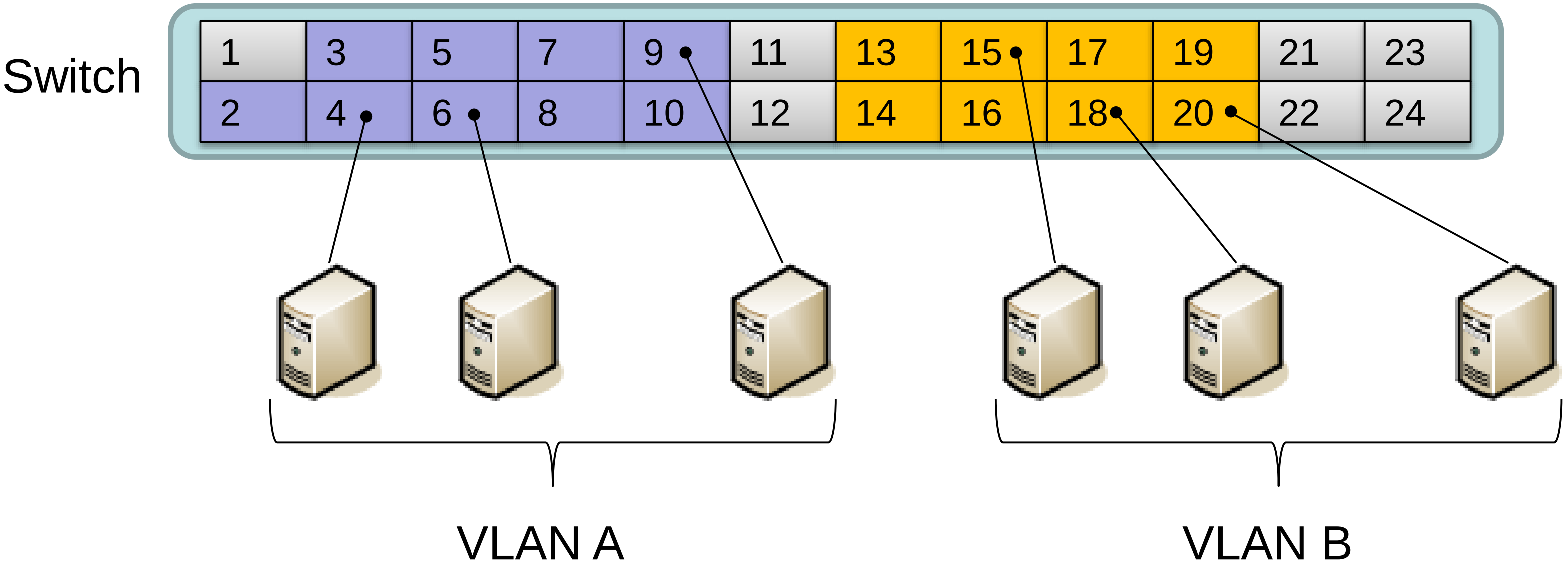
Physical View



Logical View

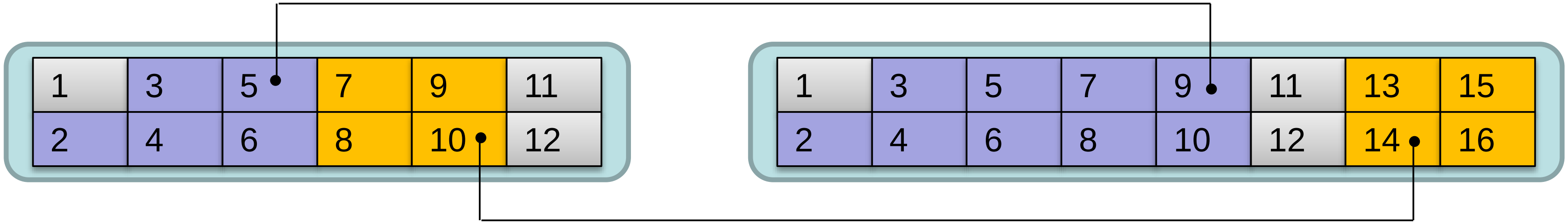
VLAN Configuration

- Switch ports are assigned with VLAN IDs
  - The switch software maintains a table of port-to-VLAN mappings



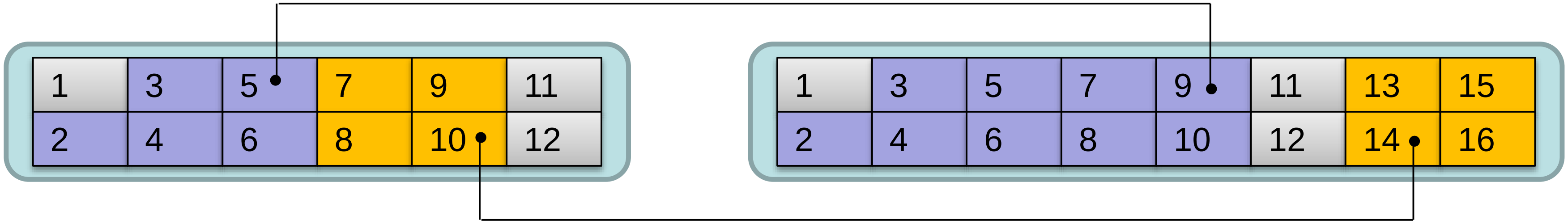
VLAN Configuration Across Multiple Switches

- Separate connection per VLAN
  - Wastes switch ports
  - Static configuration

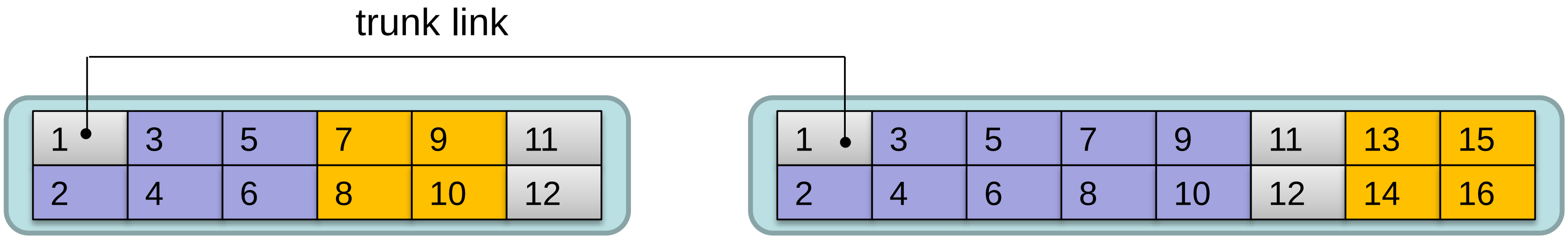


VLAN Configuration Across Multiple Switches

- Separate connection per VLAN
  - Wastes switch ports
  - Static configuration



- VLAN trunking
  - All frames (irrespective of VLAN ID) are forwarded over the trunk link



Virtual Private Networks (VPN)

- VPNs comprise a service that provides:
    - remote access to a private network via tunneling
    - secure access via authentication
    - traffic isolation

- VPNs do not offer any benefits or flexibility in terms of management (e.g., independent resource management) as virtual networks do

- VPN setup usually incurs a long delay

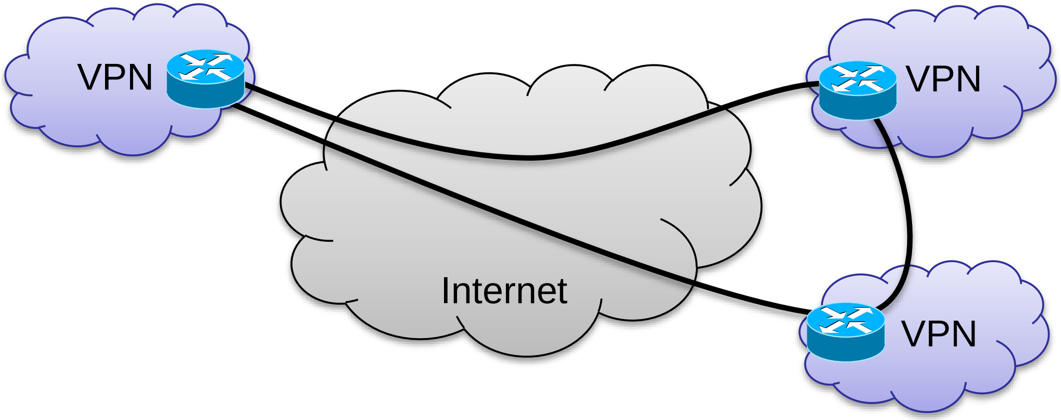
VPN Tunneling

- A tunnel is a method of sending data by encapsulating the data and its protocol information within a different transmission unit

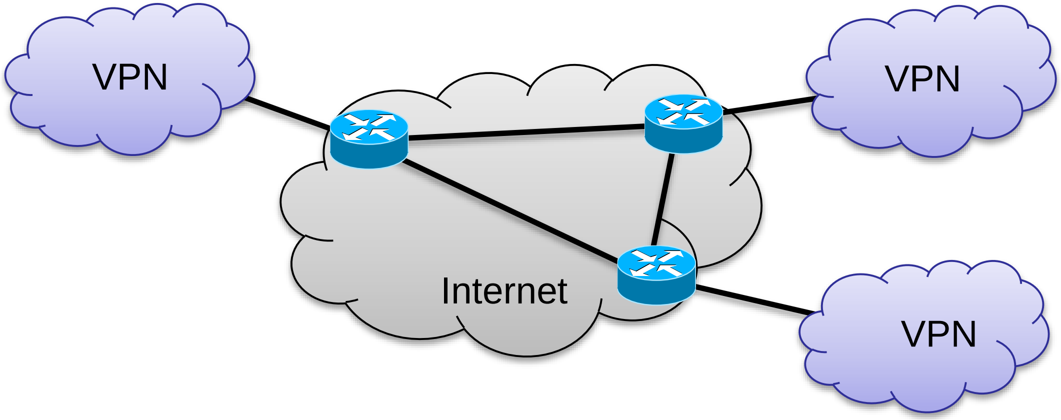
- VPNs encompass various tunneling technologies:
  - Layer 2:
    - Point-to-point:
      - MPLS, L2TP, PPTP, PPP
    - Point-to-multipoint:
      - Virtual Private LANs
  - Layer 3:
    - IP-in-IP
    - Generic Routing Encapsulation (GRE)
    - IPsec

VPN Service Models

- Customer model

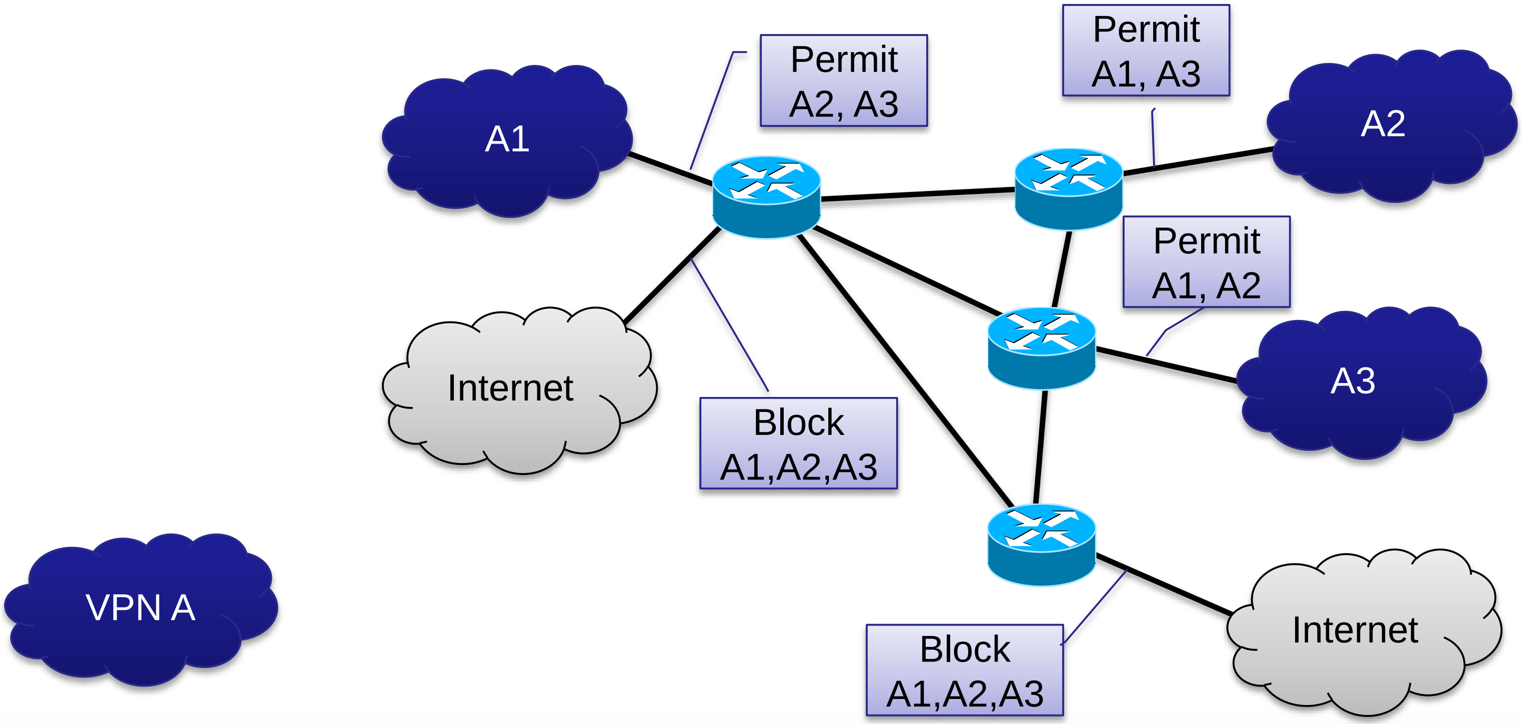


- Provider-provisioned VPN (PPVPN), e.g., BGP/MPLS VPN
  - Logical routers with own:
    - Routing protocol
    - Forwarding table
    - Route filtering



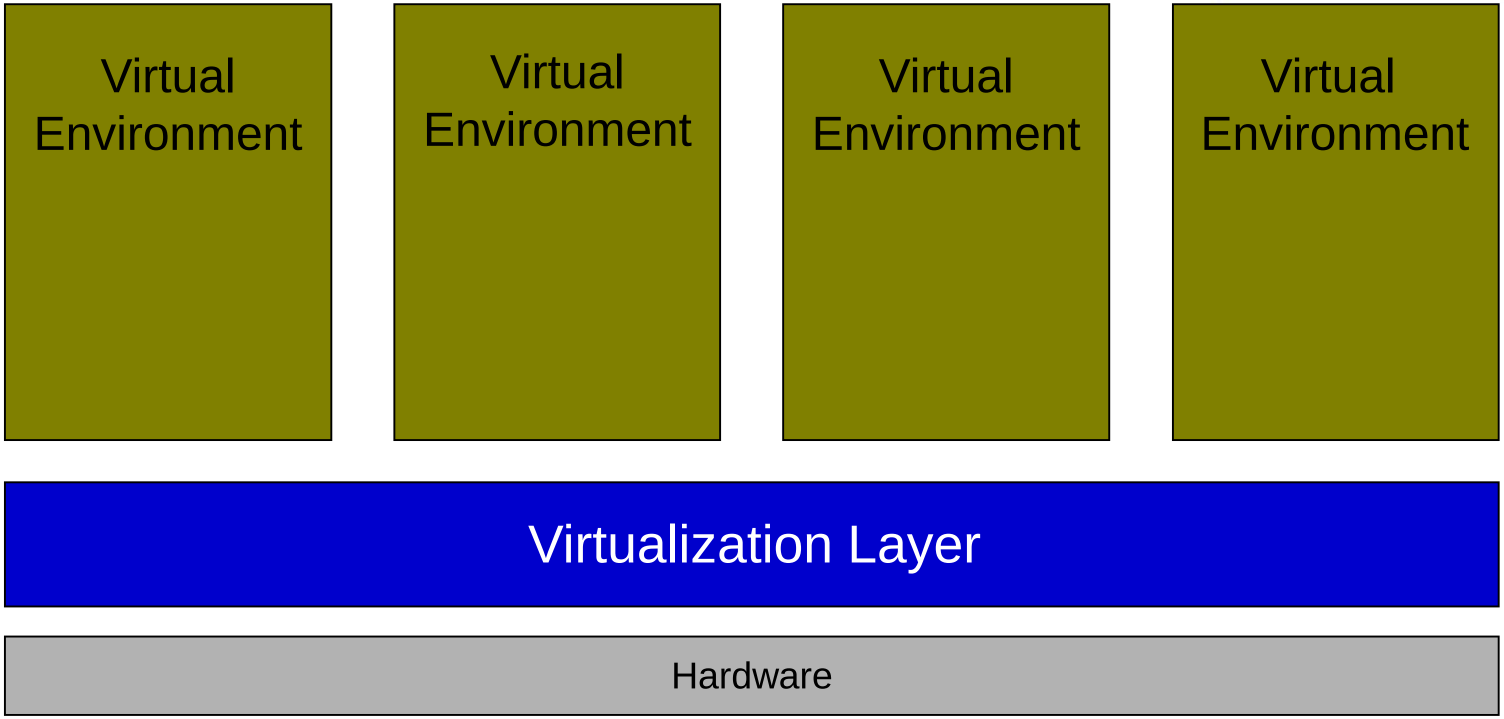
VPN Route Filtering

- Route filtering
  - Controls route propagation so that:
    - networks within a VPN receive route advertisements for other networks in the same VPN
    - networks not in same VPN do not receive these advertisements



Server Virtualization

Server Virtualization Overview



Virtual Environment

Virtual Environment

Virtual Environment

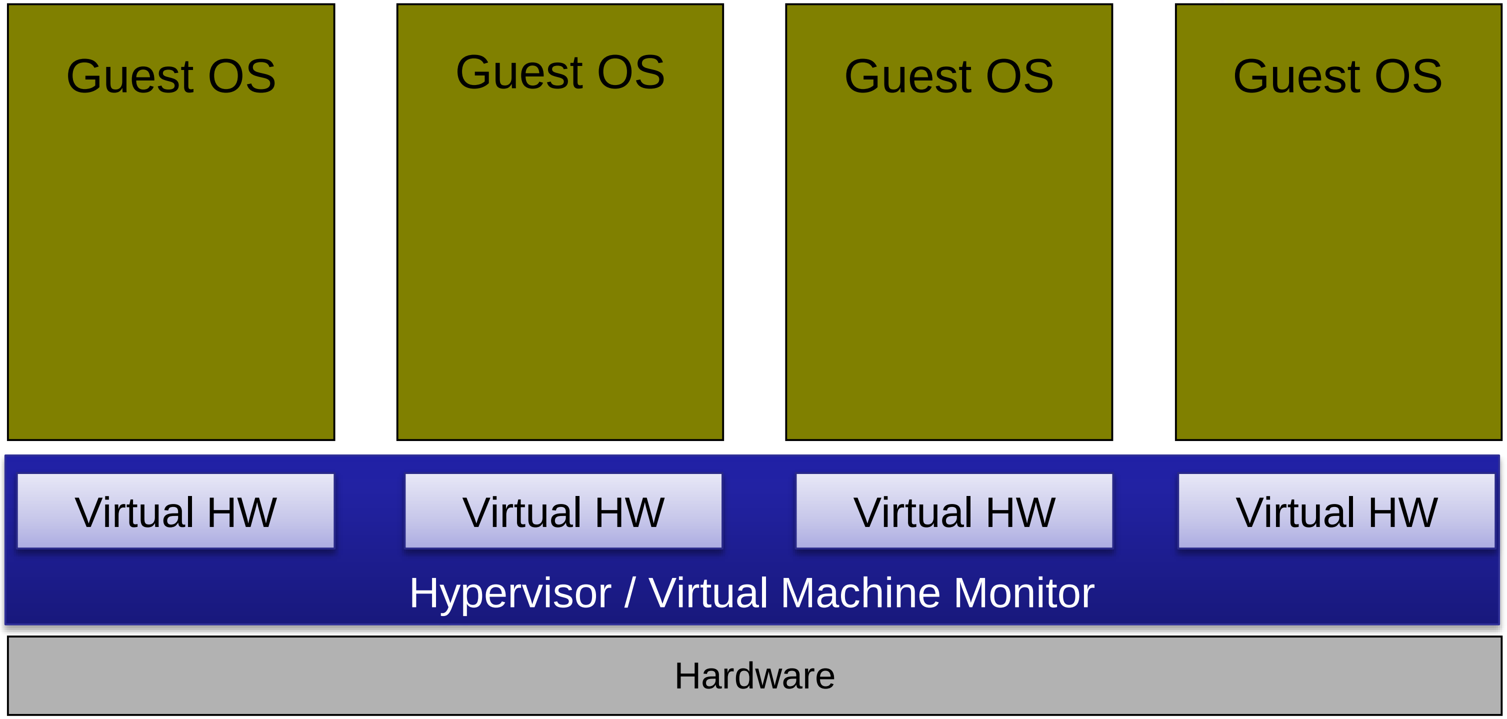
Virtual Environment

Virtualization Layer

Hardware

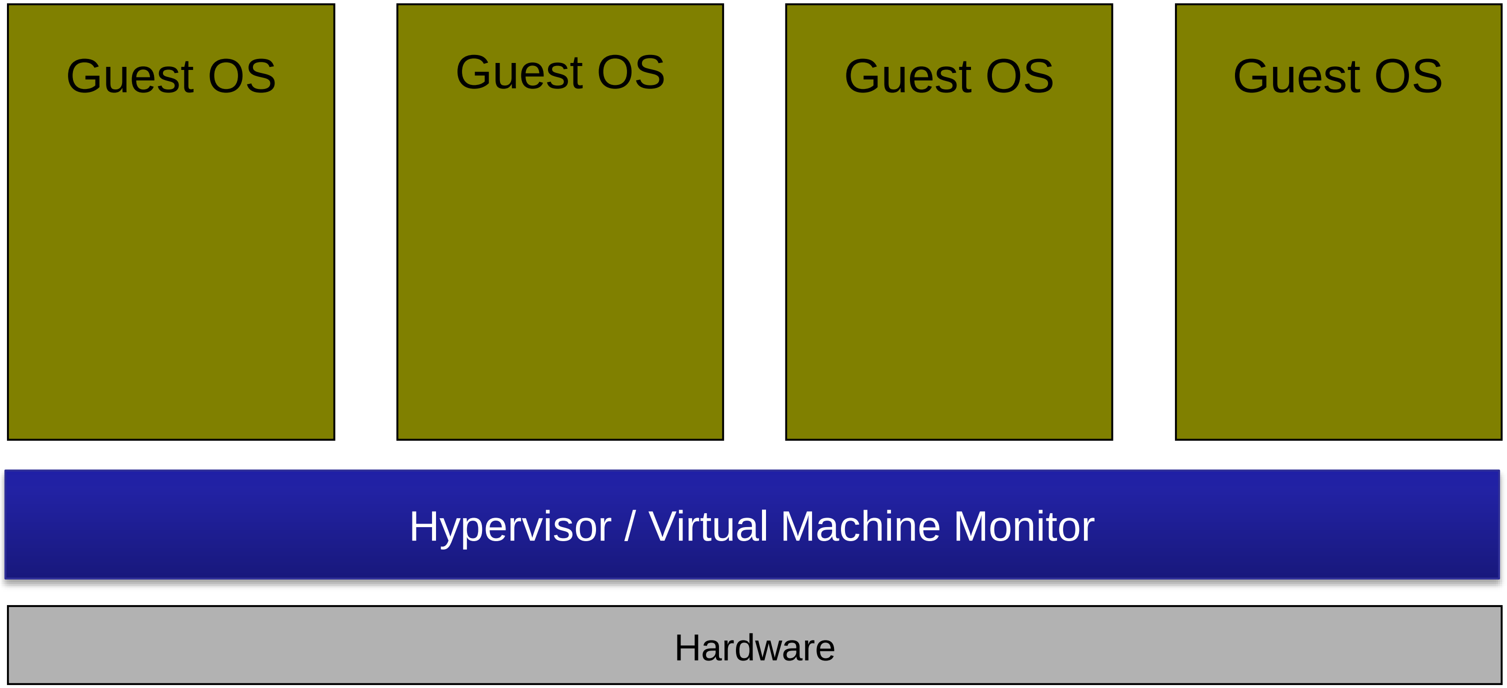
- Server virtualization technologies:
  - aim to provide efficient resource (e.g., CPU, memory) sharing and isolation
  - allow multiple operating systems (OS) to run concurrently on a single host

Full Virtualization



- Full Virtualization (e.g., KVM):
  - provides fully emulated virtual machines (BIOS, devices, memory management) for running multiple OSes
  - allows guest domains to directly execute privileged operations
  - accommodates unmodified guest OSes

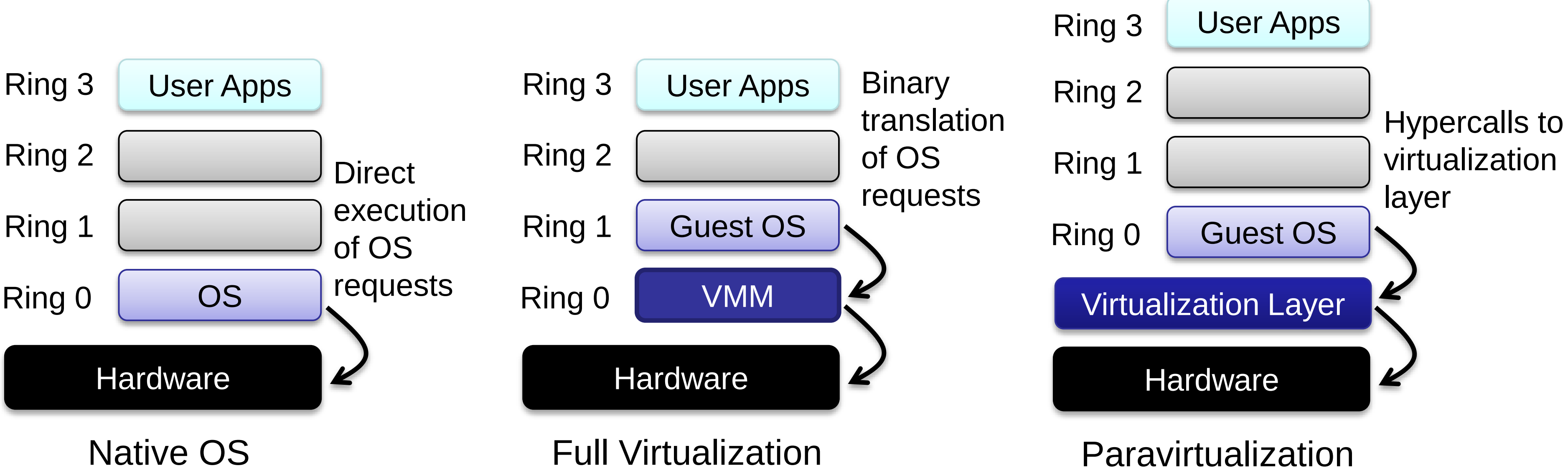
Paravirtualization



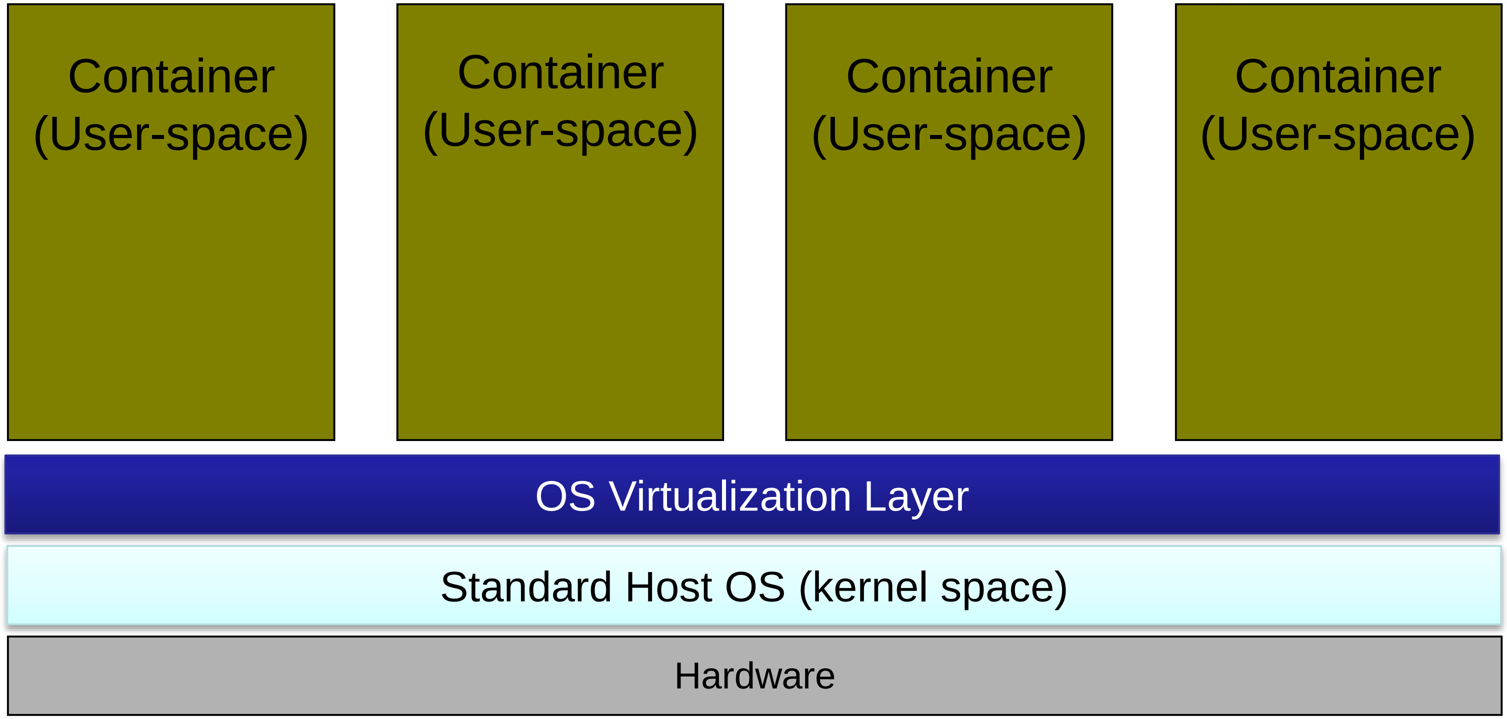
- Paravirtualization (e.g., Xen):
  - provides a hypervisor that runs on top of the hardware
  - does not allow guest domains to directly execute privileged operations (permission is required via hypercalls)
  - requires modified guest OSes

Full Virtualization vs. Paravirtualization

- Protection rings represent hierarchical levels of privilege within the architecture of a computer system
- In full virtualization, VMM provides emulation to handle and modify privileged operations made by unmodified guest OS kernels (as running at Ring 0).
- In paravirtualization, the guest OS kernel is modified to run on the hypervisor
  - Privileged operations that will only run at Ring 0 are replaced with calls to the hypervisor (hypercalls)



OS-based Virtualization



- OS-based virtualization (e.g., OpenVZ, VServer):
  - creates multiple partitions of operating system resources (containers)
  - provides a single kernel instance for all containers

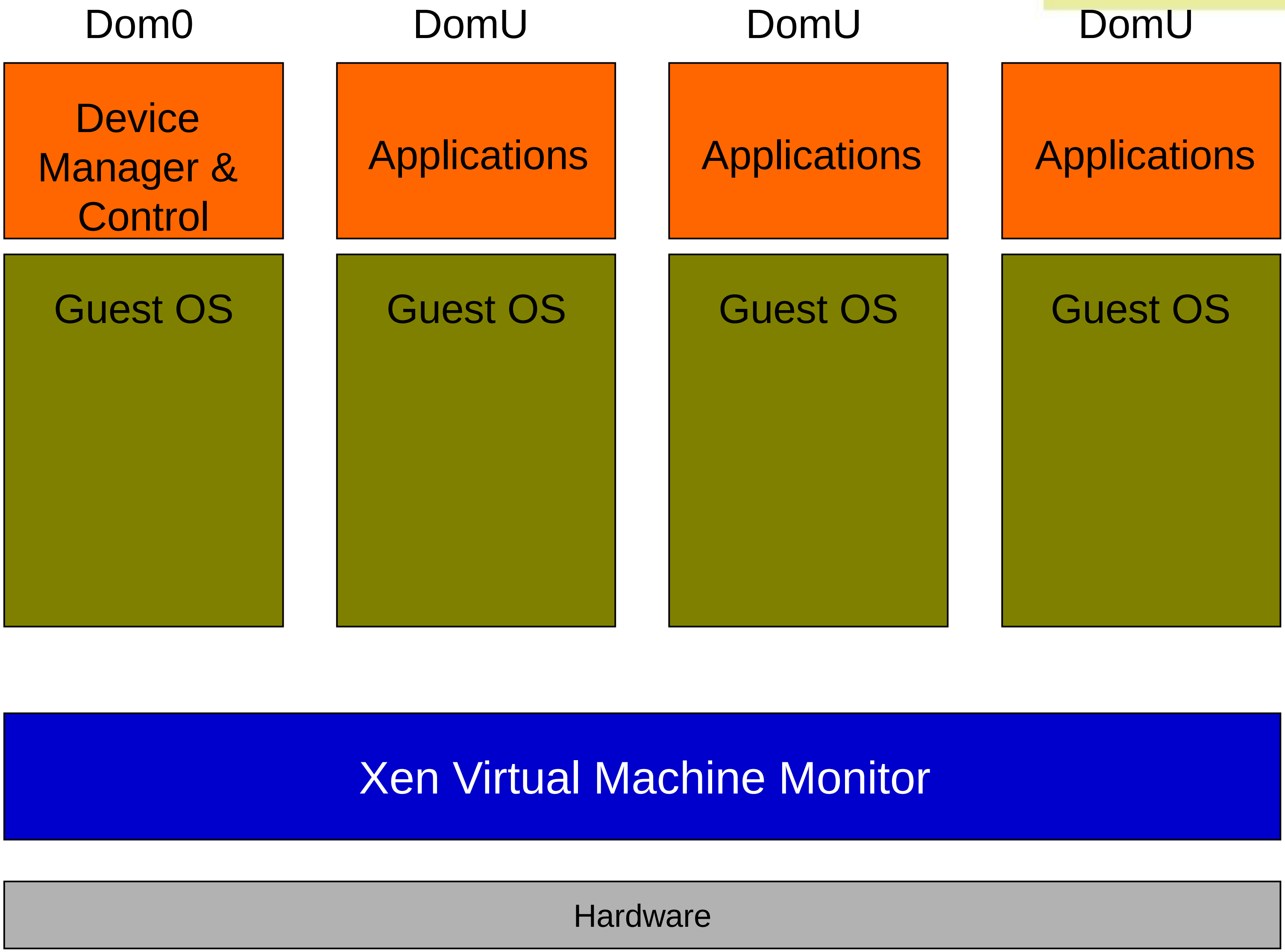
Comparison of Server Virtualization Technologies

- Full Virtualization
  - Maximum isolation
  - Performance penalty

- Paravirtualization (e.g., Xen, Denali)
  - High flexibility (e.g., different guest OSes)
  - Adequate isolation
  - Sub-optimal performance due to overhead

- OS-based virtualization
  - High performance due to small overhead
  - Limited flexibility (single instance of kernel)
  - Lower level of isolation

Xen Overview



Dom0

DomU

DomU

DomU

Device Manager & Control

Applications

Applications

Applications

Guest OS

Guest OS

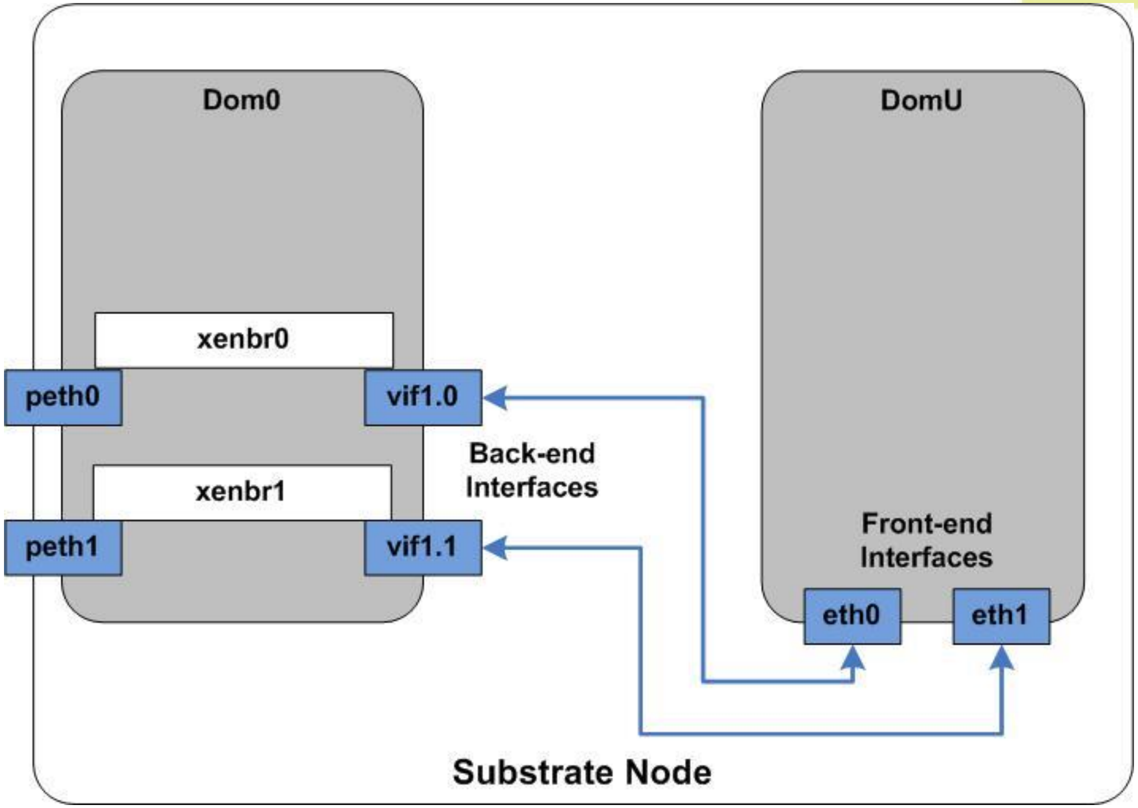
Guest OS

Guest OS

Xen Virtual Machine Monitor

Hardware

Xen Inter-Domain Communication



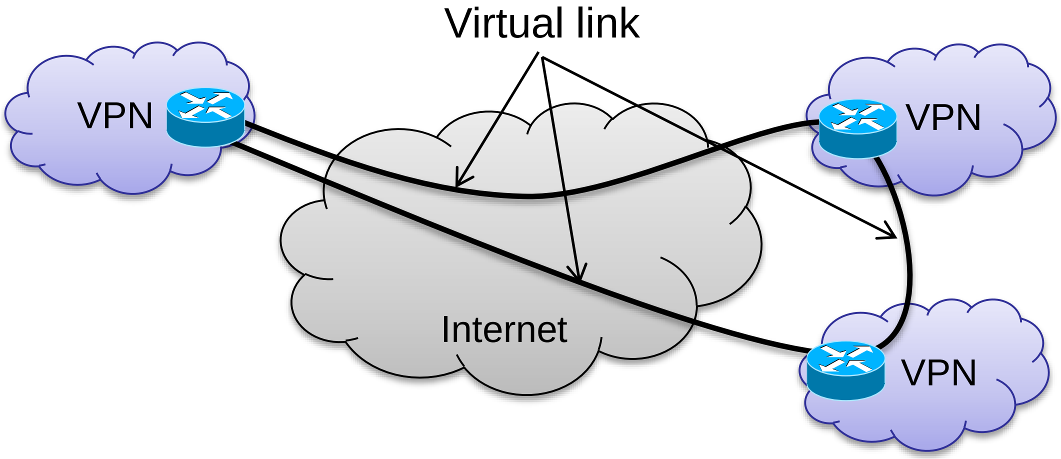
- A NIC driver is split into a front-end and back-end driver
- Xen creates I/O channels to connect front-end with back-end interfaces
- A Linux bridge typically connects a virtual back-end with a physical interface

Link Virtualization

Link Virtualization

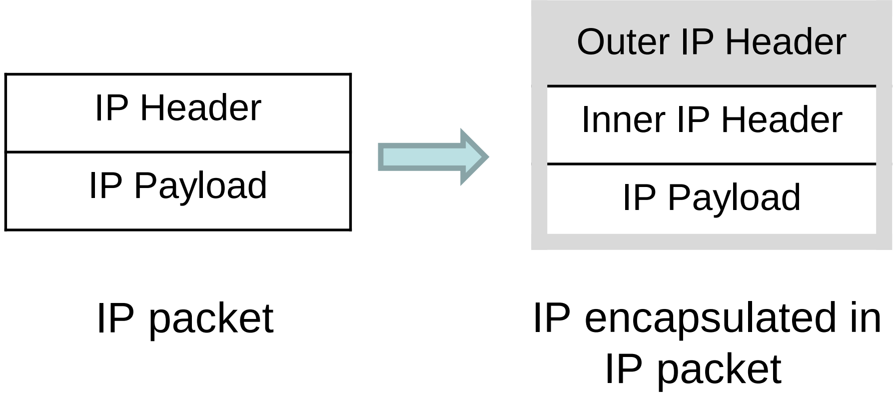
- Virtual links (a.k.a. tunnels) give the appearance of a single link ("pseudo-wire"), although they might span multiple hops

- Various technologies for link virtualization:
  - IP-in-IP
  - Multi-Protocol Label Switching (MPLS)
  - Generic Routing Encapsulation (GRE)
  - Ethernet over IP (EtherIP)

