CS60203: Design Optimization of Computing Systems

Network Virtualization

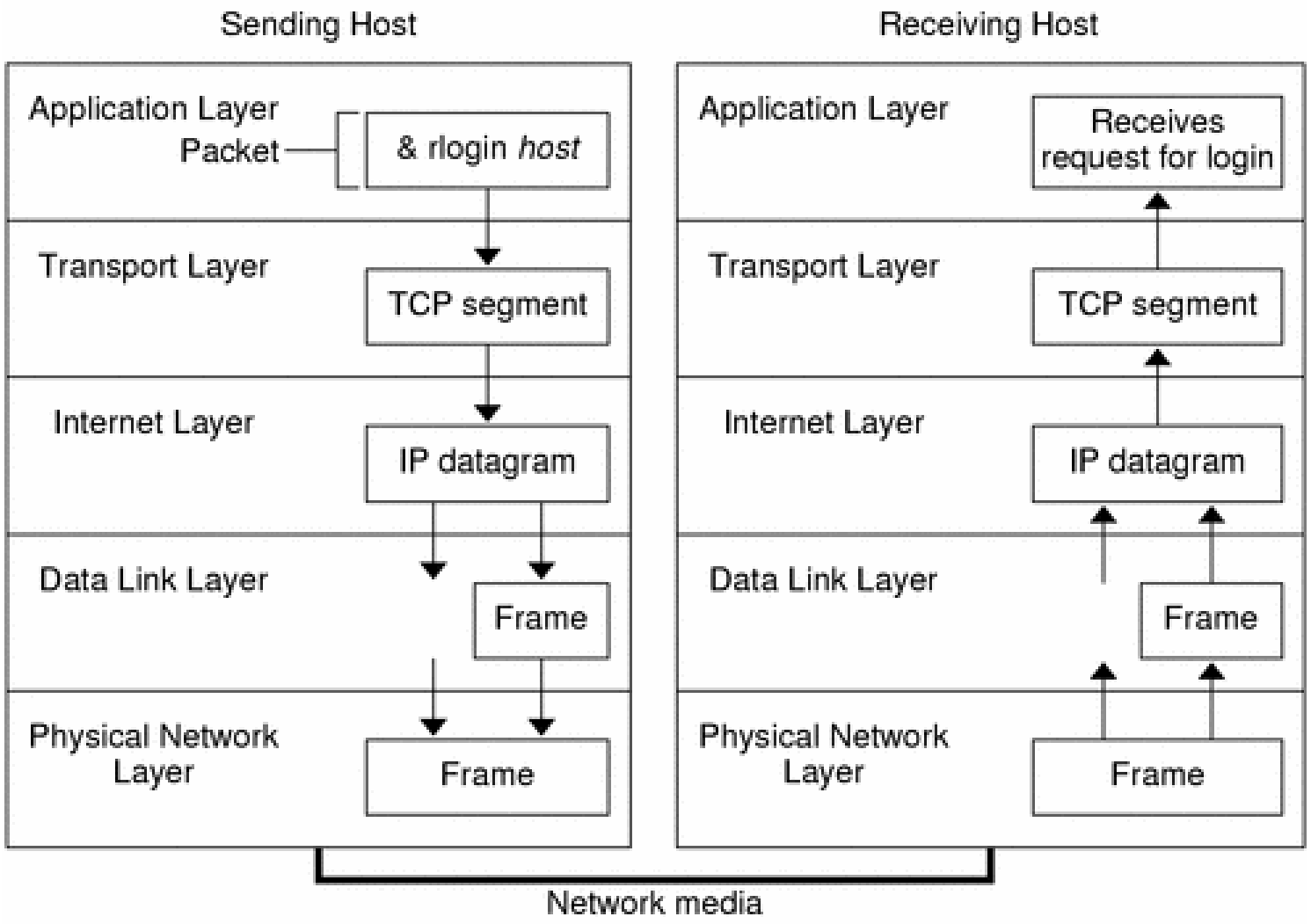
Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty

Mainack Mondal

Network Interfacing in the TCP/IP Stack



- **Data Plane vs Control Plane**
- **Basic task of the protocol stack:** To transfer data across the hosts (**Data Plane**)
  - However, the network needs to perform several control/ management tasks to ensure that the data is delivered reliably to the intended host (**Control Plane**)

Network Interfacing in the TCP/IP Stack

- **Data Plane tasks:**
  - Read a packet header
  - Find out the destination IP/MAC
  - Decide the next hop interface
  - Write the packet back to the intended interface

- **Control Plane tasks:**
  - Prepare the table to consult (routing/forwarding tables) for the next hop
  - Control the input/output buffer for flow management

Revisit Routers and Routing Functionalities

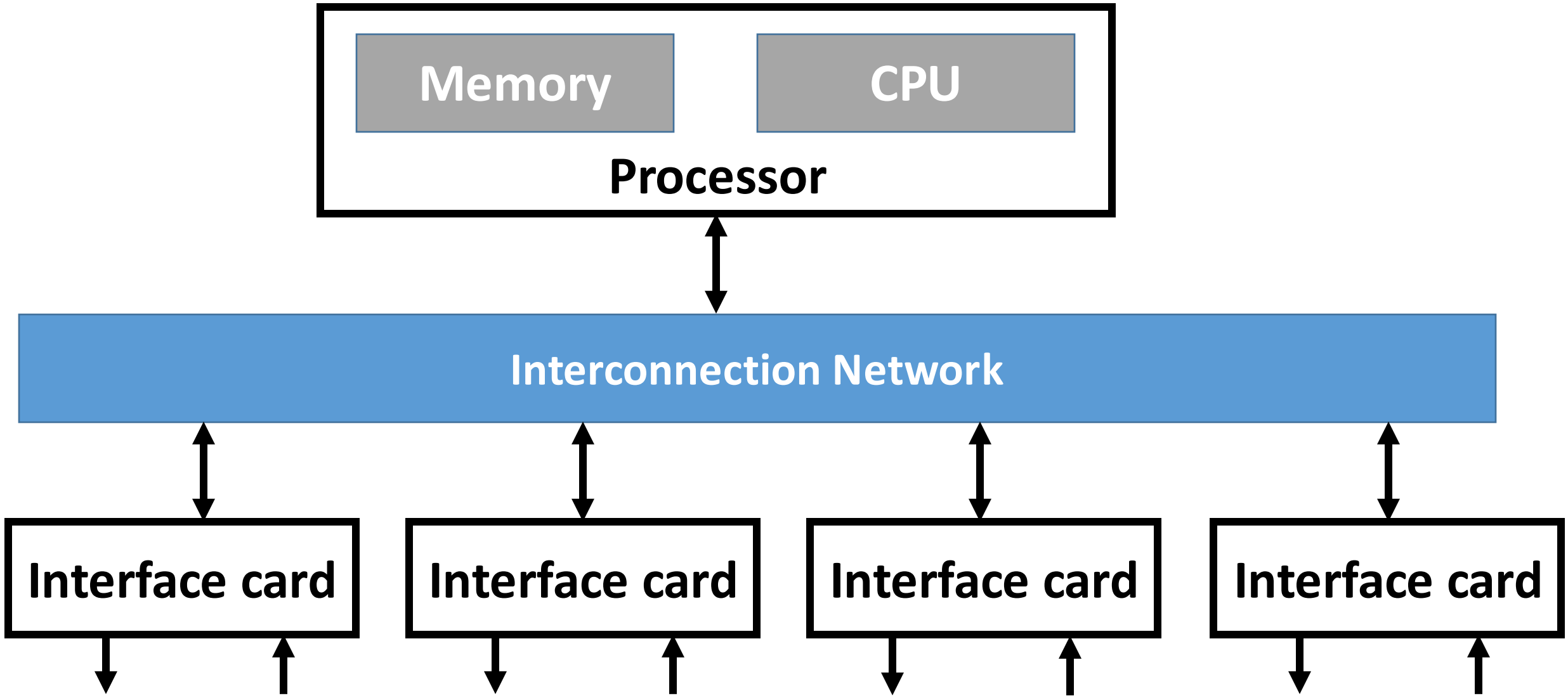
Performance Issues and Optimizations in Routers

