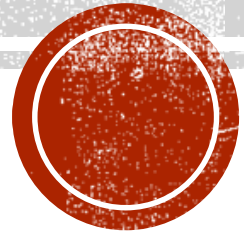
SOFTWARE DEFINED VIRTUAL NETWORKING SECURITY

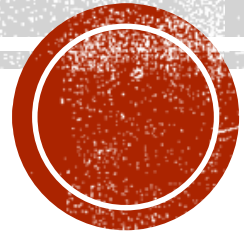
CHAPTER 2 VIRTUAL NETWORKING



Dijiang Huang, Ankur Chowdhary, and Sandeep Pisharody

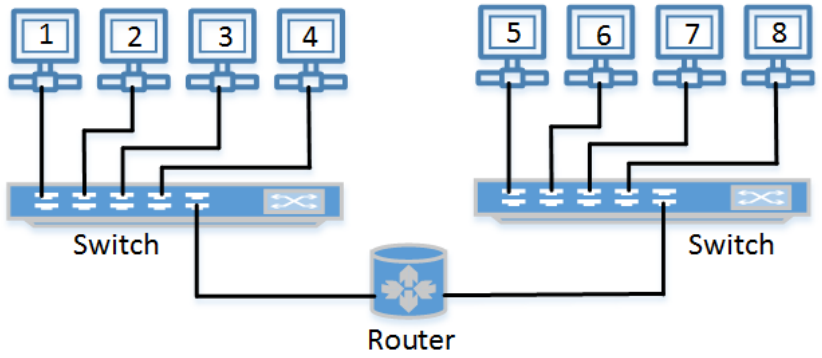
VIRTUAL NETWORKS

Physical, Logical, Virtual, and Overlay Networks

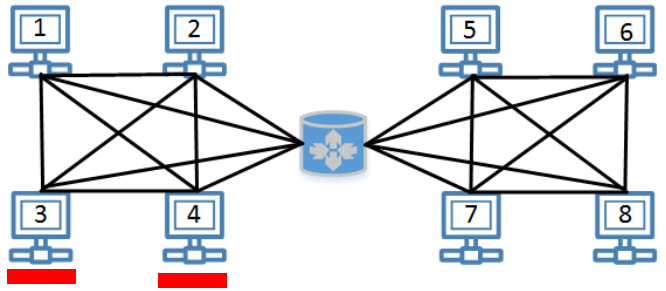


PHYSICAL/LOGICAL/VIRTUAL NETWORKS

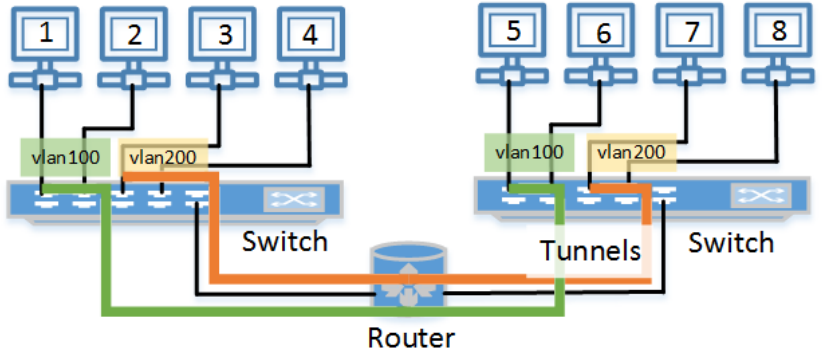
- Physical Network without virtual networks



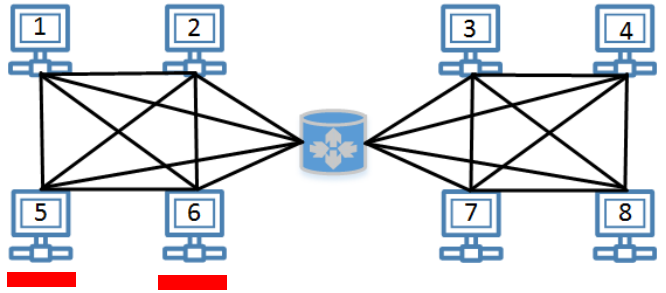
Layer-2 Logical View



- Physical Network with Layer-2 Tunnels (e.g., GRE/VxLAN)

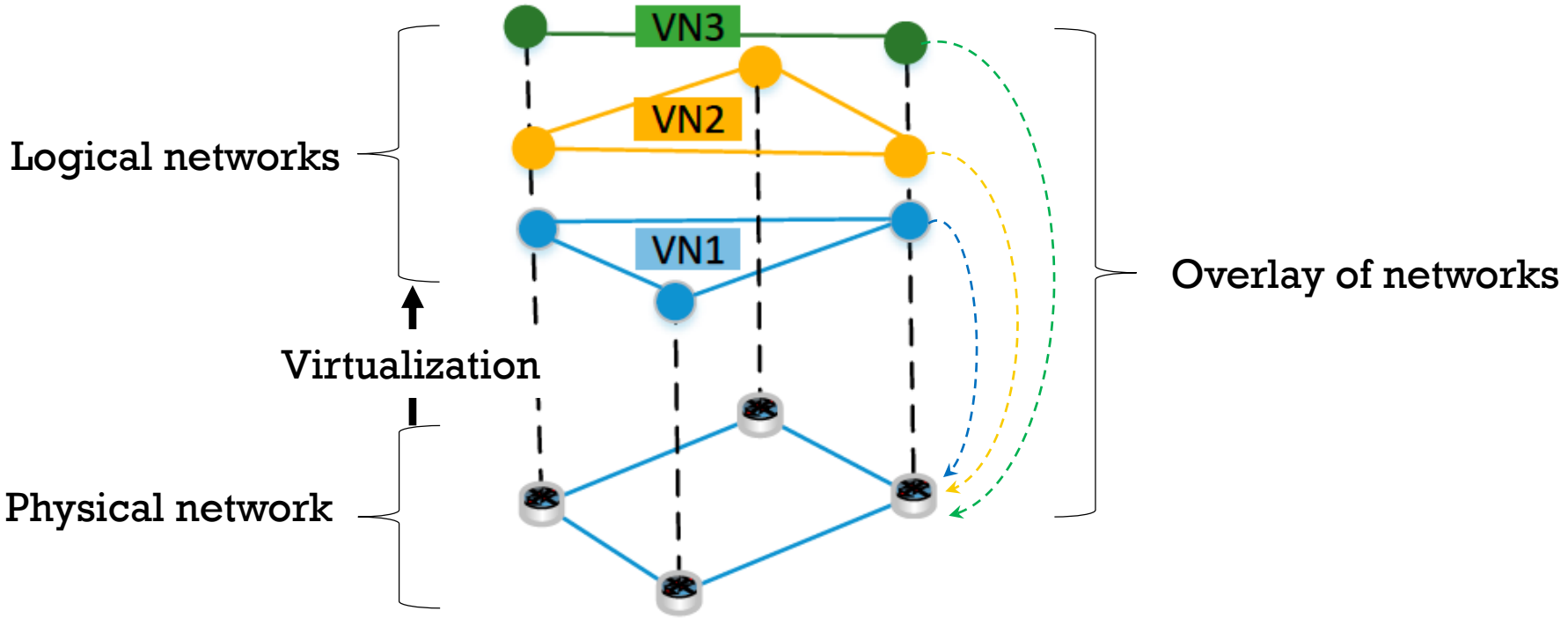


Layer-2 Logical View



PHYSICAL, LOGICAL, VIRTUAL, AND OVERLAY

- Network Virtualization focuses on creating a overplayed networking solutions to logically isolate multiple networks that are physically sharing the same set or subset of networking resources.

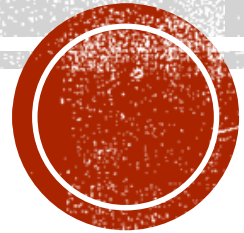


BENEFIT OF VIRTUALIZING NETWORKS

- **Resource optimization**: Tunnels and virtual networks can isolate or restrict network traffic based on the application needs, which providers in-network control capability compared to traditional best effort computer networking solutions.
- **Multiple execution environments**: Virtual networks enable an isolated multiple application running environment that can bring multi-faceted benefits such as isolating or restricting malicious network traffic, providing redundancy for backup, load sharing features, etc.
- **Debugging and intrusion detection**: Slicing is an isolation approach to reserve certain amount of networking resources for a particular application.
- **Mobility**: Virtual networking provides life-migration support for continuous data/service availability.
- **Appliance (security)**: Security functions plug-and-play feature and working as appliances that can be easily enabled and disabled, and applied at any network segment.
- **Testing/Quality assurance**: Virtual networks offer software developers isolated, constrained, test environments.

VIRTUAL NETWORKS

Orchestration, Management, and Virtual Network Embedding



ORCHESTRATION AND MANAGEMENT OF VIRTUAL NETWORKS

- A virtual assembly process needs to be included into operations practices, which describe how abstract services, features and even devices are realized on real infrastructure.
- Business support systems and network management systems may be merging into a cloud-related operations model
- Software Defined Networking (SDN) is an approach virtual network management.
  - Isolation of control and data planes
  - Build-in programmability

VIRTUAL NETWORKING EMBEDDING (VNE) PROBLEMS

- How to optimally allocate virtual networks and their associated networking resources is called Virtual Network Embedding (VNE) problem, which is an NP hard problem.
- VNE deals with the allocation of virtual resources both in nodes and links. Therefore, it can be divided in two sub-problems:
  - *Virtual Node Mapping (VNoM)*, where virtual nodes have to be allocated in physical nodes; and
  - *Virtual Link Mapping (VLiM)*, where virtual links connecting these virtual nodes have to be mapped to paths connecting the corresponding nodes in the substrate network.

VIRTUAL NETWORKING EMBEDDING (VNE) PROBLEMS - CONTINUE

- How the virtualized resources should be realized by the substrate resources?
  - A virtual link with capacity w cannot be mapped to a path containing a substrate link with capacity v, where $w>v$.
  - The CPU power requested by a virtual node has to be less than (or equal to) the CPU power actually provided by a substrate node.
  - Virtual resources are first mapped to candidate substrate resources. Only if all virtual resources can be mapped, the entire network is then embedded and substrate resources are actually spent.