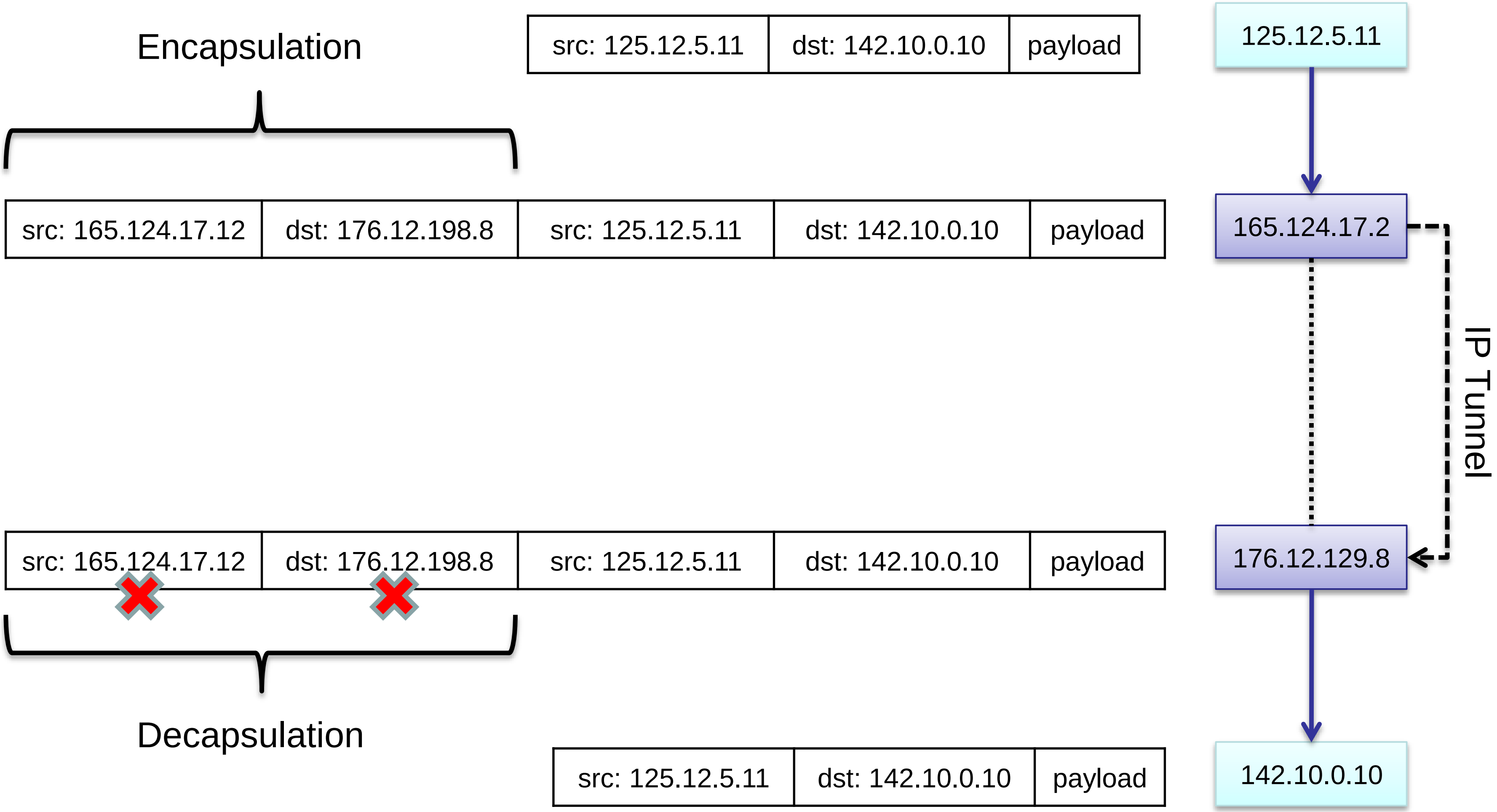


IP-in-IP

- IP-in-IP encapsulates IP packets into other IP packets (v4 or v6)
  - ✓ Widely available
  - ✗ Encapsulation overhead (20 bytes for IPv4)

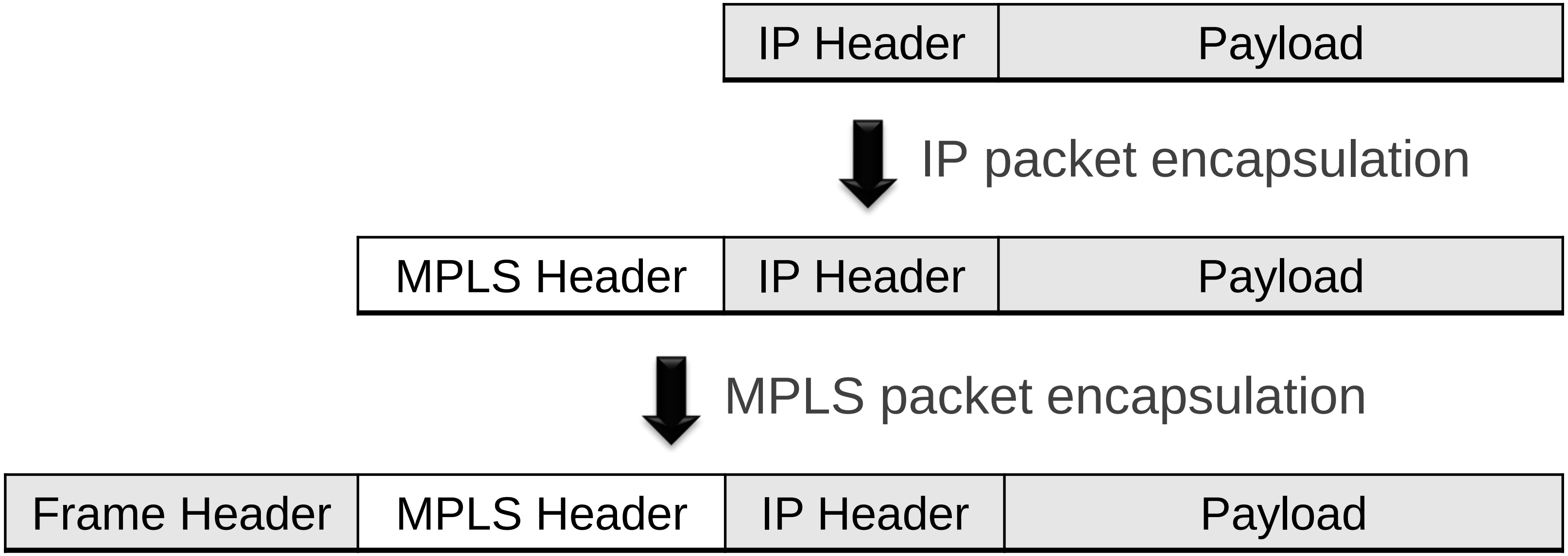


IP-in-IP Example



Multi-Protocol Label Switching (MPLS)

- MPLS encapsulates IP packets into frames (Ethernet, PPP)
  - ✓ Reliable link virtualization
  - ✓ Fast lookup based on a fixed-length label
  - ✓ Deployed by ISPs



IP Header	Payload

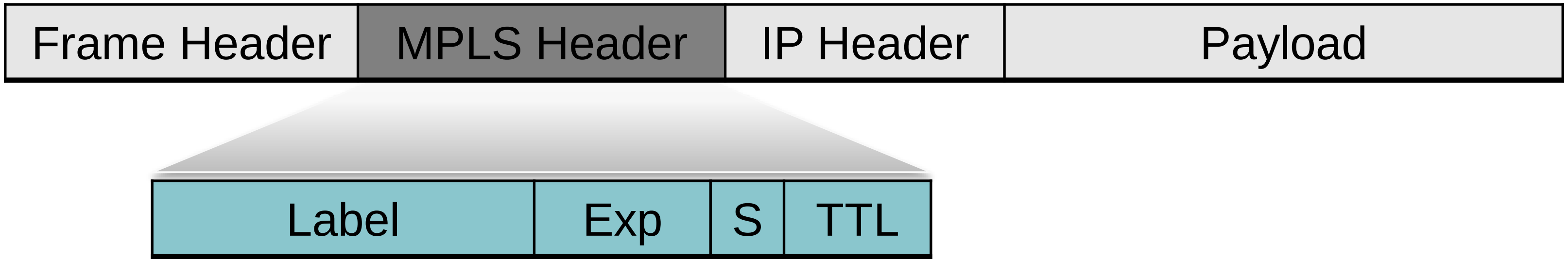
IP packet encapsulation

MPLS Header	IP Header	Payload

MPLS packet encapsulation

Frame Header	MPLS Header	IP Header	Payload

MPLS Header



- MPLS header fields:
  - 20-bit Label value
  - 3-bit Traffic Class field for QoS
  - 1-bit field for stacked MPLS headers
  - 8-bit TTL (Time-to-Live)

MPLS Switching

- An MPLS-capable router (a.k.a. label switch router – LSR) maintains MPLS forwarding tables:

in label	out label	prefix	output interface
8	13	15.1.3	1
12	4	15.4.6	2

- An LSR forwards packets as follows:
  - examines the packet label value
  - performs a label lookup to determine the “out label” and the output interface
  - swaps the packet label value with the “out label” value
  - forwards the packet to the output interface

MPLS Switching at the Edge

in label	out label	prefix	output interface
–	1	15.1.3	1
–	5	15.4.6	2

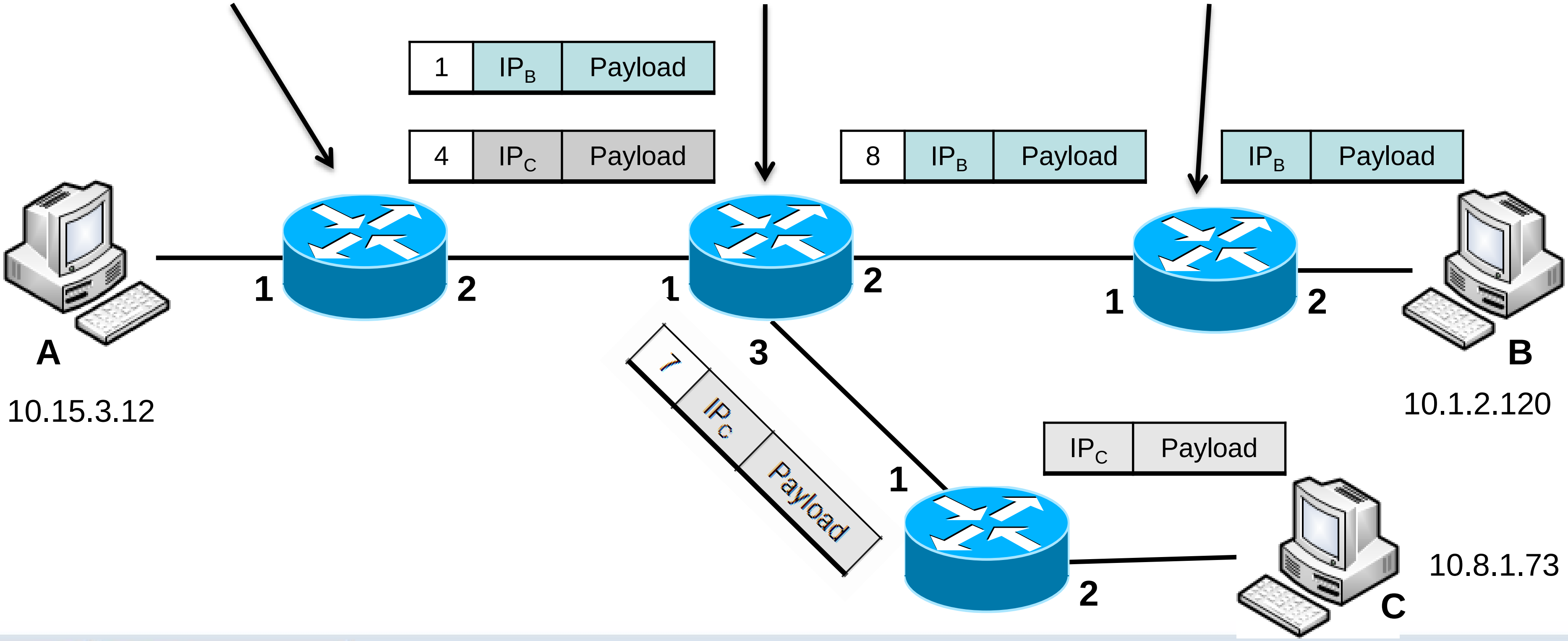
- An edge MPLS-capable router (a.k.a. label edge router – LER) forwards packets as follows:
  - examines the packet IP header
  - performs an IP lookup to determine the “out label” and the output interface
  - sets the “out label” value to the packet label value
  - forwards the packet to the output interface

MPLS Example

in label	out label	prefix	output interface
–	1	10.1.2	2
–	4	10.8.1	2

in label	out label	prefix	output interface
1	8	10.1.2	2
4	7	10.8.1	3

in label	out label	prefix	output Interface
8	–	10.1.2	2



Generic Routing Encapsulation (GRE)

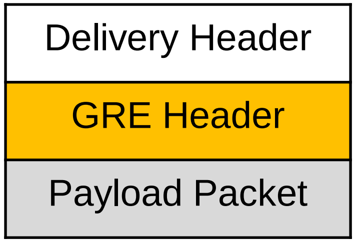
- Generic Routing Encapsulation (GRE)
  - ✓ General packaging protocol (can package any protocol's packets)
  - ✓ 4-byte (optional) key for de-multiplexing
  - ✓ Small encapsulation overhead (4 bytes without optional header fields)

32 bits

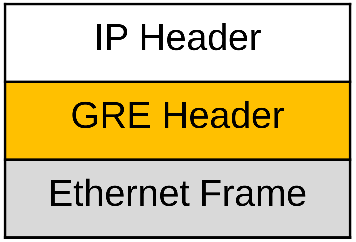
<table>
<tr><td>Reserved</td><td>Recur</td><td>Flags</td><td>Version</td><td>Protocol Type</td></tr>
<tr><td colspan="4">Checksum (optional)</td><td>Offset (optional)</td></tr>
<tr><td colspan="5">Key (optional)</td></tr>
<tr><td colspan="5">Sequence Number (optional)</td></tr>
<tr><td colspan="5">Routing (optional)</td></tr>
</table>

GRE Encapsulation

- A GRE encapsulation packet has the form:

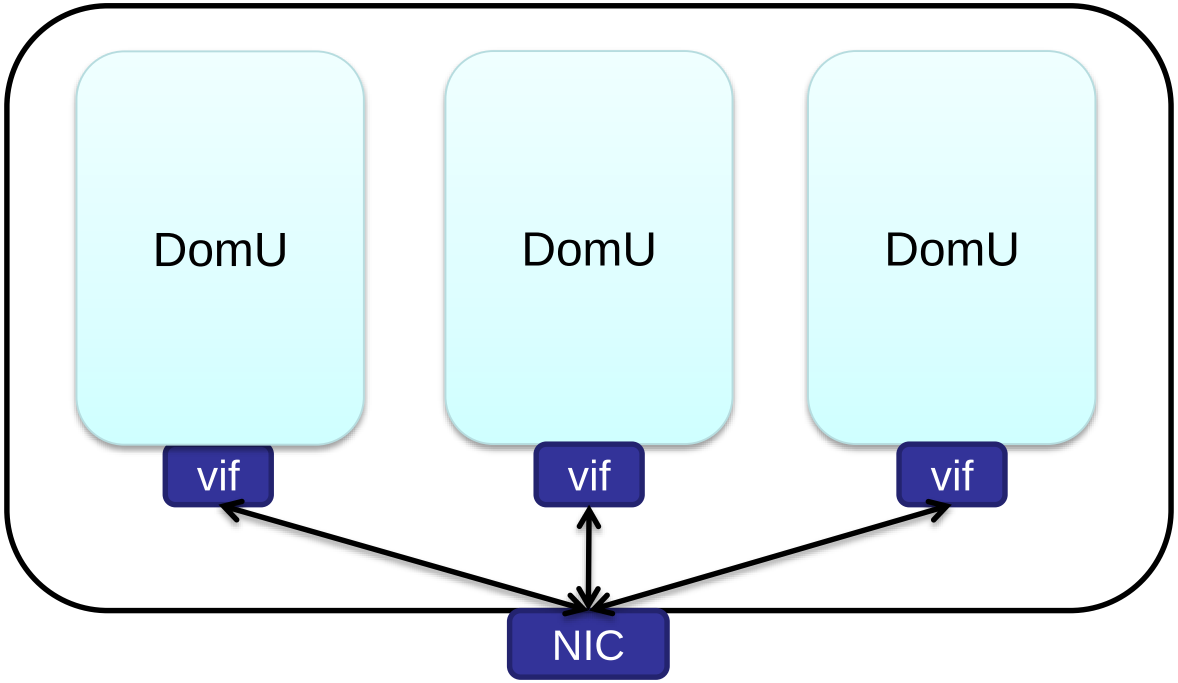


- Ethernet over GRE (EGRE):



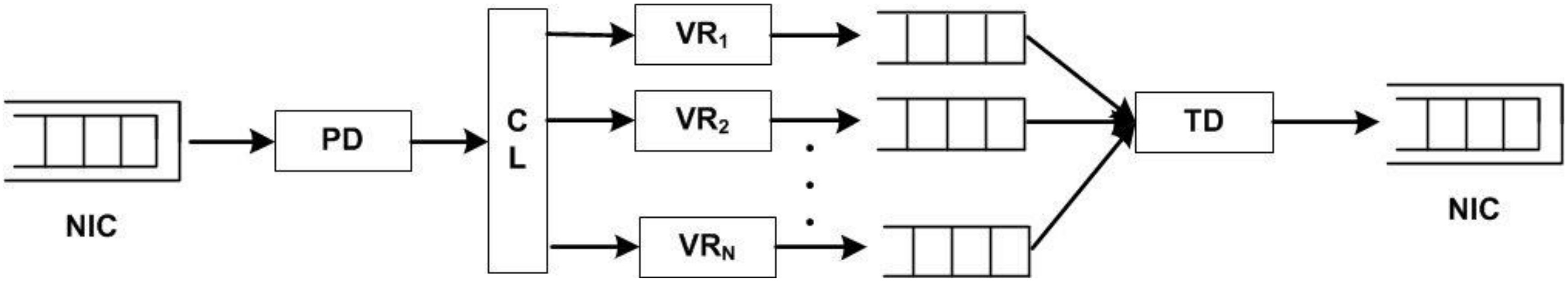
Network Interface Virtualization

Network Interface Sharing



- Multiple virtual nodes/routers might have to share a physical network interface
  - Packets should be classified among the virtual nodes/routers

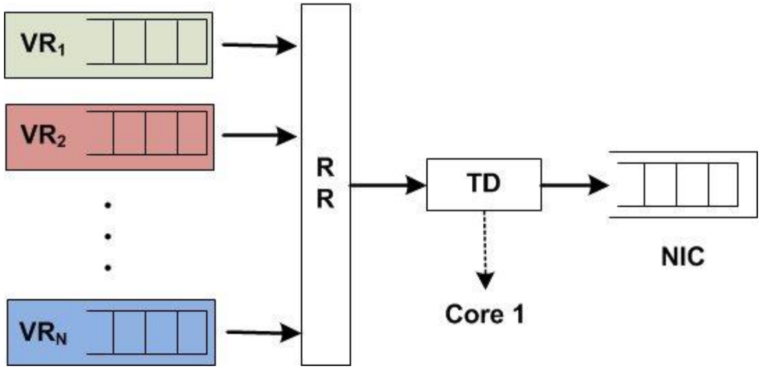
Packet Classification in Software



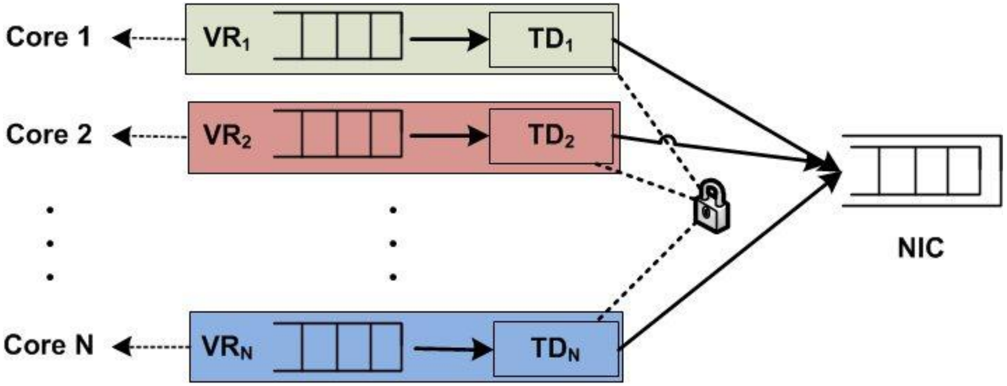
- ✓ Widely available (e.g., Click Modular Router, OpenvSwitch)
- ✓ Easy to enforce policies (e.g., weighted fair queuing)
- ✘ Resource utilization with multi-core CPUs

Packet Classification in Software with Multi-Core

- Use Case: Output processing

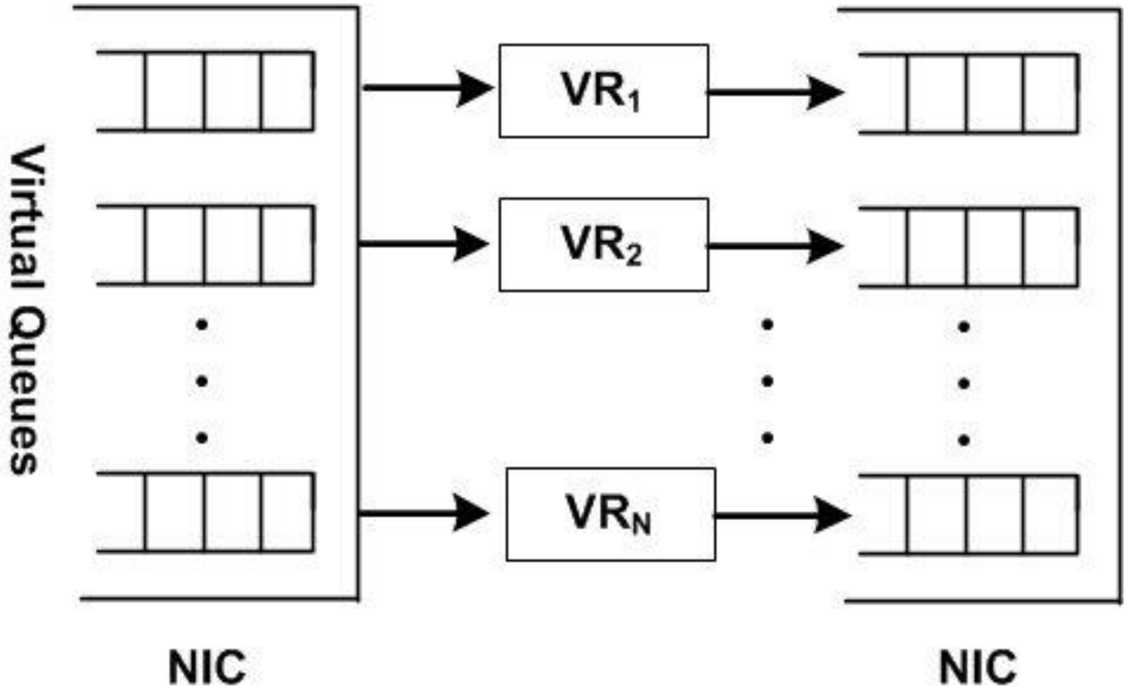


CPU-limited



concurrency-limited

Packet Classification in Hardware



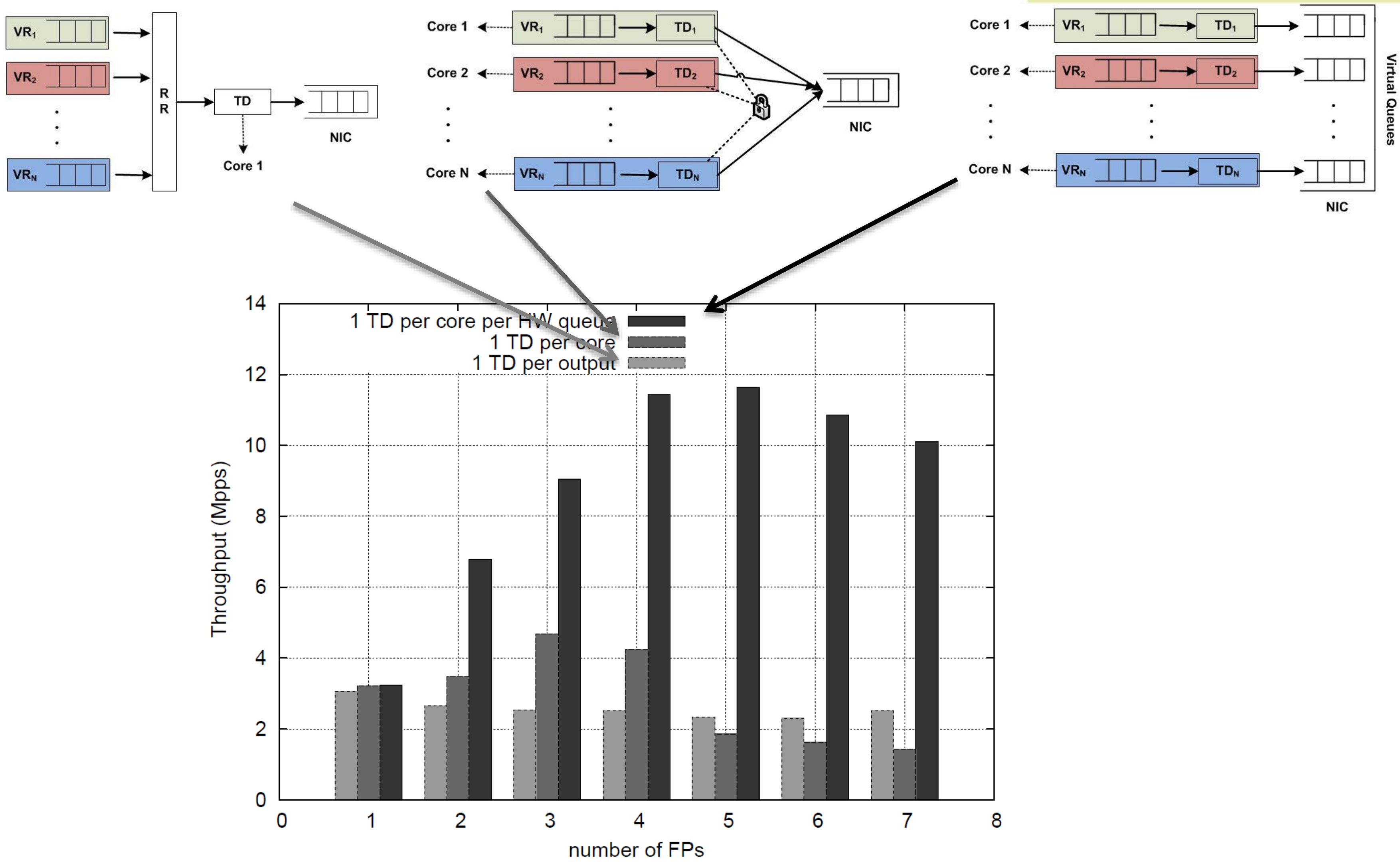
- ✓ Packet classification in hardware, off-loading the CPU
- ✘ Currently only round-robin scheduling is available

NIC Multi-Queuing Technologies

- Intel’s VMDq:
  - Classification in multiple queues based on the destination MAC address
  - Possible packet reordering

- Microsoft’s Receiver-side Scaling (RSS):
  - Classification in multiple queues using a hash function
  - Flow-based classification

Performance with Output Processing

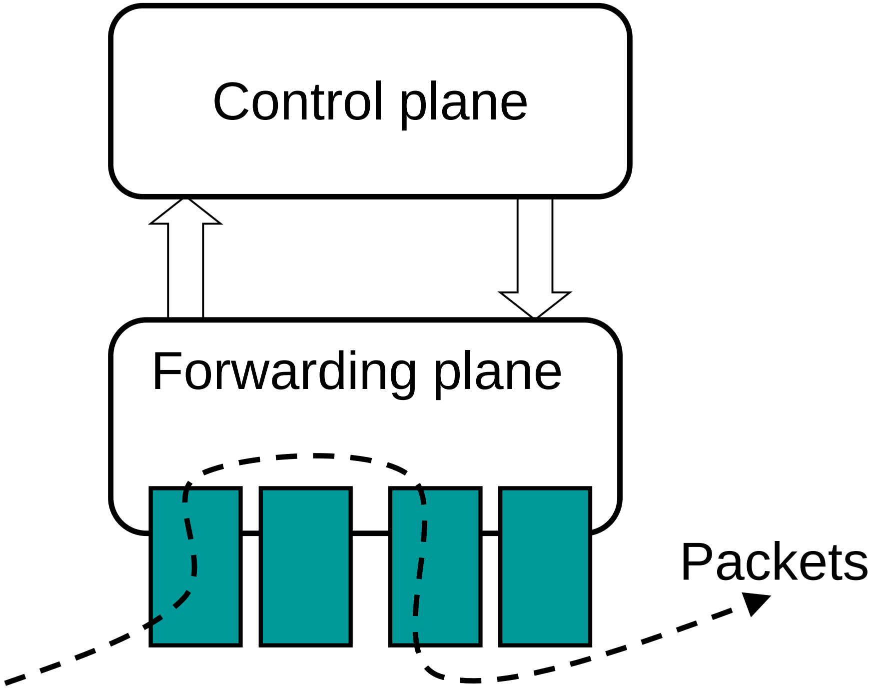


Router Virtualization

Software Routers on Commodity Hardware

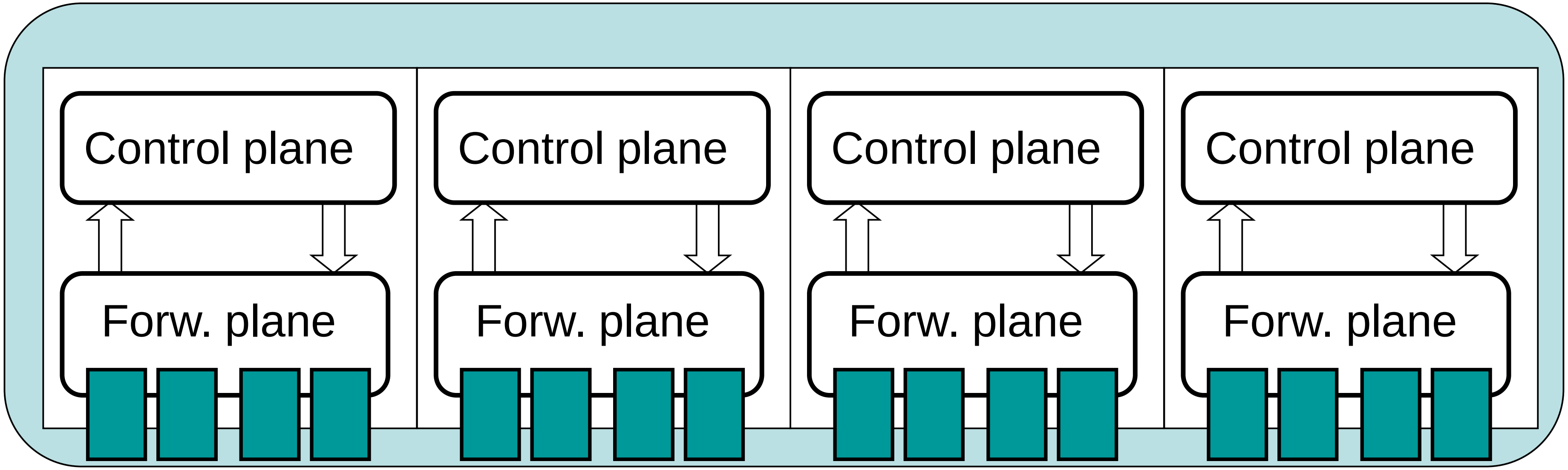
- Programmability:
  - Fully programmable forwarding plane (e.g., Click Modular Router)
  - Extensible control plane (e.g., XORP)

- High Performance:
  - Forwarding small packets at several Mpps
  - Limited by main memory accesses
  - Plenty of spare CPU cycles: much potential for virtualization



Virtual Routers on Commodity Hardware

- One box fulfills the role of multiple routers
  - Dedicated forwarding and control planes for each virtual router
  - Server virtualization technologies can be used to virtualise and isolate software routers



Router Virtualization Benefits (1)

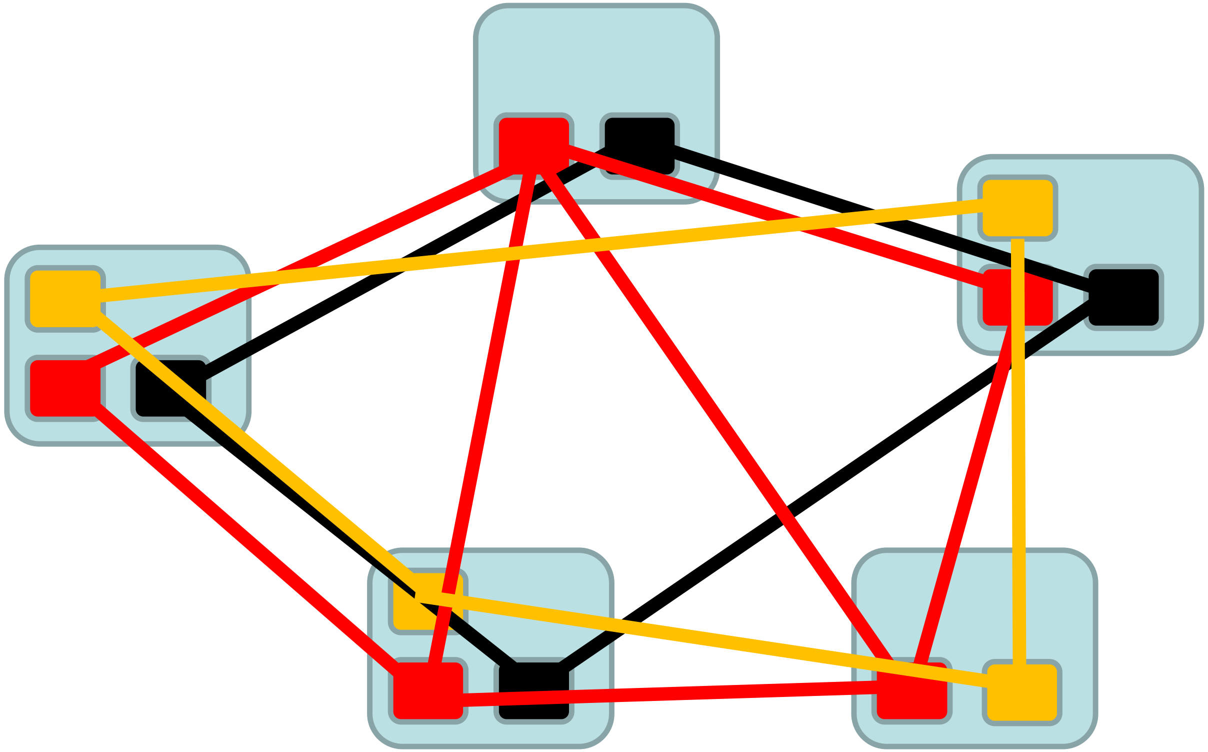
- One box fulfils the role of multiple routers
  - Independent and flexible management, leasing the Internet
  - Resource sharing (many available CPU cycles)
  - Lower hardware and support cost

    e.g., Small businesses within the same building sharing
    the same router, each managing its own VR

- Excellent platform for experimentation
  - Rolling out new and unstable solutions without risk

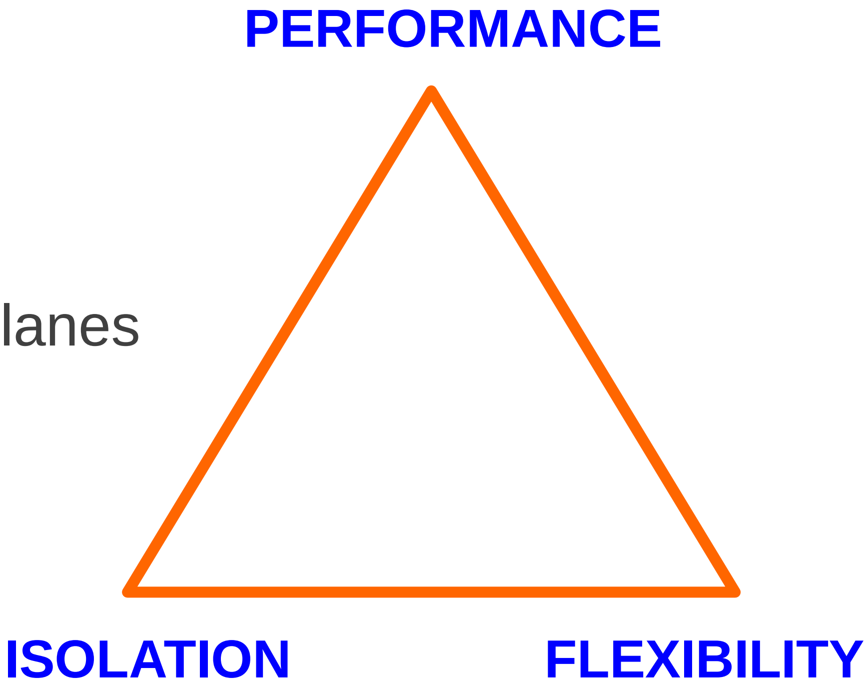
Router Virtualization Benefits (2)

- Innovation in network architecture design
  - Each virtual network has dedicated virtual routers with customized forwarding planes and routing protocols



Challenges in Router Virtualization

- Isolating the virtual routers from each other
- Minimizing the overhead of virtualization
- Highly configurable forwarding planes

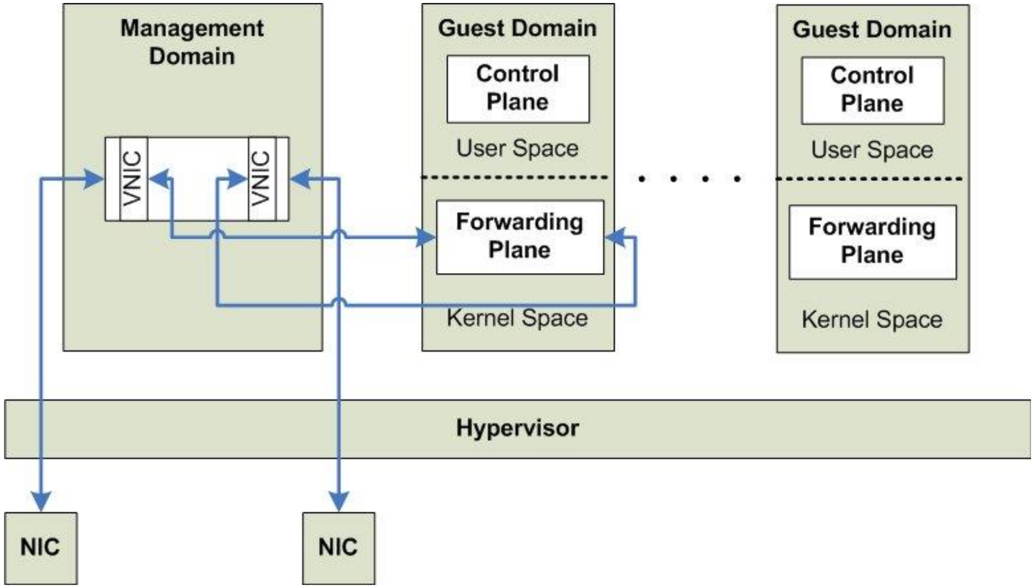


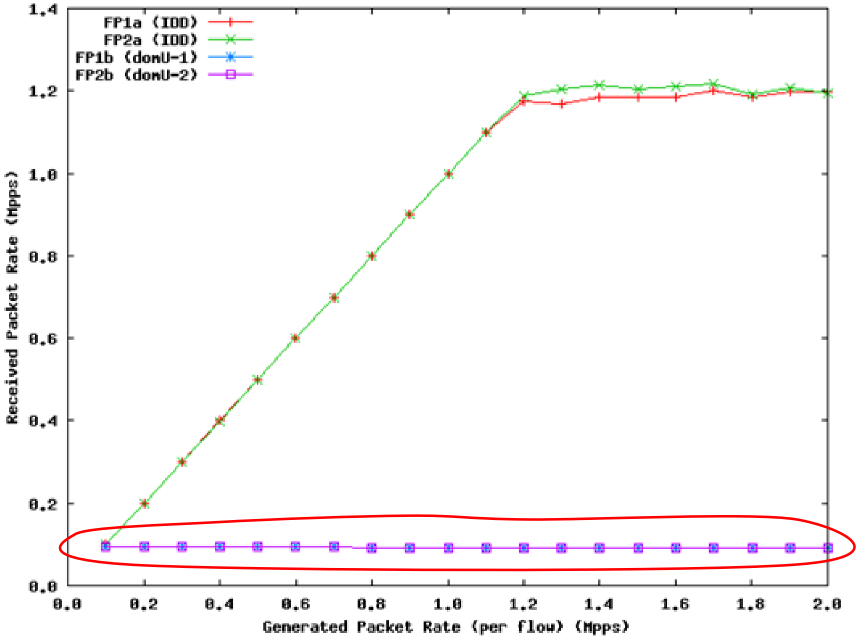
Case Study: Xen-based Virtual Routers

- Hosting forwarding planes
  - Determine the most suitable virtual forwarding configuration for high performance, flexibility and isolation

- Virtual forwarding scenarios:
  - Forwarding using I/O channels
  - Common forwarding plane
  - Forwarding with direct-mapped network interfaces

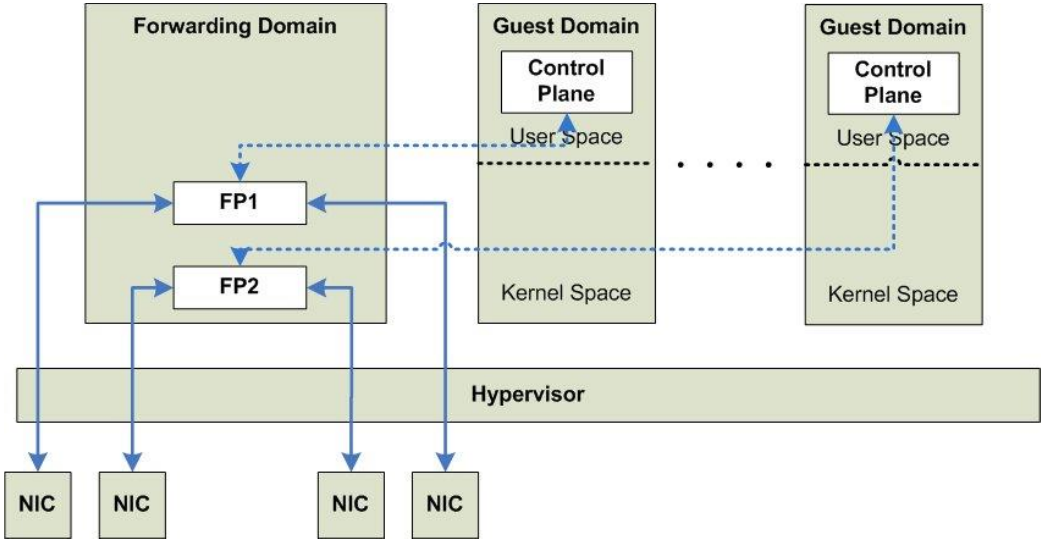
Forwarding using I/O Channels

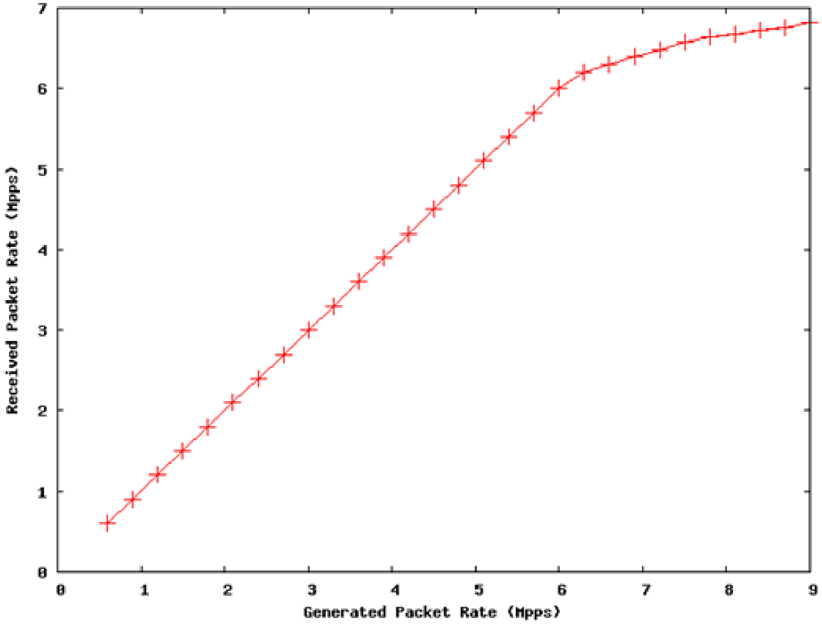




- ✓ Isolation
- ✗ Low performance (due to costly hypervisor domain switching)

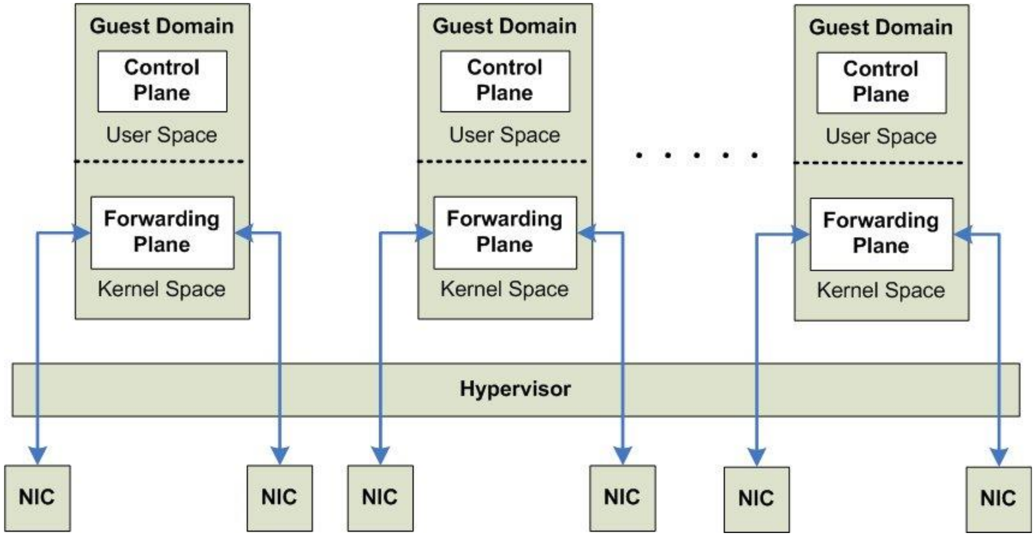
Common Forwarding Plane





- ✓ High performance
- ✘ Isolation (it should be enforced with proper core allocation and scheduling)

Forwarding with Direct-Mapped NICs



Guest Domain

Control Plane

User Space

Forwarding Plane

Kernel Space

Guest Domain

Control Plane

User Space

Forwarding Plane

Kernel Space

. . . . .