The Life of a Router

- Do
  - Find Path
  - Forward, forward, forward, forward, forward, ...
  - Find Path
  - Forward, forward, forward, forward, forward, ...
- Repeat until powered off

- Two basic operations –
  - Construct the routing table **– the control plane**
  - Do a routing match and forward the packet to a dedicated interface – **the data plane**

Basic Architectural Components of a Router



Router Hardware

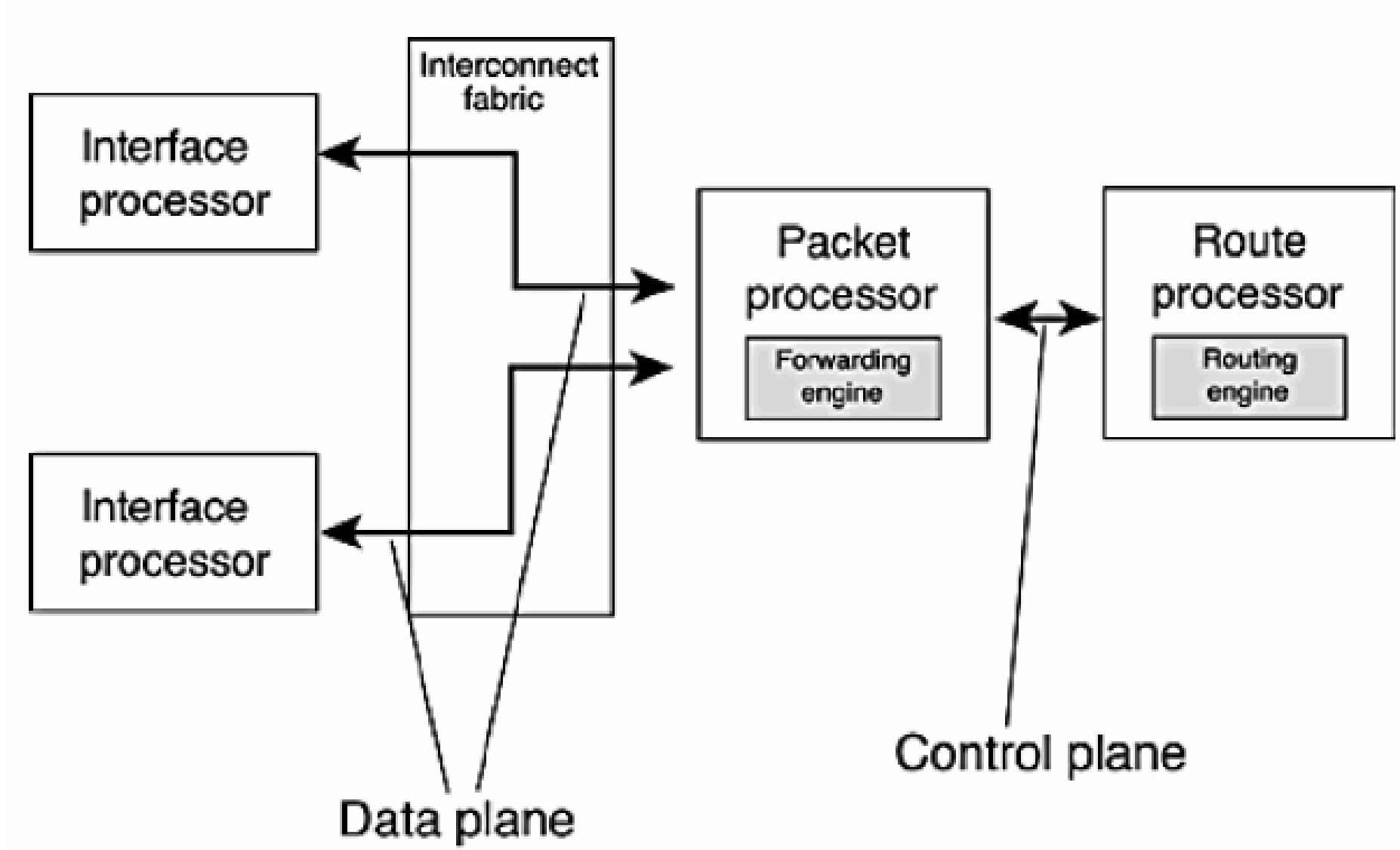
- Processor is responsible for control functions **(route processors)**
  - Construct the routing table based on the routing algorithm

- Forwarding is done at the interface card
  - Route match needs to be very fast
  - Specialized hardware – Ternary Content-Addressable Memory (TCAM)