VNE FORMAL DEFINITION

- Notations

Term	Description
$SN = (N, L)$	SN is a substrate network, consisting of nodes N and links L
$VNR^i = (N^i, L^i)$	VNR^i denotes the i^{th} Virtual Network Request, consisting of nodes N^i and links L^i
$\dot{\mathrm{R}}=\prod_{j=1}^{m} R_j$	$\dot{\mathrm{R}}$ contains resource vectors for all resources $R_1, ..., R_m$
$cap : N \cup L \rightarrow \dot{\mathrm{R}}$	The function cap assigns a capacity to an element of the substrate network (either node or link)
$dem_i : N^i \cup L^i \rightarrow \dot{\mathrm{R}}$	The function demi assigns a demand to an element of VNR^i (either a node or a link)
$f_i : N^i \rightarrow N$	f_i is the function that maps a virtual node of VNR^i to a substrate node (VNoM)
$g_i : L^i \rightarrow SN' \subseteq SN$	g_i is the function that maps a virtual link of VNR^i to a path in the substrate network (VLiM)

VNE FORMAL DEFINITION

- VNoM (Node mapping function f_i)
  - $f_i: N^i \rightarrow N,$ for each VNR^i Such that

$$\forall n^i \in N^i:\ dem_i(n^i) \leq cap\left(f_i(n^i)\right)$$

$$and$$

$$\forall l^i \in L^i: \forall l \in g_i(l^i): dem_i(l^i) \leq cap(l)$$

- VLiM (Link mapping function g_i)
  - $g_i: L^i \rightarrow SN' \subseteq SN,$ for each VNR^i Such that

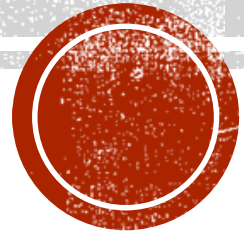
$$\forall n^i \in N^i:\ dem_i(n^i) \leq cap\left(f_i(n^i)\right)$$

$$and$$

$$\forall l^i \in L^i: \forall l \in g_i(l^i): dem_i(l^i) \leq cap(l)$$

- Solving the VNE problem is NP hard, thus heuristic approach is preferred.

VIRTUAL NETWORKS



Layer-2 Virtual Networking, Linux Bridge

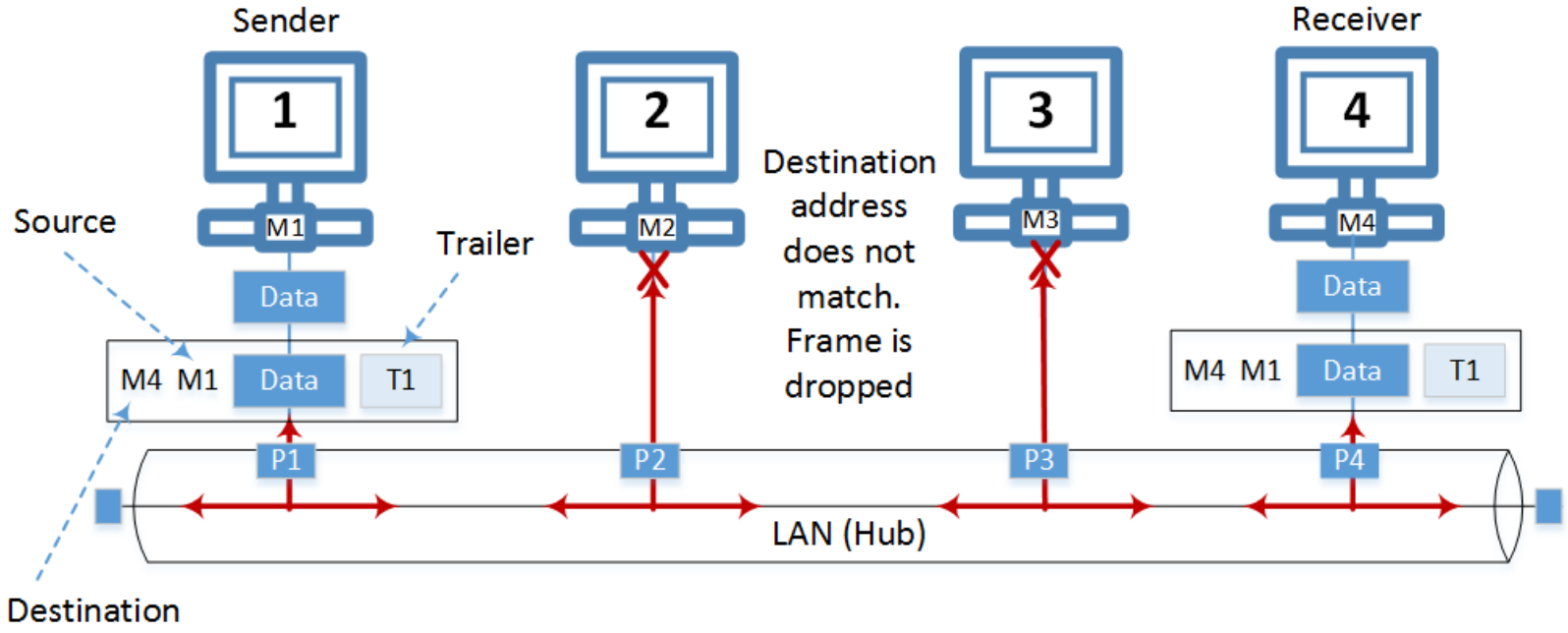
ETHERNET FRAME

802.3 Ethernet frame structure

Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interframe gap
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	42[note 2]–1500 octets	4 octets	12 octets
	← 64–1518 octets (68-1522 octets for 802.1Q tagged frames) →							
← 84–1538 octets (88-1542 octets for 802.1Q tagged frames) →								

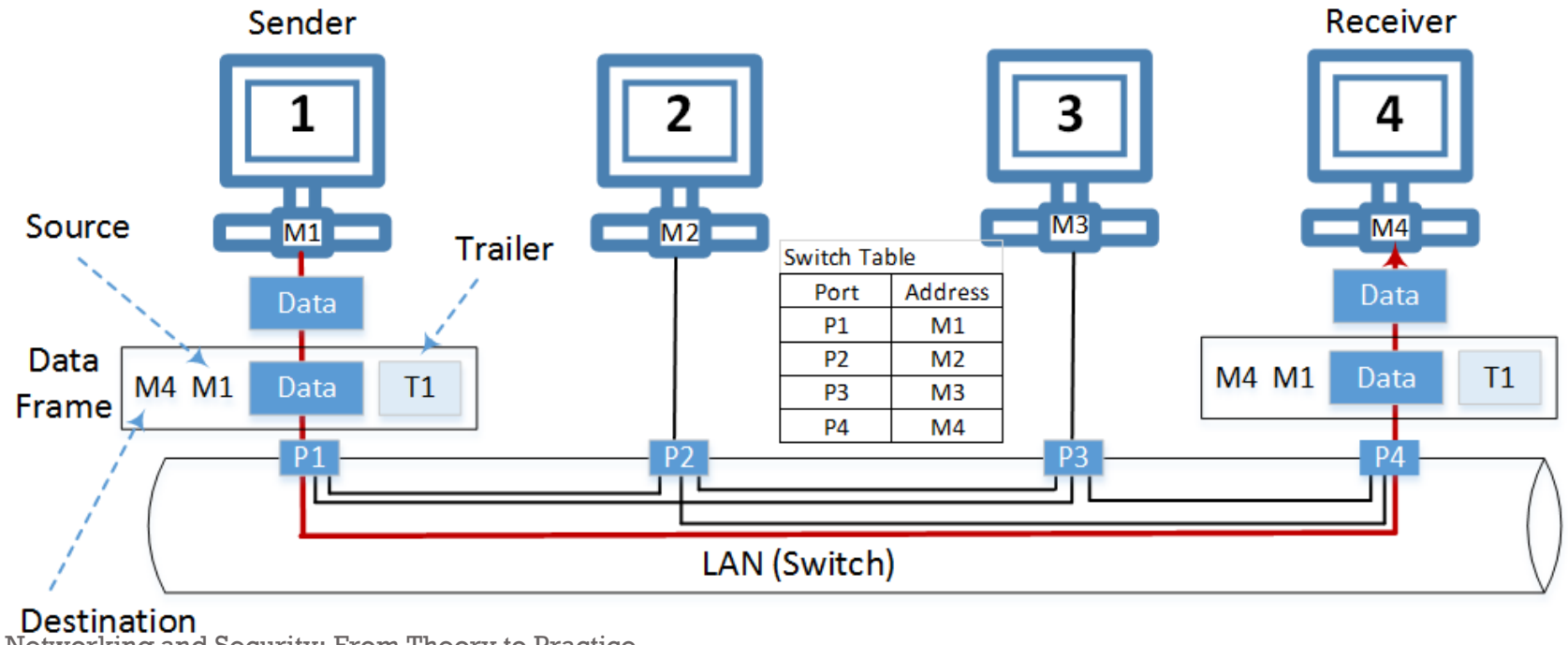
HUB, SWITCH, AND BRIDGE

- Hub is a physical-layer device, where a frame is passed along or broadcast to every port.



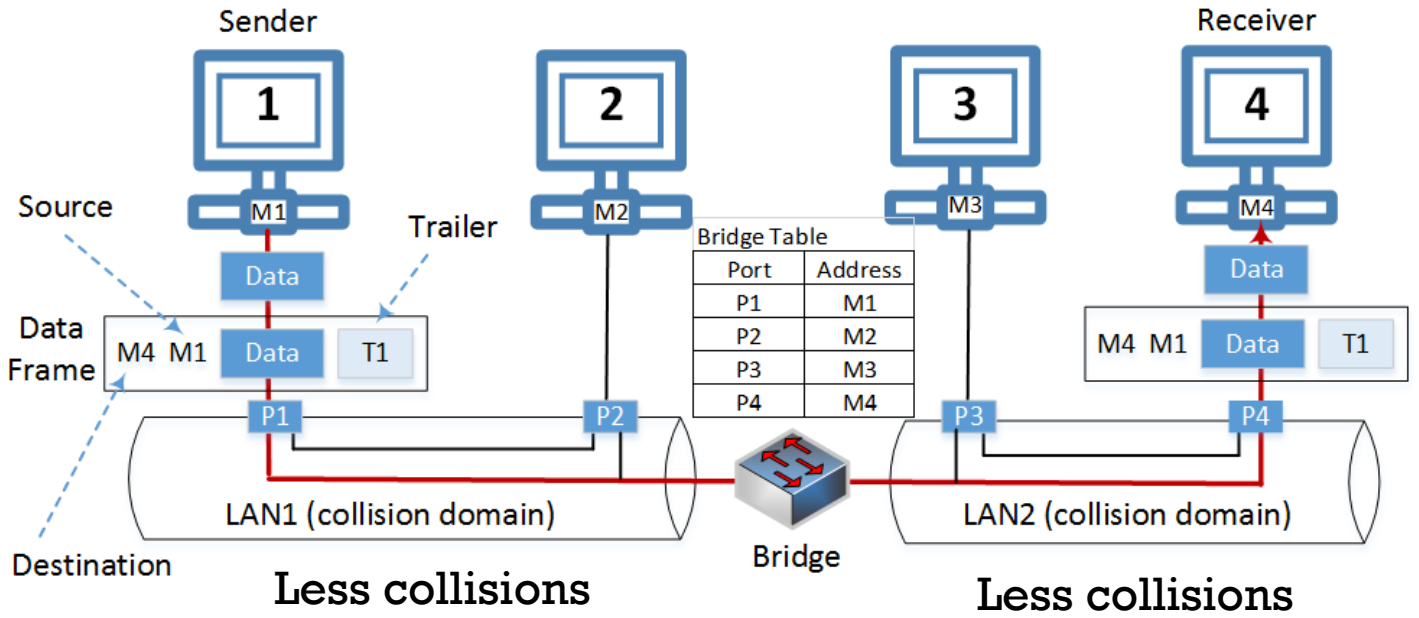
HUB, SWITCH, AND BRIDGE

- Switch is a data-link layer device, where it recodes all the MAC addresses of all the devices connected to it, then create a dedicated link when forward data from one port to another port.