Guest Domain

Control Plane

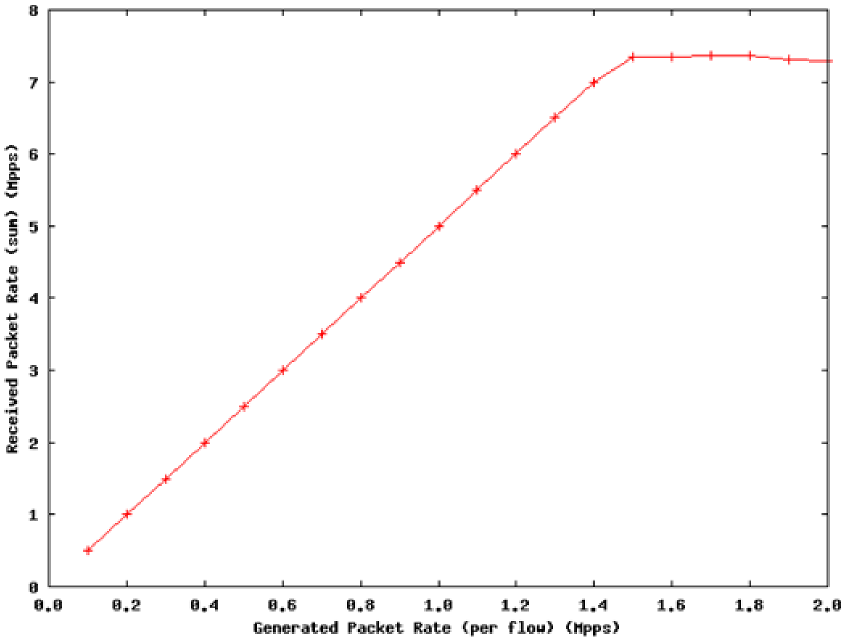
User Space

Forwarding Plane

Kernel Space

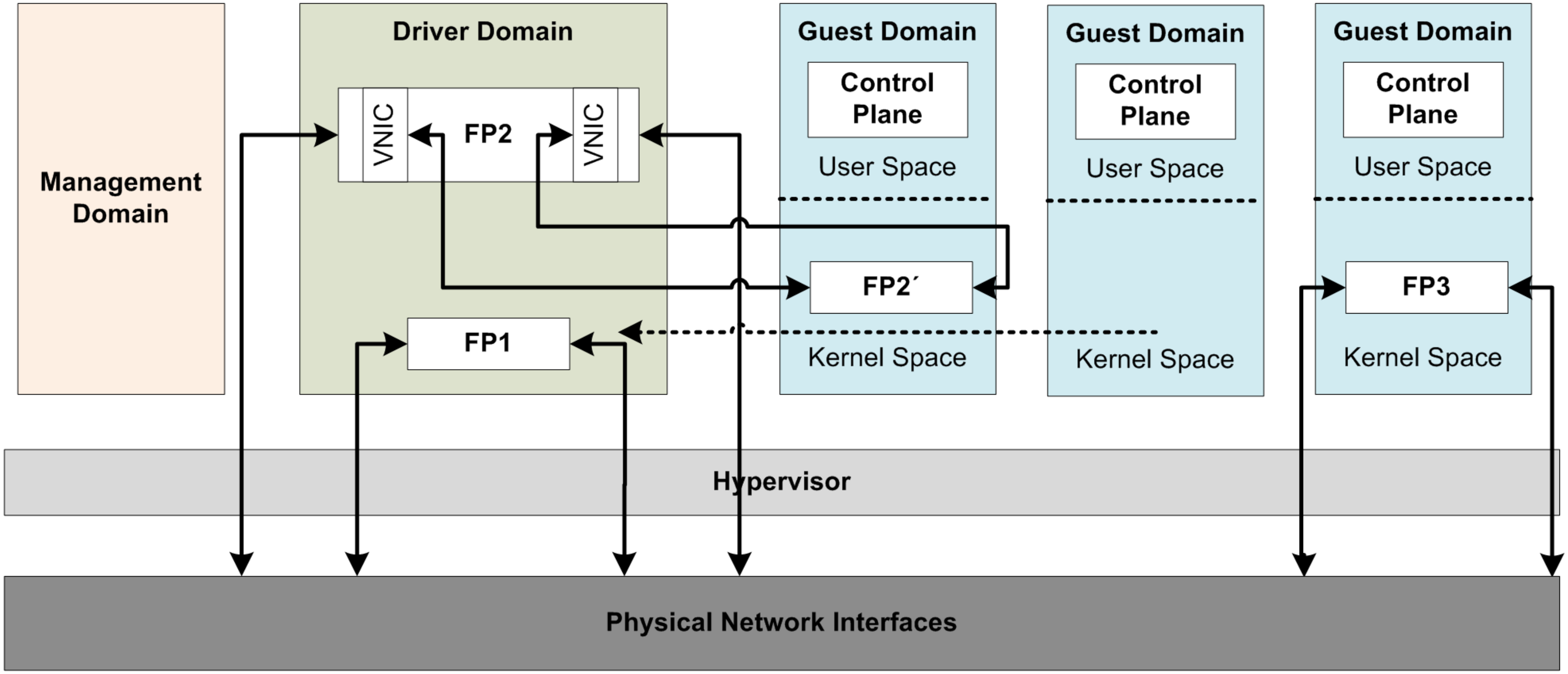
Hypervisor

NIC NIC NIC NIC NIC NIC



- ✓ High performance
- ✓ Isolation
- ✗ Requires NIC multi-queuing to scale

A Virtual Router Platform on Commodity HW



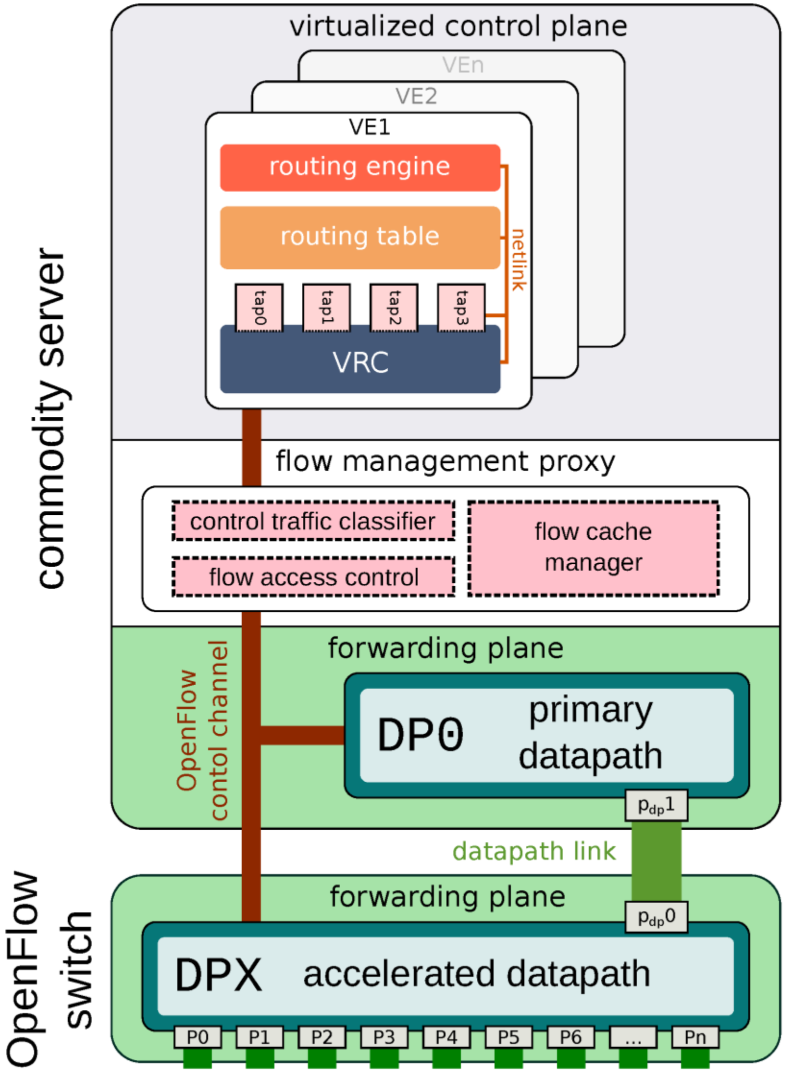
- Forwarding within a common domain (FP1)
- Forwarding with I/O Channels (FP2 ↔ FP2´)
- Forwarding in Guest Domains with direct NIC mapping (FP3)

Accelerated Software Virtual Routers

Accelerating Software Virtual Routers

- Features:
  - Control-plane extensibility
  - Forwarding-plane programmability
  - High performance
  - High port density

- Main components:
  - Forwarding plane composed of primary and accelerated datapath
  - Virtualized control plane
  - Flow management proxy:
    - Transparent layer between the forwarding and control plane



commodity server

virtualized control plane

VEn

VE2

VE1

routing engine

routing table

netlink

tap0

tap1

tap2

tap3

VRC

flow management proxy

control traffic classifier

flow access control

flow cache manager

forwarding plane

OpenFlow contol channel

DP0 primary datapath

$p_{dp}1$

datapath link

OpenFlow switch

forwarding plane

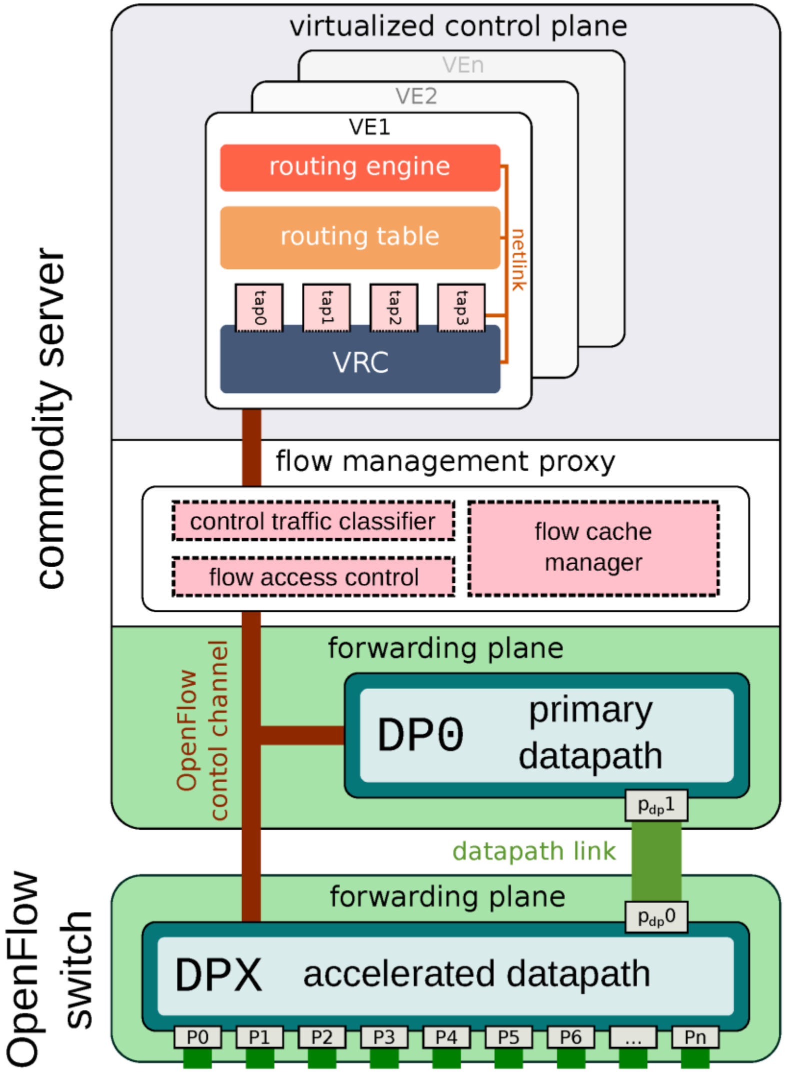
$p_{dp}0$

DPX accelerated datapath

P0 P1 P2 P3 P4 P5 P6 ... Pn

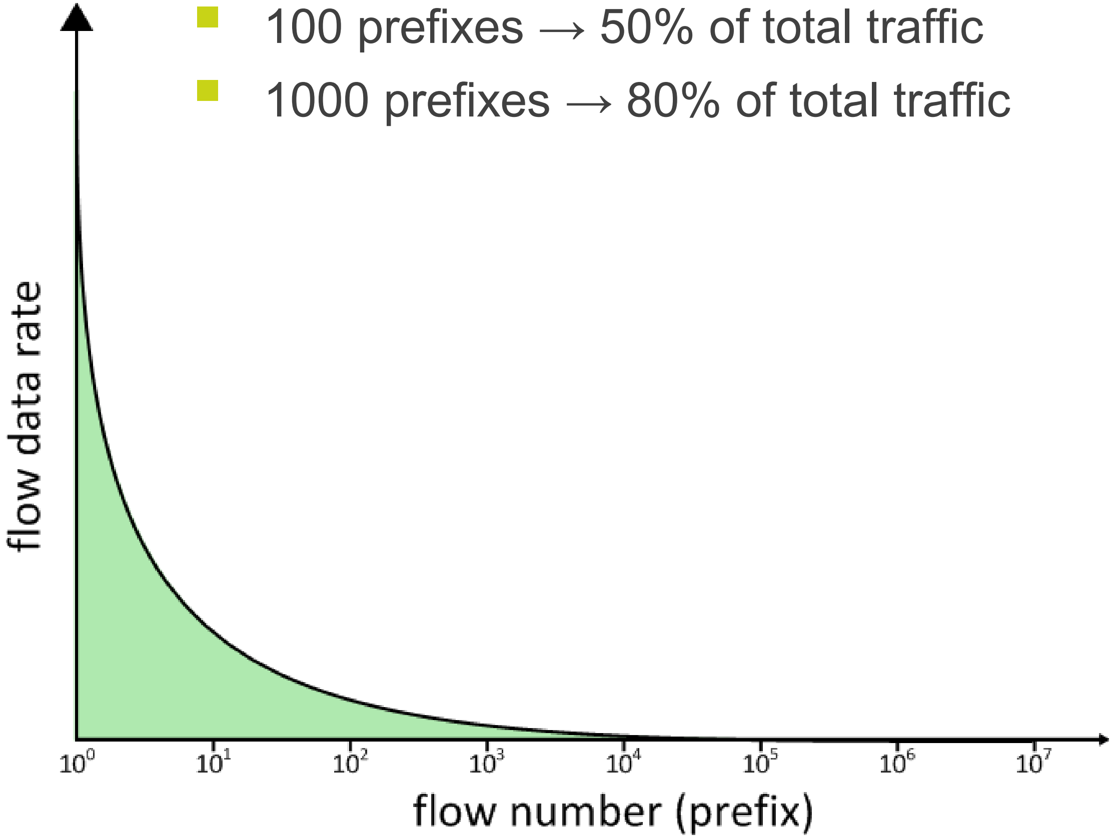
Flow Table Size Limitation

- Main limitation:
  - OpenFlow switch flow table has very small size (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 400K entries



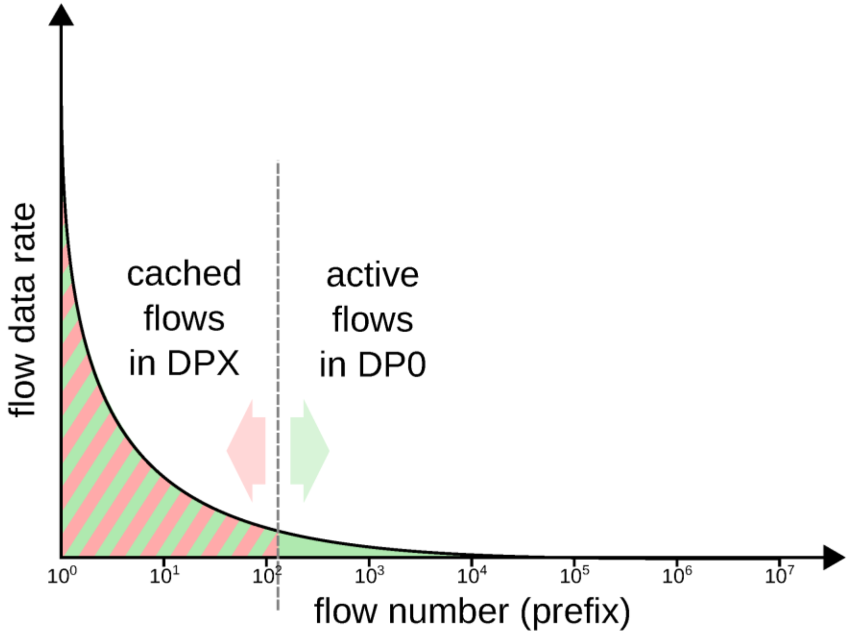
Flow Distribution in the Internet

- Flow distribution in the Internet:
  - A small subset of flows carries most of Internet traffic
    - Traffic statistics from an access router at a large European ISP
      - 100 prefixes → 50% of total traffic
      - 1000 prefixes → 80% of total traffic



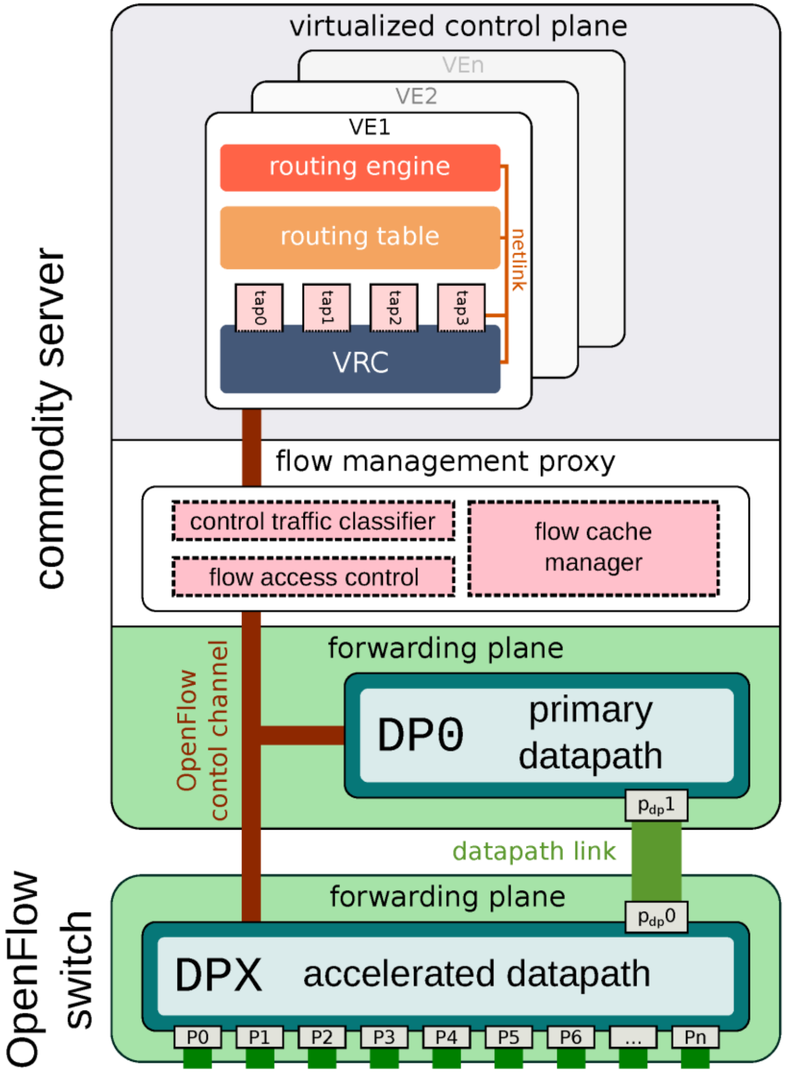
Towards an Accelerated Software Router

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



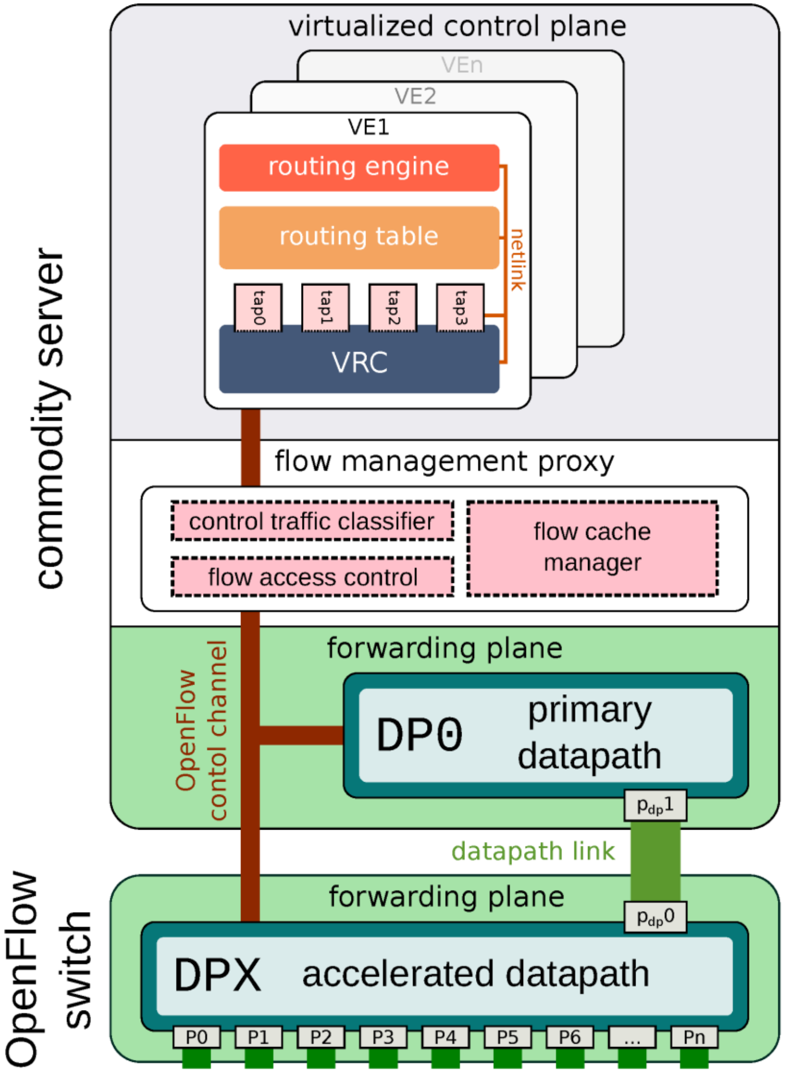
Virtualized Control Plane

- Virtual interfaces mapped to the physical ports:
    - Mapping registered with FMP
- Virtual Router Controller (VRC):
    - Forwards control messages to virtual interfaces
    - Generates flow insertion/ deletion commands
        - Listens on Netlink socket for routing table or ARP updates

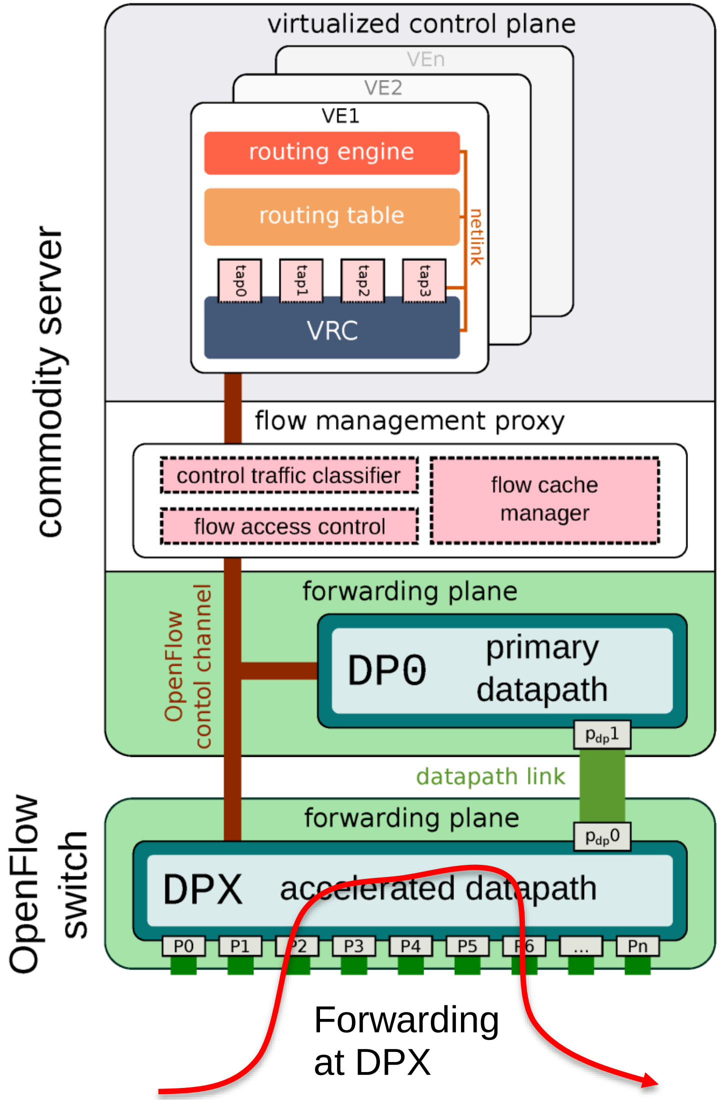


Flow Management Proxy

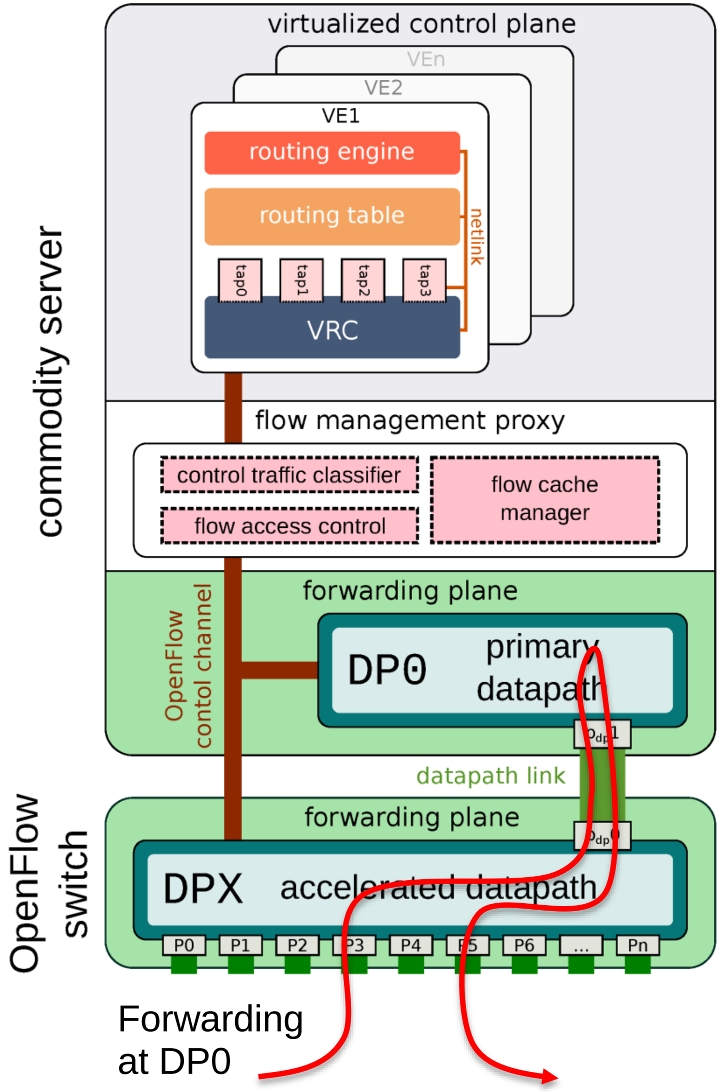
- Control message classification among virtualized control planes
- Access control on flow insertion/ deletion commands generated by a VRC
- Flow caching in the DPX
  - Least recently used (LRU)
  - Least frequently used (LFU)



Forwarding Plane



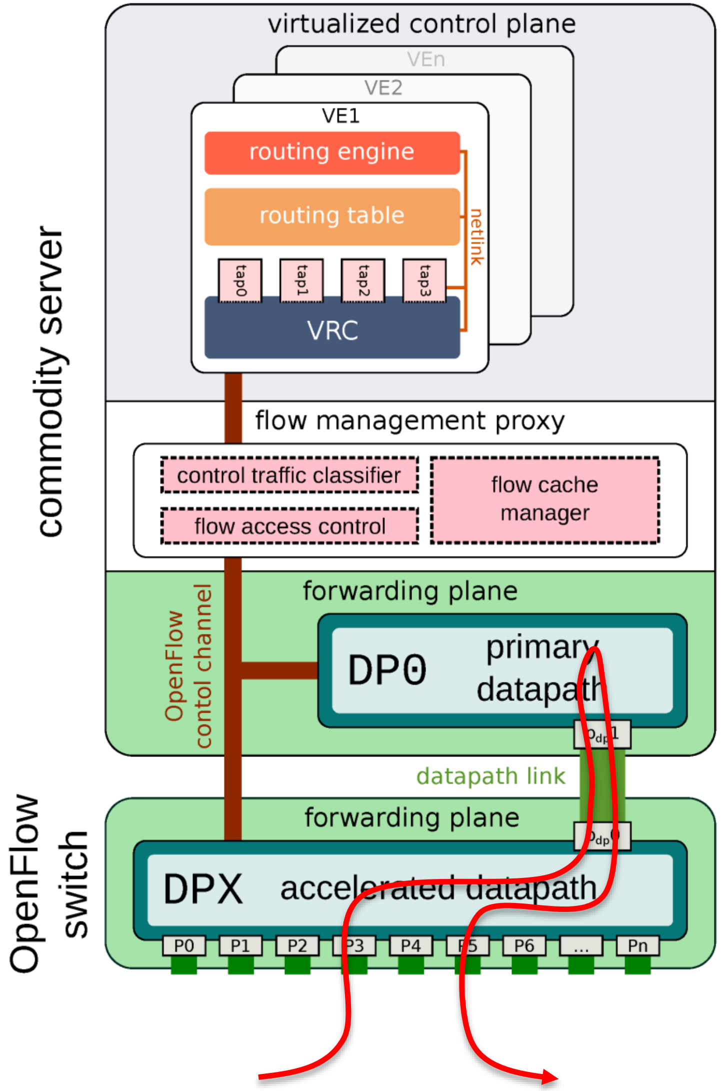
Forwarding at DPX



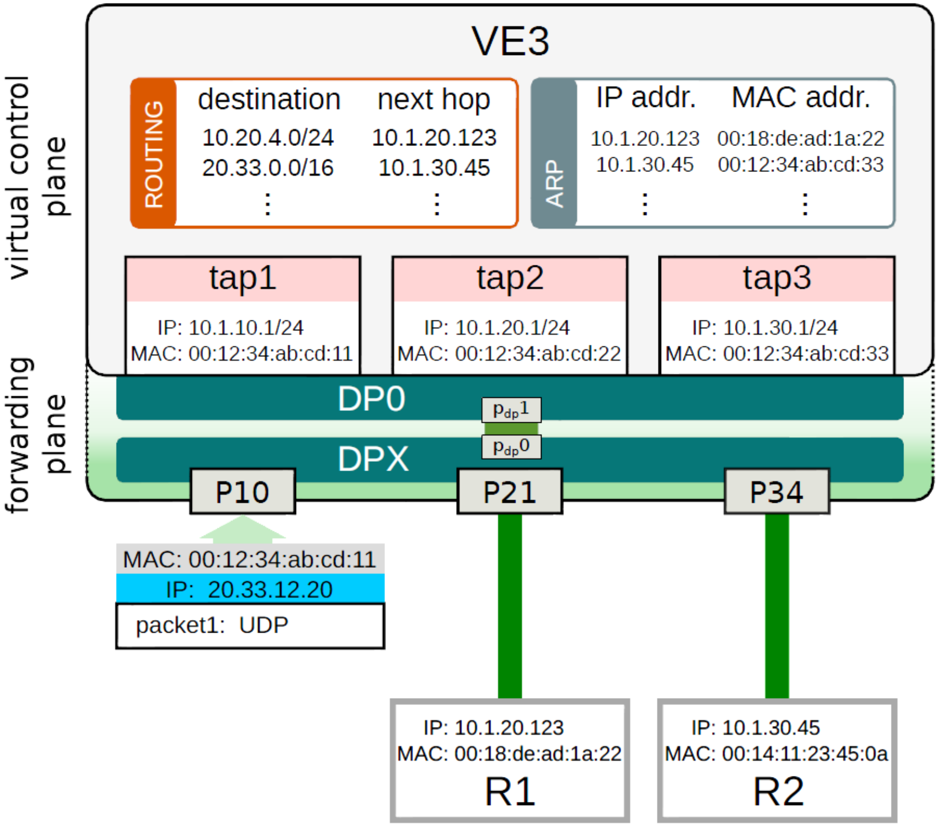
Forwarding at DP0

Flow Entry Types at DPX

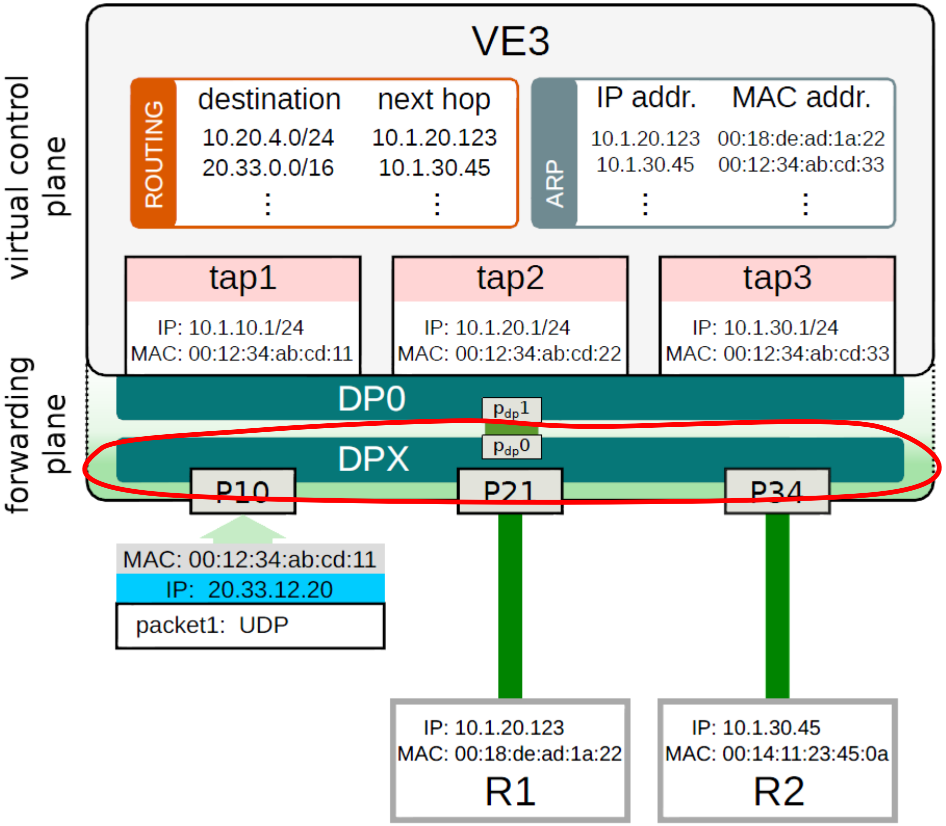
- Input redirection flow entries
- Output flow entries
- Cached routing flow entries
- Local control flow entries:
  - Routing updates
  - ARP resolution