Routing Table Construction

Forwarding

Router Internals



Functional Components

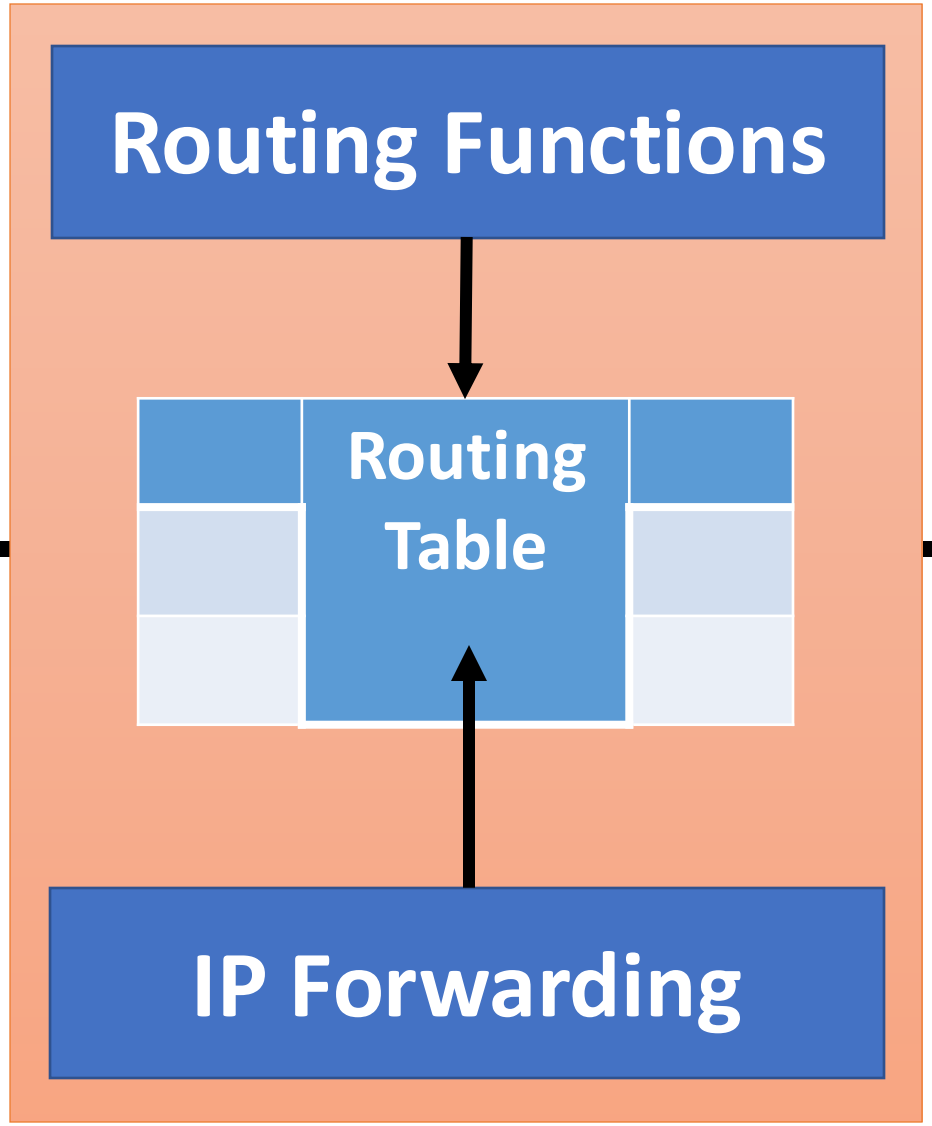


Table Update

Route Lookup

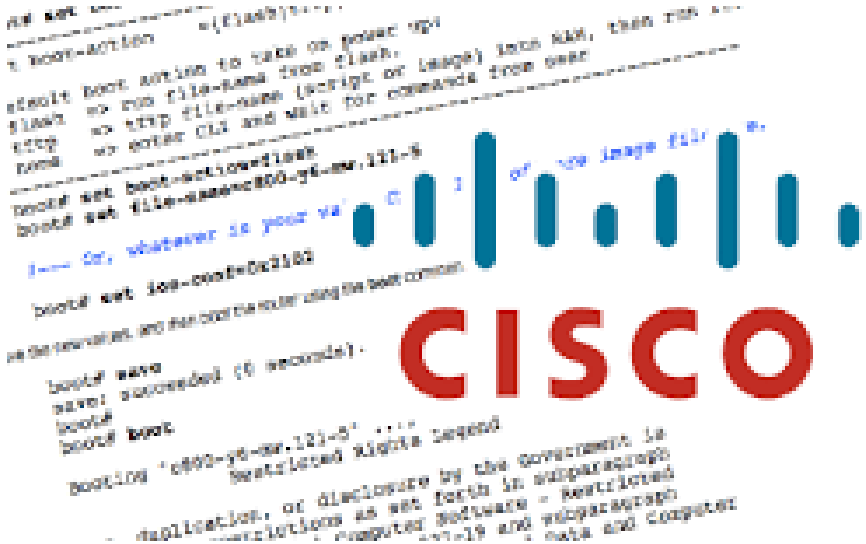
Control

Datapath

Per Packet Processing

Control Plane in a Router

- Note that a router is a **special purpose computer**
- Implemented as a software (router OS) that supports the basic computing functionalities to run a router along with routing functionalities
- Routing protocols are implemented in the router OS
- Example: Cisco IOS



Routing Functions

- **Route Calculation**
- **Maintenance of the routing table**
- **Execution of the routing protocol**

- On commercial routers, routing functions are handled by a single general-purpose processor, called the **route processor**