HUB, SWITCH, AND BRIDGE

- Bridge is a data-link layer device, where it usually connects two LANs to reduce the size of collision domains when the data-link protocol share the communication media such as Ethernet. A switch can be viewed as a multiport bridge.

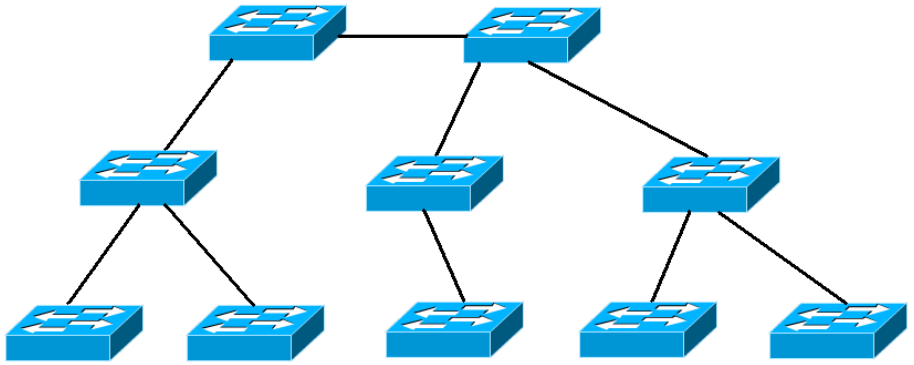


LINUX BRIDGE

- Four components of a switch/bridge
  - A set of network ports
  - A control plane
  - A forwarding plane
  - A MAC learning database
- Linux bridge use case examples
  - Create a bridge
    - *strace brctl addbr br1*
  - Adding an interface
    - *strace brctl addif br0 beth0*

SPANNING TREE PROTOCOL

- Used by switches to turn a redundant topology into a spanning tree
- Disables unwanted links by blocking ports
- STP defined by IEEE 802.1d
- Rapid STP defined by IEEE 802.1w
- Switches run STP by default – no configuration needed.

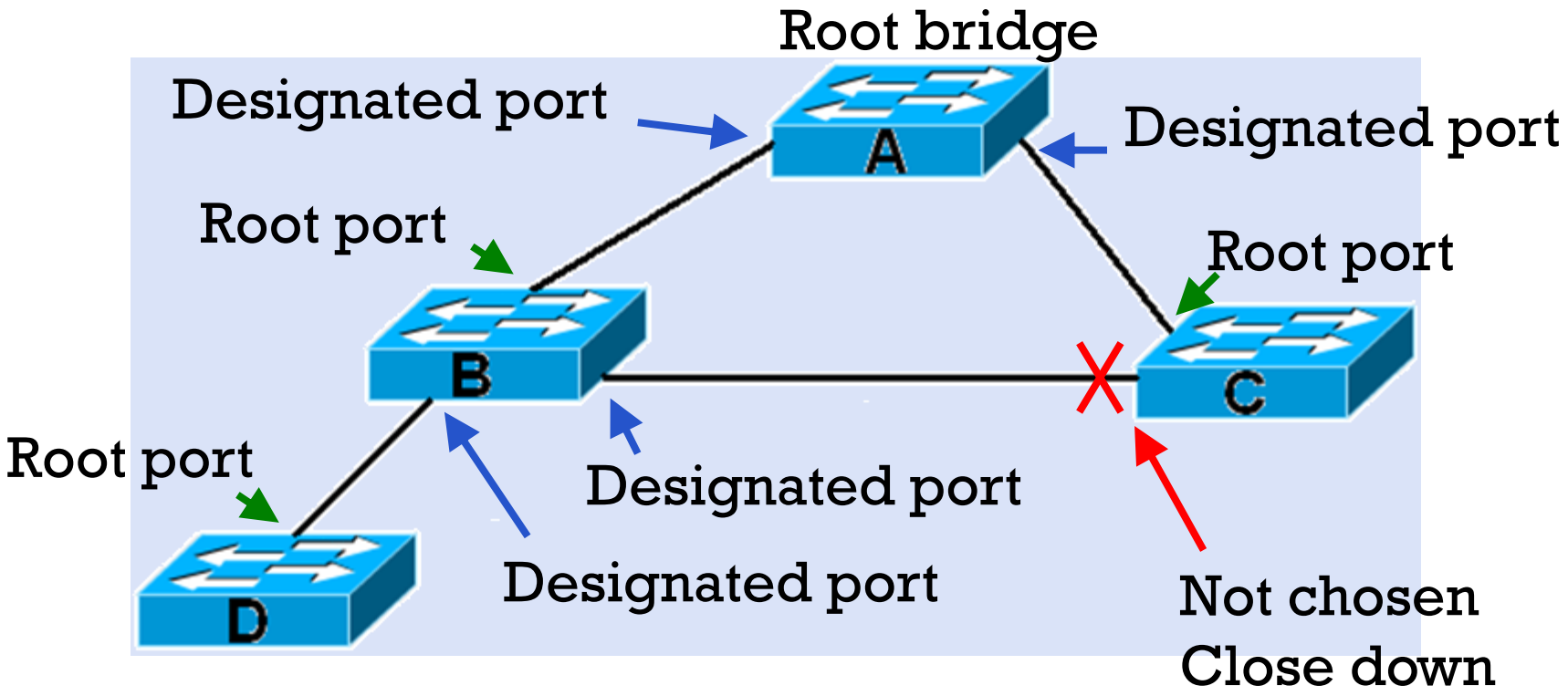


SPANNING TREE ALGORITHM

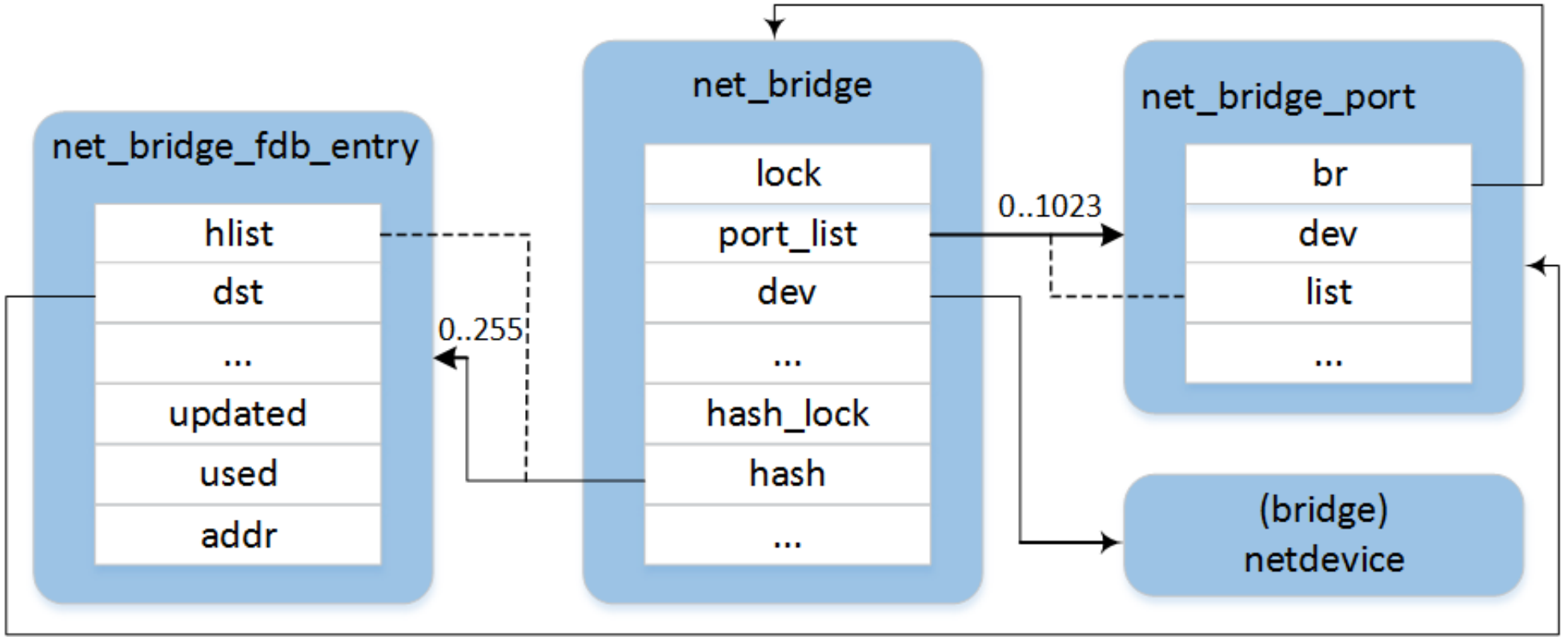
The switches use this algorithm to decide which ports should be shut down.

1. Choose one switch to be "root bridge"
2. Choose a "root port" on each other switch
3. Choose a "designated port" on each segment.
4. Close down all other ports.