Exemplary Packet Forwarding at DP0



Exemplary Packet Forwarding at DP0

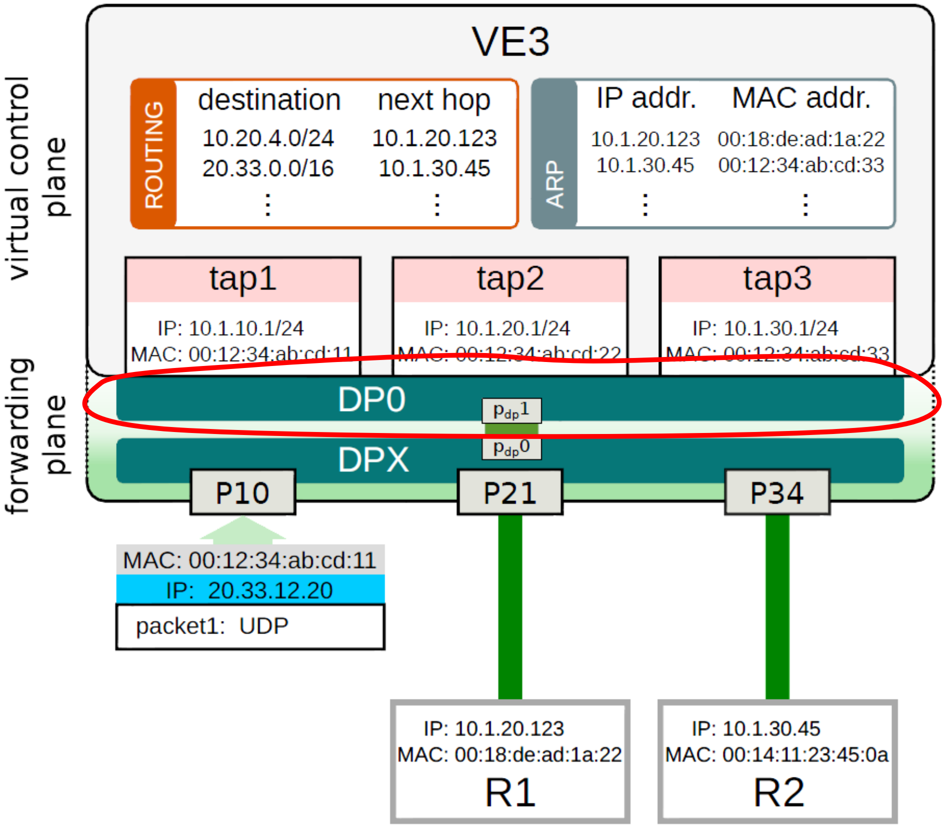


packet1

DST_MAC = 00:12:34:ab:cd:11	SET_VLAN_ID: 03 OUTPUT: P_{dp0}

Redirection flow entries (DPX)

Exemplary Packet Forwarding at DP0



packet1

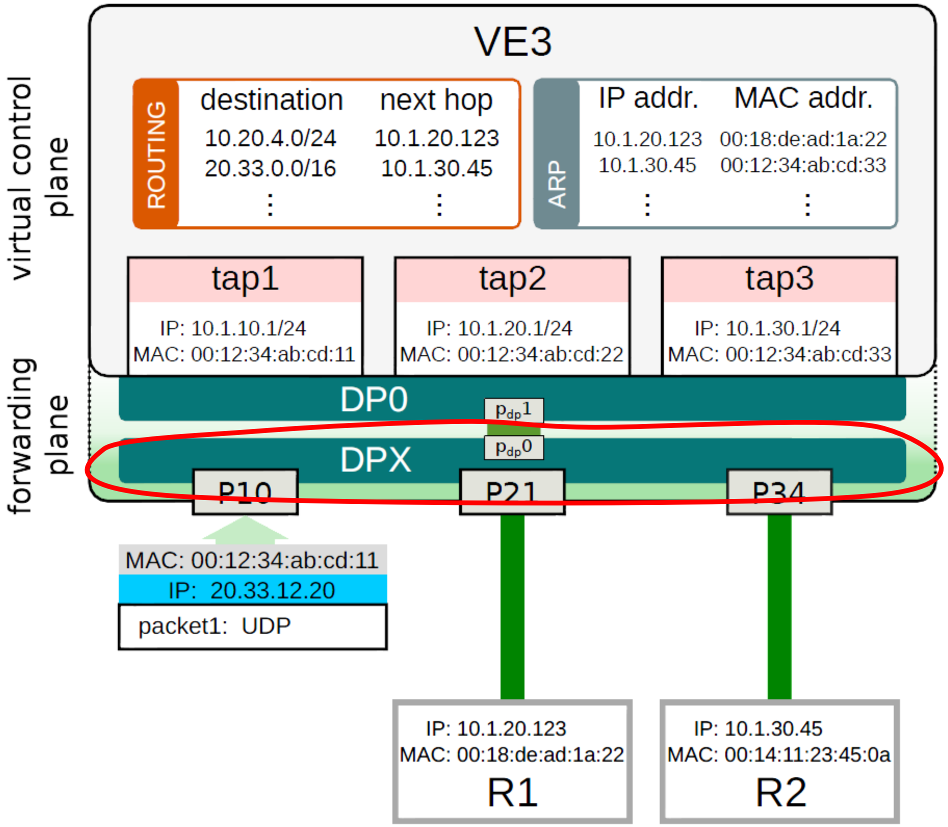
DST_MAC = 00:12:34:ab:cd:11	SET_VLAN_ID: 03 OUTPUT: P_{dp0}

Redirection flow entries (DPX)

IN_PORT = P_{dp1} VLAN_ID = 03 DST_IP = 20.33.0.0/16	SET_SRC_MAC: 00:12:34:ab:cd:33 SET_DST_MAC: 00:14:11:23:45:0a SET_VLAN_ID: 34 OUTPUT: P_{dp1}

Routing flow entries (DP0)

Exemplary Packet Forwarding at DP0



packet1

DST_MAC = 00:12:34:ab:cd:11	SET_VLAN_ID: 03 OUTPUT: P_{dp0}

Redirection flow entries (DPX)

IN_PORT = P_{dp1} VLAN_ID = 03 DST_IP = 20.33.0.0/16	SET_SRC_MAC: 00:12:34:ab:cd:33 SET_DST_MAC: 00:14:11:23:45:0a SET_VLAN_ID: 34 OUTPUT: P_{dp1}

Routing flow entries (DP0)

IN_PORT = P_{dp0} VLAN_ID = 21	STRIP_VLAN OUTPUT: P_{21}
IN_PORT = P_{dp0} VLAN_ID = 34	STRIP_VLAN OUTPUT: P_{34}

Output flow entries (DPX)

DPX Flowspace Requirements

- One input redirection entry per virtual port ($\sum_{i=1}^{N} M_i$ entries)
- One local control entry per virtual port ($\sum_{i=1}^{N} M_i$ entries)
- One output entry per physical port (P entries)

$$2\sum_{i=1}^{N} M_i + P$$

$\downarrow$

Hundreds of entries

N: number of virtual routers

M_i: number of virtual ports for i^{th} virtual router

P: number of physical ports

Performance Evaluation

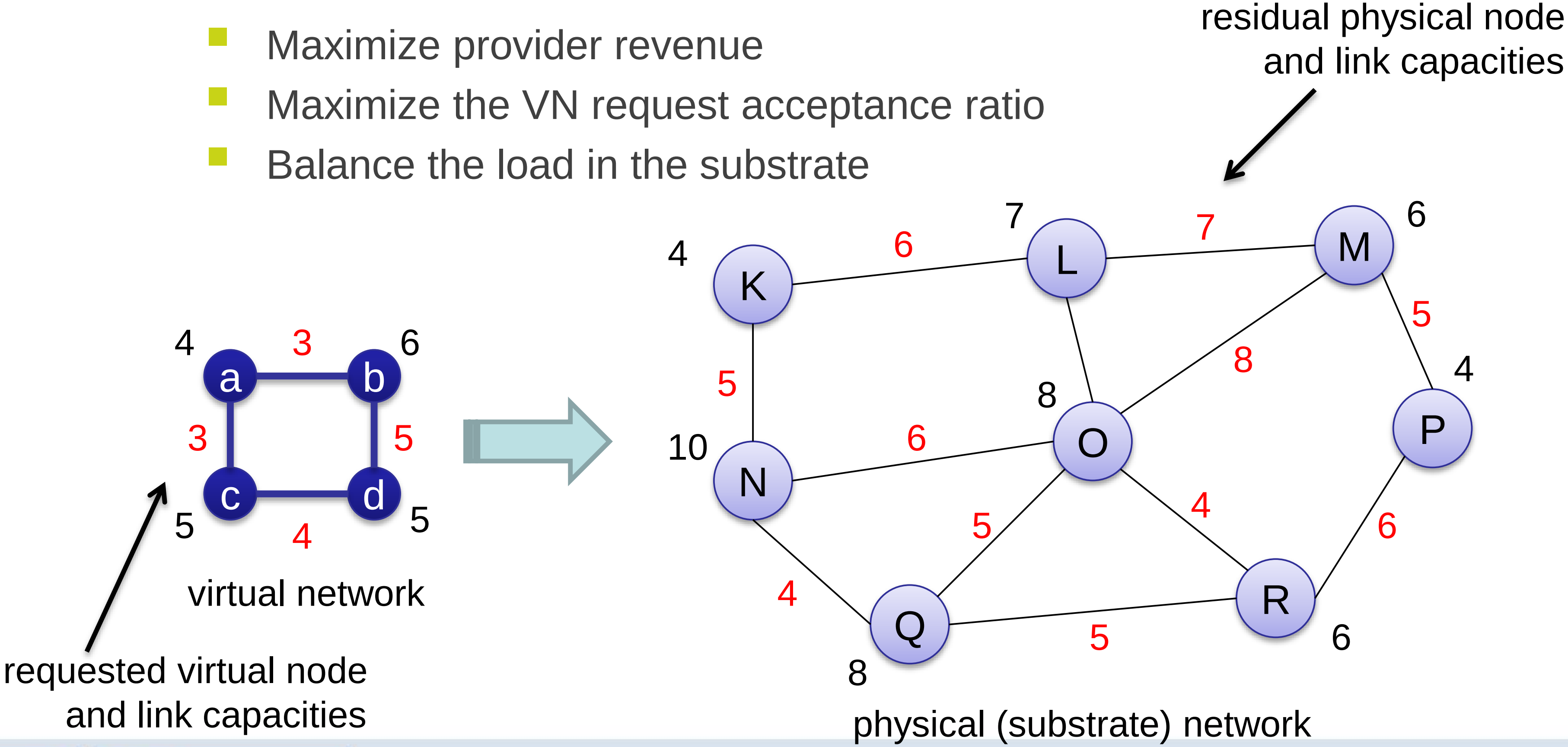
- DP0 (Pronto Switch)
  - Forwarding at line rate
  - Latency: 4 µs
  - Flow insertion rate: ~1000 flows/sec

- DPX (Server)
  - Forwarding rate: ~1Mpps per core
  - Latency: 15 µs
  - Flow insertion rate: ~6000 flows/sec

Virtual Network Embedding

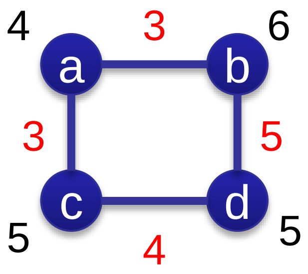
Virtual Network Embedding

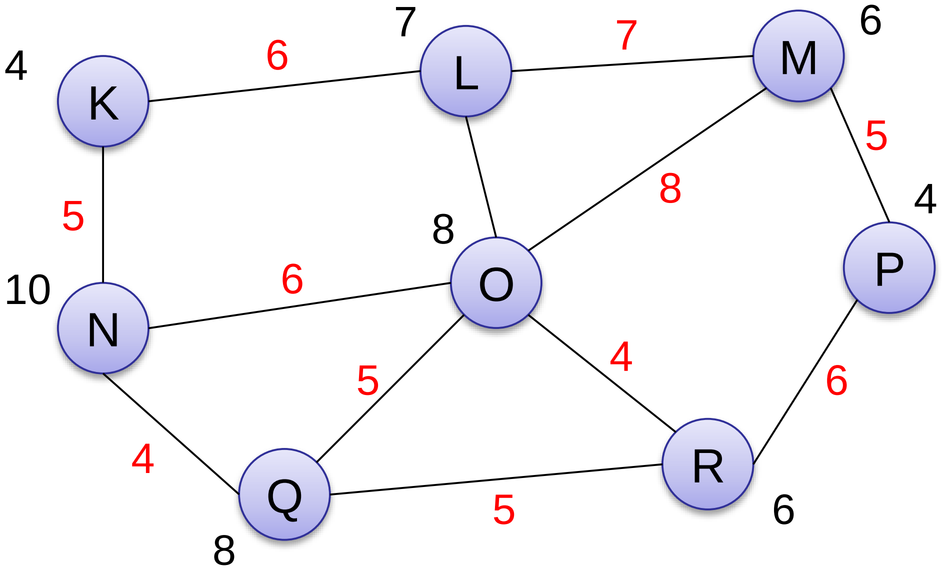
- Virtual network embedding:
  - Compute a mapping of a virtual network onto a physical network given an objective, e.g.,:
    - Maximize resource utilization
    - Maximize provider revenue
    - Maximize the VN request acceptance ratio
    - Balance the load in the substrate



Virtual Network Embedding Algorithm

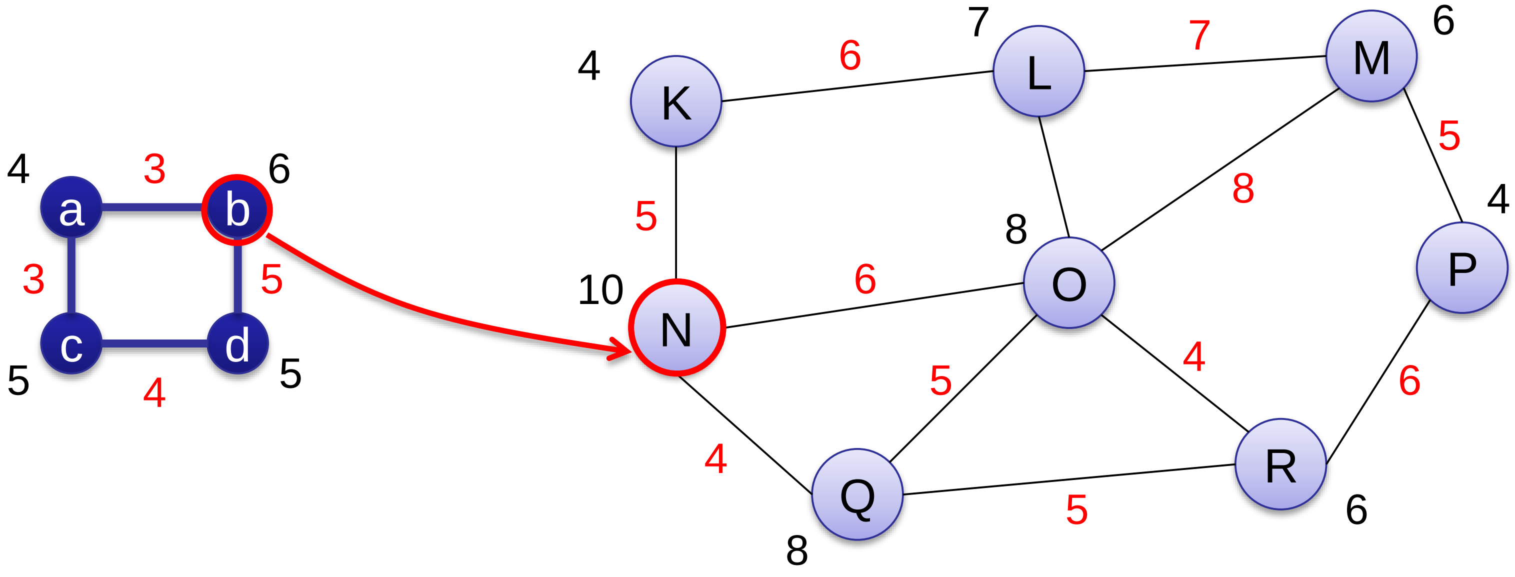
- VN embedding algorithm with simultaneous virtual node and link assignment
  - Objective: Load balancing across the substrate





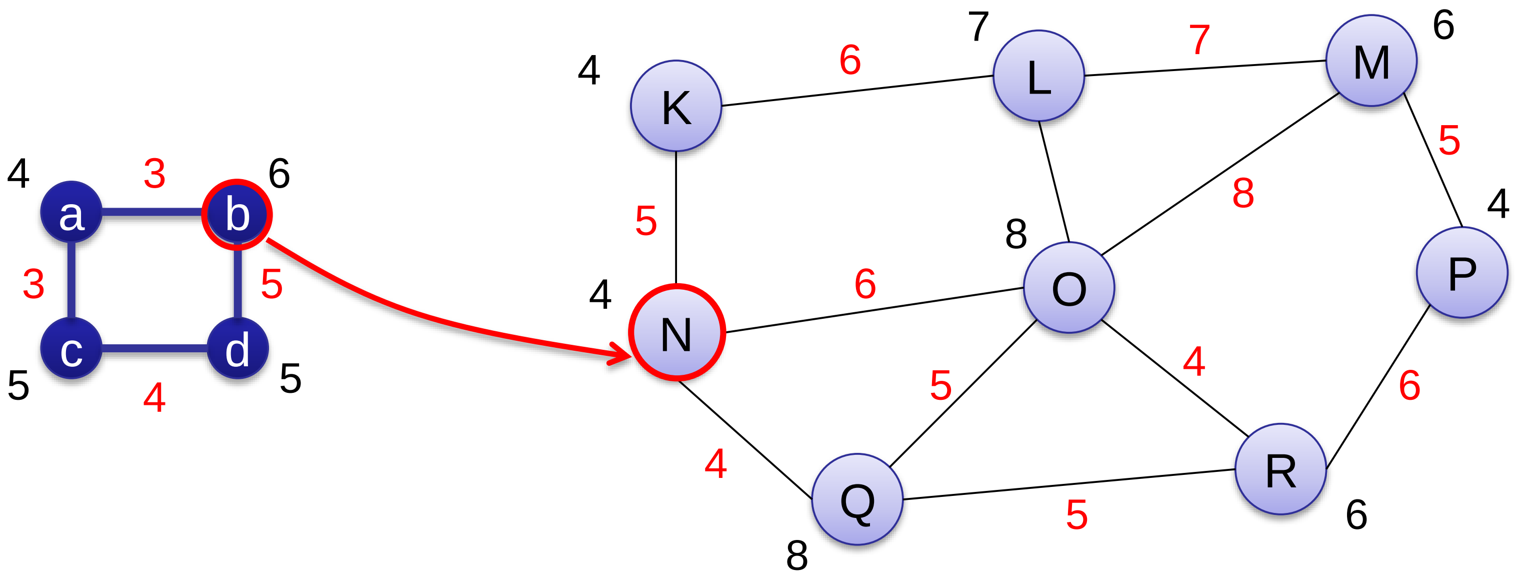
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
  - Objective: Load balancing across the substrate



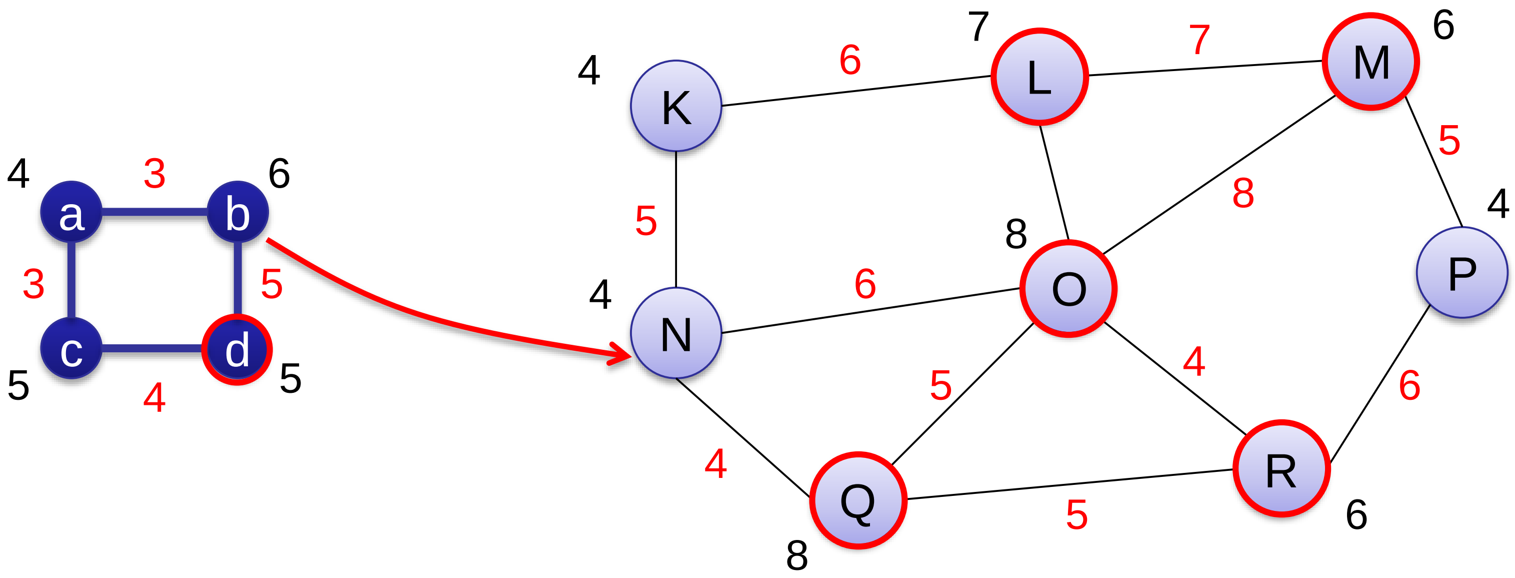
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
  - Objective: Load balancing across the substrate



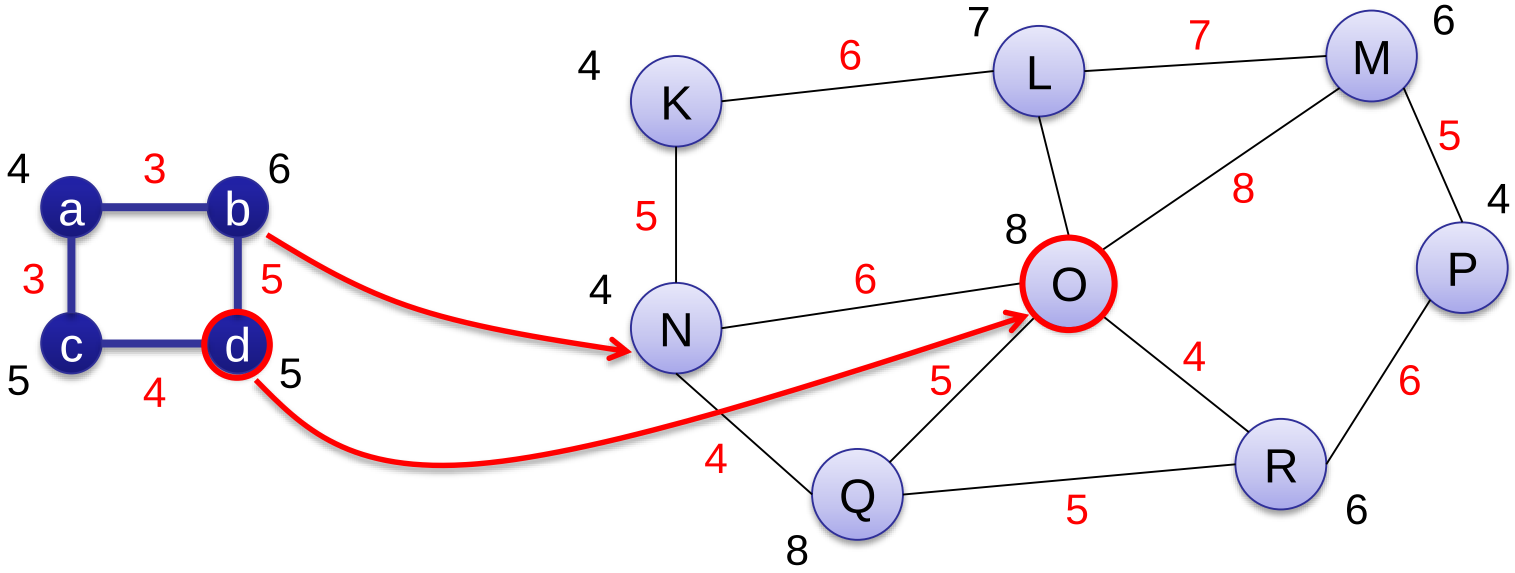
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
  - Objective: Load balancing across the substrate



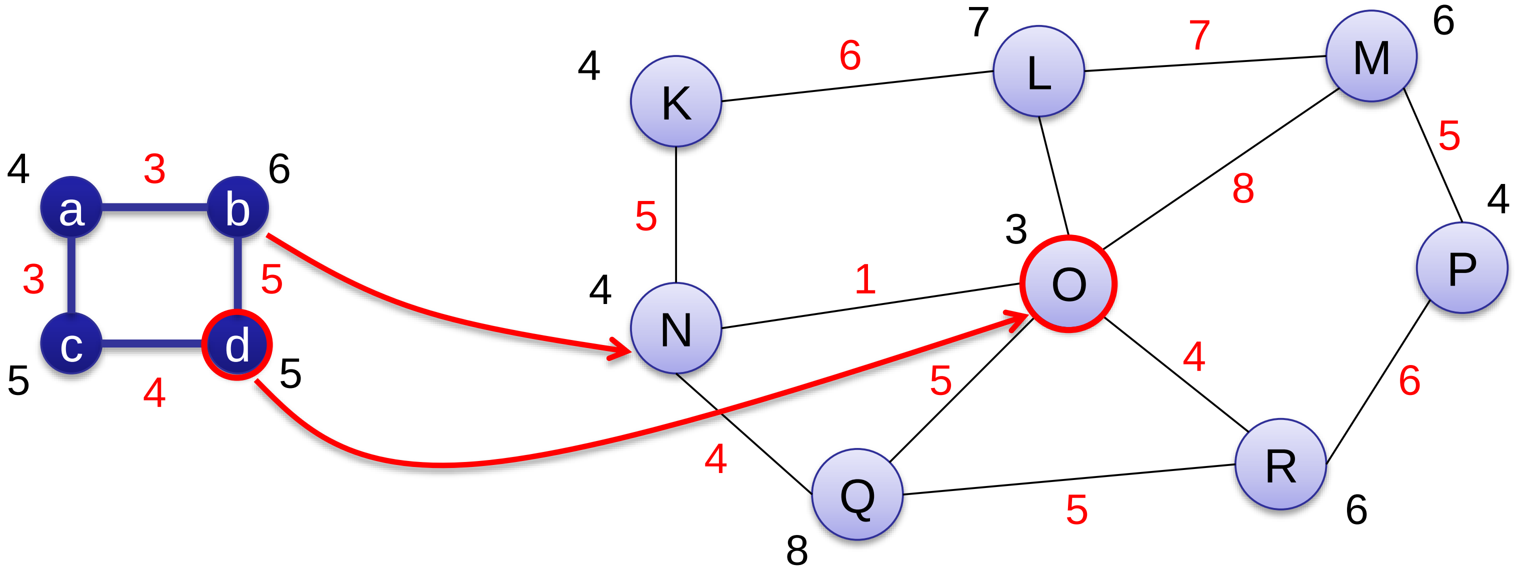
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
  - Objective: Load balancing across the substrate



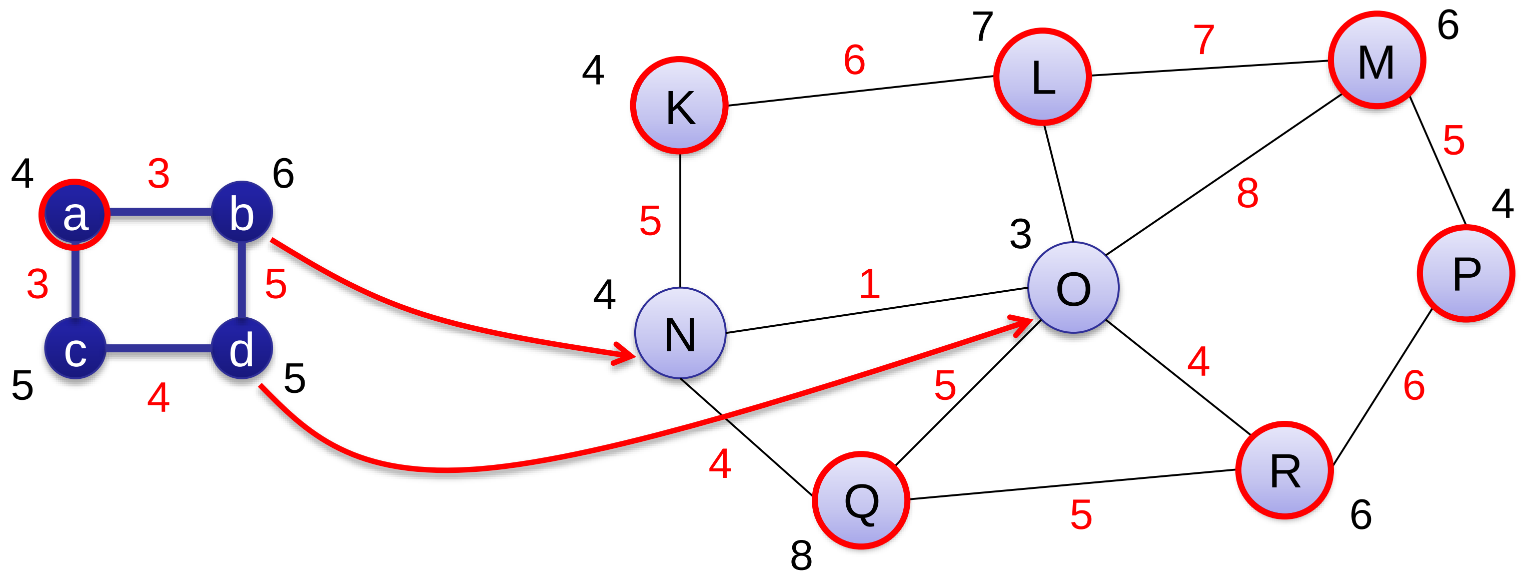
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
    - Objective: Load balancing across the substrate



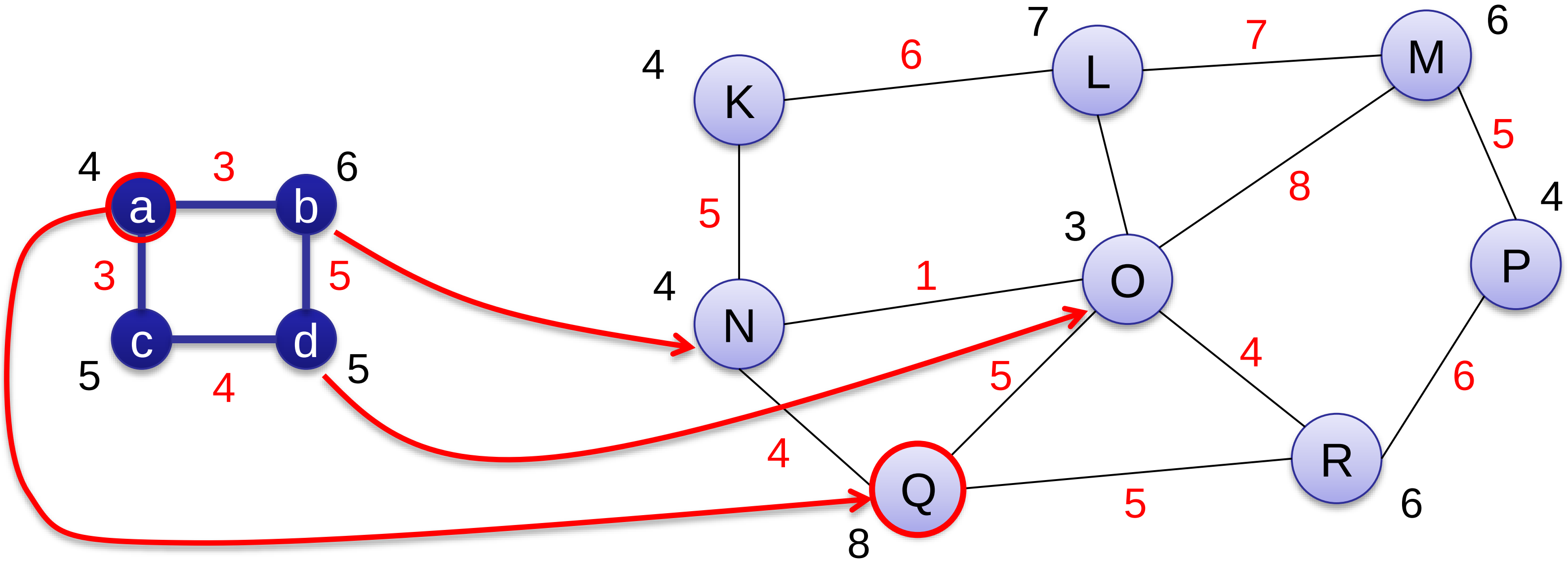
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
    - Objective: Load balancing across the substrate



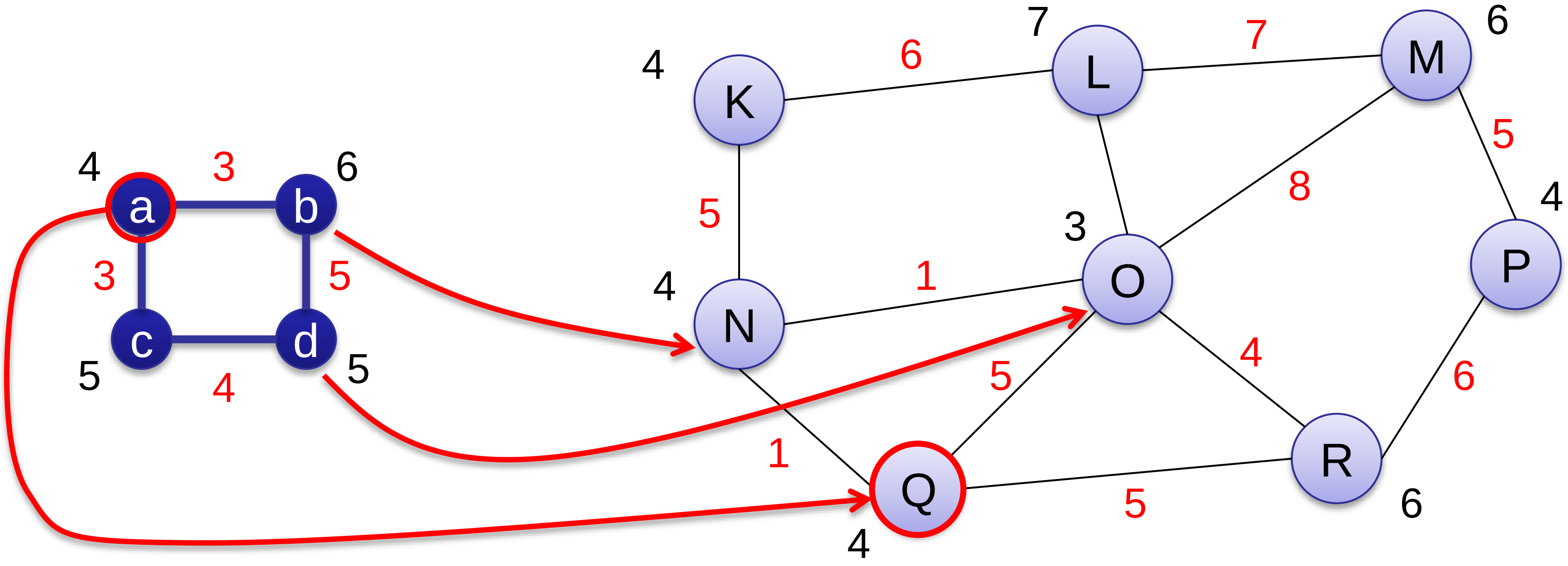
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
  - Objective: Load balancing across the substrate



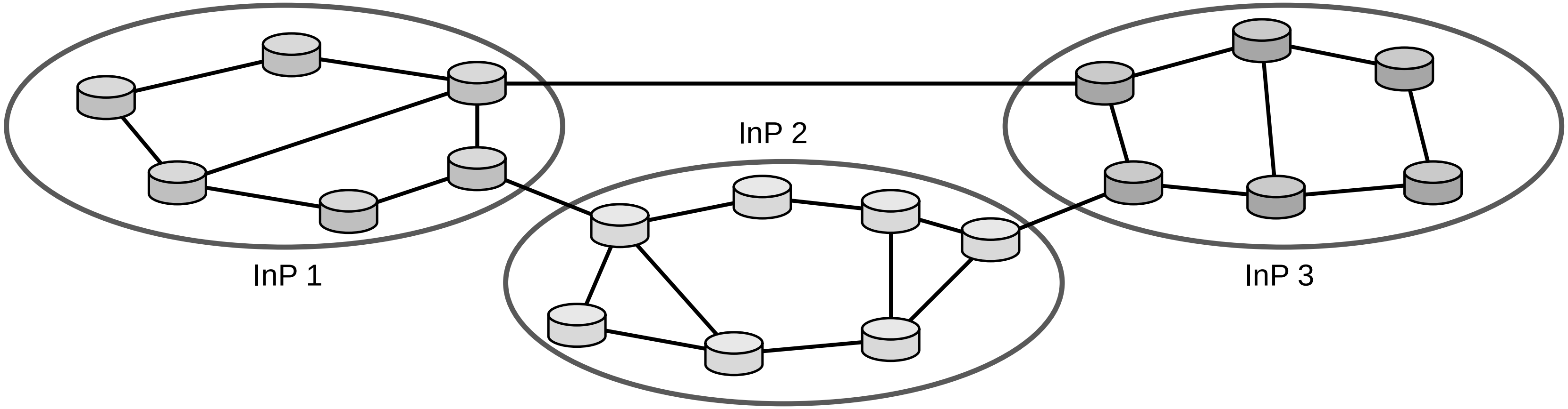
Virtual Network Embedding Algorithm

- VN embedding algorithm with simultaneous virtual node and link assignment
  - Objective: Load balancing across the substrate

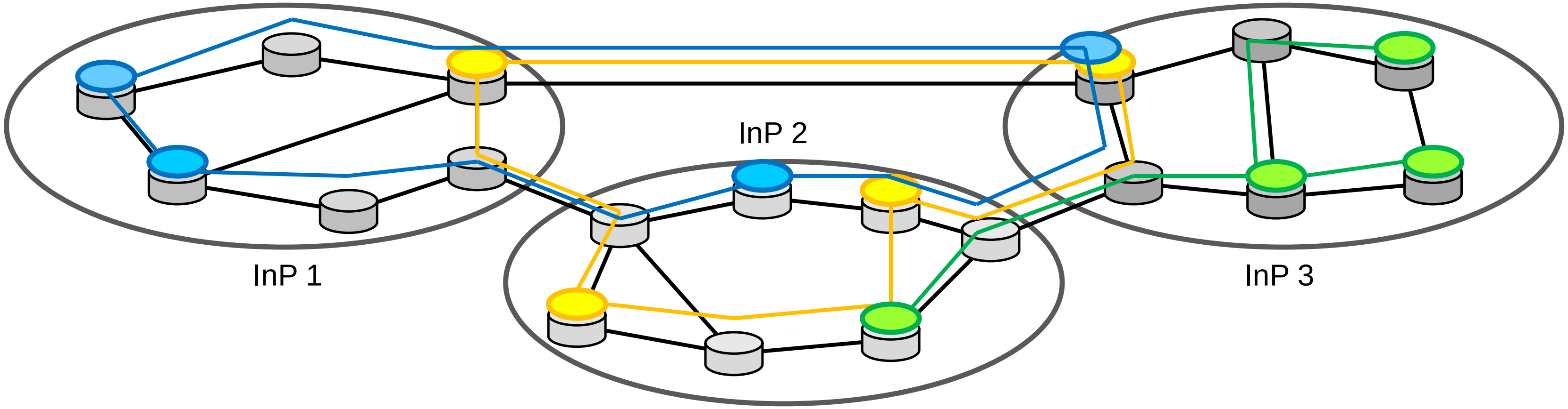


Virtual Network Embedding Across Multiple Substrate Networks

Wide-Area Virtual Network Deployment



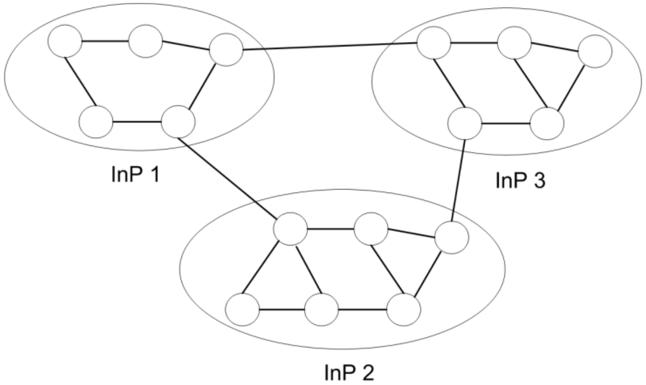
Wide-Area Virtual Network Deployment



Roles and Actors

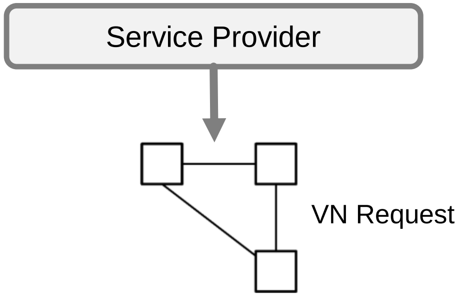
- Service Provider (SP)
  - Deploys services on VNs

- Infrastructure Provider (InP)
  - Owns and manages the physical infrastructure
  - Leases resources for VNs