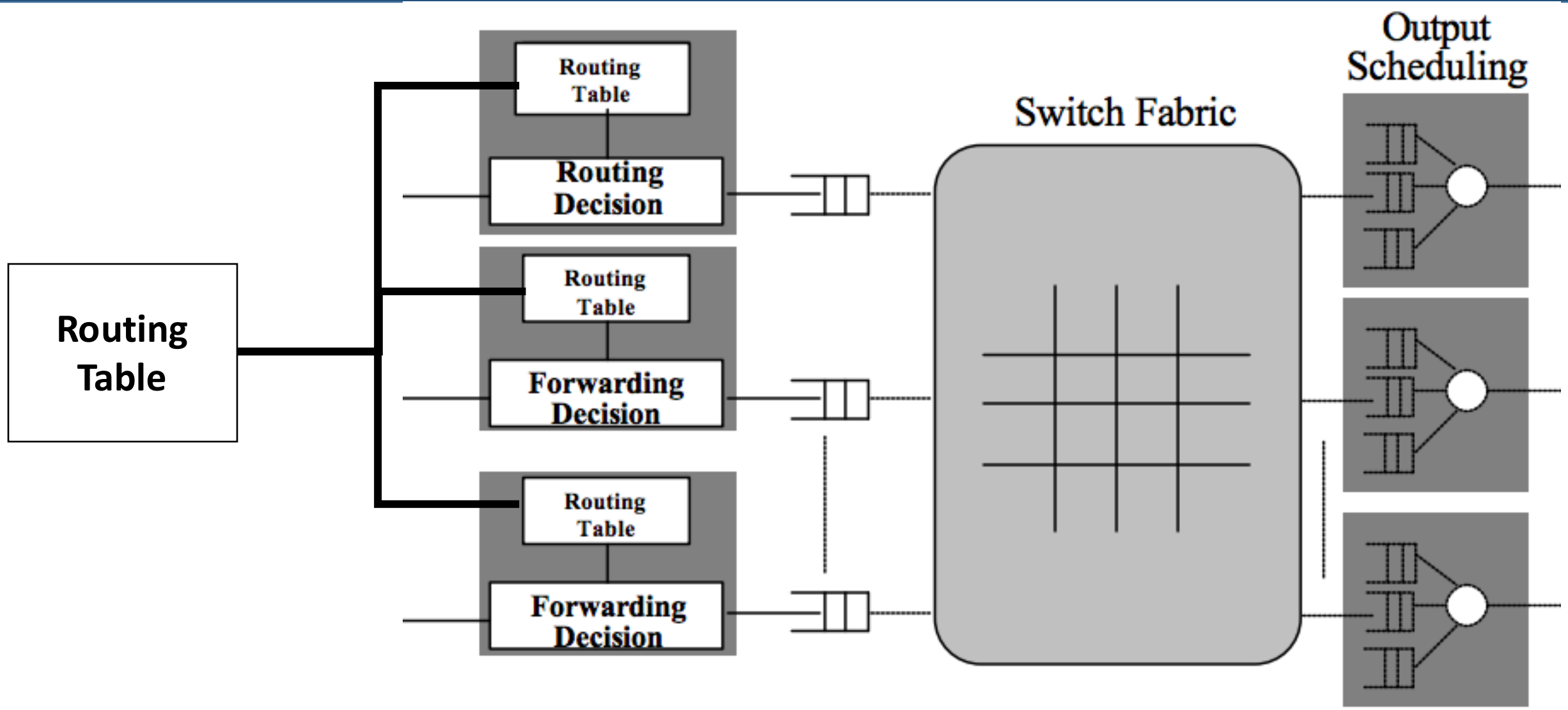
Data Plane of a Router

- Implement forwarding functionalities – make a route lookup and forward the packet at the destination interface
- Functionality is similar to a L2 switch – use switch fabric (the mapping from input ports to output ports) to forward the packet from one interface to another
- Maintains interface buffer – to implement store and forward functionality

IP Forwarding

- Per packet processing of the IP packets
- IP forwarding is distributed, handled by individual interface controllers
- Special hardware devices are used - TCAM

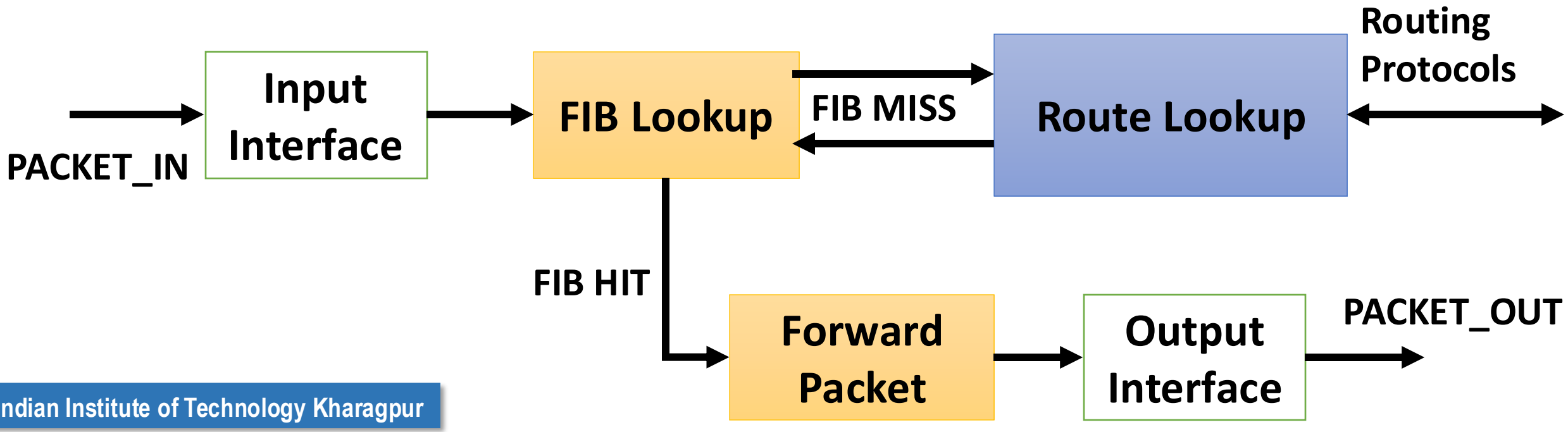
Per Packet Processing – Basic Architectural Components



Forwarding Information Base (FIB)

- The interfaces maintains a *forwarding information base* (FIB) – a mapping from input interface to output interface

- A replica of the routing table used at the interfaces for making the forwarding decision



Difference between RIB and FIB

- **Routing Information Base (RIB)** – The routing table, implemented in software, is maintained at the control plane

- **Forwarding Information Base (FIB)** – The copy of the required routes maintained in interface TCAM hardware

- RIB is dynamic and maintains entire routing information, FIB is updated whenever required

RIB and FIB

The RIB

172.16.1.0	255.255.255.0	172.16.1.2	Eth0
172.16.2.0	255.255.255.0	172.16.2.2	Eth1
10.3.0.0	255.255.0.0	10.3.1.1	Eth3
10.9.0.0	255.255.0.0	10.9.1.1	Eth4

172.16.2.0	255.255.255.0	172.16.2.2	Eth1
10.3.0.0	255.255.0.0	10.3.1.1	Eth3
10.9.0.0	255.255.0.0	10.9.1.1	Eth4