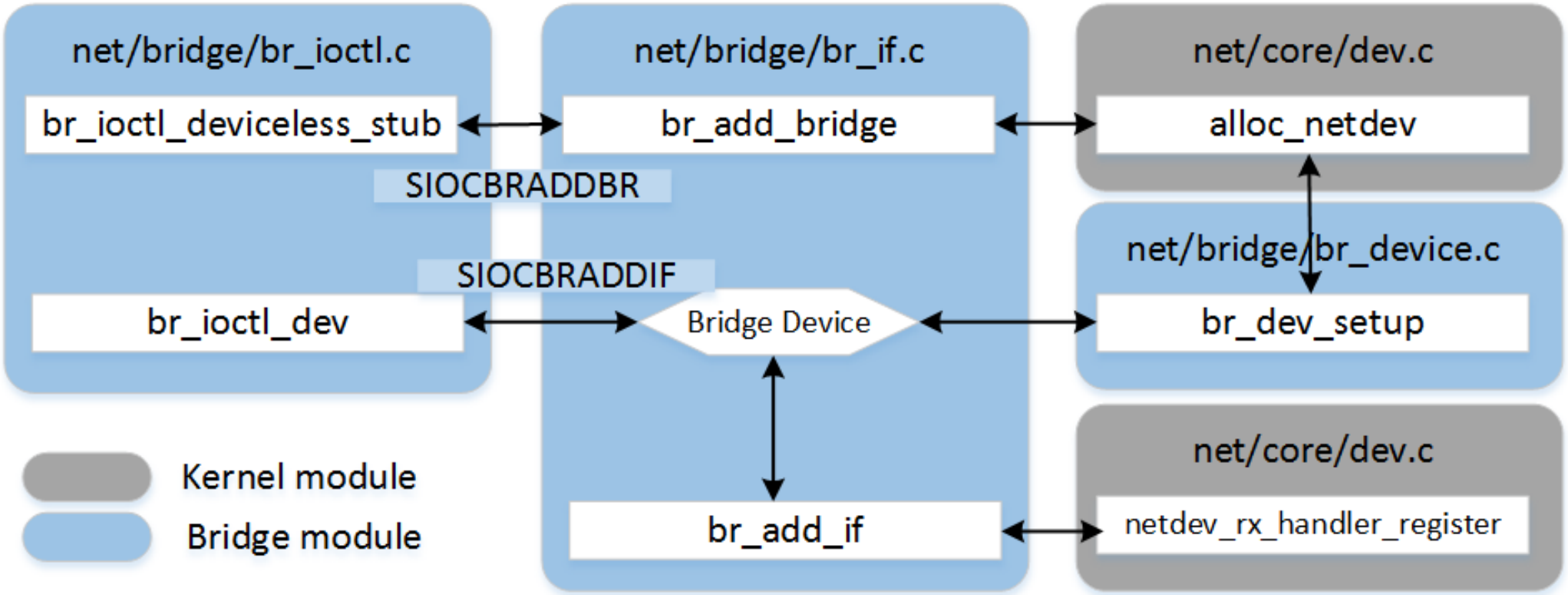
OUTLINE OF PROCESS



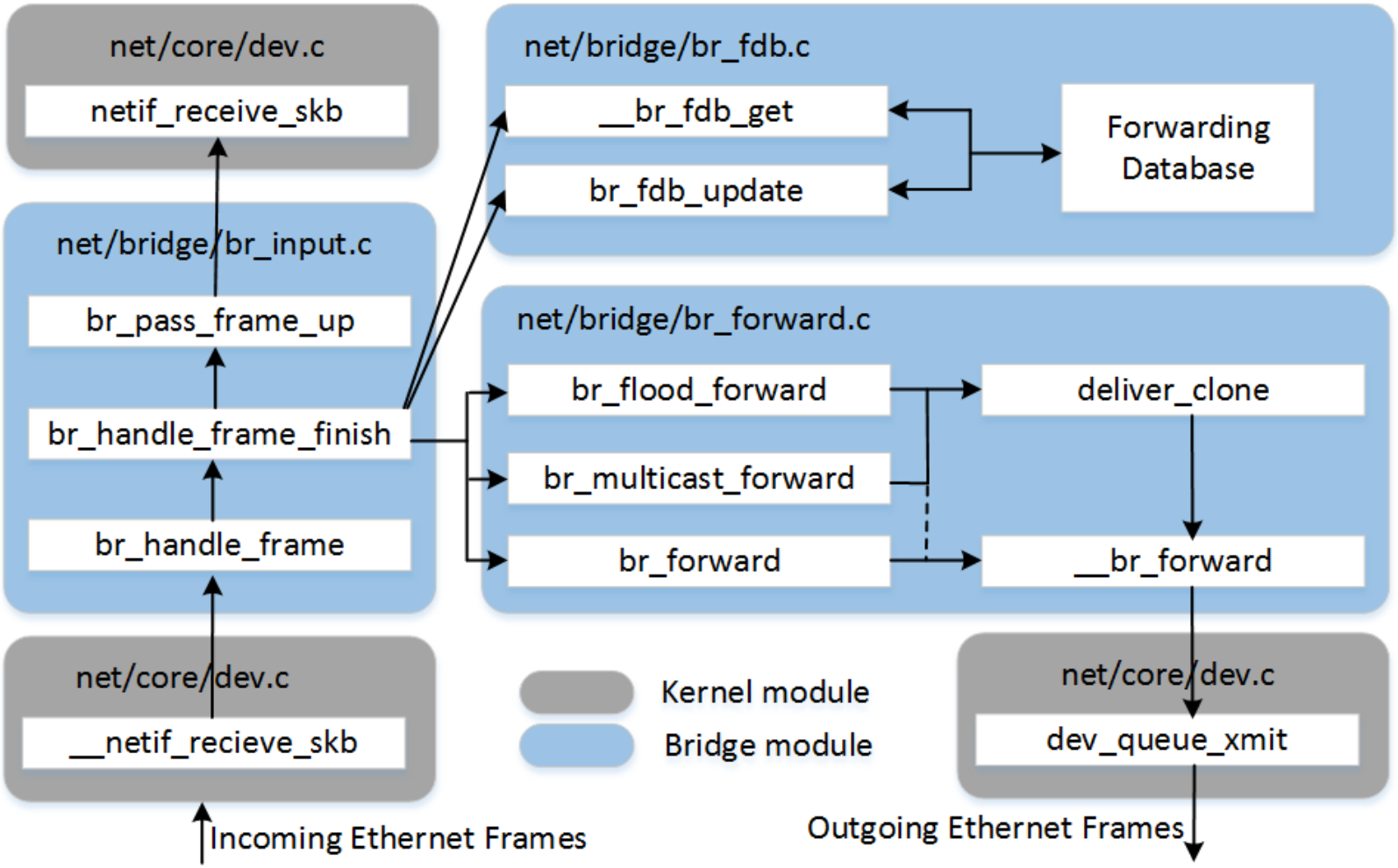
LINUX BRIDGE DATA STRUCTURE



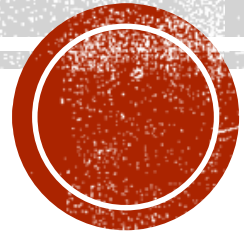
LINUX BRIDGE CONFIGURATION



LINUX BRIDGE FRAME PROCESSING



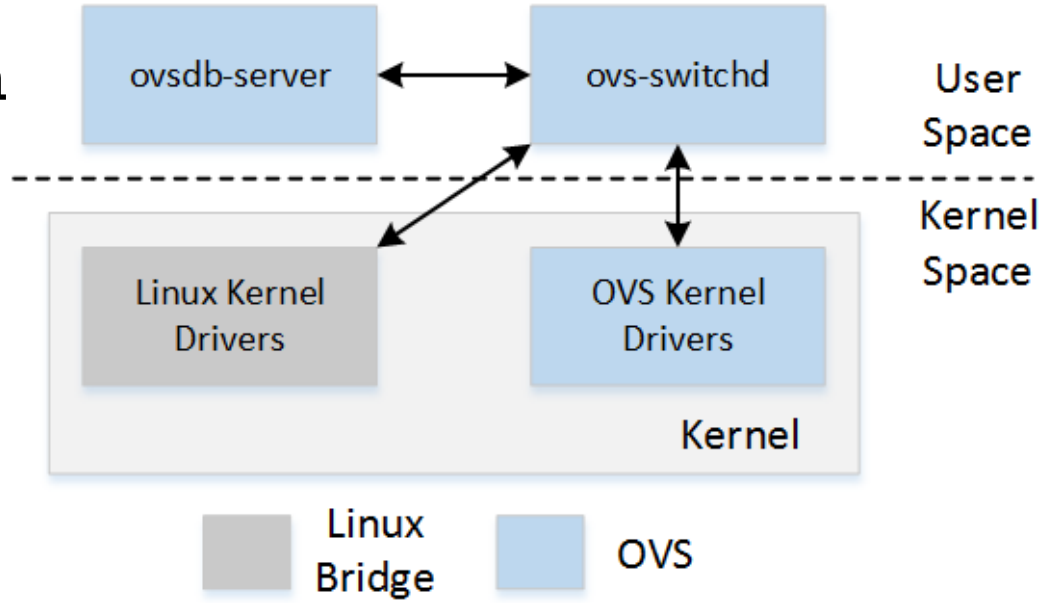
VIRTUAL NETWORKS



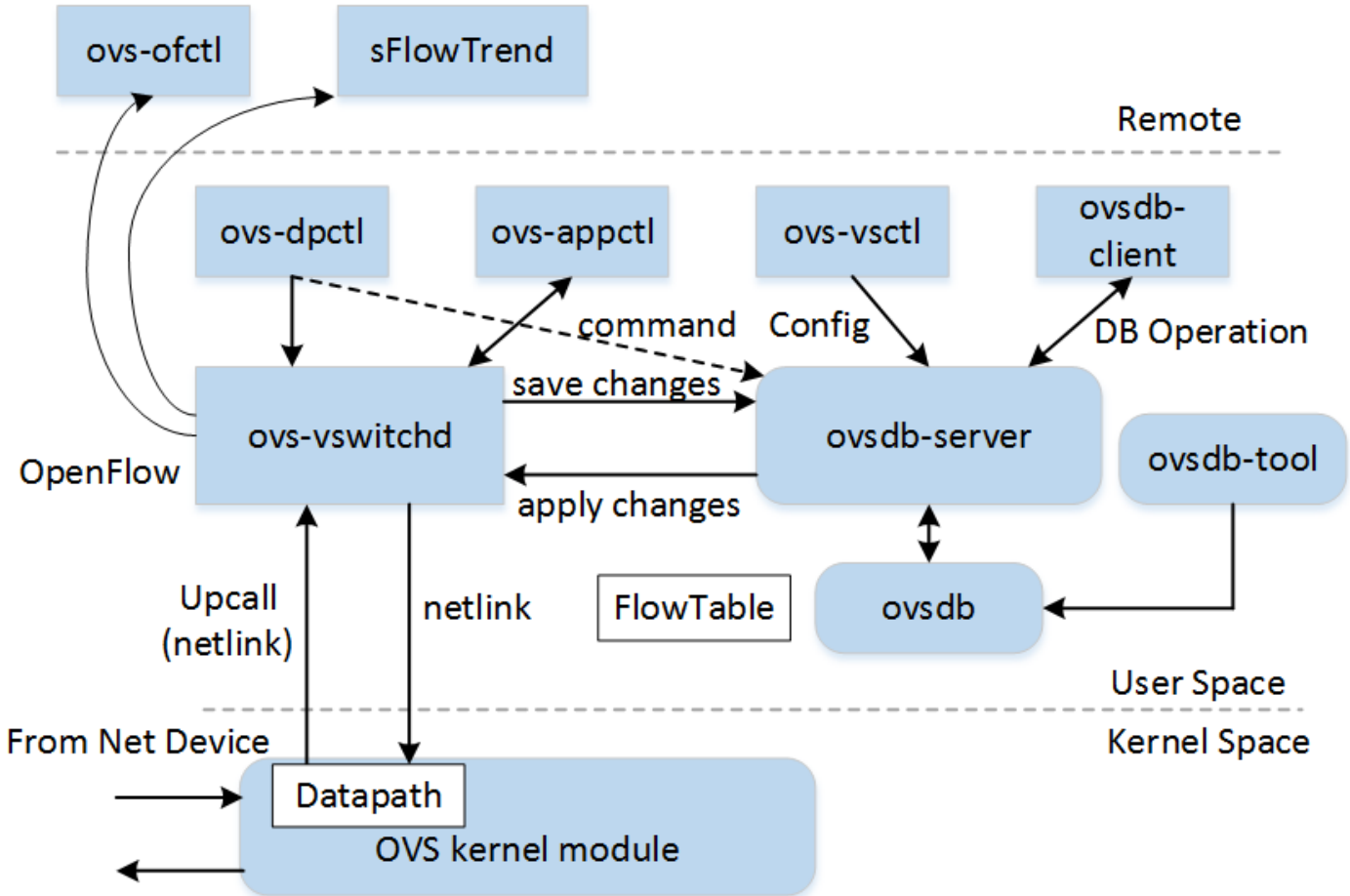
Open Virtual Switch (OVS)

OPEN VIRTUAL SWITCH (OVS)

- Open vSwitch (OVS) is a multilayer software switch licensed under the open source Apache 2 license.
- It was designed to support distribution across multiple physical servers. OVS supports multiple Linux-based virtualization technologies including Xen/XenServer, KVM, and VirtualBox.



OVS INTERNAL MODULES

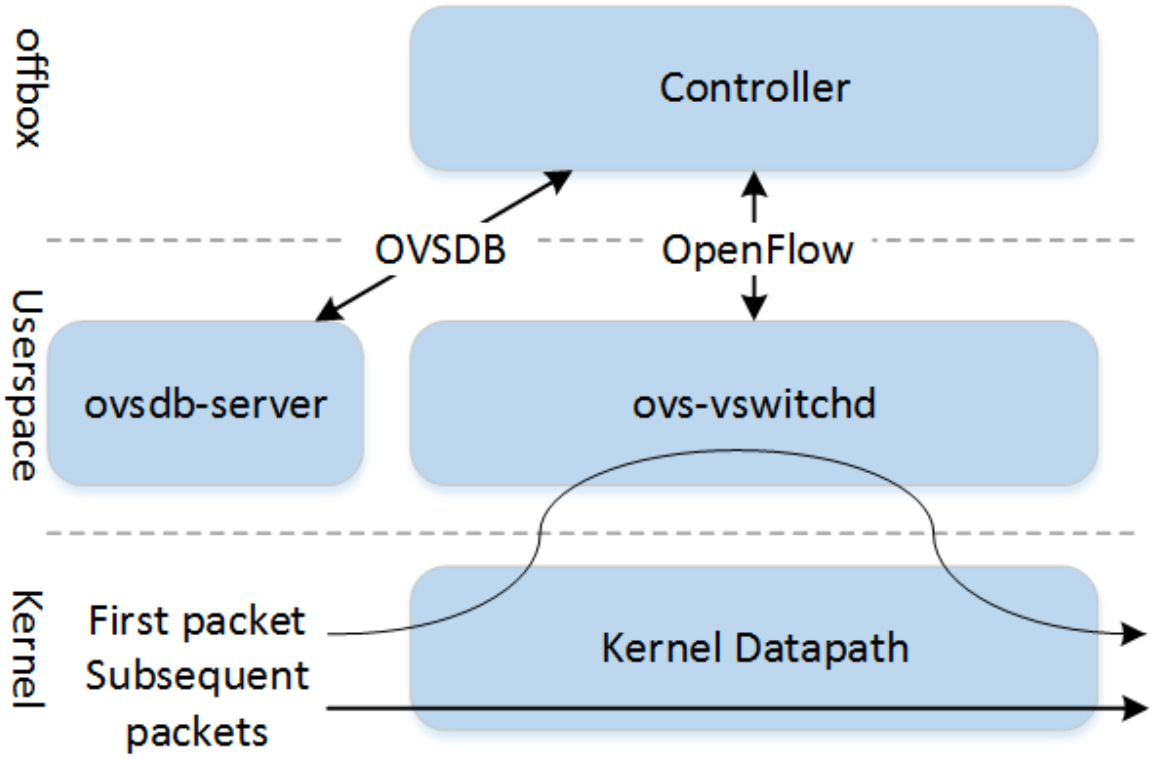


OVS INTERNAL MODULES (CONTINUE)

- *ovs-vswitchd*, a daemon that implements the switch, along with a companion Linux kernel module for flow-based switching. It talks to the kernel module through *netlink*}protocol. *ovs-vswitchd* saves and changes the switch configuration into a database and talks to *ovsdb-server* that manages *ovsdb*.
- *ovsdb-server*, a lightweight database server that *ovs-vswitchd* queries to obtain its configuration.
- *ovs-dpctl,* the OVS *datapath* management utility. It configures the switch kernel module.
- *ovs-vsctl*, a utility for querying and updating the configuration of *ovs-vswitchd*. It manages the switch through interaction with *ovsdb-server*.
- *ovs-appctl*, a utility that sends commands to running Open vSwitch daemons.
- *ovsdb*, persists the data across reboots; configures *ovs-vswitchd*.