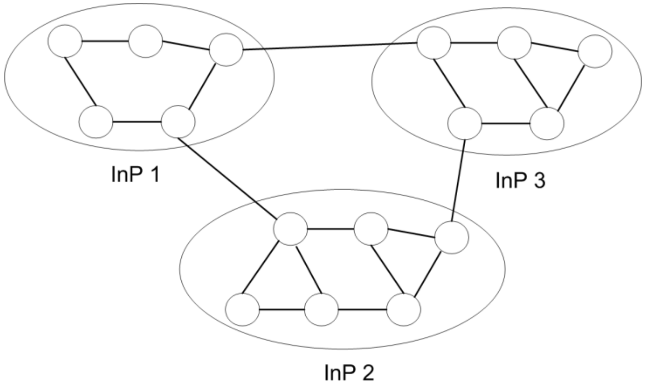


Roles and Actors

- Service Provider (SP)
  - Deploys services on VNs

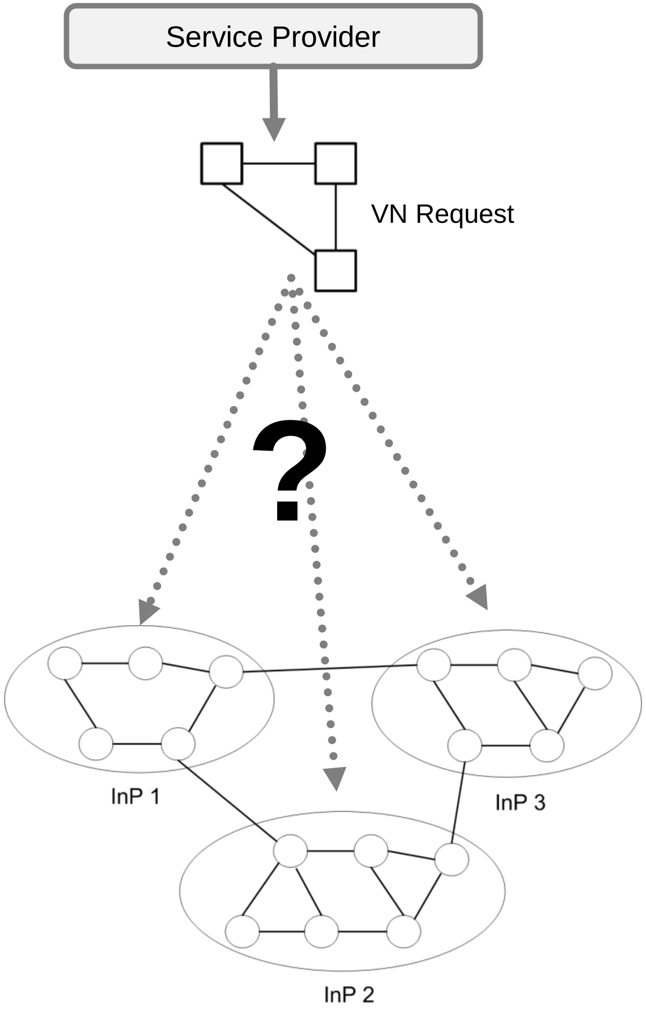


- Infrastructure Provider (InP)
  - Owns and manages the physical infrastructure
  - Leases resources for VNs



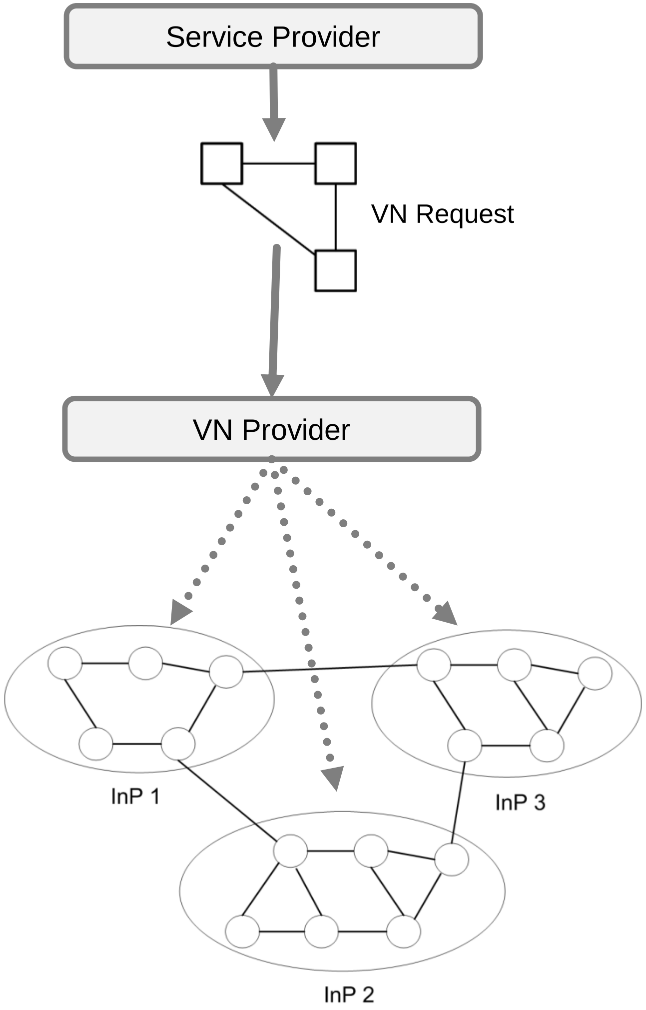
Roles and Actors

- Service Provider (SP)
  - Deploys services on VNs
- Infrastructure Provider (InP)
  - Owns and manages the physical infrastructure
  - Leases resources for VNs



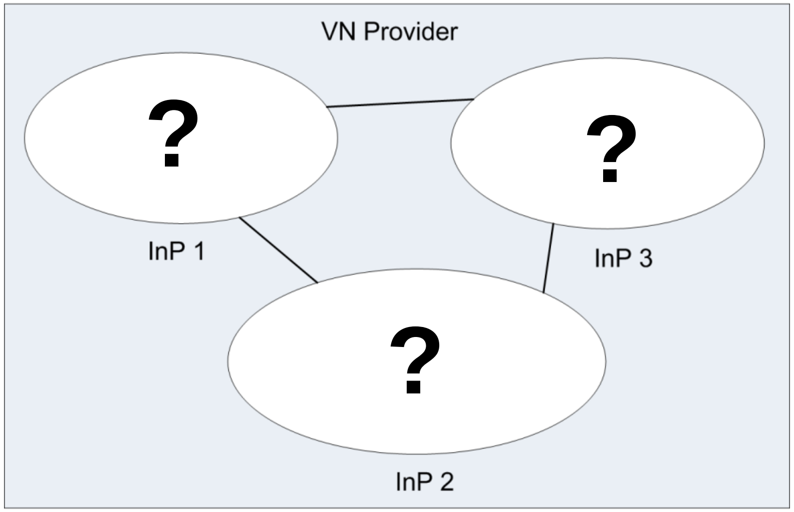
Roles and Actors

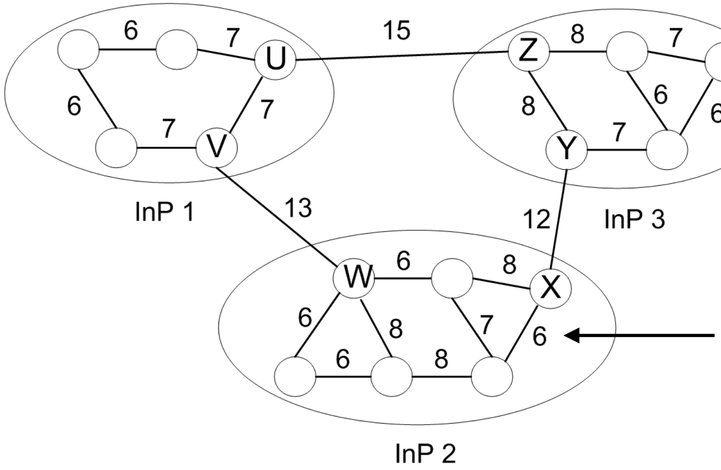
- Service Provider (SP)
  - Deploys services on VNs
- Virtual Network Provider (VNP)
  - Assembles resources from one or multiple InPs into a VN
- Infrastructure Provider (InP)
  - Owns and manages the physical infrastructure
  - Leases resources for VNs



Substrate Network Visibility

- VN Provider‘s visibility on substrate network topology and resources is limited to:

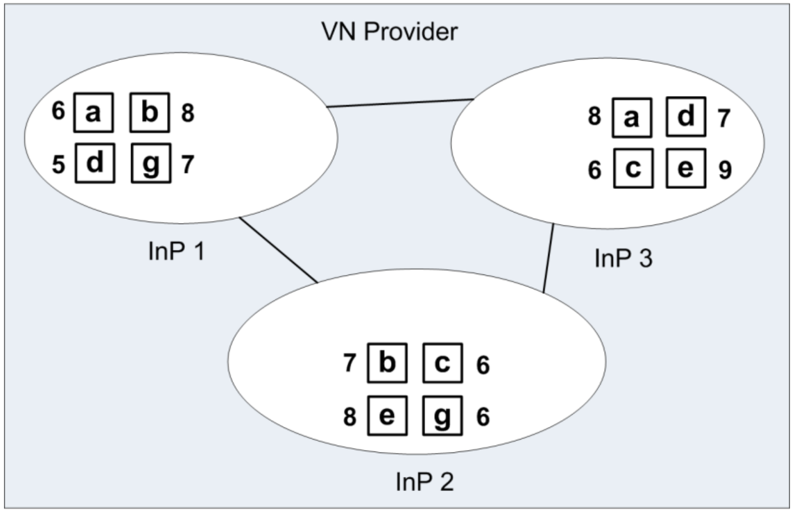


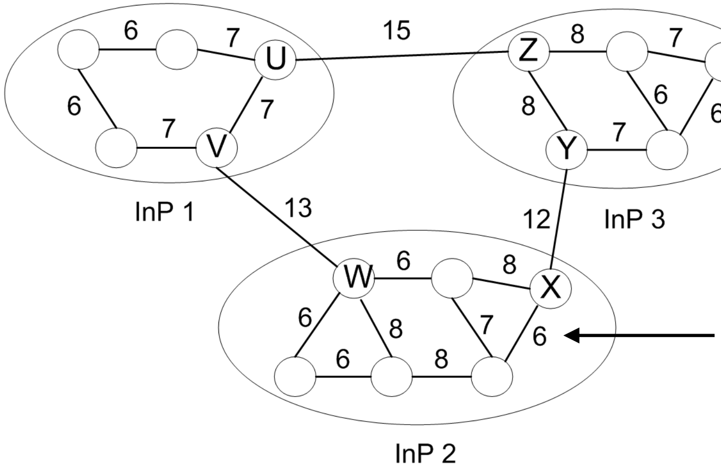


bandwidth costs

Substrate Network Visibility

- VN Provider‘s visibility on substrate network topology and resources is limited to:
  - Offered virtual node types (similar to Amazon EC2)

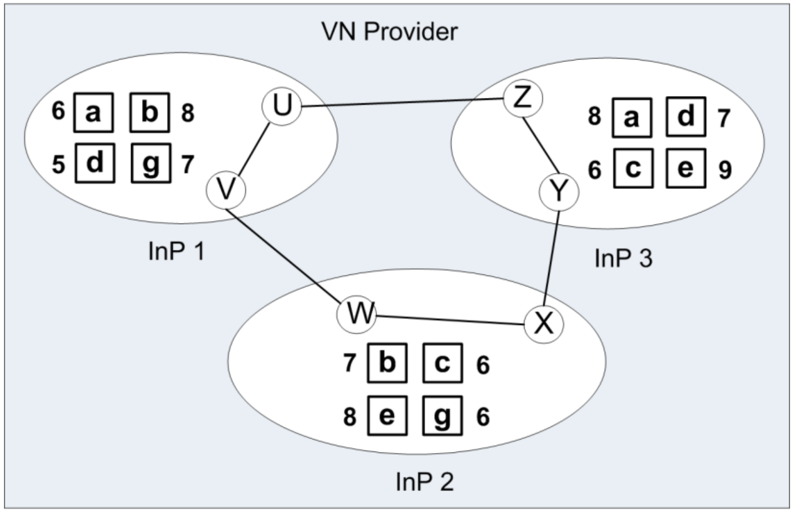


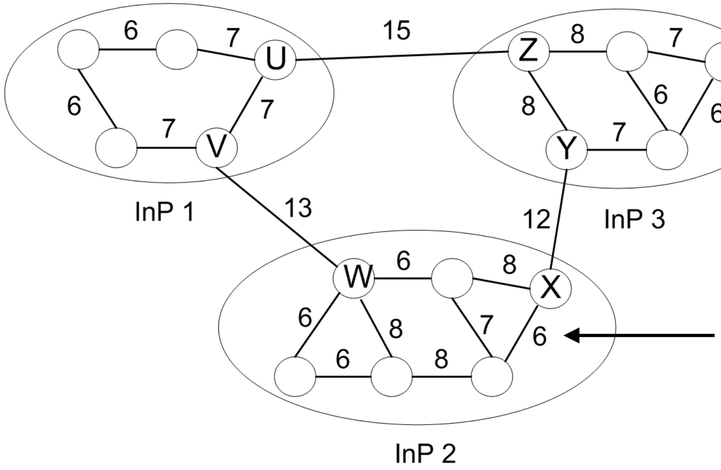


bandwidth costs

Substrate Network Visibility

- VN Provider‘s visibility on substrate network topology and resources is limited to:
  - Offered virtual node types (similar to Amazon EC2)
  - Location of peering nodes

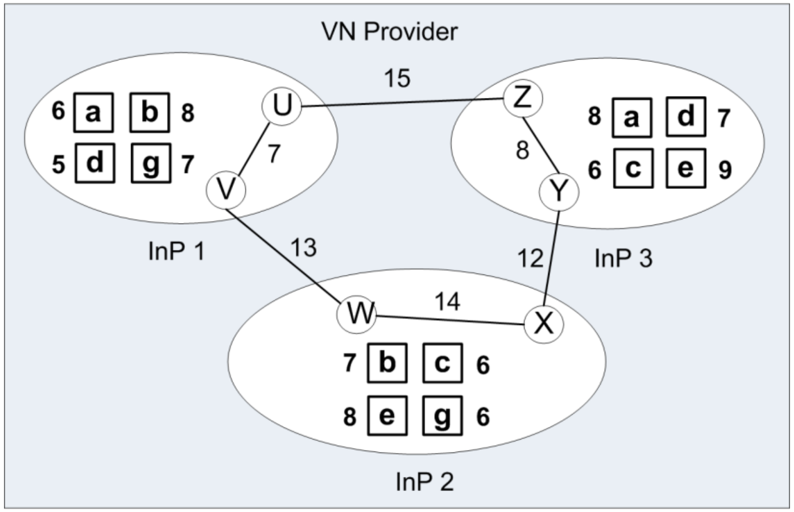


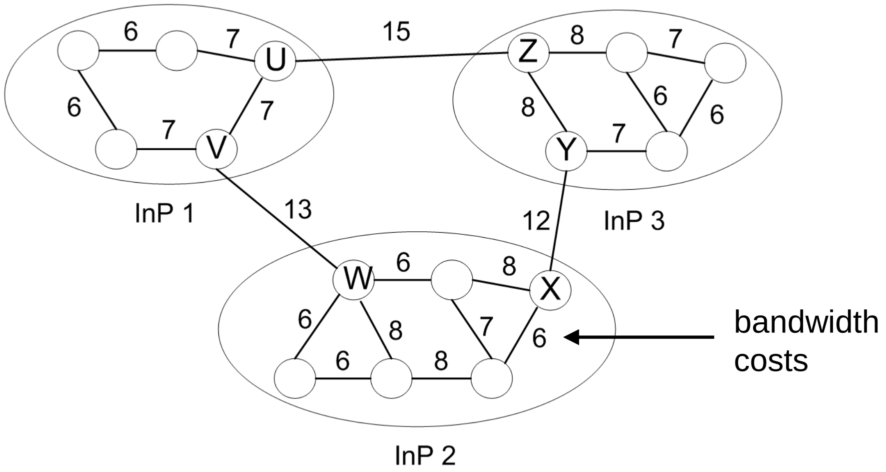


bandwidth costs

Substrate Network Visibility

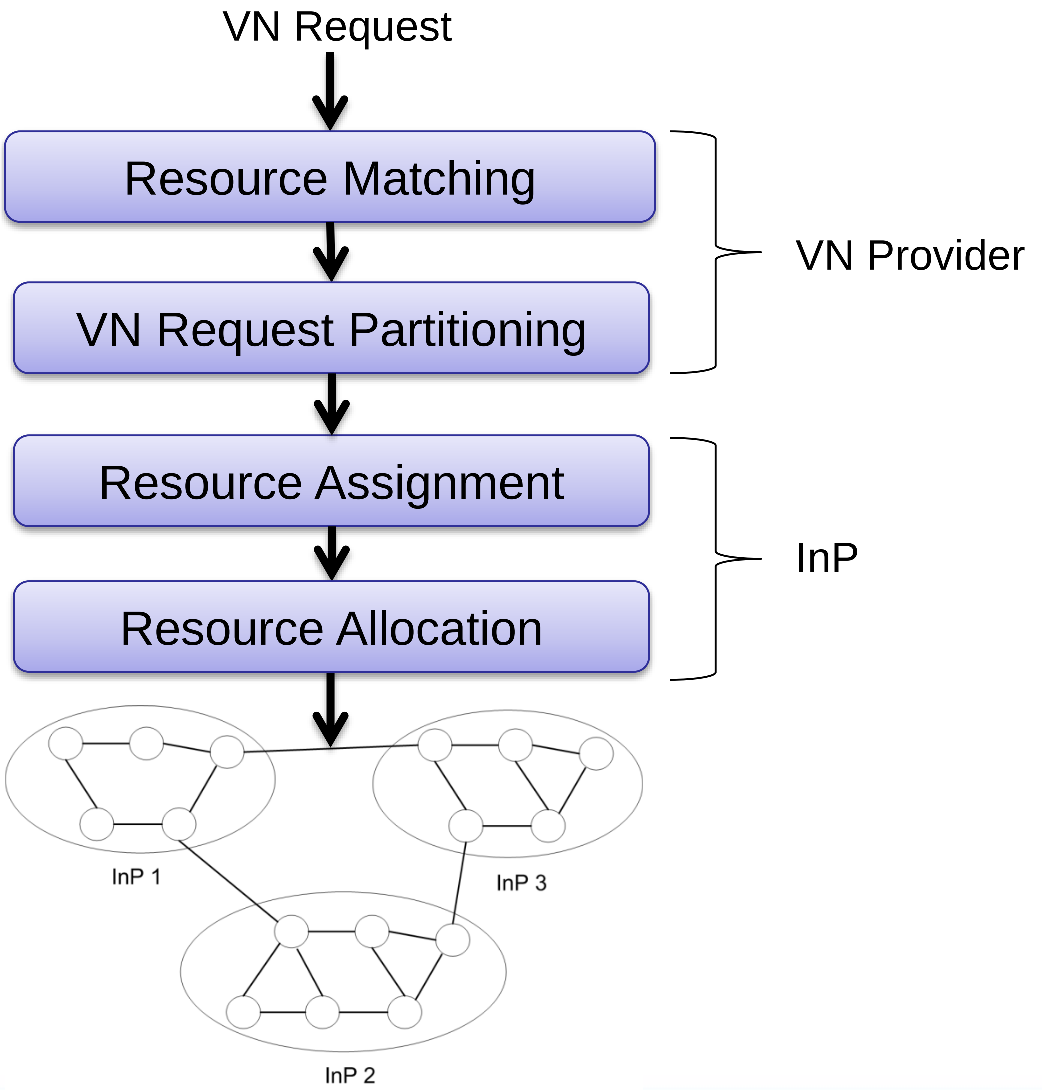
- VN Provider‘s visibility on substrate network topology and resources is limited to:
  - Offered virtual node types (similar to Amazon EC2)
  - Location of peering nodes
  - Bandwidth cost over links between peering nodes





bandwidth costs

Multi-Domain Virtual Network Embedding



VN Request

Resource Matching

VN Request Partitioning

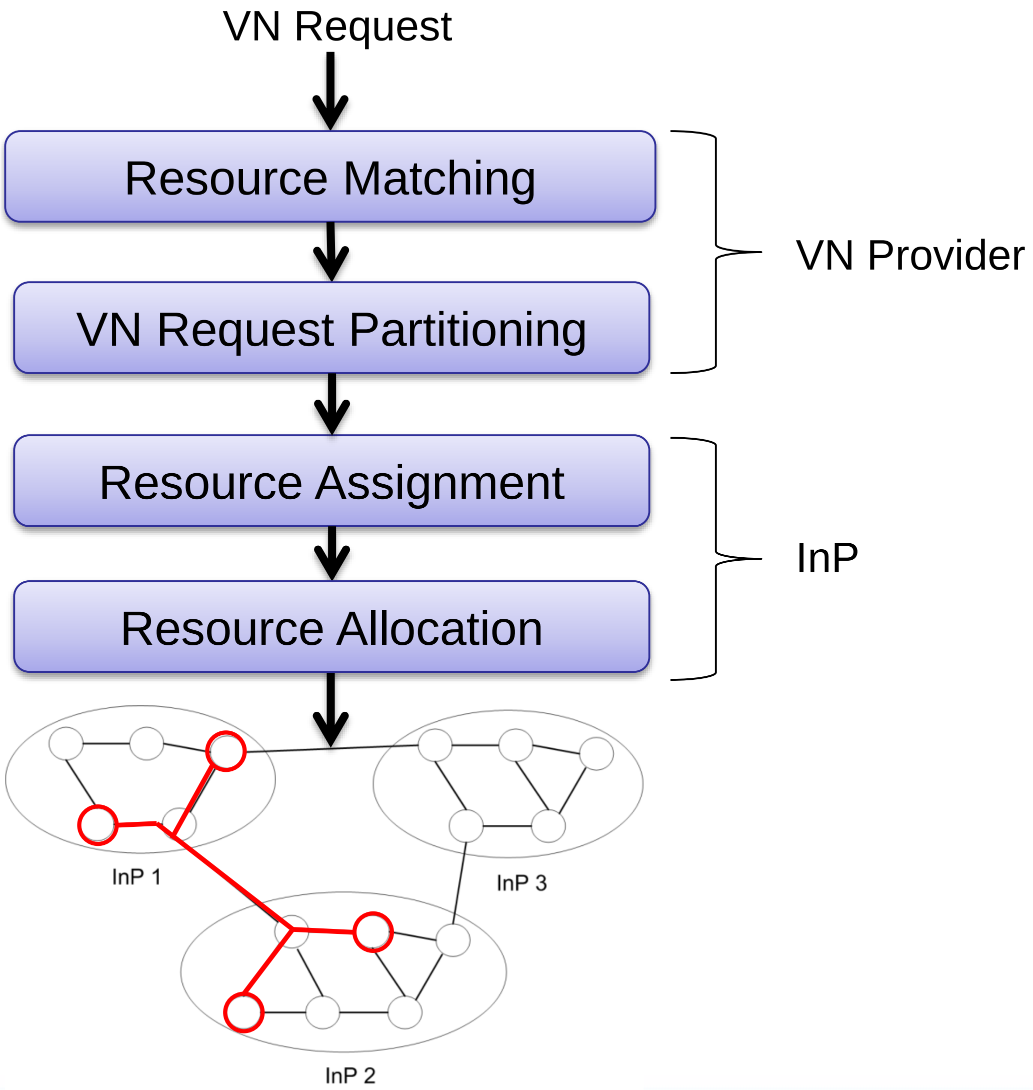
Resource Assignment

Resource Allocation

VN Provider

InP

Multi-Domain Virtual Network Embedding



VN Request

Resource Matching

VN Request Partitioning

Resource Assignment

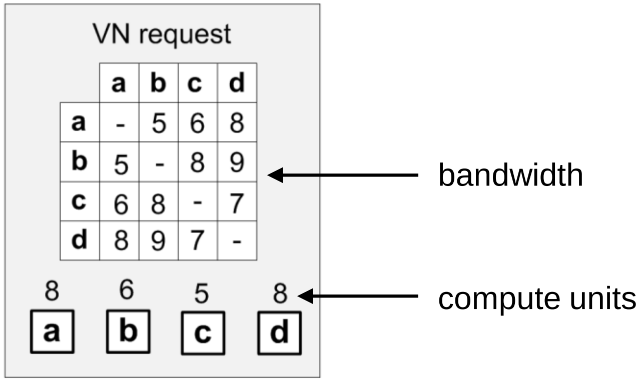
Resource Allocation

VN Provider

InP

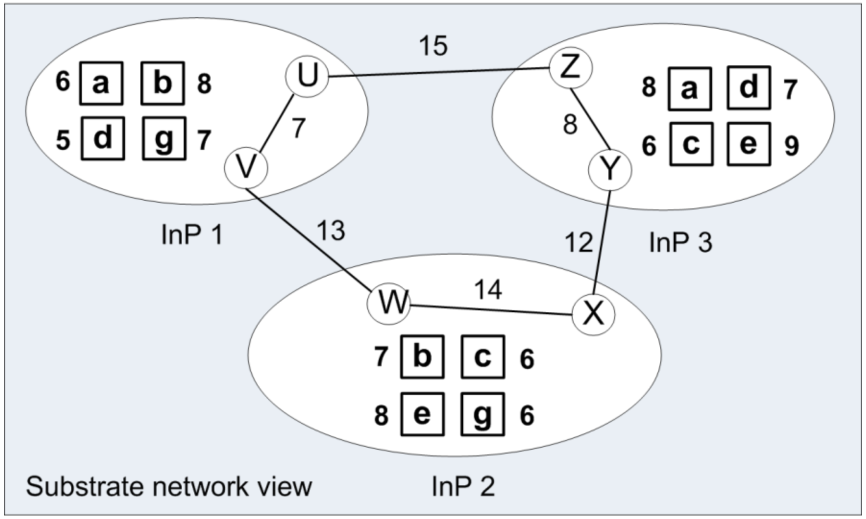
Resource Matching

- VN Provider matches requested to advertised resources
    - Candidates for each requested resource are identified



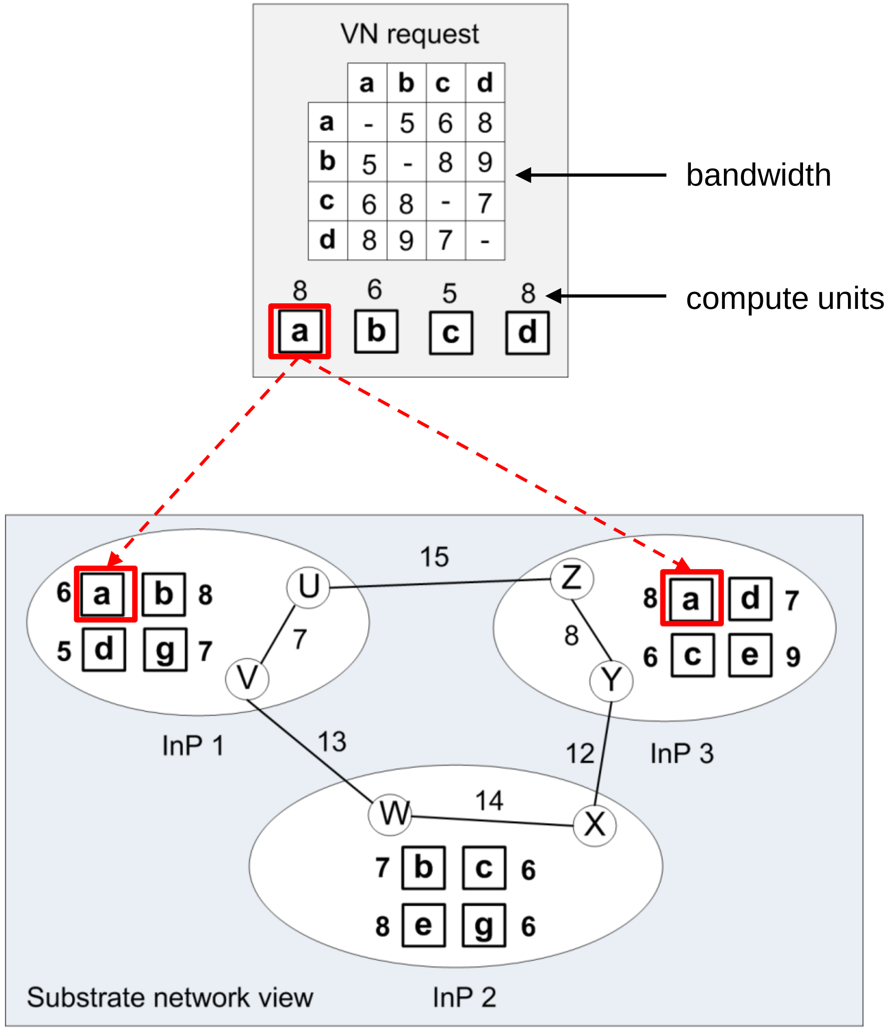
VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-



Resource Matching

- VN Provider matches requested to advertised resources
  - Candidates for each requested resource are identified
    - `a: {InP1, InP3}`

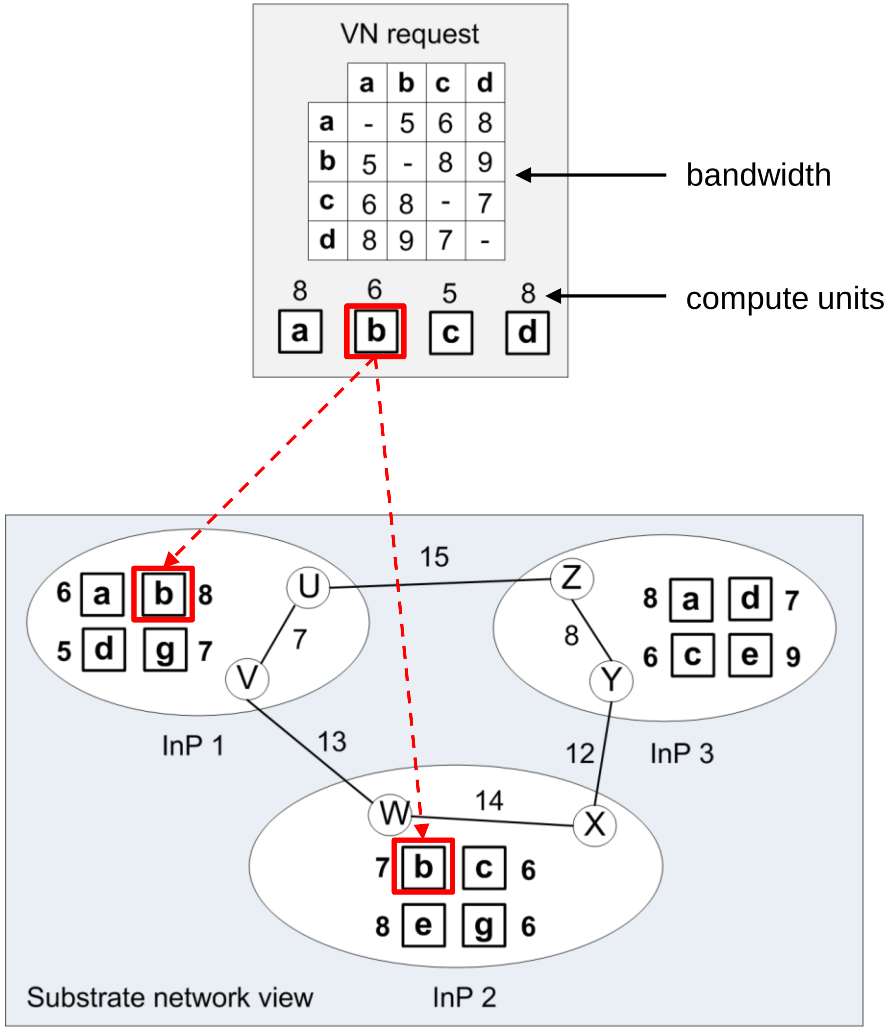


VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

Resource Matching

- VN Provider matches requested to advertised resources
  - Candidates for each requested resource are identified
    - `a: {InP1, InP3}`
    - `b: {InP1, InP2}`

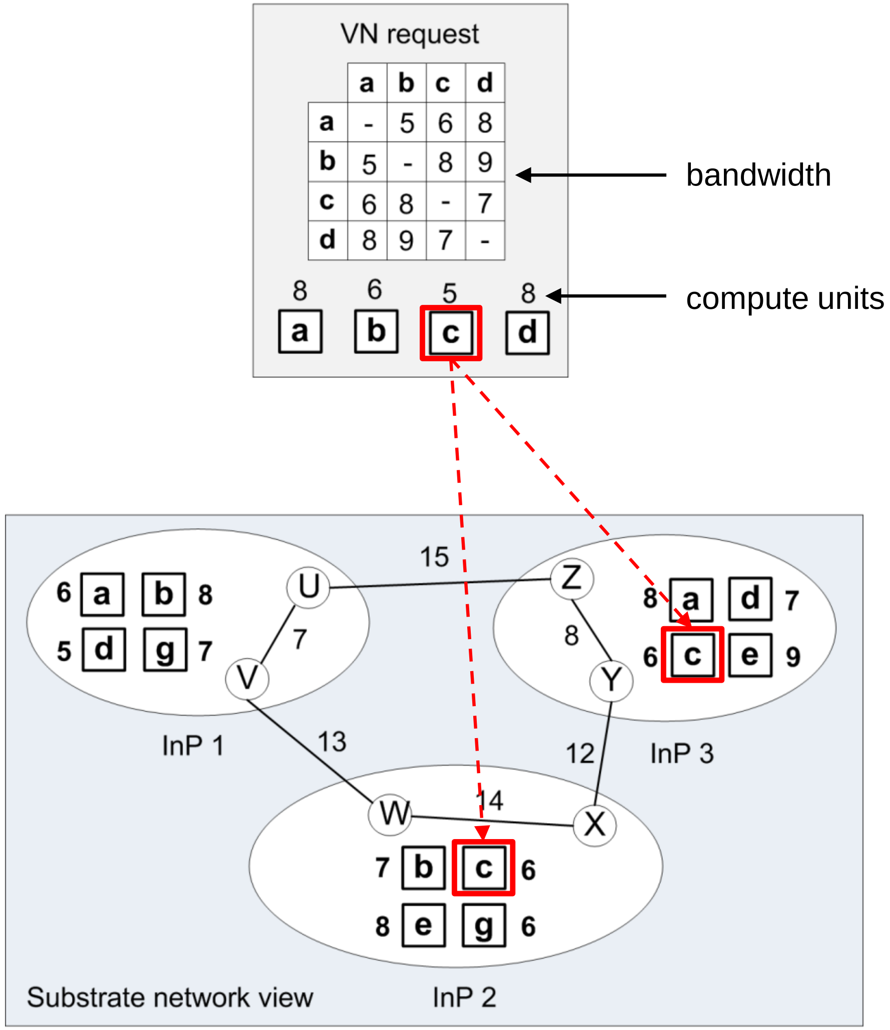


VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

Resource Matching

- VN Provider matches requested to advertised resources
  - Candidates for each requested resource are identified
    - `a: {InP1, InP3}`
    - `b: {InP1, InP2}`
    - `c: {InP2, InP3}`

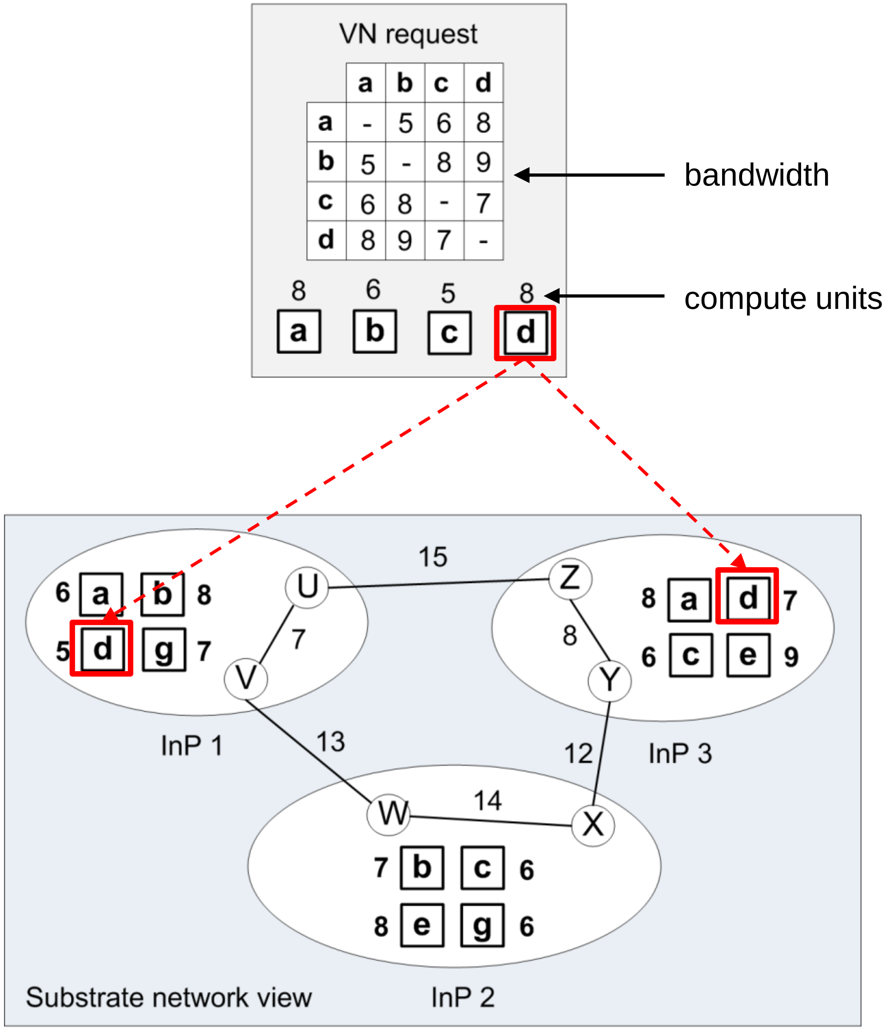


VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

Resource Matching

- VN Provider matches requested to advertised resources
  - Candidates for each requested resource are identified
    - `a: {InP1, InP3}`
    - `b: {InP1, InP2}`
    - `c: {InP2, InP3}`
    - `d: {InP1, InP3}`

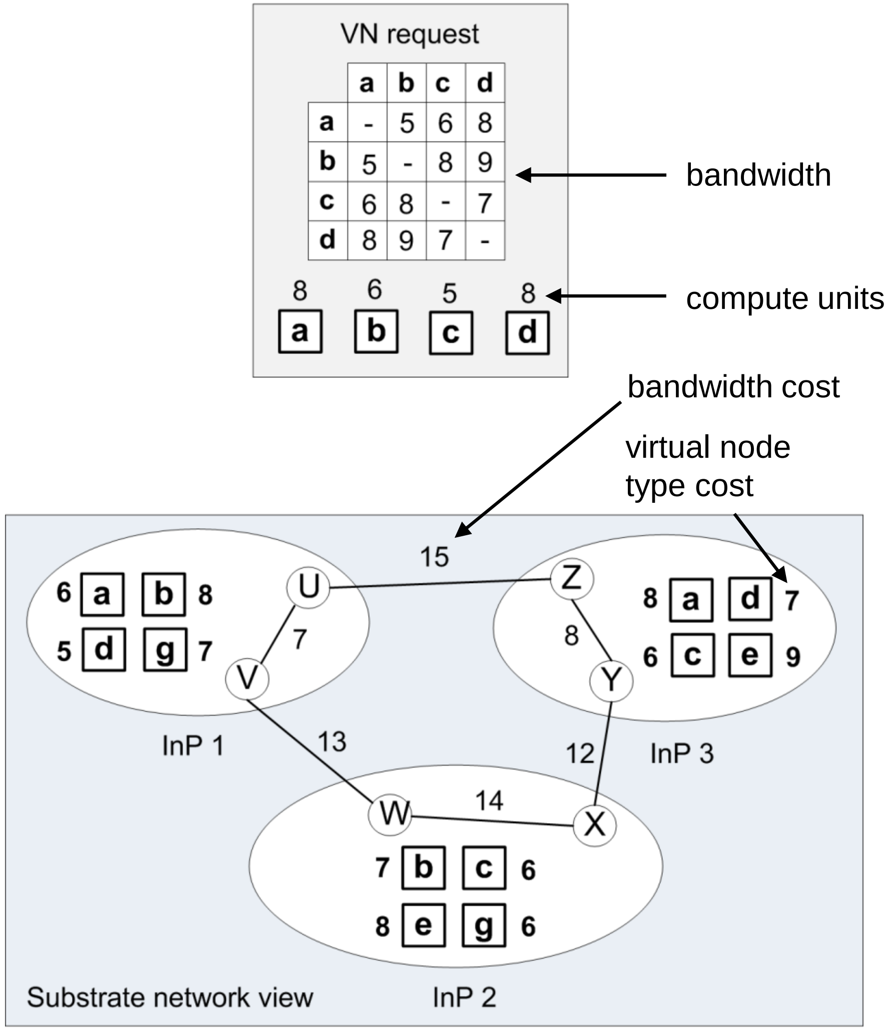


VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

VN Request Partitioning

- Binary integer program:
  - Objective:
    - Minimize the cost for the Service Provider