FIB at Eth0

FIB at Eth1

172.16.1.0	255.255.255.0	172.16.1.2	Eth0
10.3.0.0	255.255.0.0	10.3.1.1	Eth3

RIB Feeds FIB

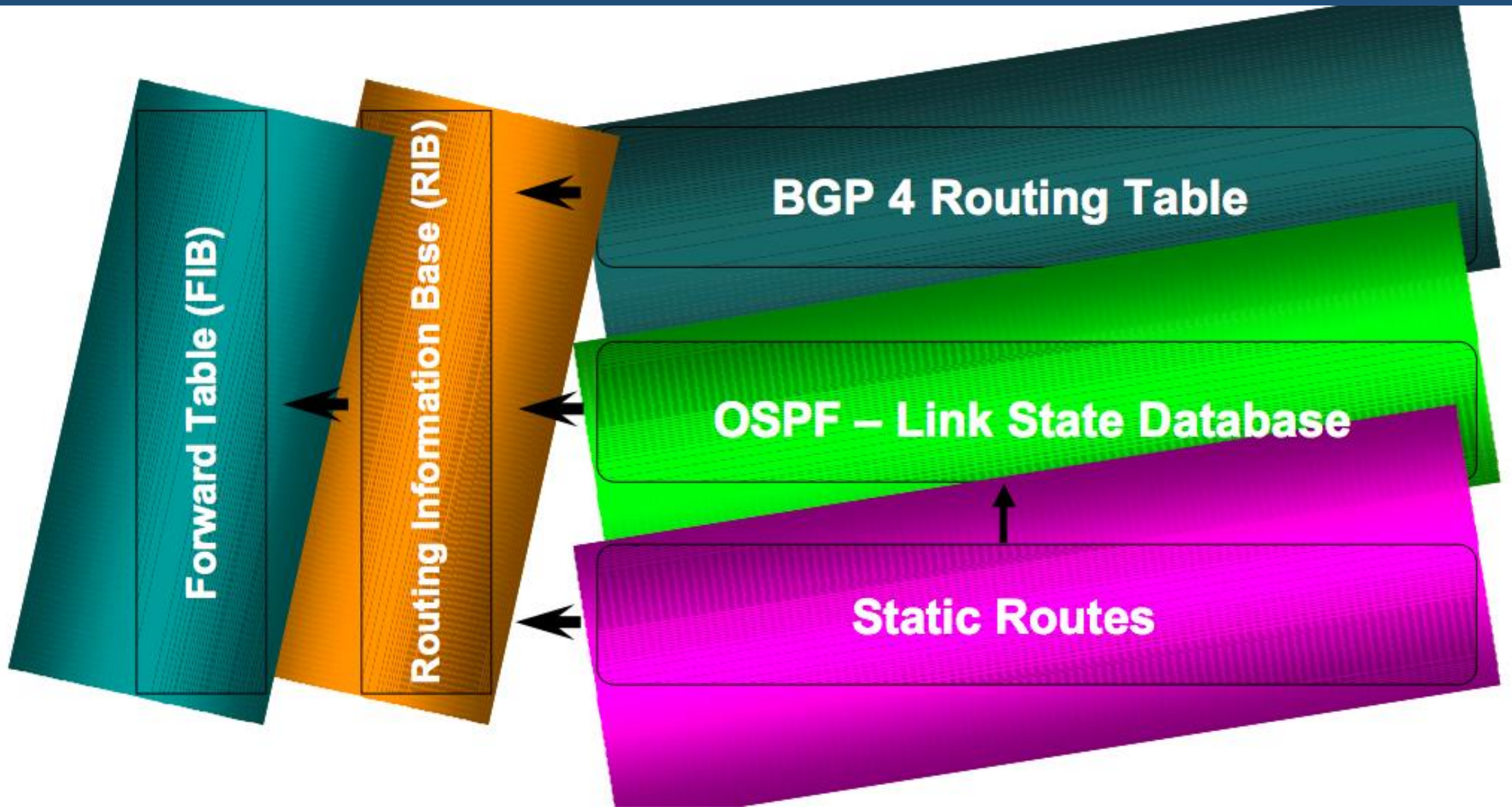
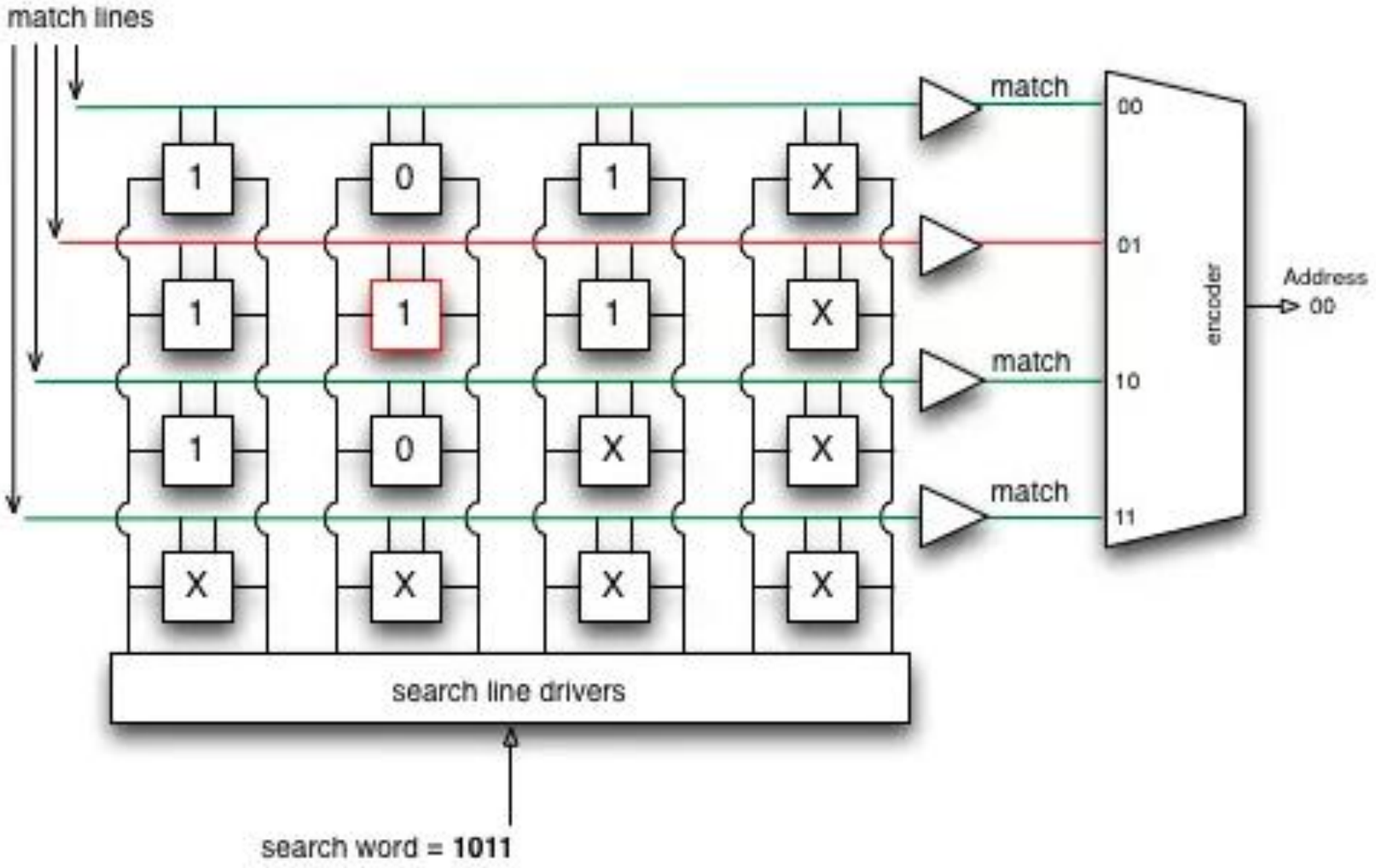