- *OVS kernel module*, it is designed to be fast and simple, and implements tunnels and caches flows. It handles switching and tunneling without knowledge of Openflow. If a flow found, actions are executed otherwise the flow is passed to the user space.

PACKET PROCESSING IN OVS

- The *datapath module* in the kernel receives the packets first from the NIC
- If *ovs-vswitchd* has instructed the *datapath* how to handle packets, the *datapath module* transfers the packet accordingly.
- Or, the *datapath module* transfers the packet to *ovs-vswitchd* for how to handle the packet.



OVS COMMANDS

- Install OVS

  apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils

  apt-get install openvswitch-switch

- ovs-ofctl speaks to OpenFlow module

  ovs-ofctl show

  ovs-ofctl dump-flows

  ovs-ofctl add-flow

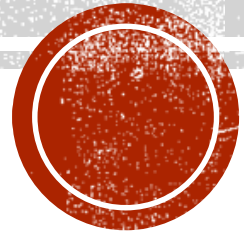
  ovs-ofctl del-flows [flow]

  ovs-ofctl snoop

- See "hidden" flows (in-band, fail-open, etc):

  ovs-appctl bridge/dump-flows

TUNNELING PROTOCOLS AND VIRTUAL PRIVATE NETWORKS (VPN)



VLAN

TUNNELING PROTOCOLS

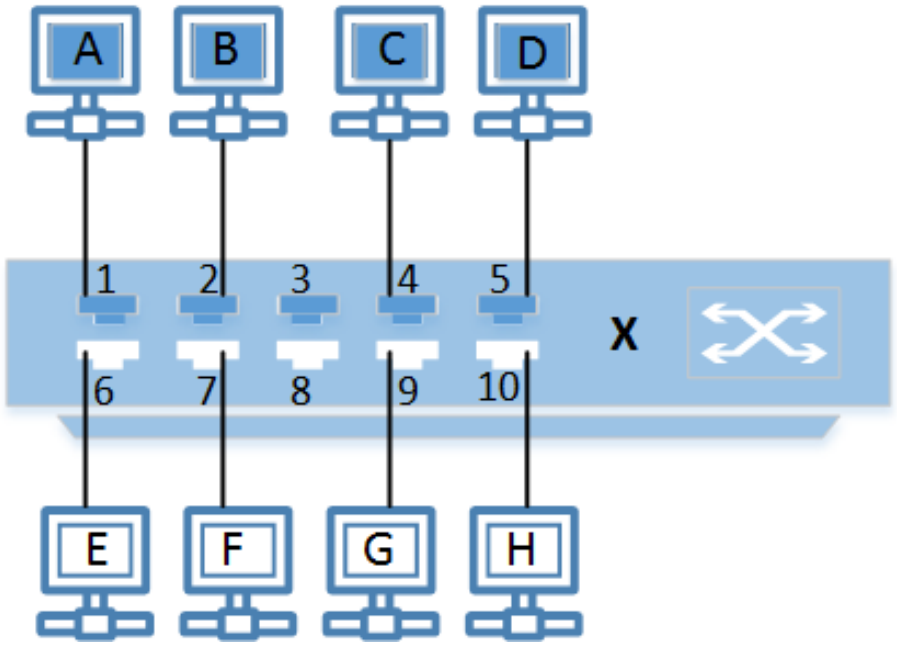
Protocol	Protocol Layer Over	Service Layer	Security	Description
L2TP	UDP	Data Link	Use with IPSec	Fast, scalable, efficient, and reliable layer-2 connections over Internet.
PPP	Point-to-Point	Data Link	Provided	Can be used over may types of physical networks, e.g., serial cable, phone line, trunk line, and provide layer-2 service to IP.
VLAN	Data Link	Data Link	N/A	Provide layer-2 overlay networks.
VXLAN	UDP	Data Link	N/A	Provide layer-2 VLAN connections over Internet.
GRE	IP	Any	N/A	Provide tunnels over IP such as ethernet, MPL, IP, etc.
SSL/TLS	TCP	Transport	Provided	Provide transportation layer VPN.
IPSec	IP	IP	Provided	Provide IP layer VPN.