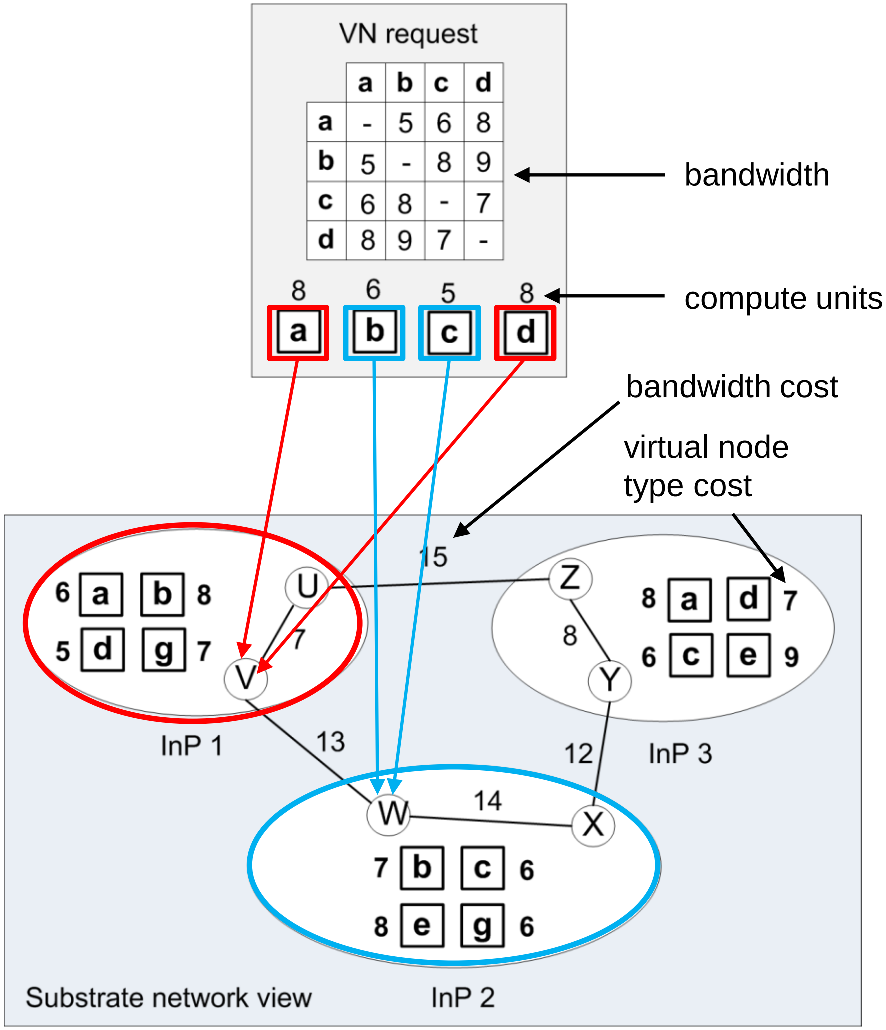


VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

VN Request Partitioning

- Binary integer program:
    - Objective:
        - Minimize the cost for the Service Provider
    - Solution:
        - Virtual node to peering node assignment

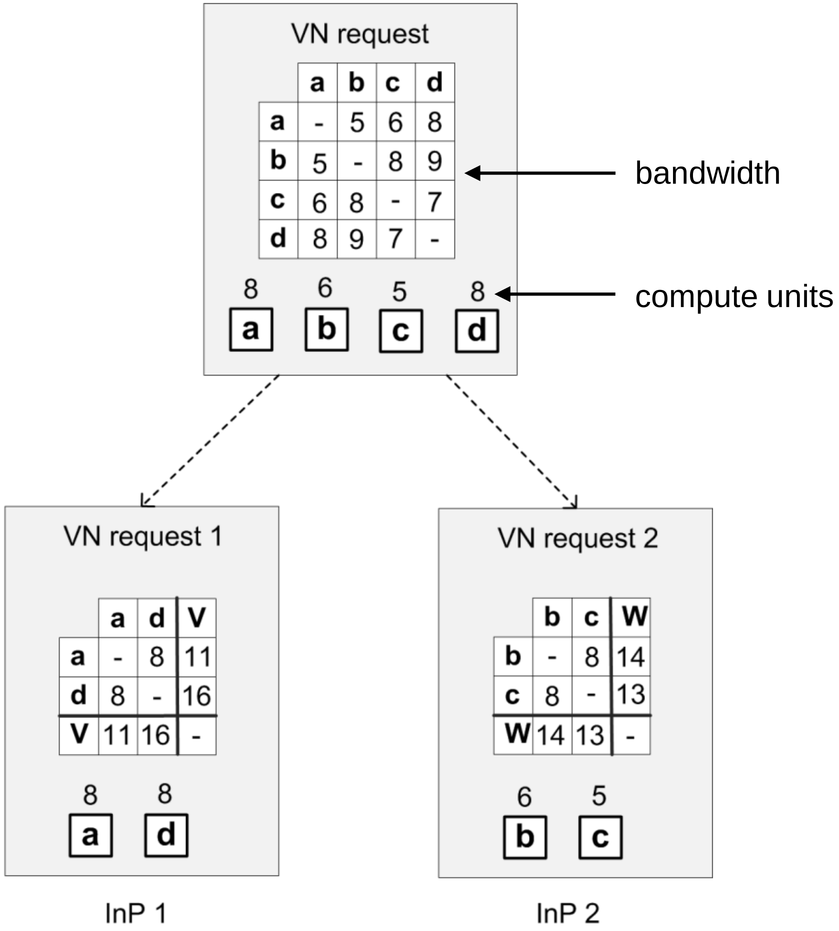


VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

VN Request Partitioning

- Binary integer program:
  - Objective:
    - Minimize the cost for the Service Provider
  - Solution:
    - Virtual node to peering node assignment
- VN segment description:
  - Virtual node specifications
  - Bandwidth demands between virtual nodes and peering nodes



VN request

	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

8	6	5	8
a	b	c	d

VN request 1

	a	d	V
a	-	8	11
d	8	-	16
V	11	16	-

8	8
a	d

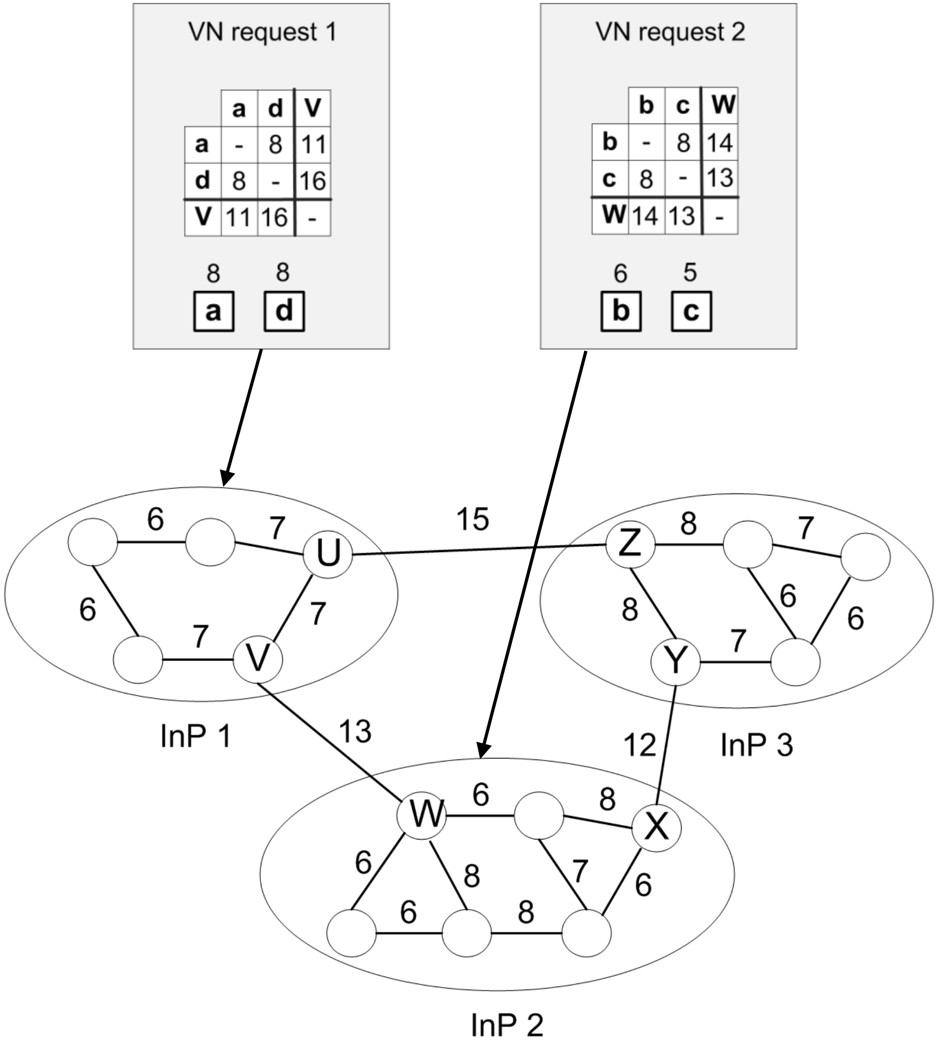
VN request 2

	b	c	W
b	-	8	14
c	8	-	13
W	14	13	-

6	5
b	c

VN Segment Mapping

- Mixed integer multi-commodity flow problem:
  - Objective:
    - Minimize the embedding cost



VN request 1

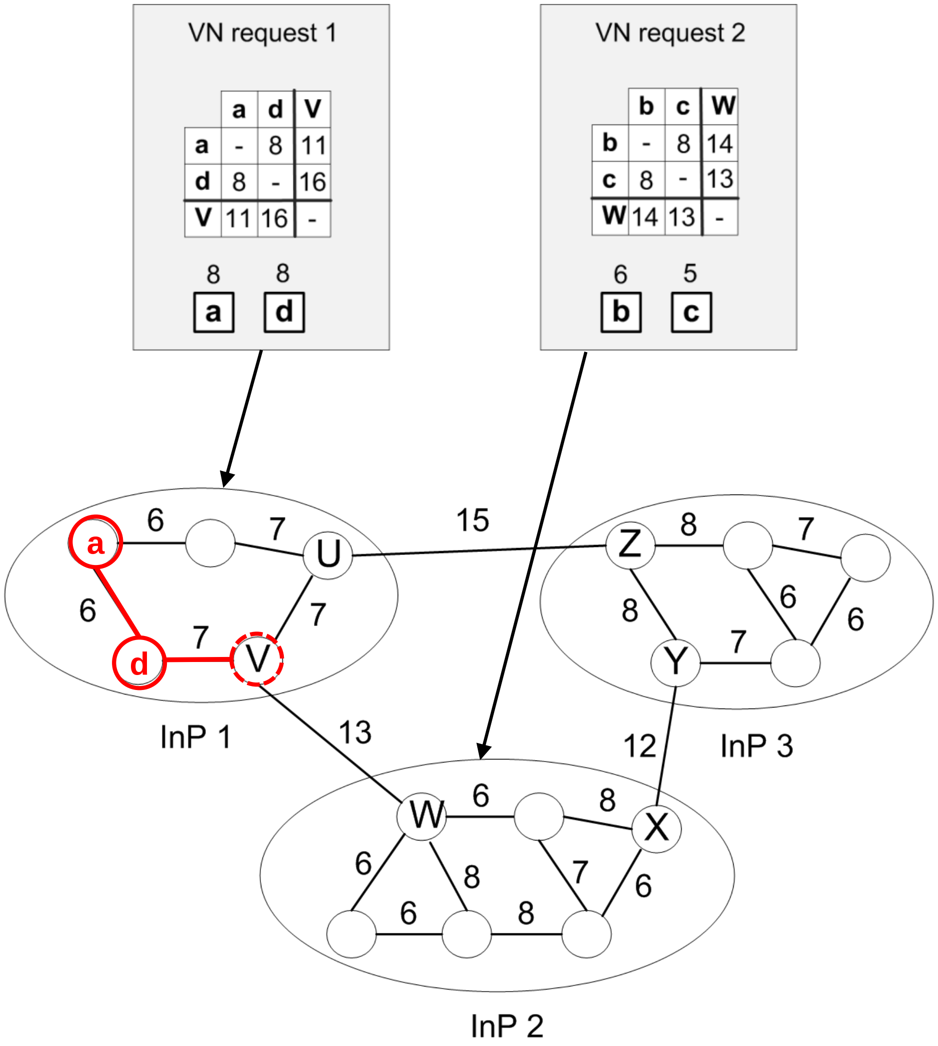
	a	d	V
a	-	8	11
d	8	-	16
V	11	16	-

VN request 2

	b	c	W
b	-	8	14
c	8	-	13
W	14	13	-

VN Segment Mapping

- Mixed integer multi-commodity flow problem:
  - Objective:
    - Minimize the embedding cost
  - Solution:
    - VN segment mapping onto the substrate network
      - Compliance with virtual node to peering node bindings



VN request 1

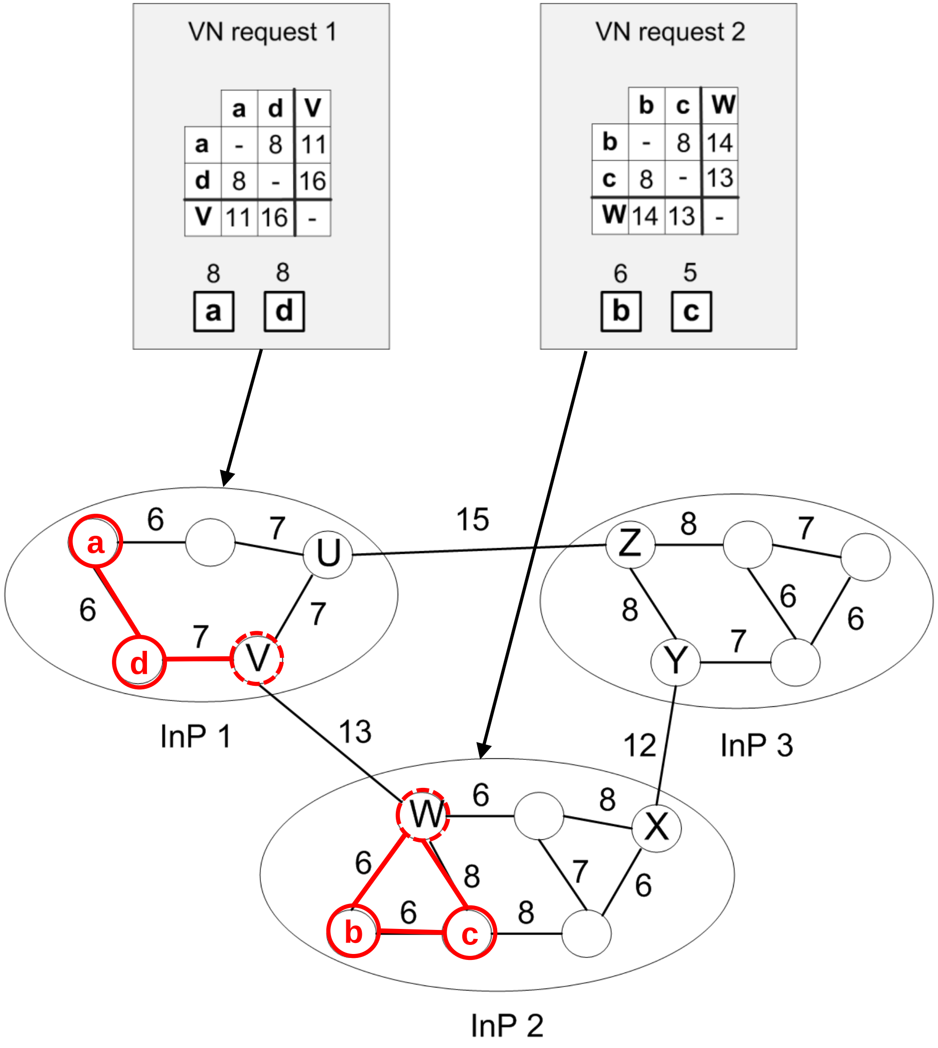
	a	d	V
a	-	8	11
d	8	-	16
V	11	16	-

VN request 2

	b	c	W
b	-	8	14
c	8	-	13
W	14	13	-

VN Segment Mapping

- Mixed integer multi-commodity flow problem:
  - Objective:
    - Minimize the embedding cost
  - Solution:
    - VN segment mapping onto the substrate network
      - Compliance with virtual node to peering node bindings



VN request 1

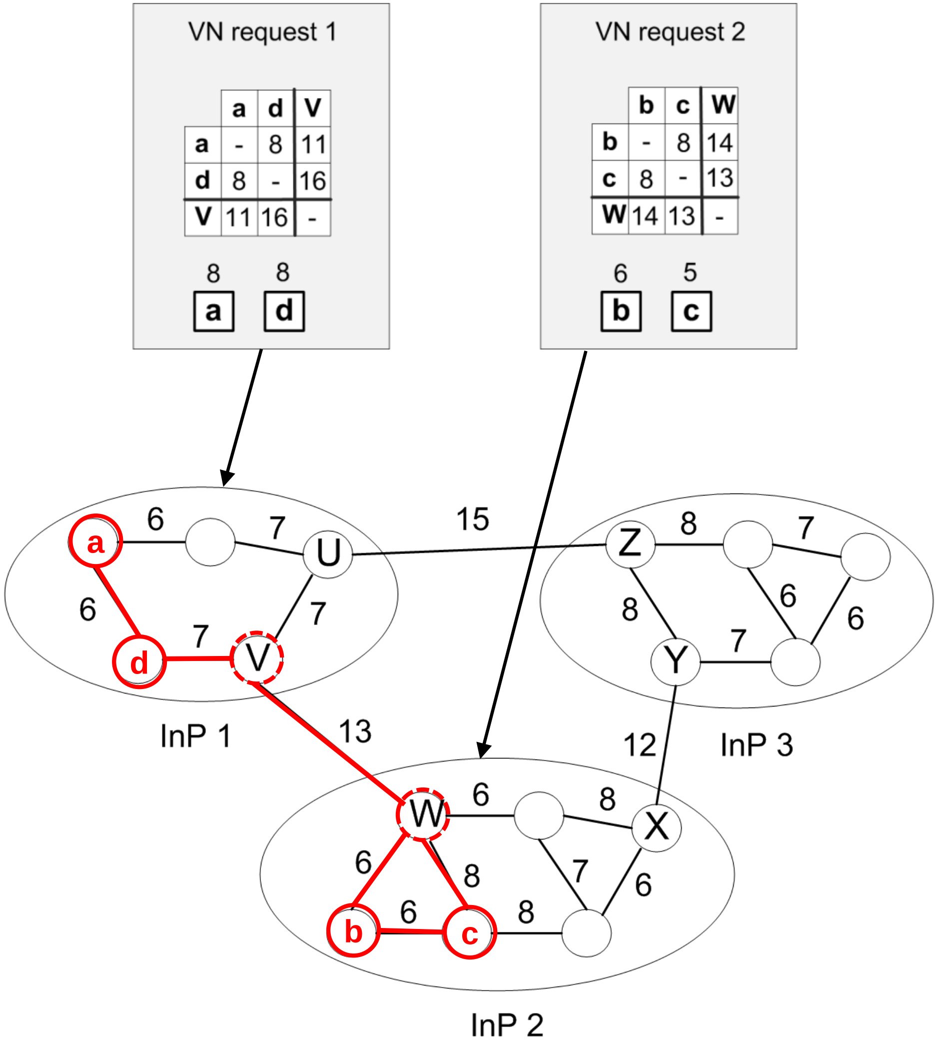
	a	d	V
a	-	8	11
d	8	-	16
V	11	16	-

VN request 2

	b	c	W
b	-	8	14
c	8	-	13
W	14	13	-

VN Segment Mapping

- Mixed integer multi-commodity flow problem:
  - Objective:
    - Minimize the embedding cost
  - Solution:
    - VN segment mapping onto the substrate network
      - Compliance with virtual node to peering node bindings



VN request 1

	a	d	V
a	-	8	11
d	8	-	16
V	11	16	-

VN request 2

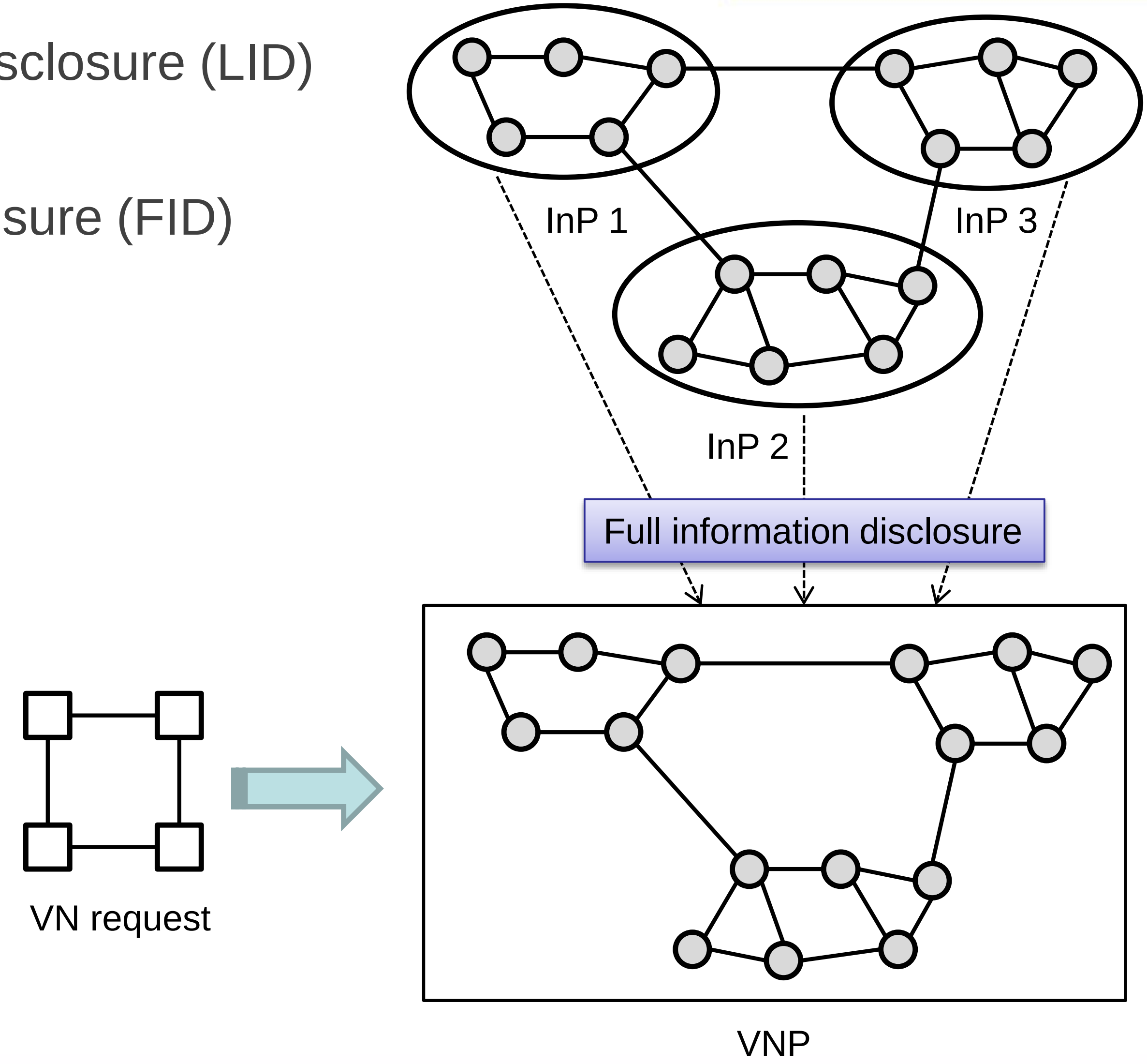
	b	c	W
b	-	8	14
c	8	-	13
W	14	13	-

VN Embedding Efficiency

Limited information disclosure (LID)

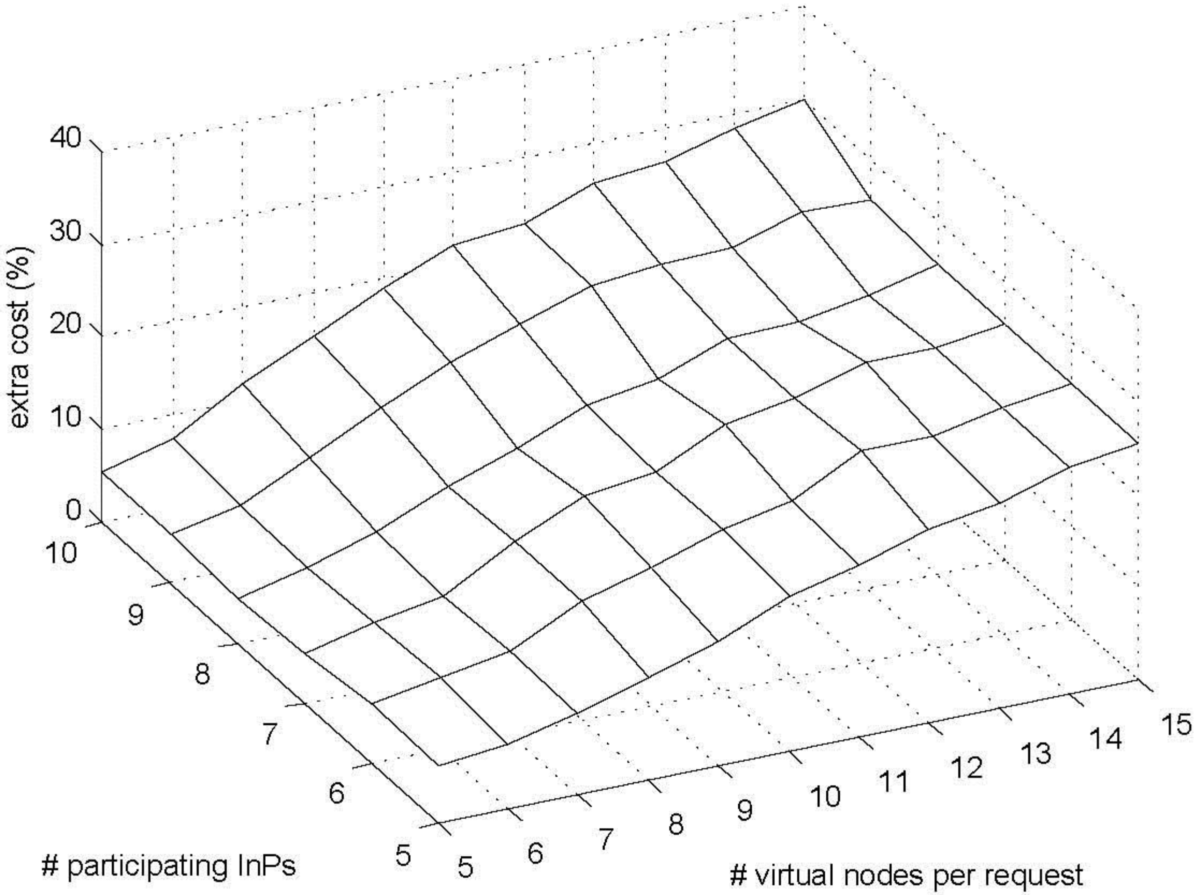
vs.

Full information disclosure (FID)



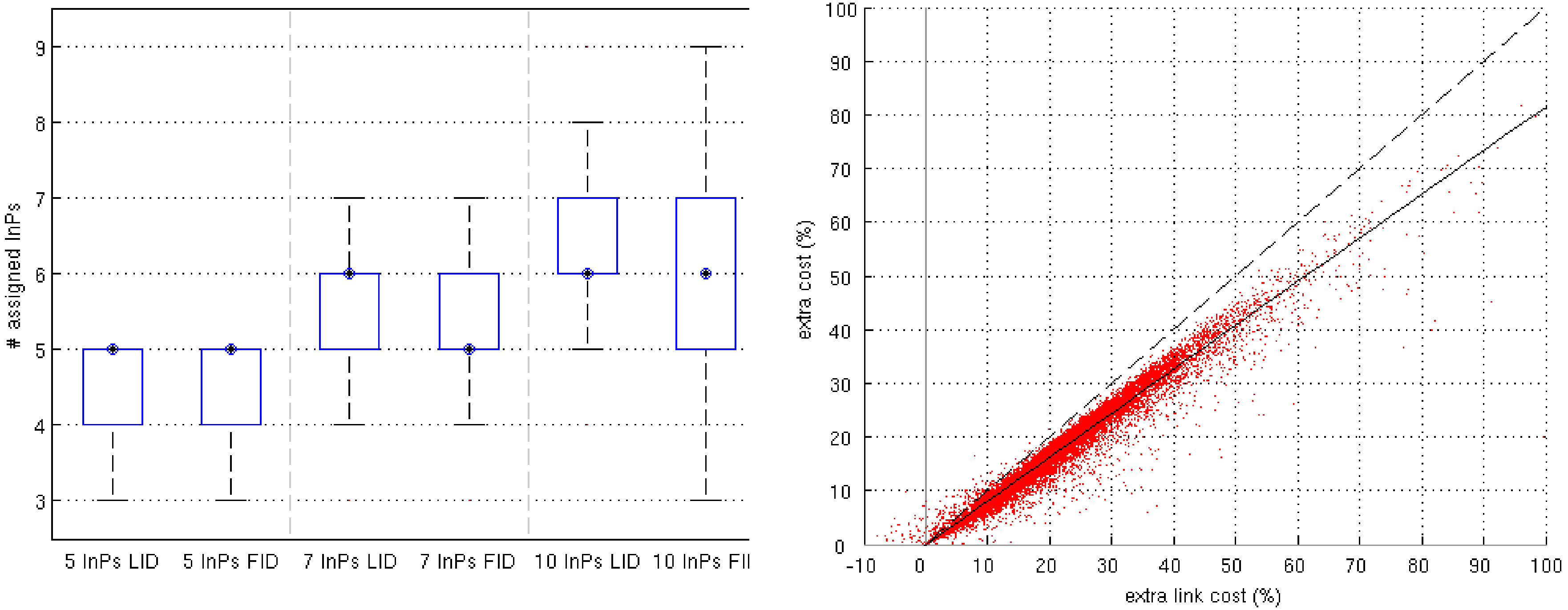
Extra Cost under LID

- LID incurs 5-30% extra cost



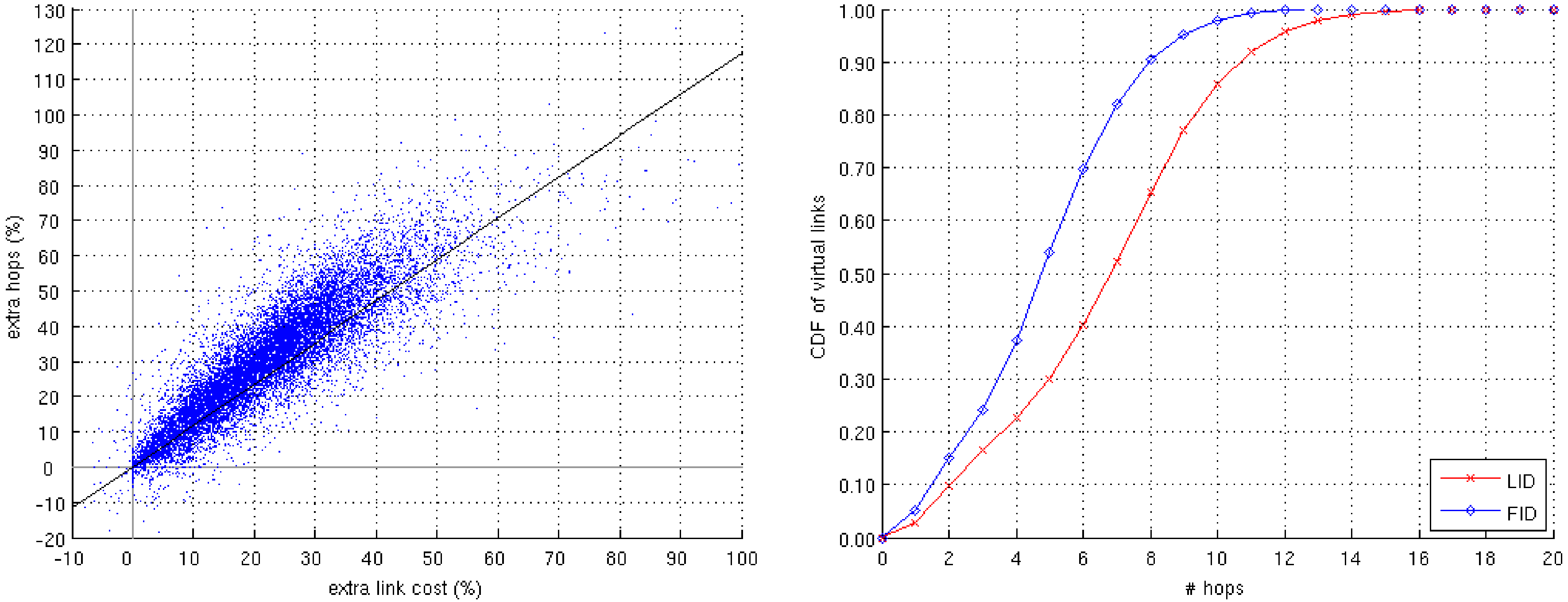
Origins of Extra Cost

- Same number of assigned InPs under LID and FID
- Extra cost is correlated with extra link cost



Origins of Extra Link Cost

- Extra link cost is due to increased hop count



Virtual Link Setup

Virtual Link Setup

- Objective:
  - Interoperable solution for virtual link setup with QoS guarantees across InPs
- Approach:
  - Couple virtual link setup signaling with QoS reservation signaling for efficiency
    - Rely on existing QoS resource reservation protocol (IETF NSIS)
    - Add new object to NSIS QoS NSLP to carry the required information for virtual link setup
- Requirements:
  - NSIS support in routers
  - IP-based substrate
  - New QoS NSLP object support (only) in virtual link end-points

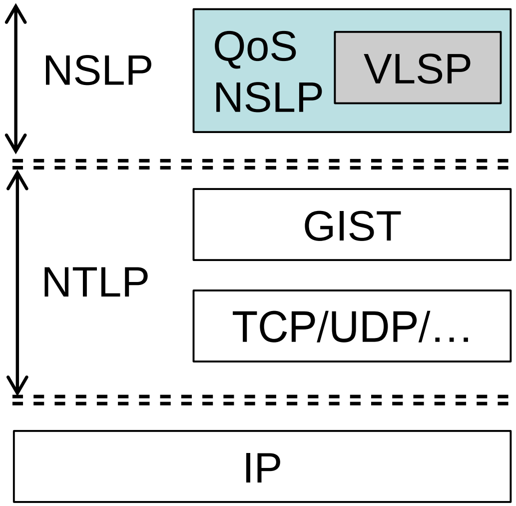
Virtual Link Setup Protocol (VLSP)

- NSIS QoS NSLP extension with new Virtual Link Setup Protocol (VLSP) object:
  - Virtual link setup at the end-points via VLSP
  - Resource reservation and QoS via NSLP object at the intermediate nodes

0 64 127

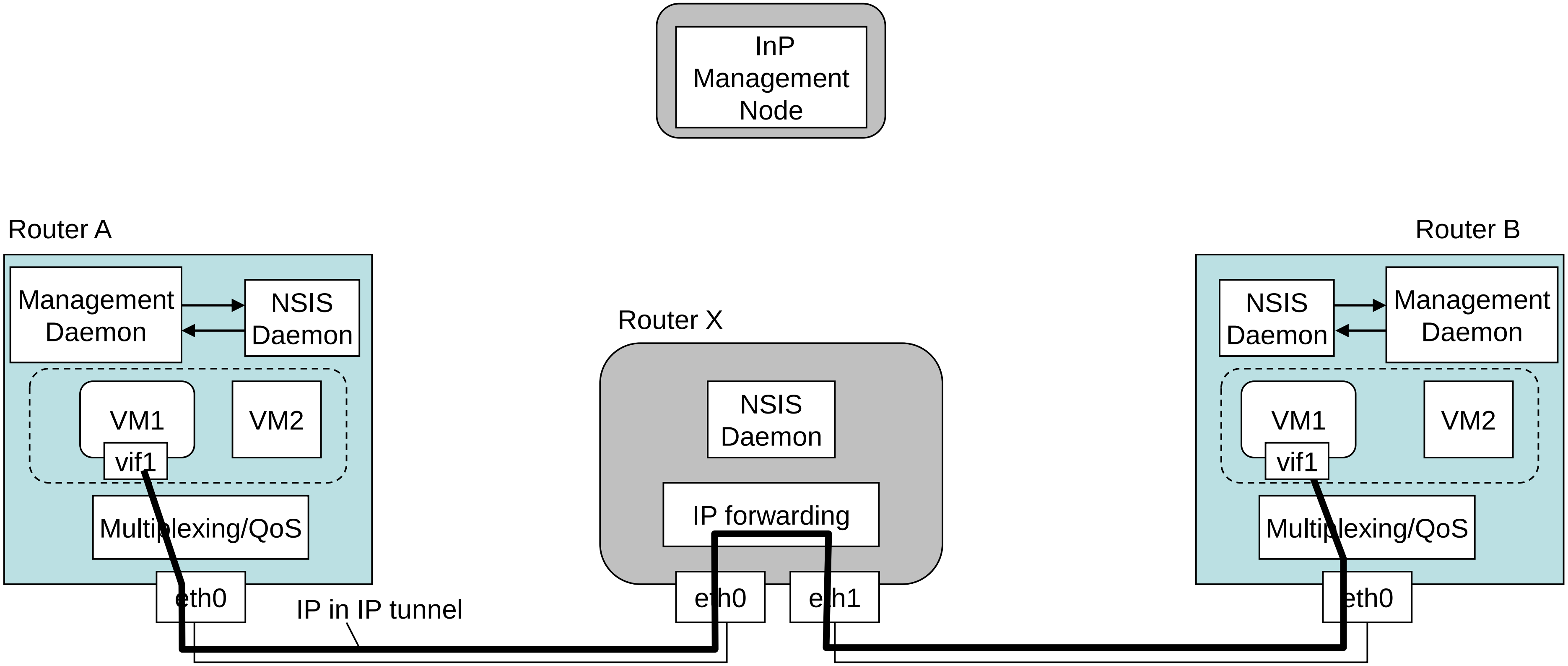
Virtual Network ID	
Source Virtual Node ID	
Destination Virtual Node ID	
Source Virtual Interface ID	Destination Virtual Interface ID
Virtual Link ID (optional)	Virtual Link Type (optional)