Image source: Cisco

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

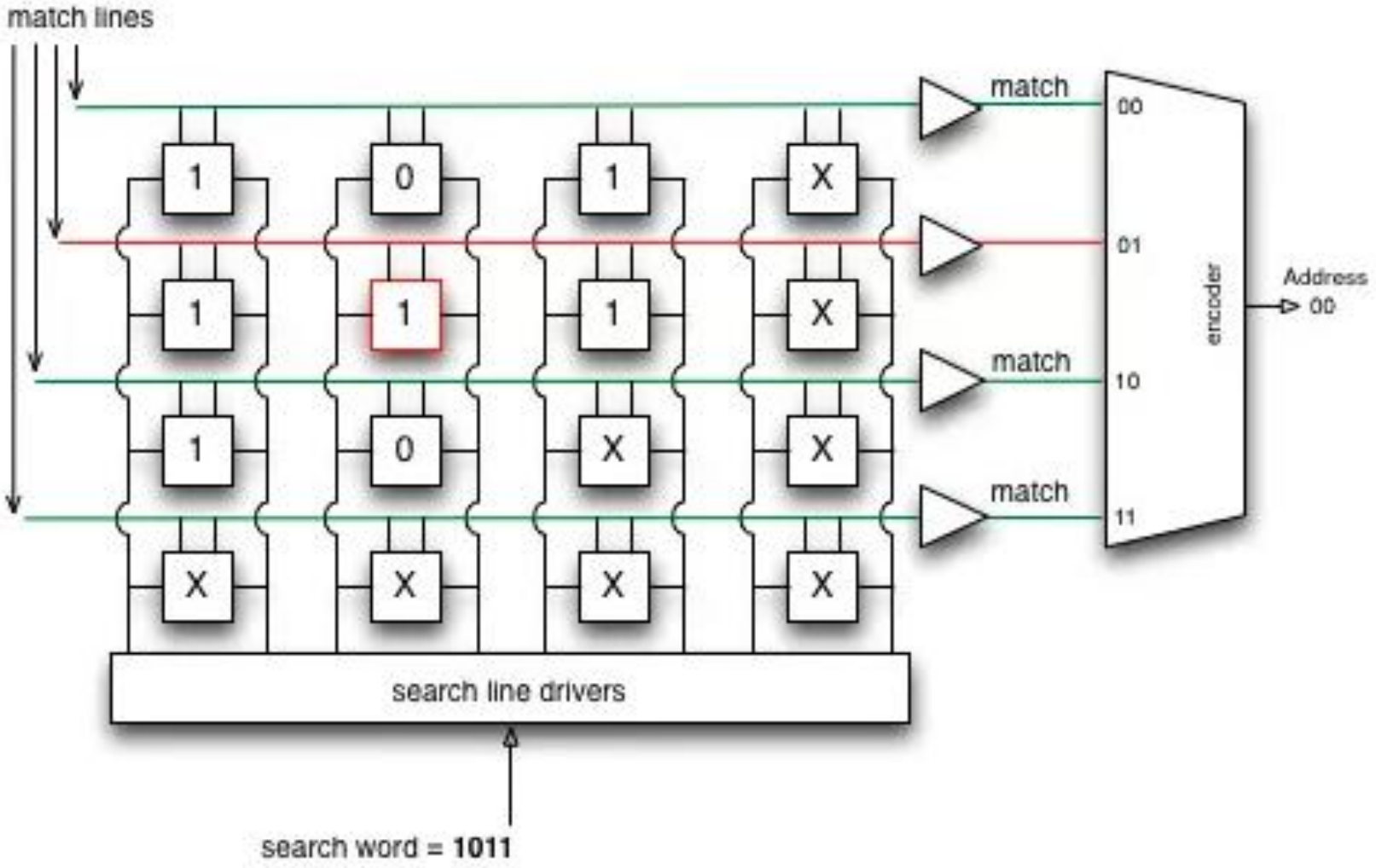
Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/



Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

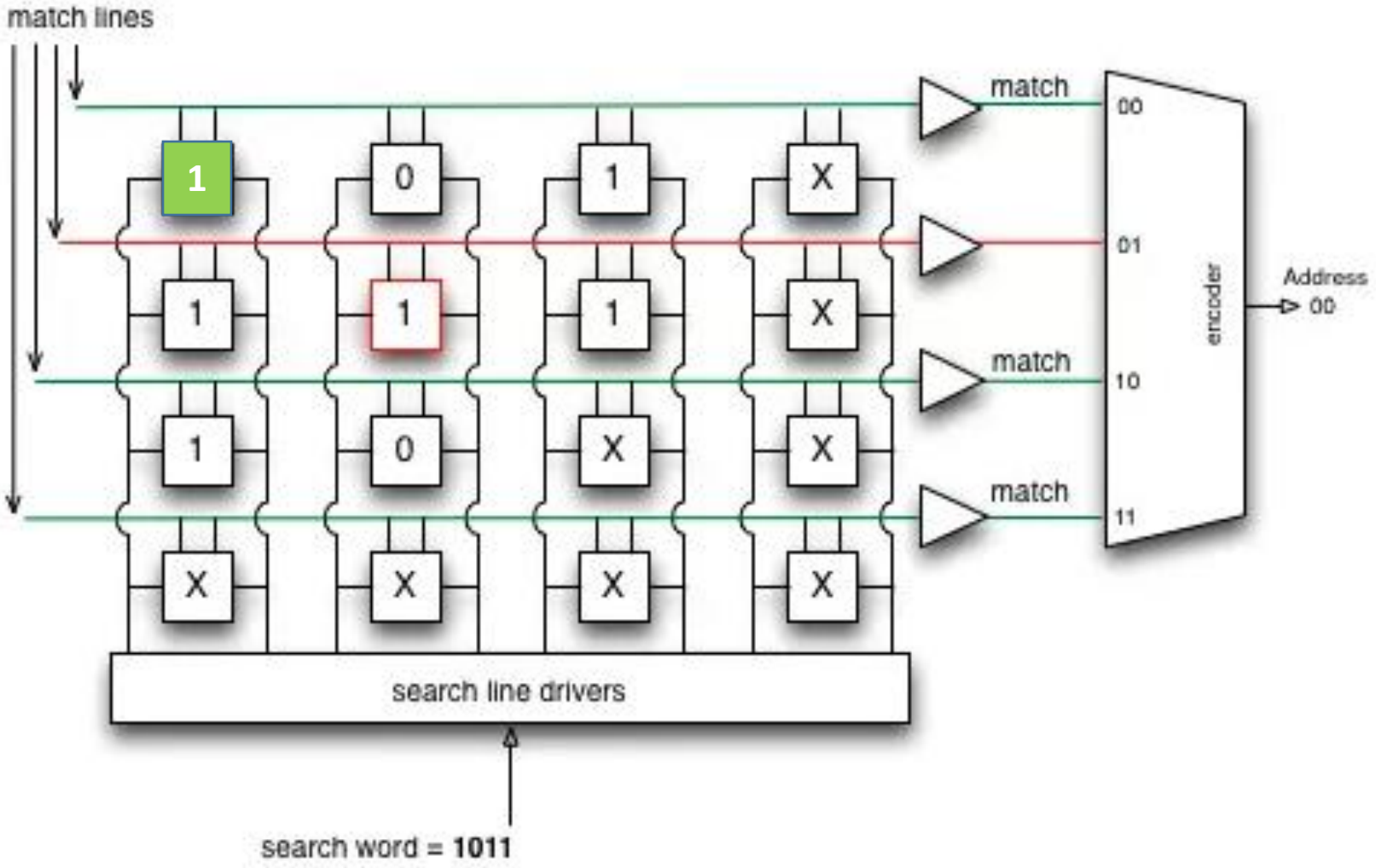


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

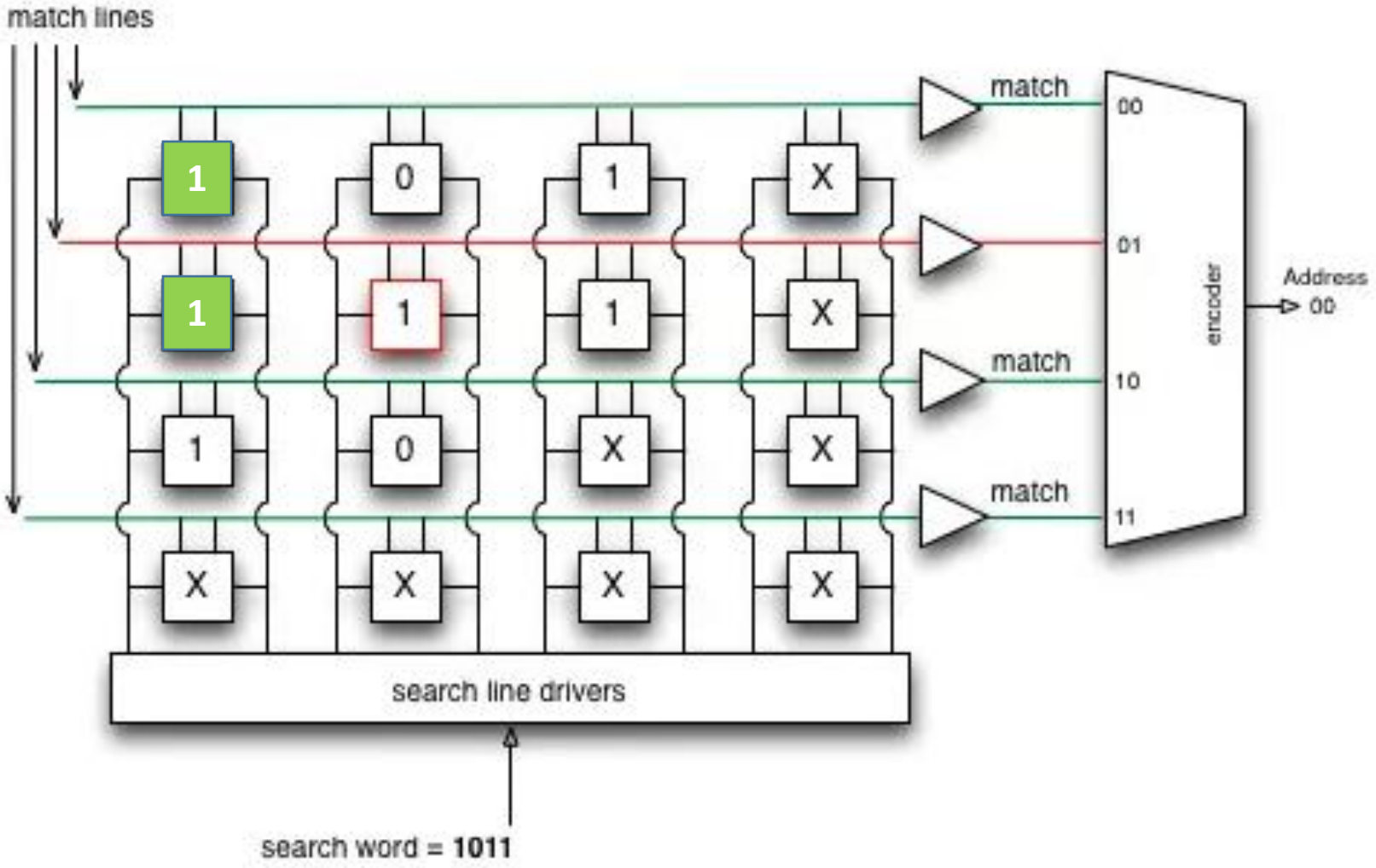