VIRTUAL LOCAL AREA NETWORKS (VLANS)

- VLANs divide a single existing physical network into multiple logical networks, in which VLANs are layer-2 overlay networking technologies.
- Two types of VLANs:
  - Port-based VLANs (untagged)
  - Tagged VLANs

PORT-BASED VLANS

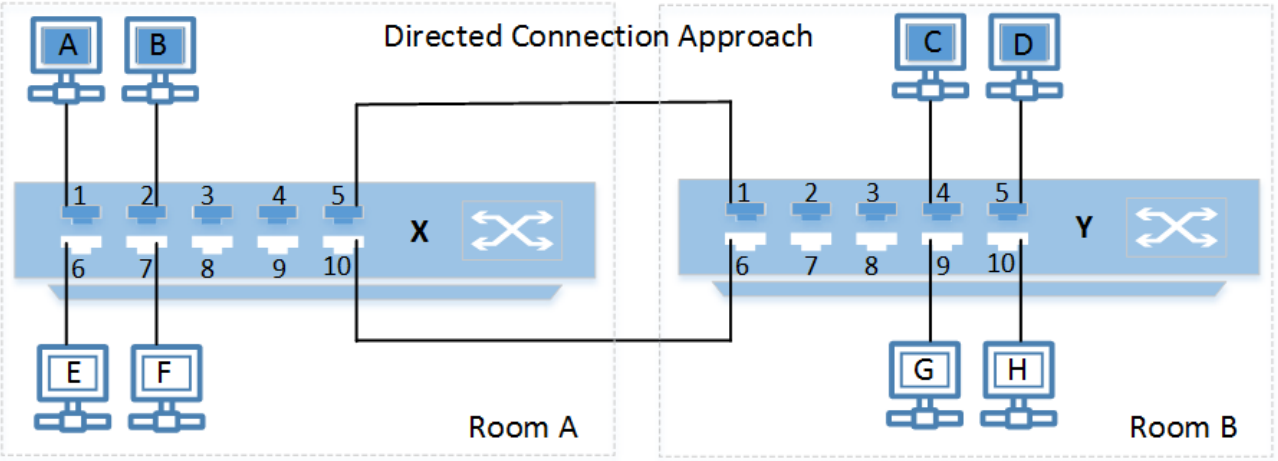
- A single physical switch X is simply divided into multiple logical switches. E.g., the following 10-port switch is divided into two logical switches (VLAN 1 and VLAN 2).



Port	VLAN ID	Connected Device
1	1	A
2		B
3		not used
4		C
5		D
6	2	E
7		F
8		not used
9		G
10		H

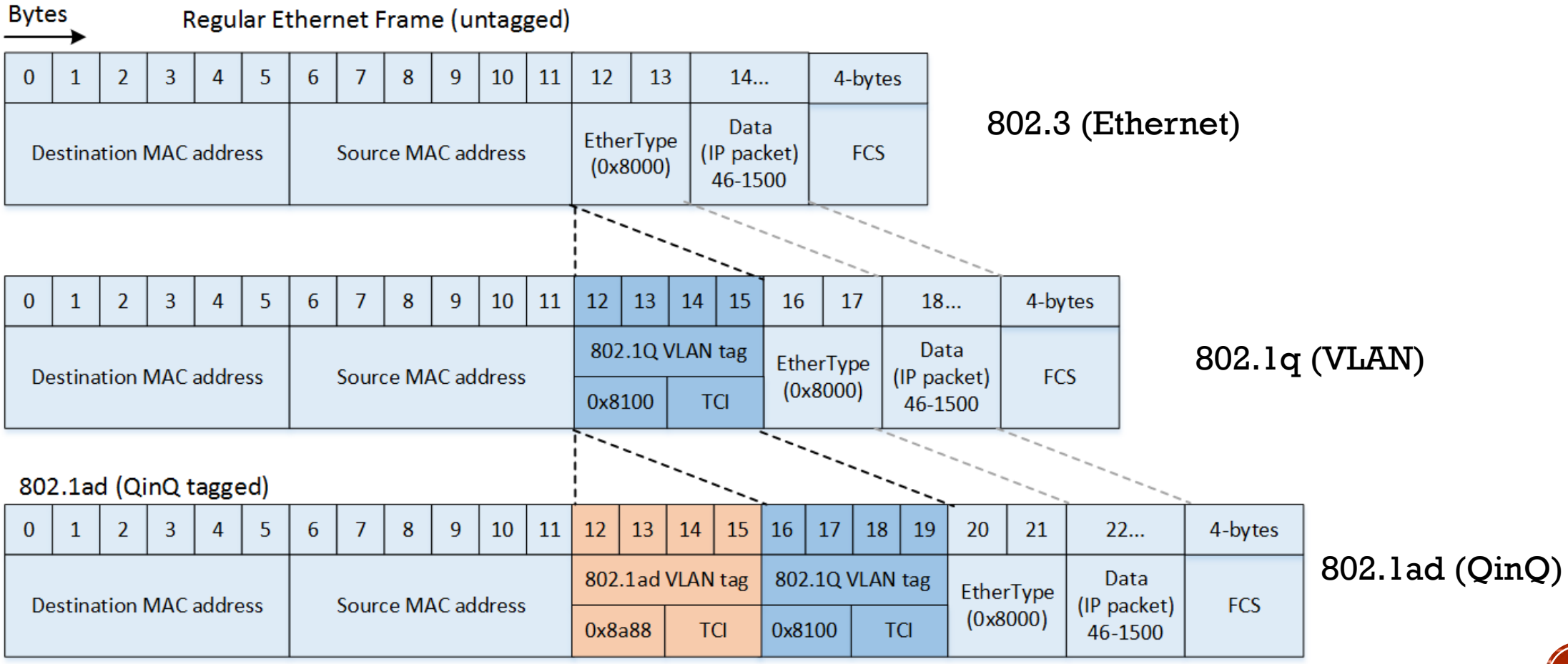
PORT-BASED VLAN ASSIGNMENT FOR TWO SWITCHES

- Use physical cables to connect two logical switches (VLANs).



Port	VLAN ID	Connected Device	Port	VLAN ID	Connected Device
1		A	1		Direct connection
2		B	2		not used
3	1	not used	3	1	not used
4		not used	4		C
5		Direct connection	5		D
6		E	6		Direct connection
7		F	7		not used
8	2	not used	8	2	not used
9		not used	9		G
10		Direct connection	10		H

VLAN TAGGING



TAG, TRUNK PORT, AND PORT CONFIGURATION

- Using tags to identify which logical switch a port belongs to.
- Using a single link to connect two ports on two switches, in which the link is called trunk-link and it forwards all data frames between two switches, thus allowing tagged traffic to be forwarded to the same VLAN physically residing on different switches.
- Port states

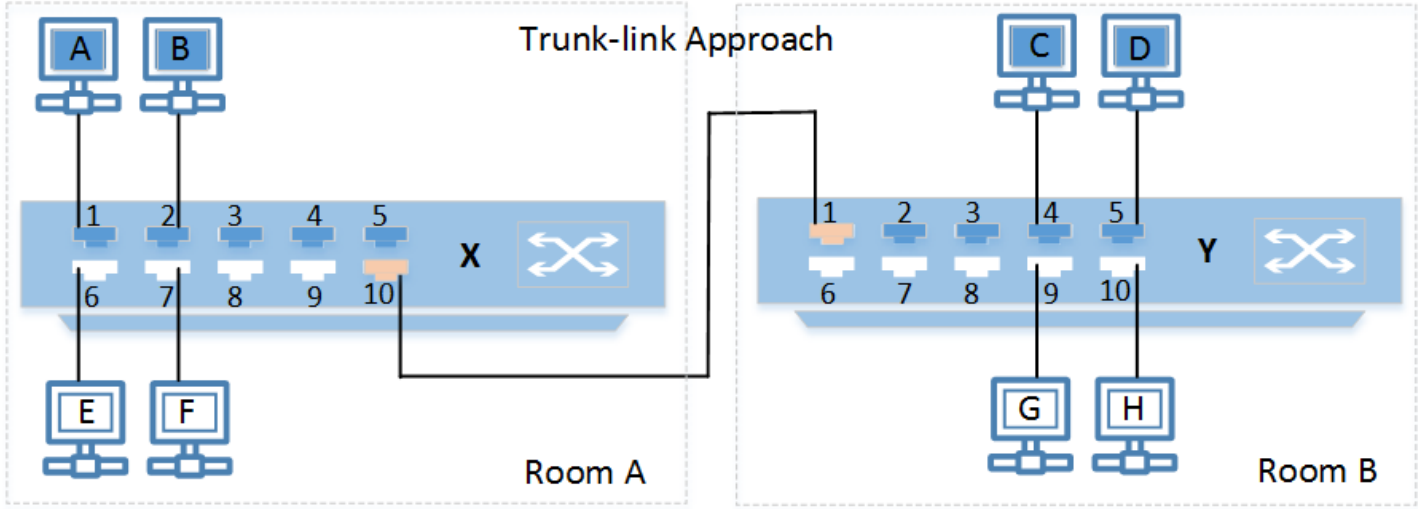
State	Ingress	Egress
tag	allowed	allowed, will be tagged
untag	allowed	allowed, will not be tagged
non-member	prohibited	prohibited

MAXIMUM TRANSMISSION UNIT (MTU)

- The standard Ethernet MTU is 1500 bytes at the network layer or 1518 bytes at the link layer:
  1500 bytes + 14 bytes (header) + 4 bytes (frame check sequence) =1518 bites
- *Add 4 bytes of tags , which can accommodated either by:*
  - *changing MTU to 1522 bytes, or*
  - *reducing the network layer MTU to 1496 bytes*

TAG-BASED VLANS

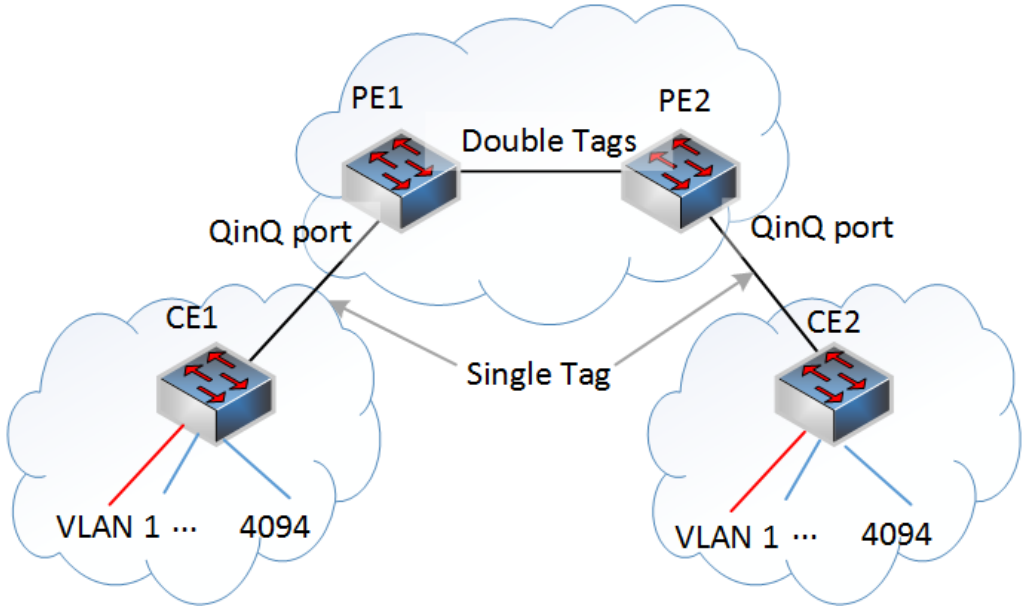
- Using tags to identify which logical switch a port belongs to.
- Using a single link to connect two ports on two switches, in which the link is called trunk-link and it forwards all data frames between two switches, thus allowing tagged traffic to be forwarded to the same VLAN physically residing on different switches.