VLSP object

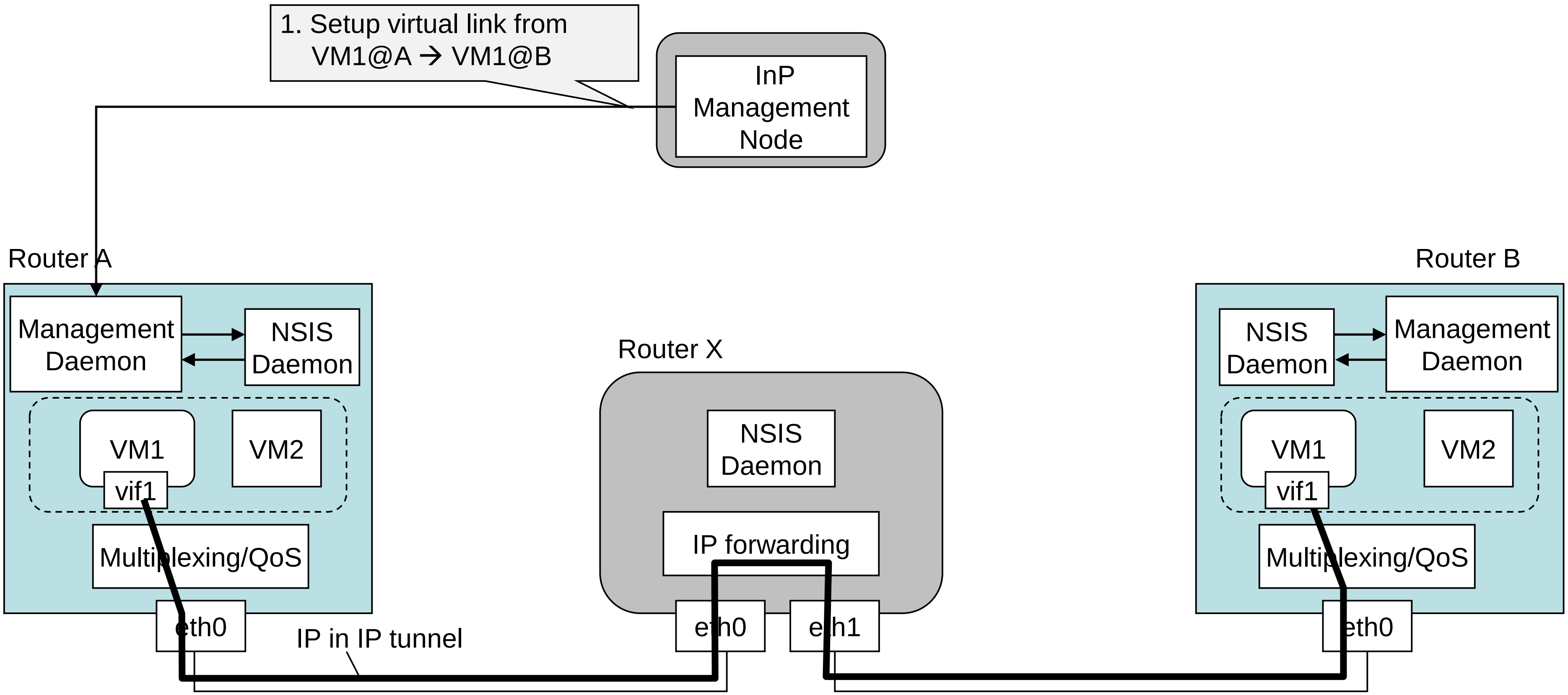


NSIS QoS NSLP/VLSP

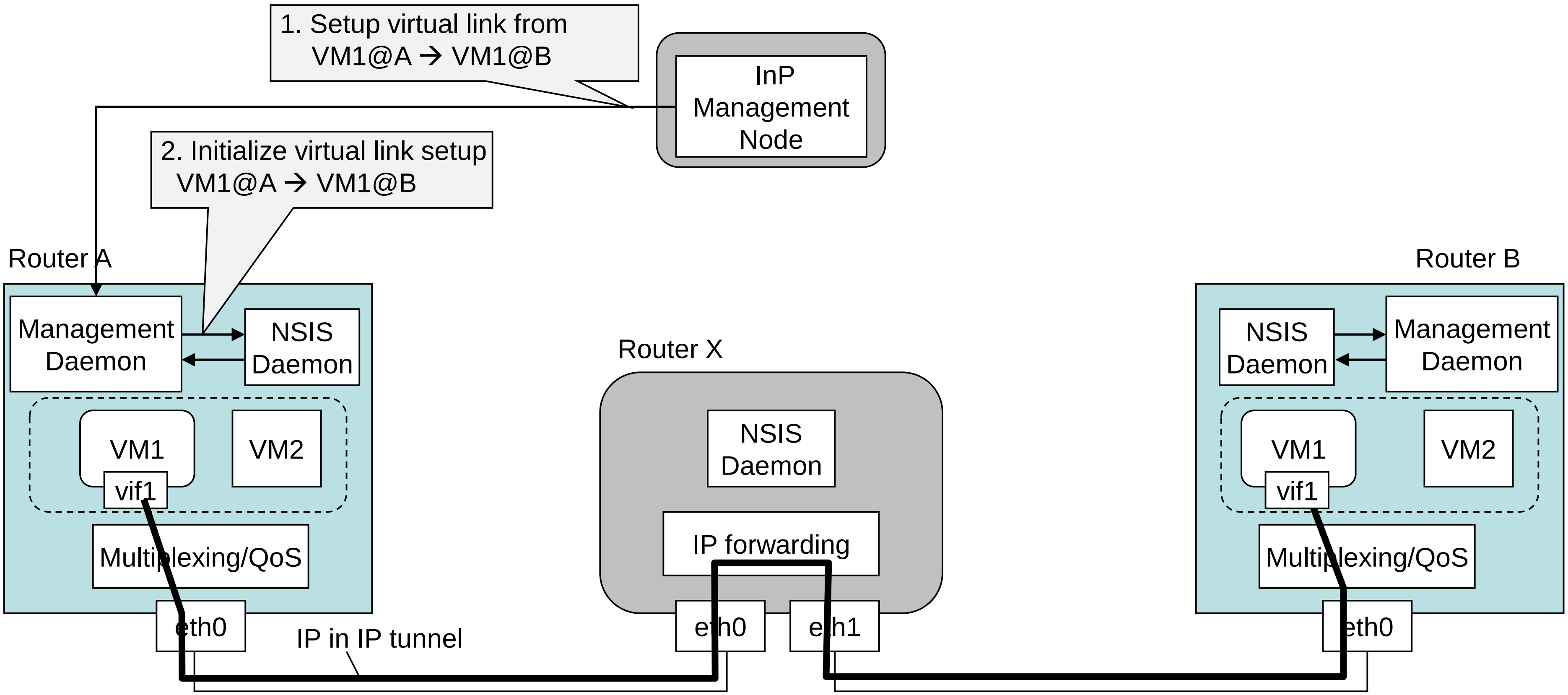
Virtual Link Setup Workflow



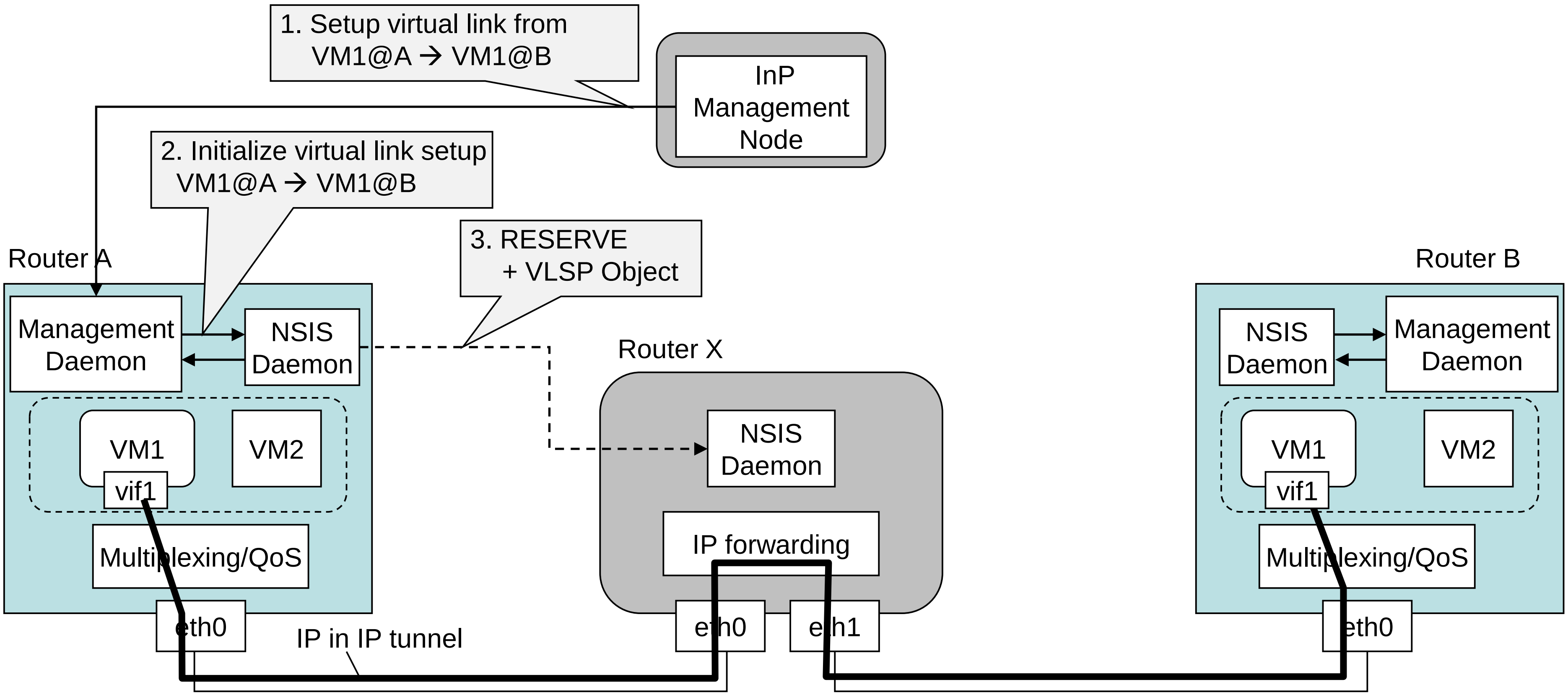
Virtual Link Setup Workflow



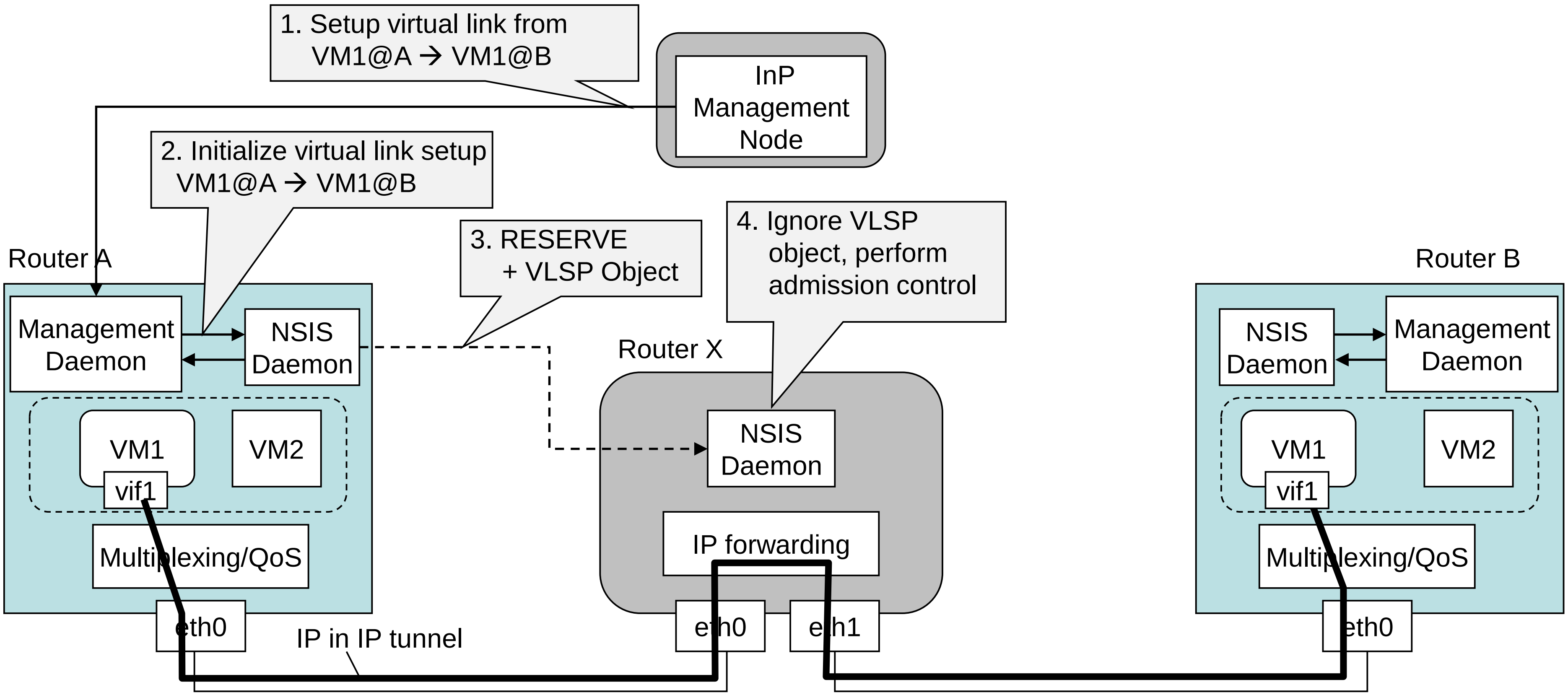
Virtual Link Setup Workflow



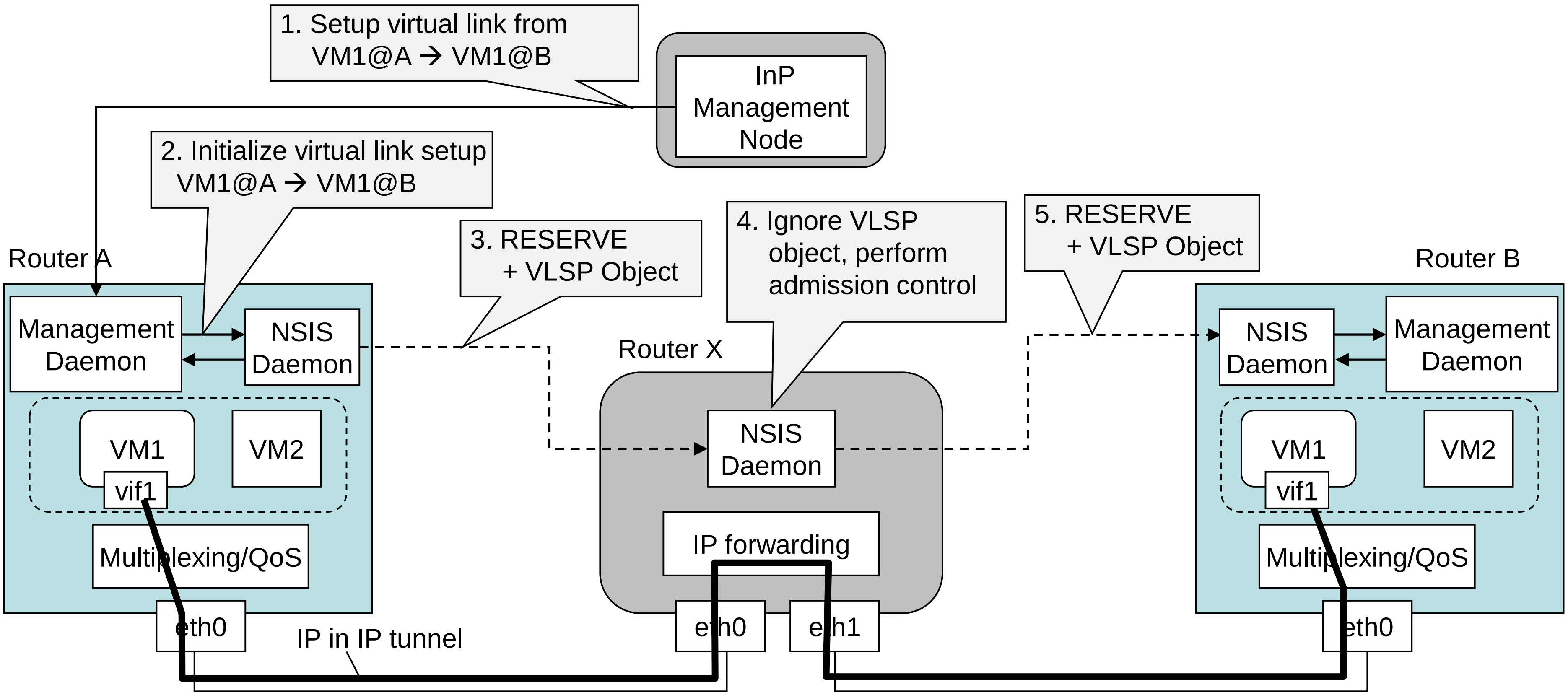
Virtual Link Setup Workflow



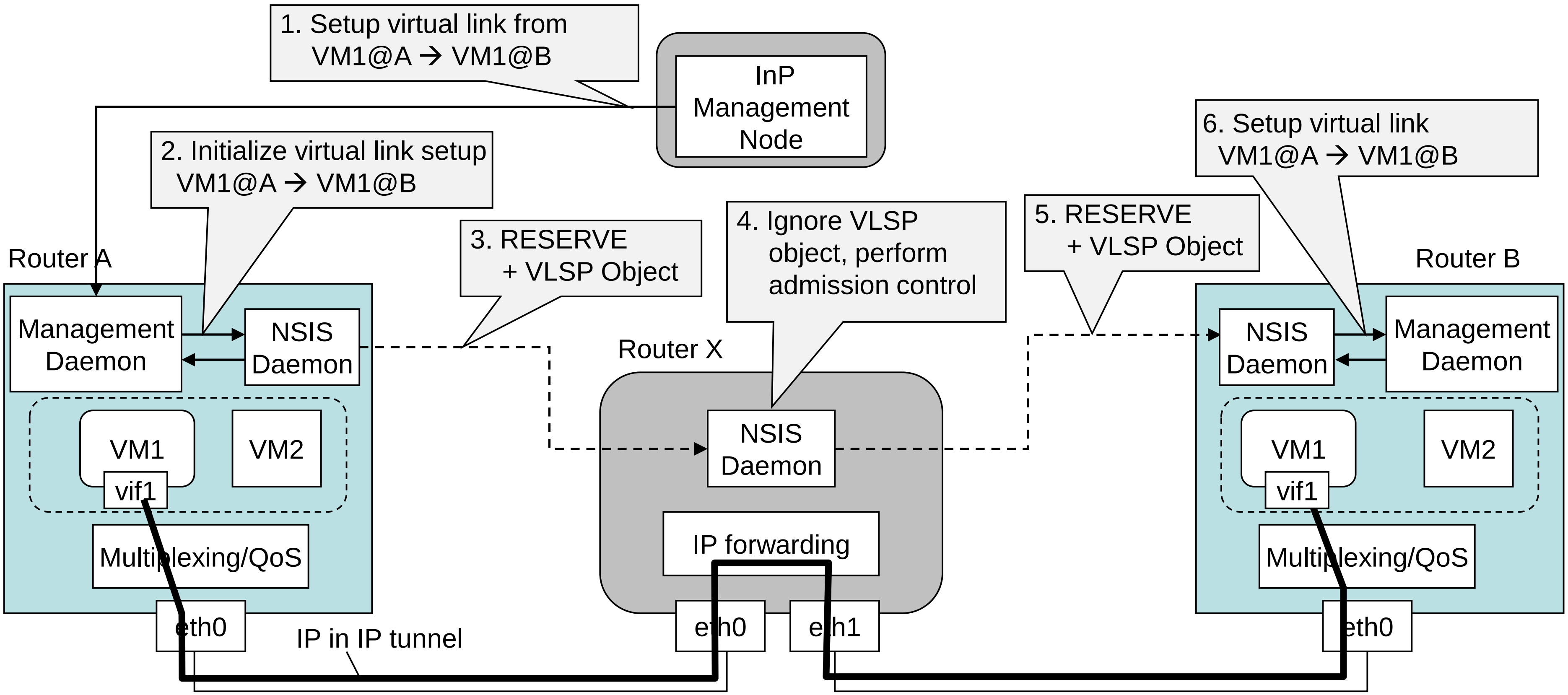
Virtual Link Setup Workflow



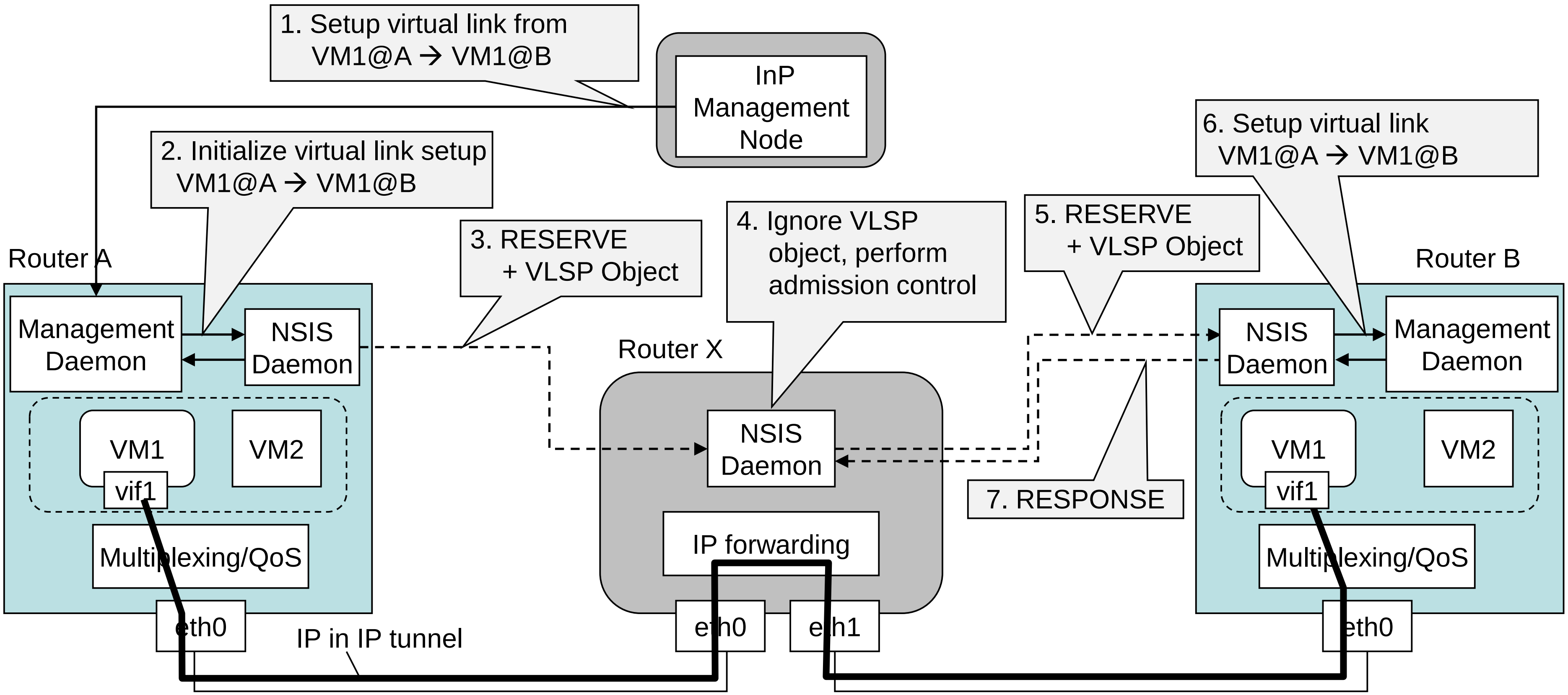
Virtual Link Setup Workflow



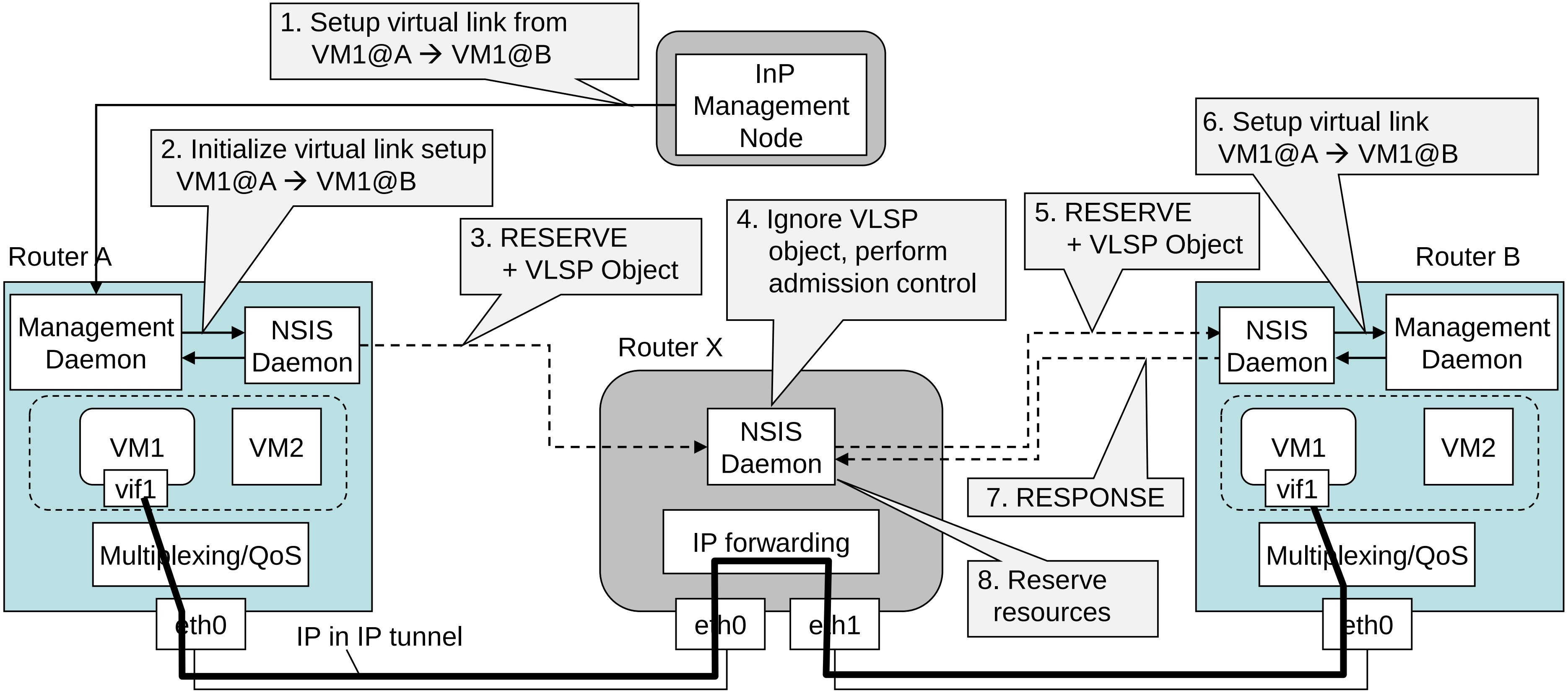
Virtual Link Setup Workflow



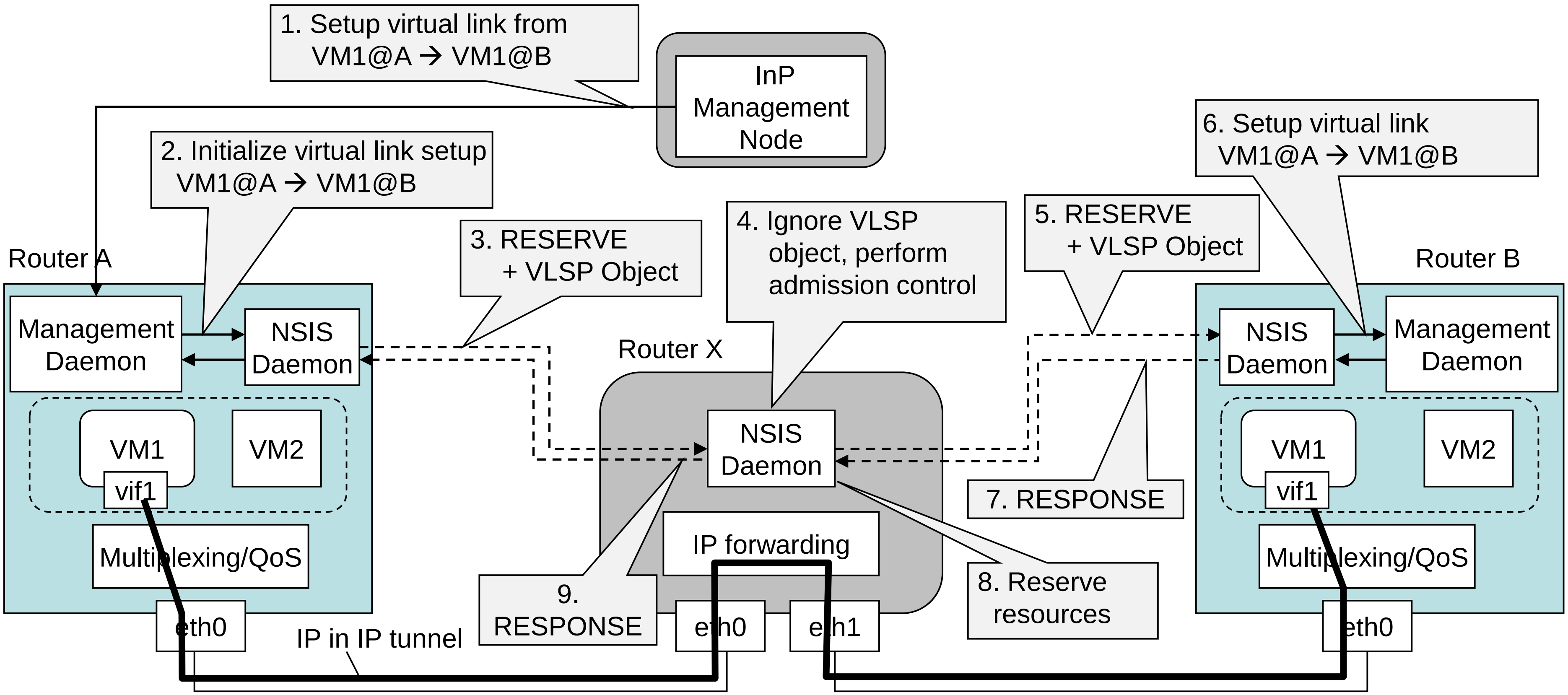
Virtual Link Setup Workflow



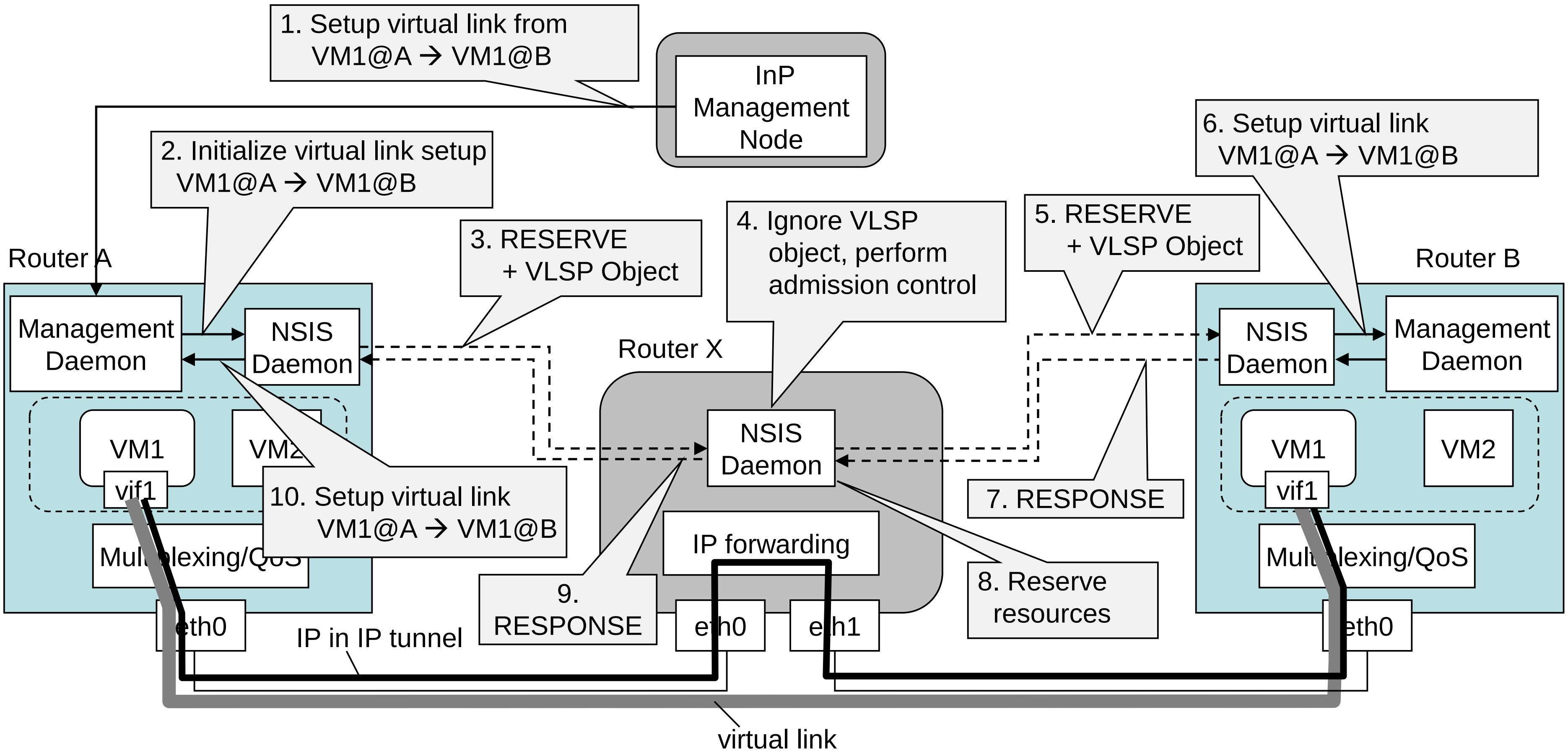
Virtual Link Setup Workflow



Virtual Link Setup Workflow



Virtual Link Setup Workflow

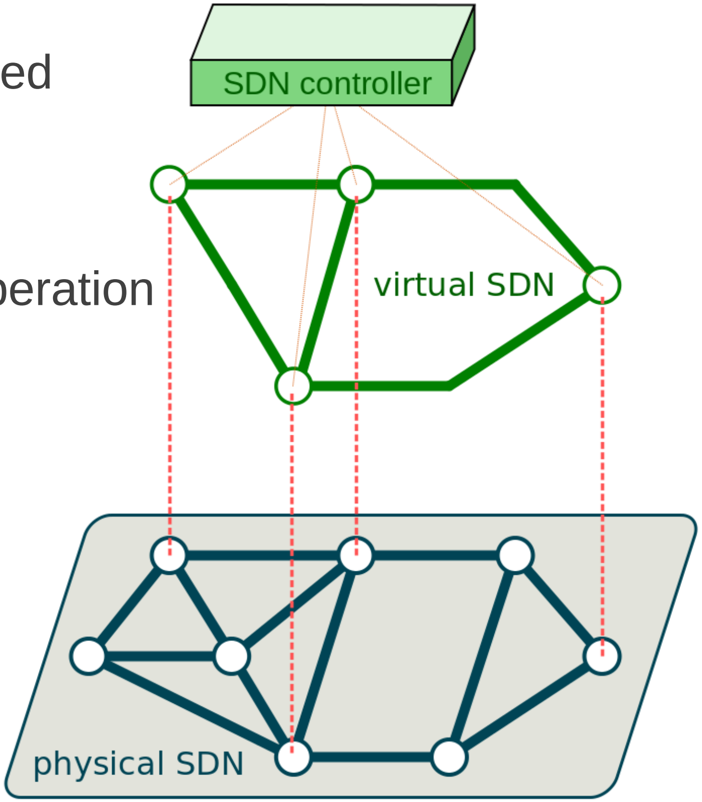


Software-Defined Network Virtualization

SDN Virtualization

- Virtual networks programmable as SDNs (vSDNs)

- Benefits:
  - Tenants:
    - Advanced control and access on virtualized network devices
  - Providers:
    - Less configuration overhead for vSDN operation
    - New cloud service model:
      - SDN as a Service (SDNaaS)



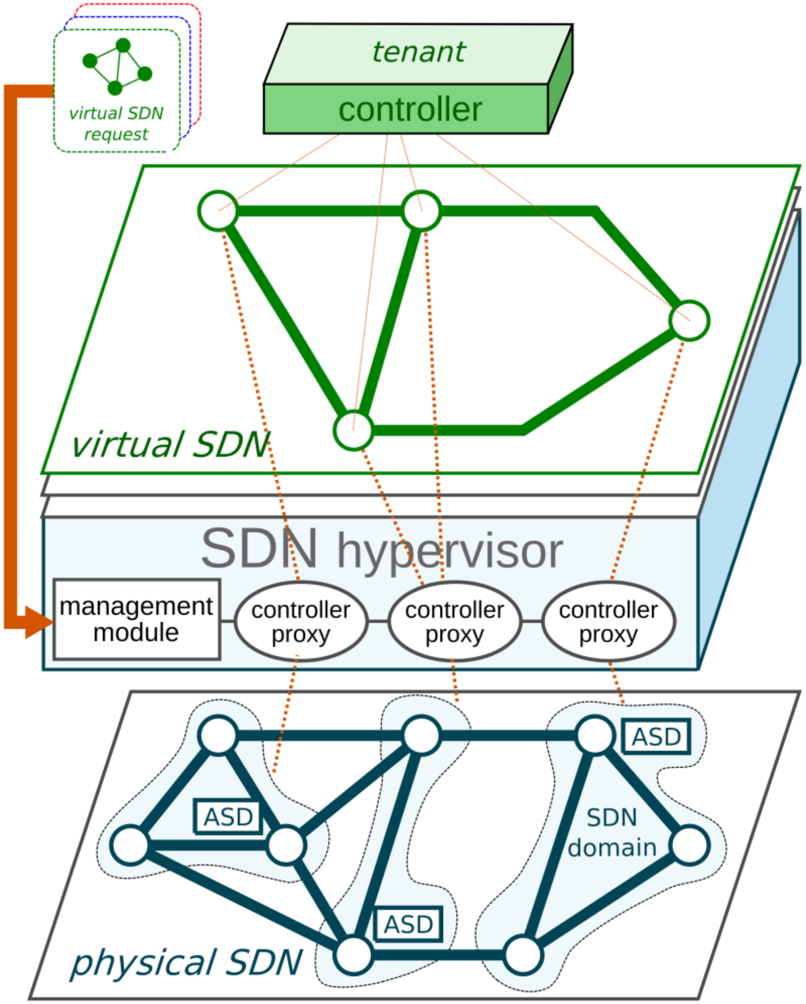
SDN Virtualization Challenges

- Automation of vSDN setup
  - vSDN mapping
  - Transparent allocation of isolated flowspaces
    - Selection of identifiers
  - Generation and installation of flow entries for packet forwarding and encapsulation
    - Binding traffic to logical context using tagging

- Deployment of arbitrary vSDN topologies
  - Mapping multiple virtual switches onto the same switch

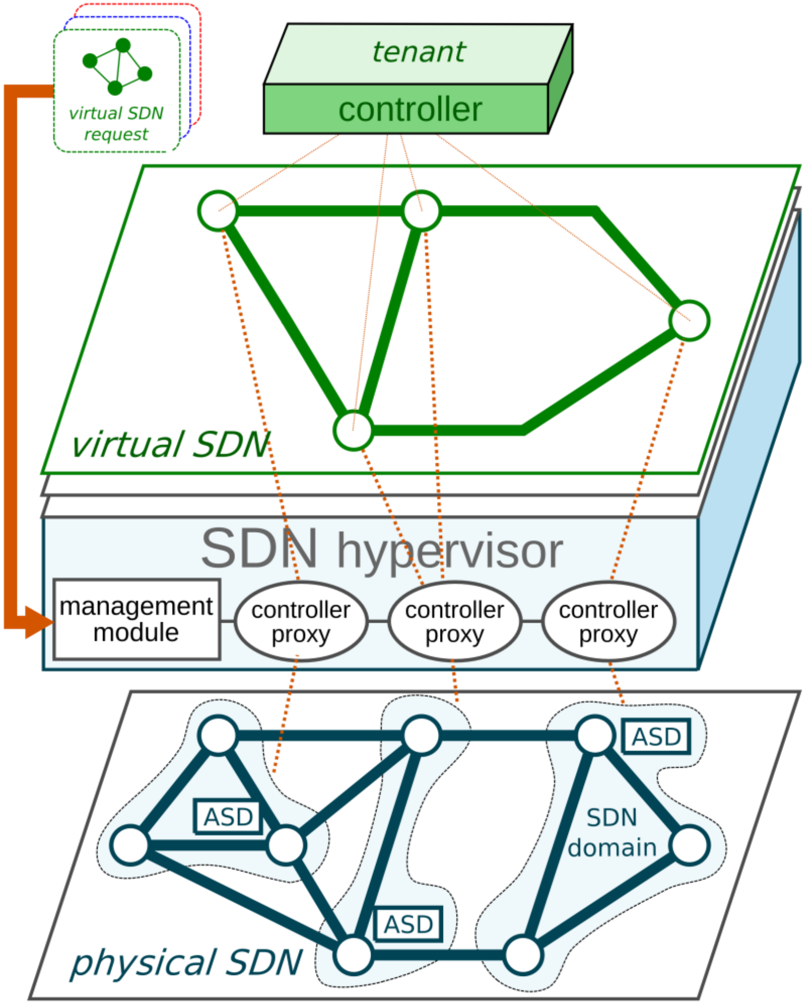
SDN Hypervisor

- Distributed SDN hypervisor
  - Multiple autonomous controller proxies (CPX)
    - Coordination by a management module (MM)
- Dataplane segmentation
  - Multiple SDN domains
  - Switches within a domain controlled by the same controller proxy



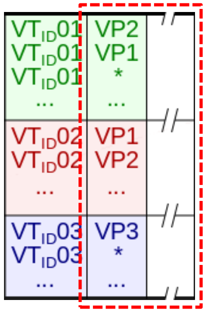
Control Plane Functions

- Management module
  - Topology embedding
  - SDN domain segmentation
  - CPX coordination for network-wide resource management
- Controller proxy
  - Infrastructure flow entry installation
  - Message translation
  - Flow cache management
  - SDN domain optimizations



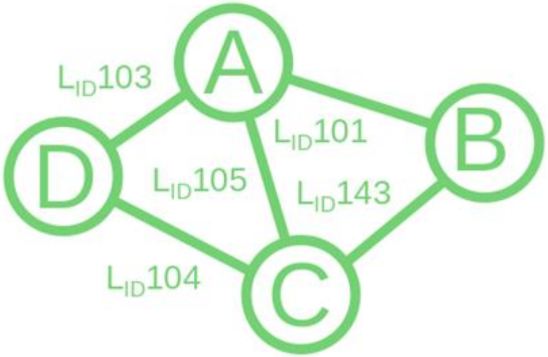
SDN Data Plane Virtualization

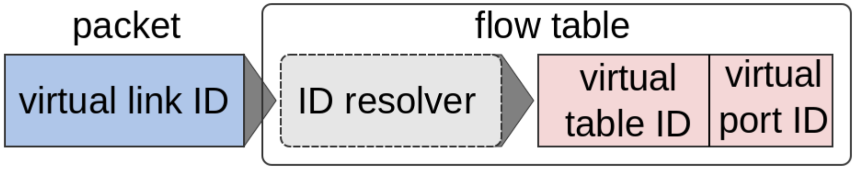
- Flow table segmentation



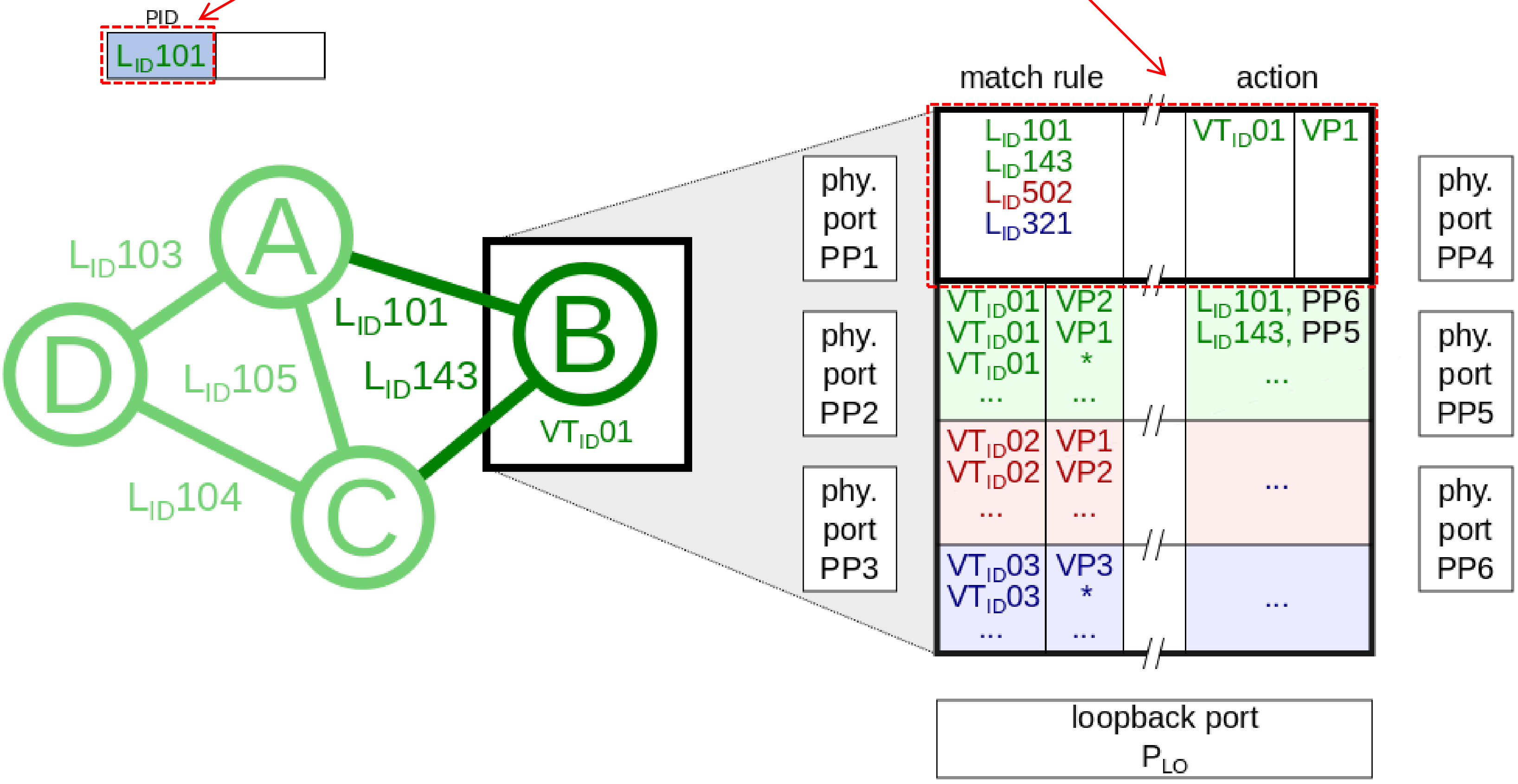
user-specified fields

- Link identifier (LID) assignment and resolution
  - Resolver table stored in the switch flow table
    - Simplified LID remapping for vSDN resource migrations