Port	VLAN ID	Connected Device	Port	VLAN ID	Connected Device
1	1	A	1	1	Trunk port
2		B	2		not used
3		not used	3		not used
4		not used	4		C
5		not used	5		D
6	2	E	6	2	not used
7		F	7		not used
8		not used	8		not used
9		not used	9		G
10		Trunk port	10		H

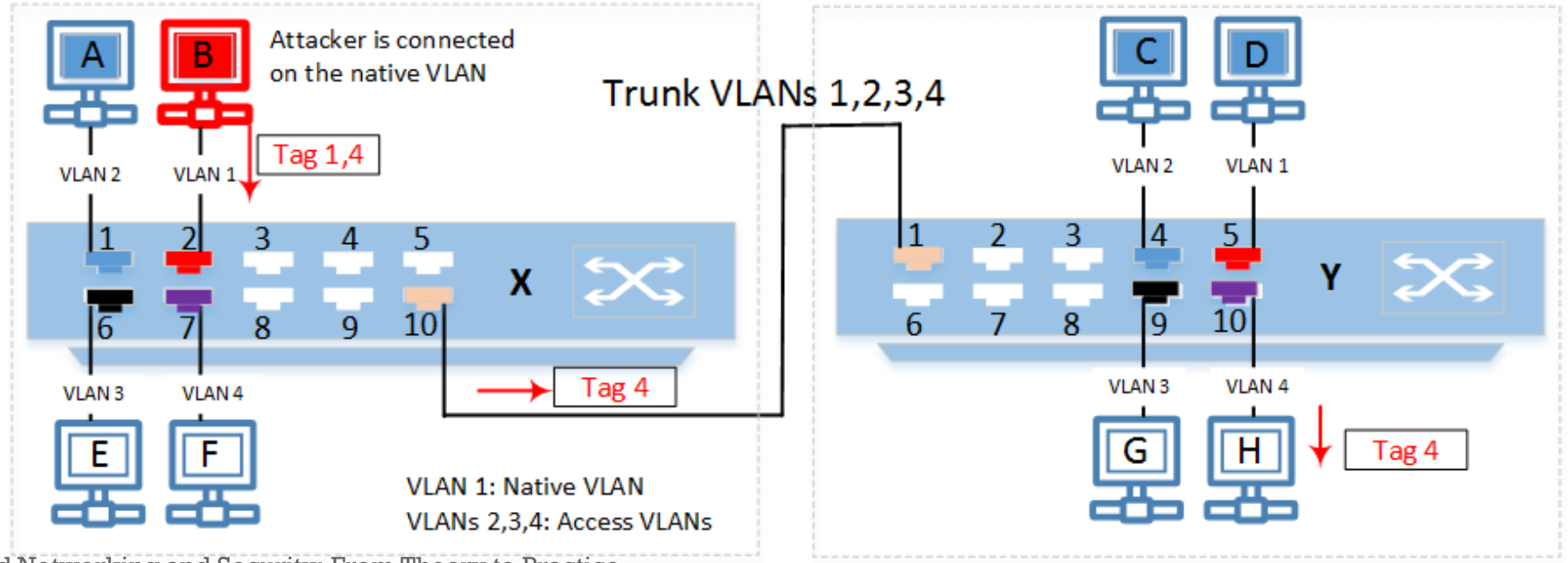
Q-IN-Q

- Basic QinQ: When receiving the packet, the QinQ port adds the VLAN TAG of the default VLAN of the port to the packet no matter whether the packet has the VLAN TAG. Before the packet is forwarded out from the QinQ port, delete the out layer of TAG and then forward it.
  - The disadvantage of the method is that the encapsulated-out layer of VLAN cannot be selected according to the VLAN TAG of the packet.
- Selective QinQ: When receiving the packet, the QinQ port adds the specified-out layer of VLAN TAG to the packet according to the VLAN TAG of the packet. If the encapsulated-out layer of VLAN TAG is not specified, add the VLAN TAG of the default VLAN of the port to the packet.

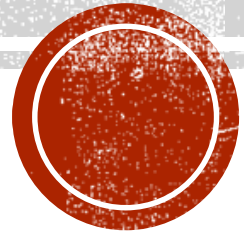


VLAN HOPPING

- VLAN hopping is usually done by
  - Double tagging: forward a double tagged frame on to a VLAN trunk with its outer tag removed but the inner tag intact and exposed. The inner tag should match the target hopping VLAN
  - Somehow persuading a switch to reconfigure an access port as a trunkport.



TUNNELING PROTOCOLS AND VIRTUAL PRIVATE NETWORKS (VPN)



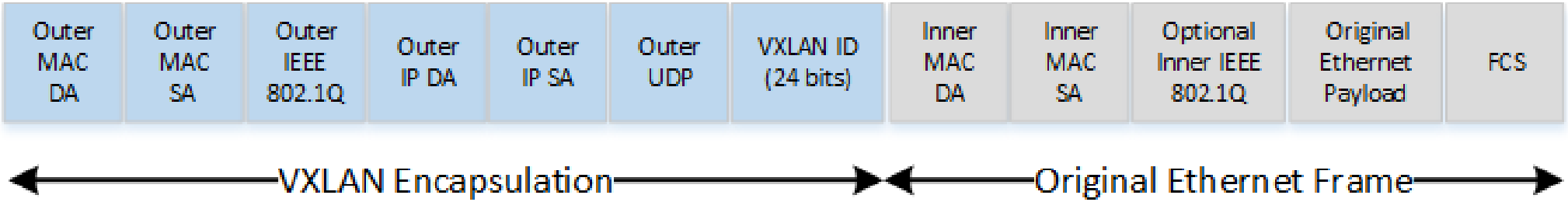
VXLAN

WHY VIRTUAL EXTENSIBLE LAN (VXLAN)?

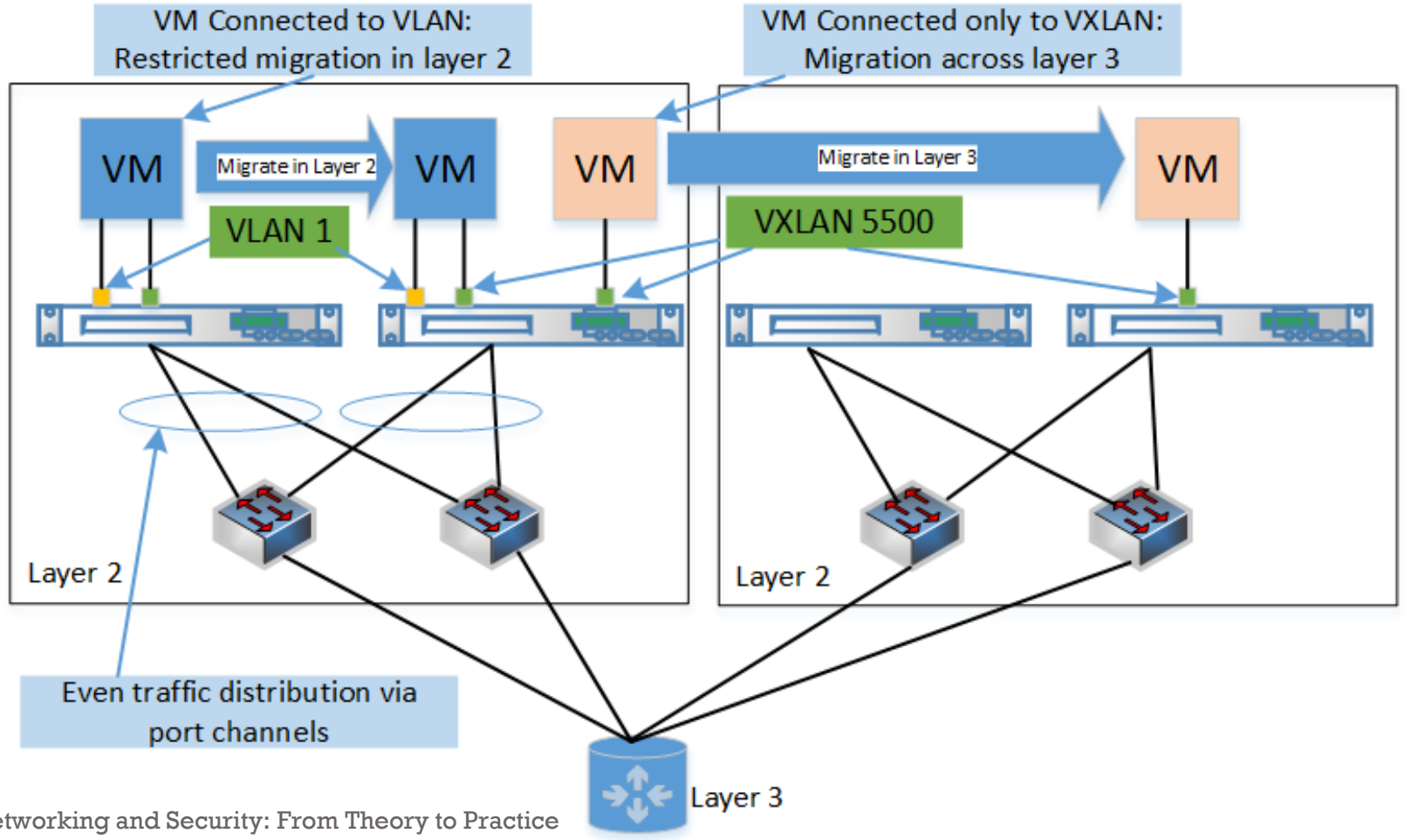
- Provide VLANs over IP networks to support VLANs over geographically remote datacenters
- Increase the length of a VLAN tag from 802.1Q's 12 bits to 24 bits
- Support migration of VMs between servers across layer-3 networks

VXLAN

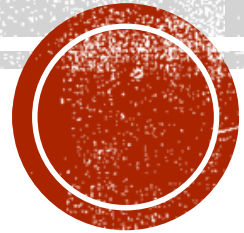
- VXLAN Frame Header



VXLAN EXAMPLE



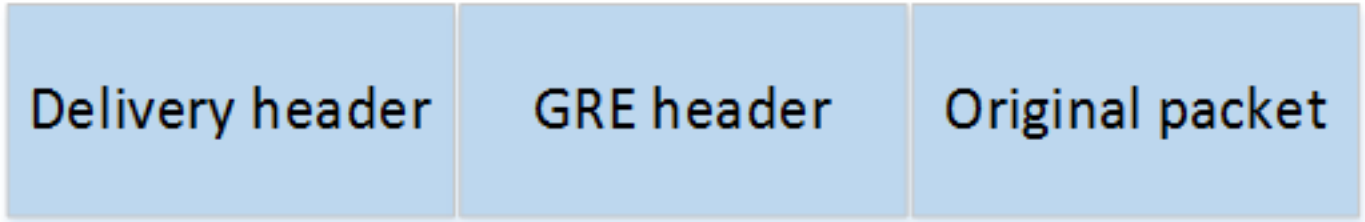
TUNNELING PROTOCOLS AND VIRTUAL PRIVATE NETWORKS (VPN)



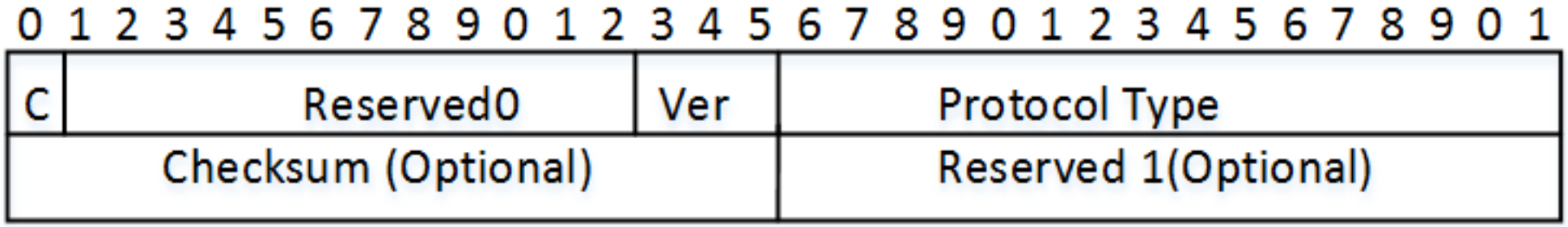
GRE

GRE HEADER

- GRE is running over IP
- GRE encapsulates the tunneled protocol layer



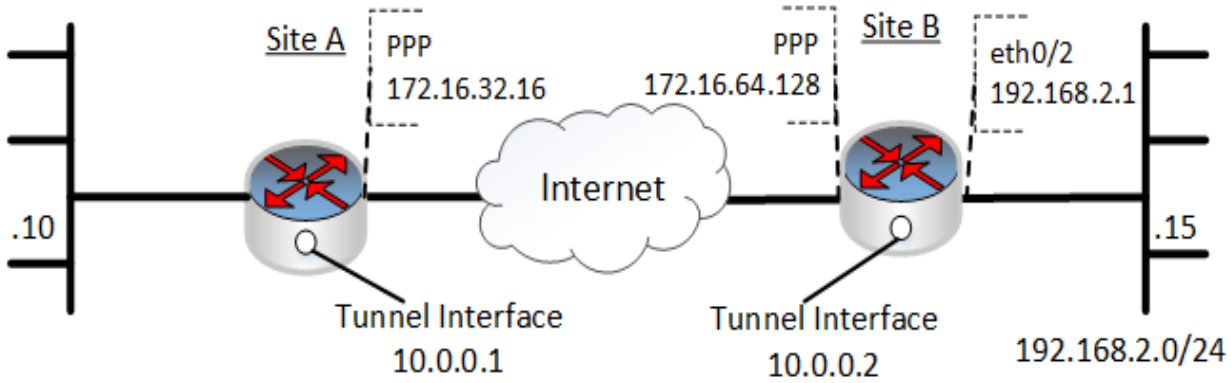
- GRE packet header



GRE PACKET FLOW

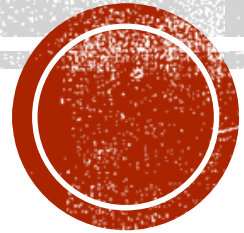
- In the following example, trace a packet from a node on the 192.168.1.0 network (192.168.1.10) to a node on the 192.168.2.0 network (192.168.2.15). The step-by-step tunnel establish procedure is presented in the following table:

Step	Description
1	Packet originates from 192.168.1.10.
2	Packet received by the router at Site A on eth 0/1 (192.168.1.1).
3	Checks routing table for routing information (destination 192.168.2.15) and determines the destination network is available through the tunnel interface.
4	Packet is encased in GRE header with source IP (172.16.32.16) and destination IP (172.30.64.128) and routed back through the route stack.
5	Checks routing table for routing information for destination IP 172.30.64.128 and routes the packet out the WAN interface.
6	The router at Site B receives the packet on the PPP interface (172.30.64.128). The system recognizes that there is a GRE header on the packet and sends it to the tunnel interface associated with the source and destination IP addresses (172.16.32.16 and 172.30.64.128).
7	GRE header is stripped and the packet is routed through the route stack with a destination IP of 192.168.2.15.
8	Packet is routed out eth 0/2 for delivery.

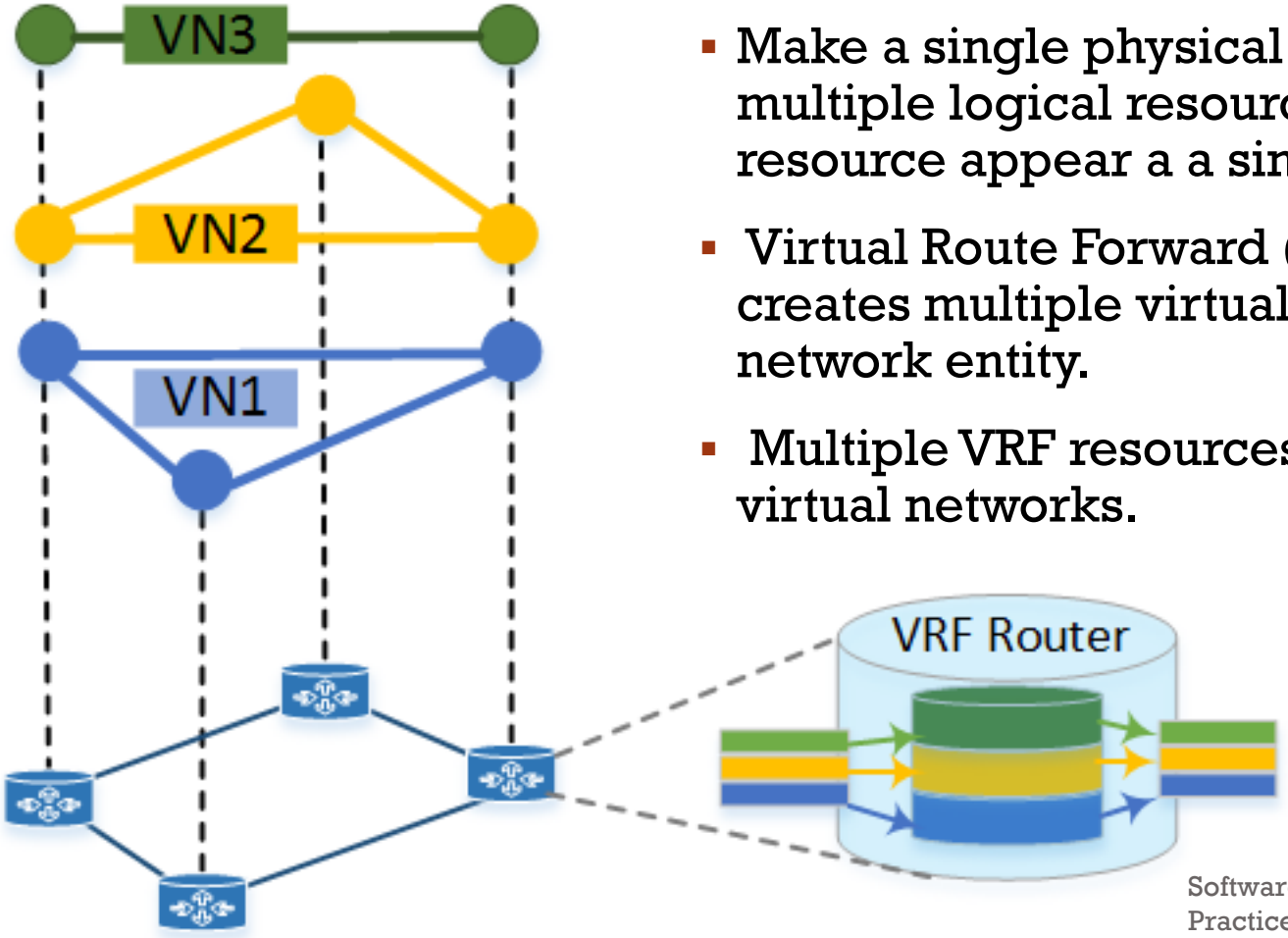


VIRTUAL ROUTING AND FORWARDING

Virtual Routing Forwarding (VRF)



VIRTUAL ROUTING FORWARDING (VRF)



- Make a single physical resource appear to function as multiple logical resources; or make multiple physical resource appear a a single logical resource.
- Virtual Route Forward (VRF) is a technique which creates multiple virtual networks within a single network entity.
- Multiple VRF resources create the isolation between virtual networks.

ADDITIONAL LEARNING MATERIALS

- OVS http://docs.openvswitch.org/en/latest/

CITE THIS WORK

```
@book{huang2018software,
title={Software-Defined Networking and Security: From Theory to Practice},
author={Huang, Dijiang and Chowdhary, Ankur and Pisharody, Sandeep},
year={2018},
publisher={CRC Press}}
```