Packet Forwarding

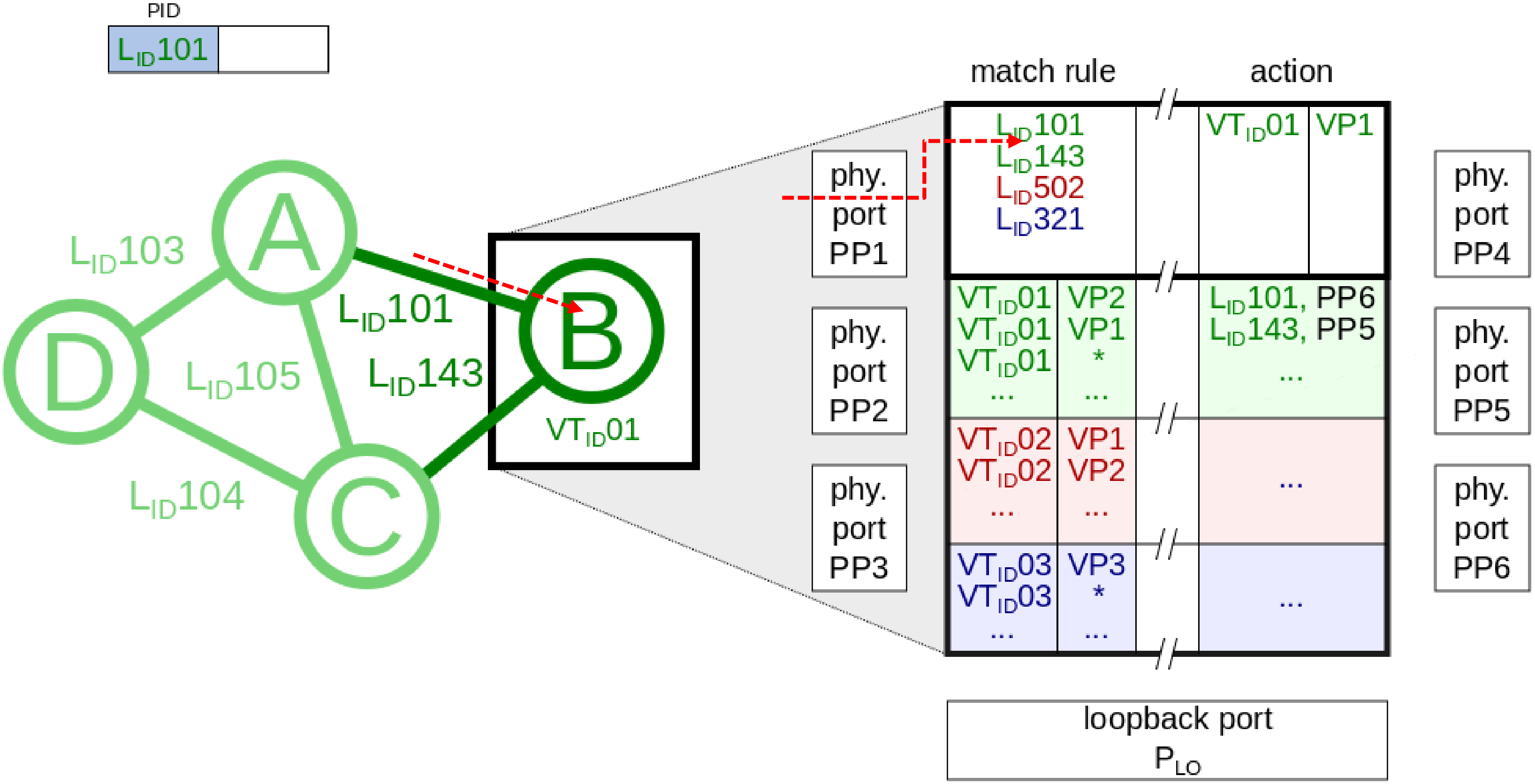


LID encoded in packet using MPLS/VLAN

resolver table

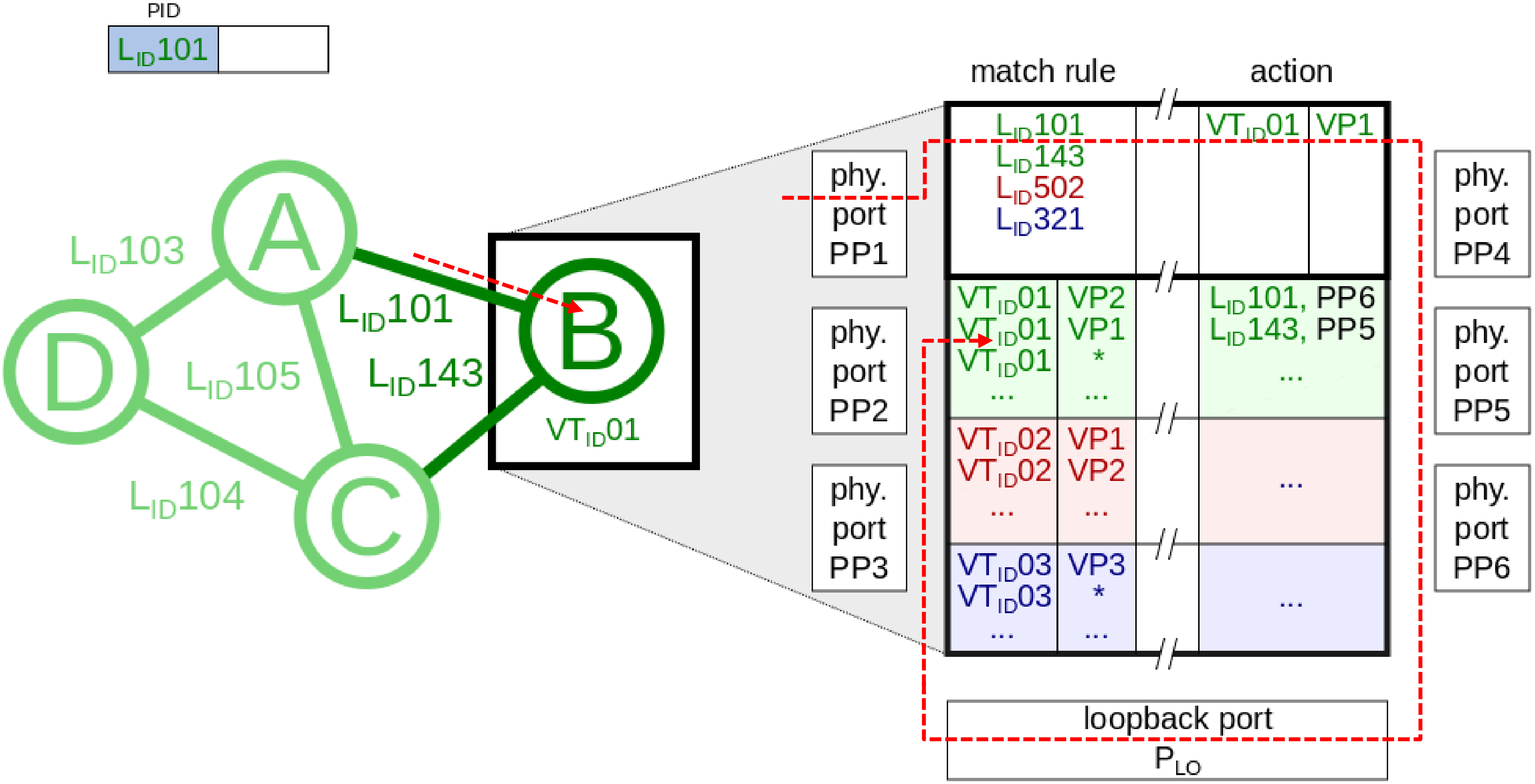
match rule		action	
$L_{ID}101$ $L_{ID}143$ $L_{ID}502$ $L_{ID}321$		$VT_{ID}01$	VP1
$VT_{ID}01$ $VT_{ID}01$ $VT_{ID}01$ ...	VP2 VP1 * ...	$L_{ID}101$, PP6 $L_{ID}143$, PP5 ...	
$VT_{ID}02$ $VT_{ID}02$ ...	VP1 VP2 ...	...	
$VT_{ID}03$ $VT_{ID}03$ ...	VP3 * ...	...	

Packet Forwarding



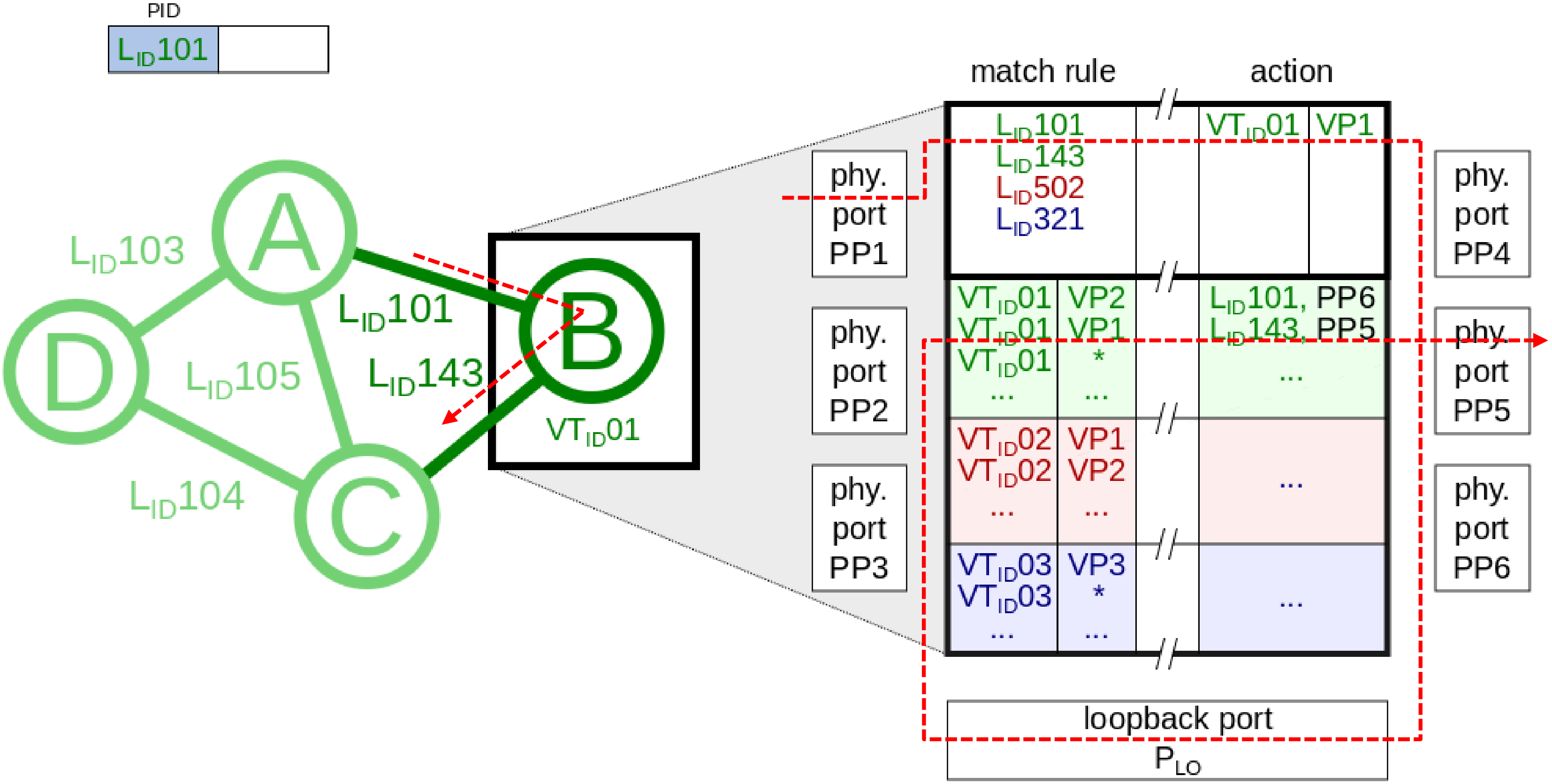
match rule		action	
$L_{ID}101$ $L_{ID}143$ $L_{ID}502$ $L_{ID}321$		$VT_{ID}01$	VP1
$VT_{ID}01$ $VT_{ID}01$ $VT_{ID}01$ …	VP2 VP1 * …	$L_{ID}101$, PP6 $L_{ID}143$, PP5 …	
$VT_{ID}02$ $VT_{ID}02$ …	VP1 VP2 …	…	
$VT_{ID}03$ $VT_{ID}03$ …	VP3 * …	…	

Packet Forwarding



match rule			action	
$L_{ID}101$ $L_{ID}143$ $L_{ID}502$ $L_{ID}321$			$VT_{ID}01$	VP1
$VT_{ID}01$ $VT_{ID}01$ $VT_{ID}01$ ...	VP2 VP1 * ...		$L_{ID}101$, PP6 $L_{ID}143$, PP5 ...	
$VT_{ID}02$ $VT_{ID}02$ ...	VP1 VP2 ...		...	
$VT_{ID}03$ $VT_{ID}03$ ...	VP3 * ...		...	

Packet Forwarding

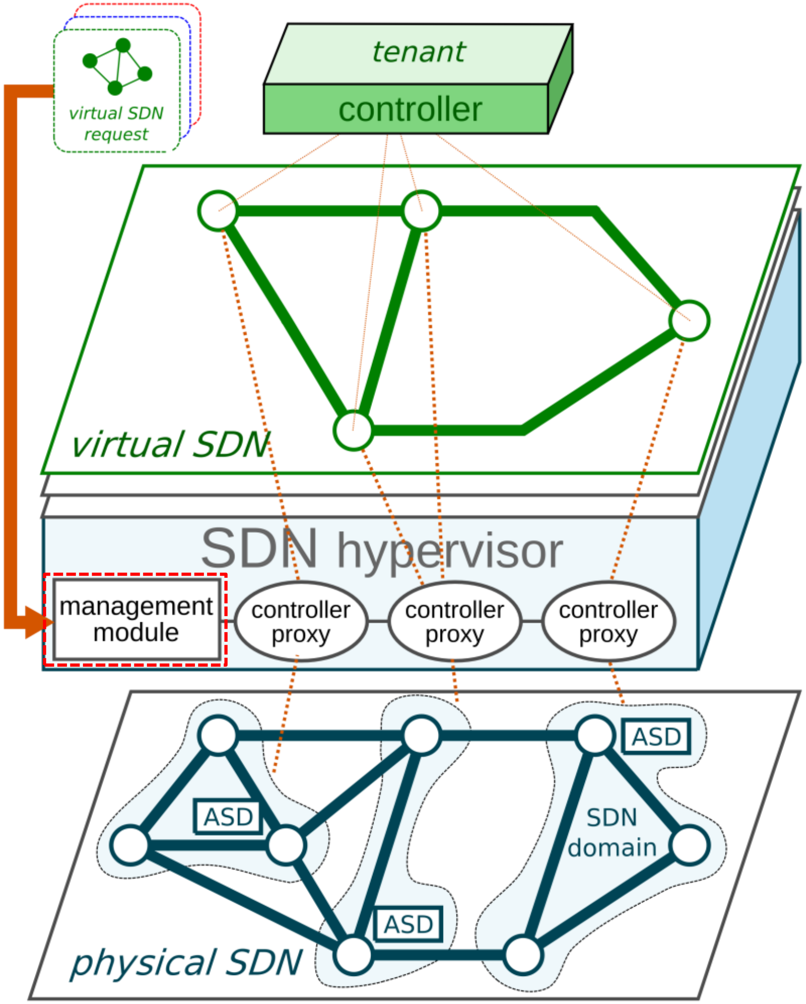


match rule		action	
$L_{ID}101$ $L_{ID}143$ $L_{ID}502$ $L_{ID}321$		$VT_{ID}01$	VP1
$VT_{ID}01$	VP2	$L_{ID}101$, PP6	
$VT_{ID}01$	VP1	$L_{ID}143$, PP5	
$VT_{ID}01$	*	...	
...	...		
$VT_{ID}02$	VP1	...	
$VT_{ID}02$	VP2		
...	...		
$VT_{ID}03$	VP3	...	
$VT_{ID}03$	*		
...	...		

- Easier to deploy with multiple flow tables

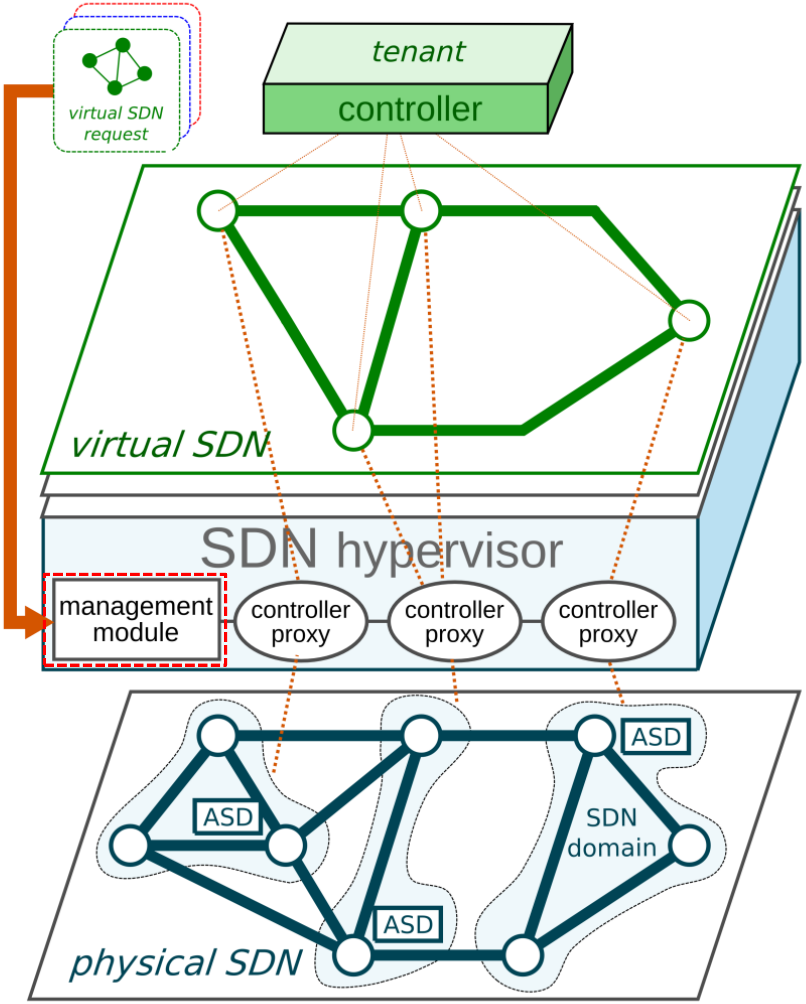
Virtual SDN Setup

1. vSDN topology mapping



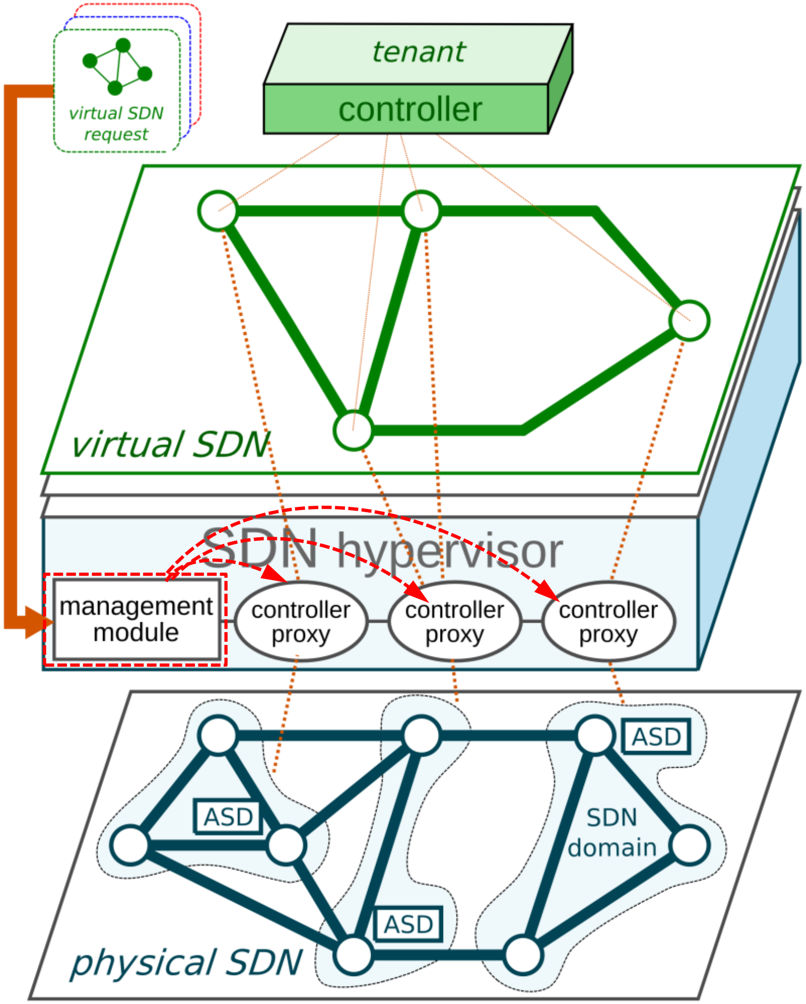
Virtual SDN Setup

1. vSDN topology mapping
2. Selection of identifiers



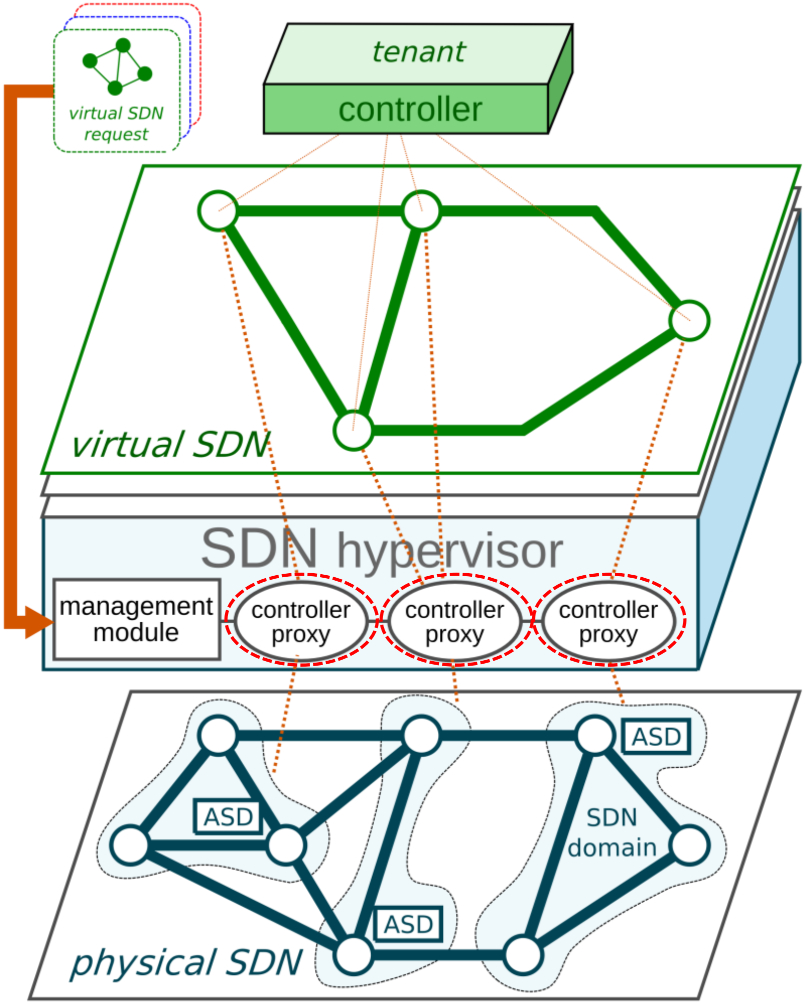
Virtual SDN Setup

1. vSDN topology mapping
2. Selection of identifiers
3. Assignment of vSDN resources and identifiers to corresponding CPXs



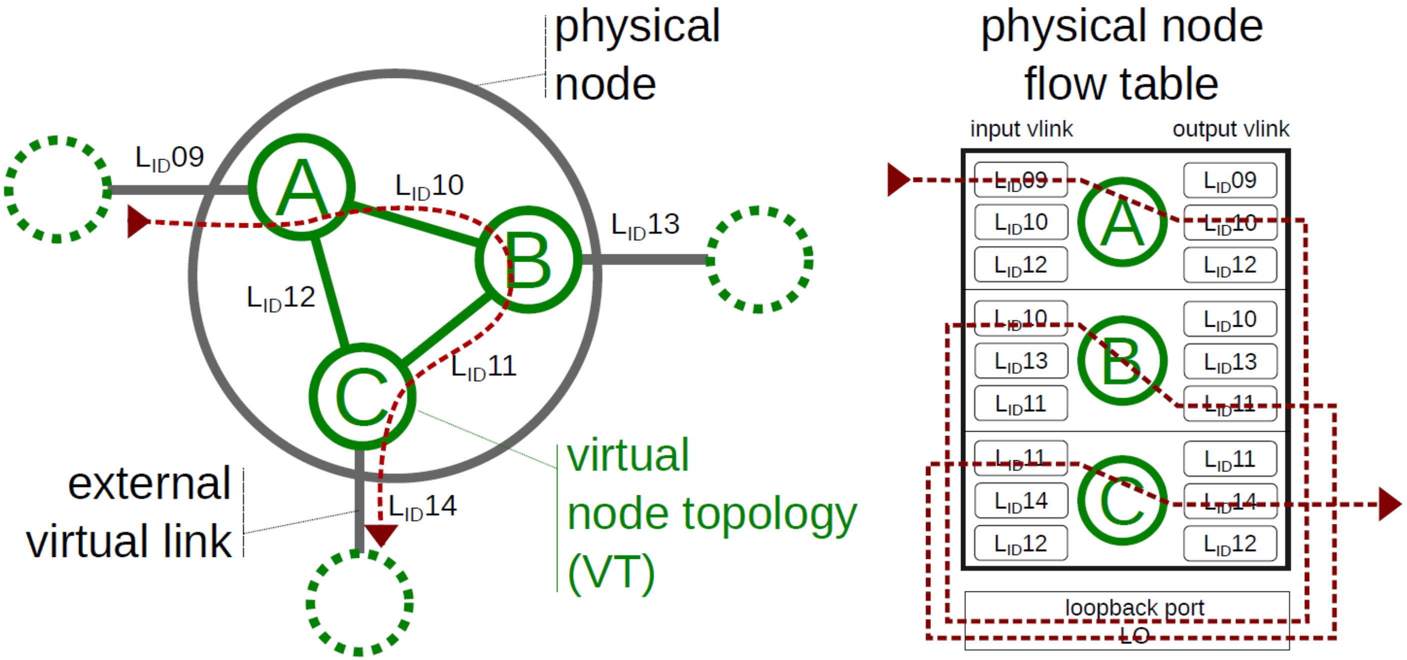
Virtual SDN Setup

1. vSDN topology mapping
2. Selection of identifiers
3. Assignment of vSDN resources and identifiers to corresponding CPXs
4. “Infrastructure” flow entry installation
    - Packet forwarding at intermediate nodes



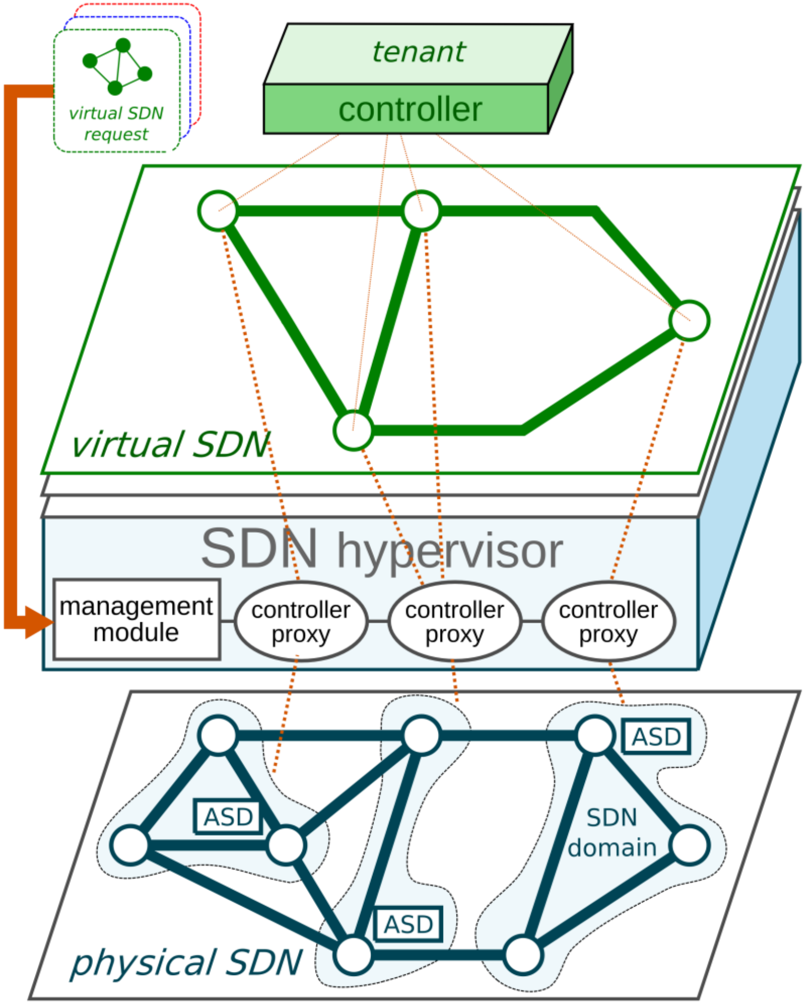
Arbitrary vSDN Topology Deployment

- Mapping multiple virtual switches onto the same physical switch
  - Multiple lookups on a single flow table using a loopback interface



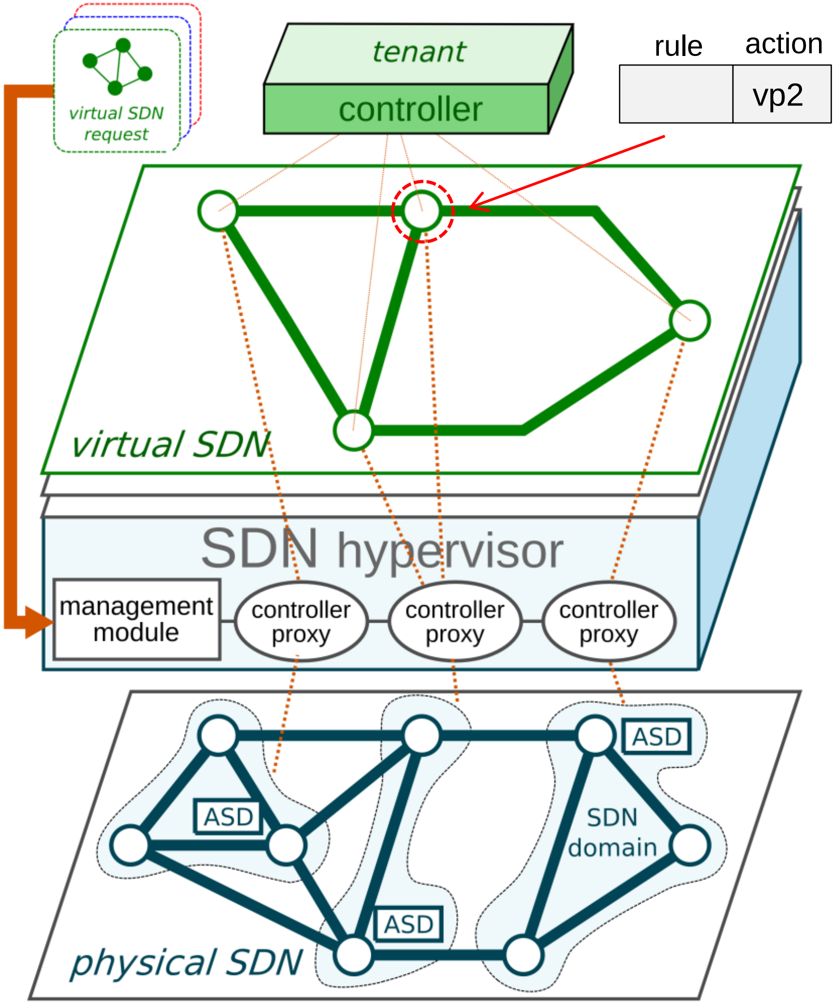
Control Message Translation

- Transparent translation of control messages:
    - Translation of references between logical and physical resource identifiers
    - Policy control to prevent access to unauthorized vSDN resources



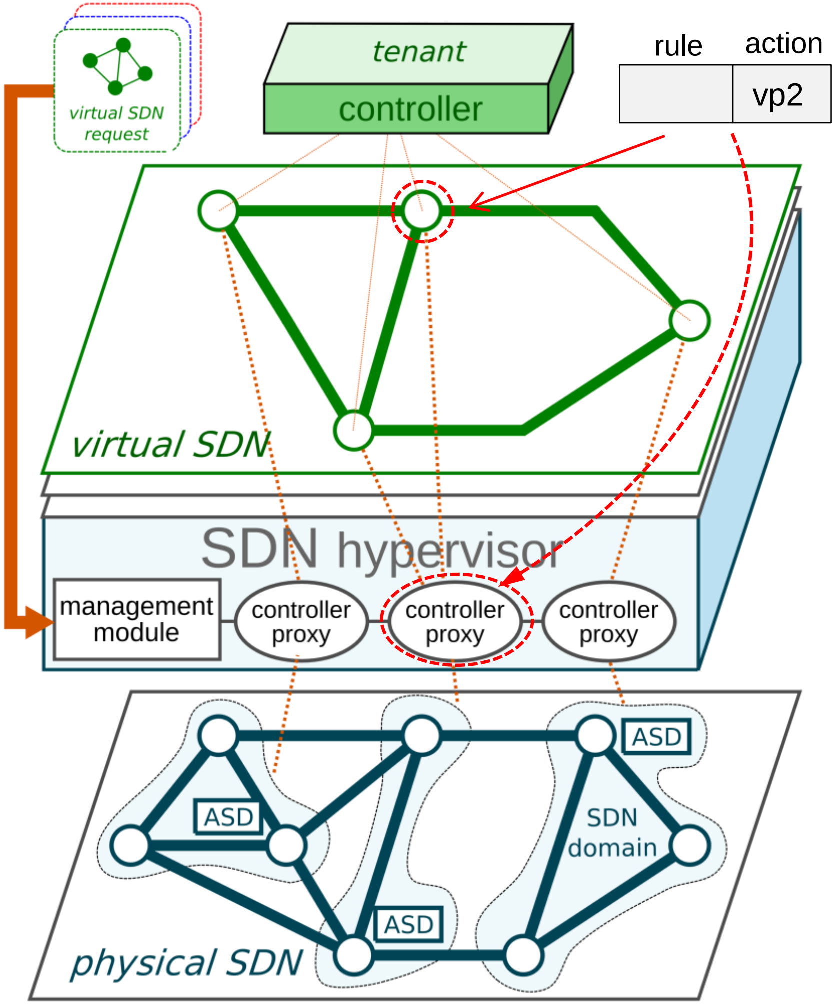
Control Message Translation

- Transparent translation of control messages:
  - Translation of references between logical and physical resource identifiers
  - Policy control to prevent access to unauthorized vSDN resources



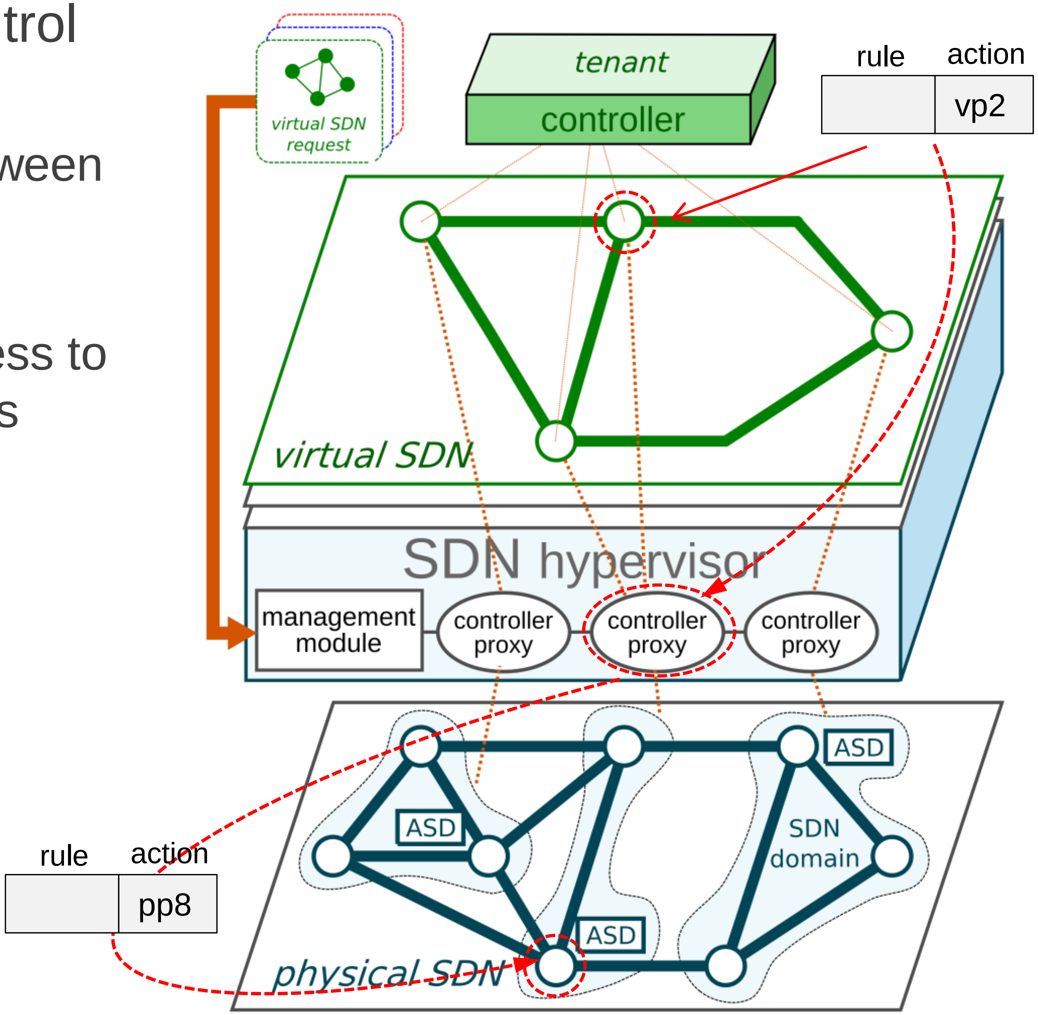
Control Message Translation

- Transparent translation of control messages:
  - Translation of references between logical and physical resource identifiers
  - Policy control to prevent access to unauthorized vSDN resources



Control Message Translation

- Transparent translation of control messages:
  - Translation of references between logical and physical resource identifiers
  - Policy control to prevent access to unauthorized vSDN resources



SDN Hypervisor Requirements

- OpenFlow:
  - Multiple tags (e.g., VLAN/MPLS) for scalability
  - Arbitrary masking for VLAN and MPLS tags

- Switching hardware:
  - Loopback interfaces or multiple flow tables for mapping multiple virtual nodes onto a single switch
  - Multiple queues per port for bandwidth isolation

References

- P. Barham, et al., **Xen and the Art of Network Virtualization**, ACM SOSP 2003
- D. Farinacci, et al., **Generic Routing Encapsulation (GRE)**, RFC 2784, IETF, 2000
- N. Egi, et al., **Towards High Performance Virtual Routers on Commodity Hardware**, ACM CoNEXT 2008
- Z. Bozakov and P. Papadimitriou, **OpenVRoute: An Open Architecture for High-Performance Programmable Virtual Routers**, IEEE HPSR 2013
- G. Schaffrath, et al., **Network Virtualization Architecture: Proposal and Initial Prototype**, ACM SIGCOMM VISA 2009
- D. Dietrich, et al., **Multi-Domain Virtual Network Embedding with Limited Information Disclosure**, IFIP Networking 2013
- P. Papadimitriou, et al., **Towards Large-Scale Network Virtualization**, IFIP WWIC 2012
- Z. Bozakov and P. Papadimitriou, **Towards a Scalable SDN Virtualization Platform**, IFIP/IEEE SDNMO 2014