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

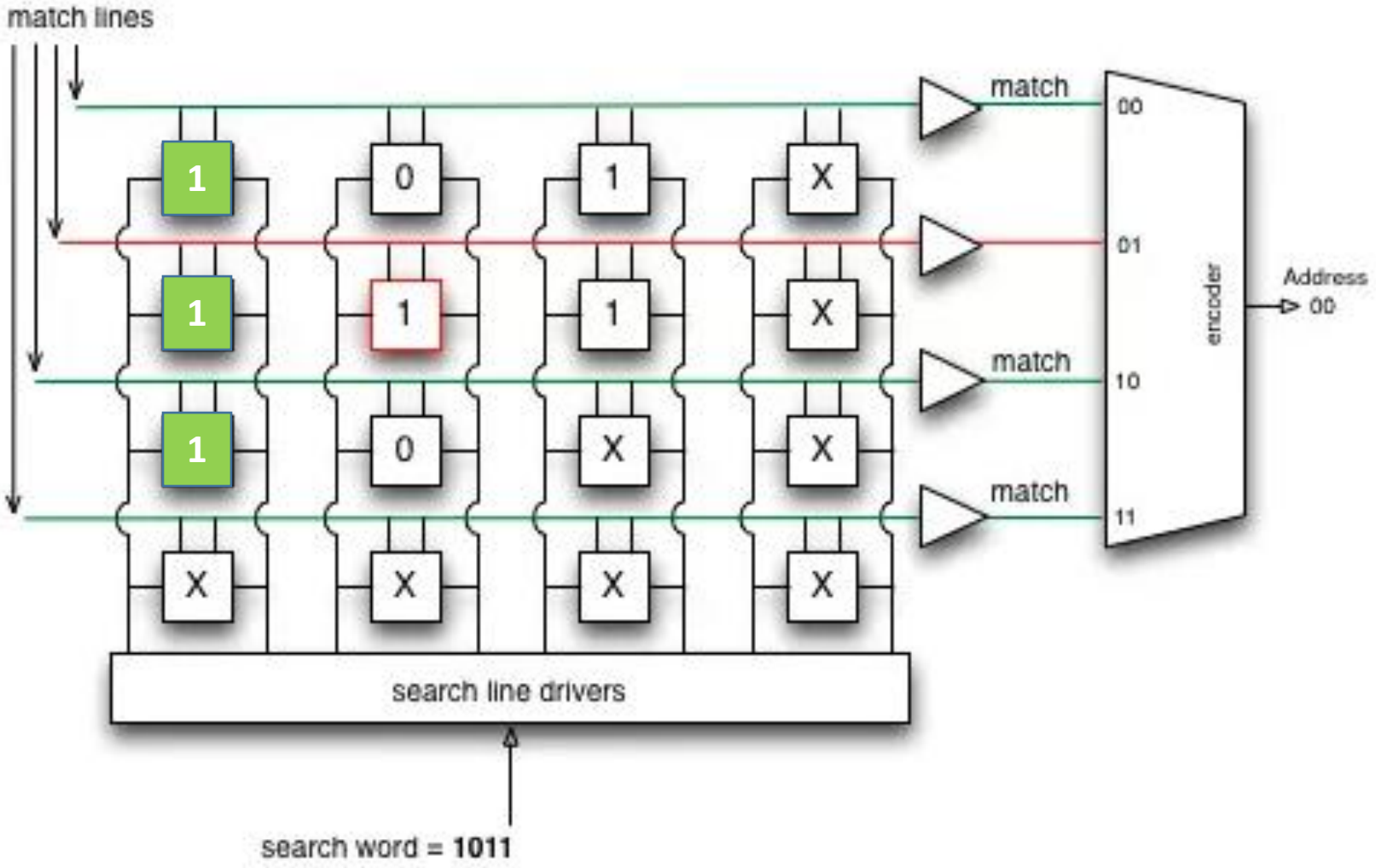


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

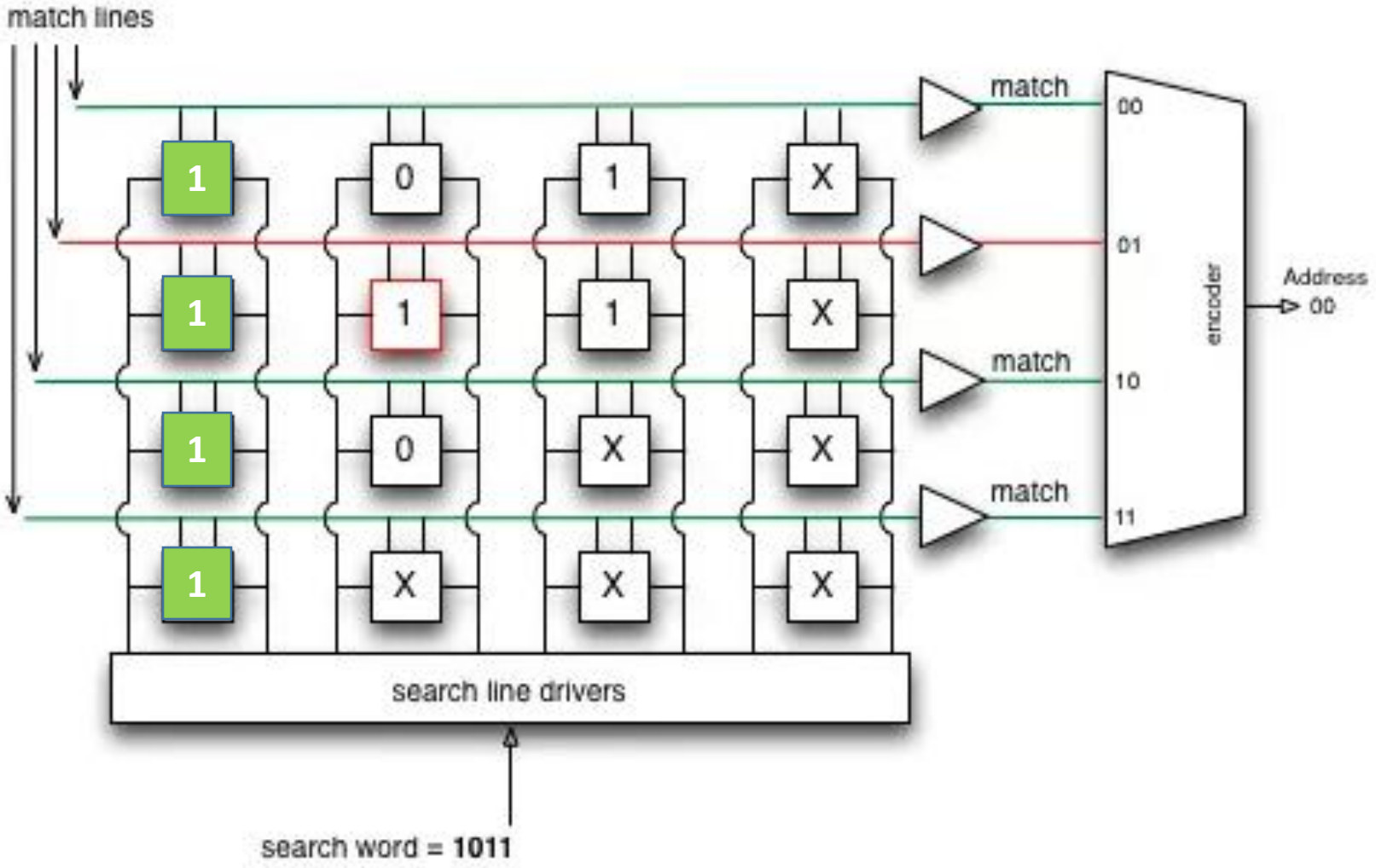


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

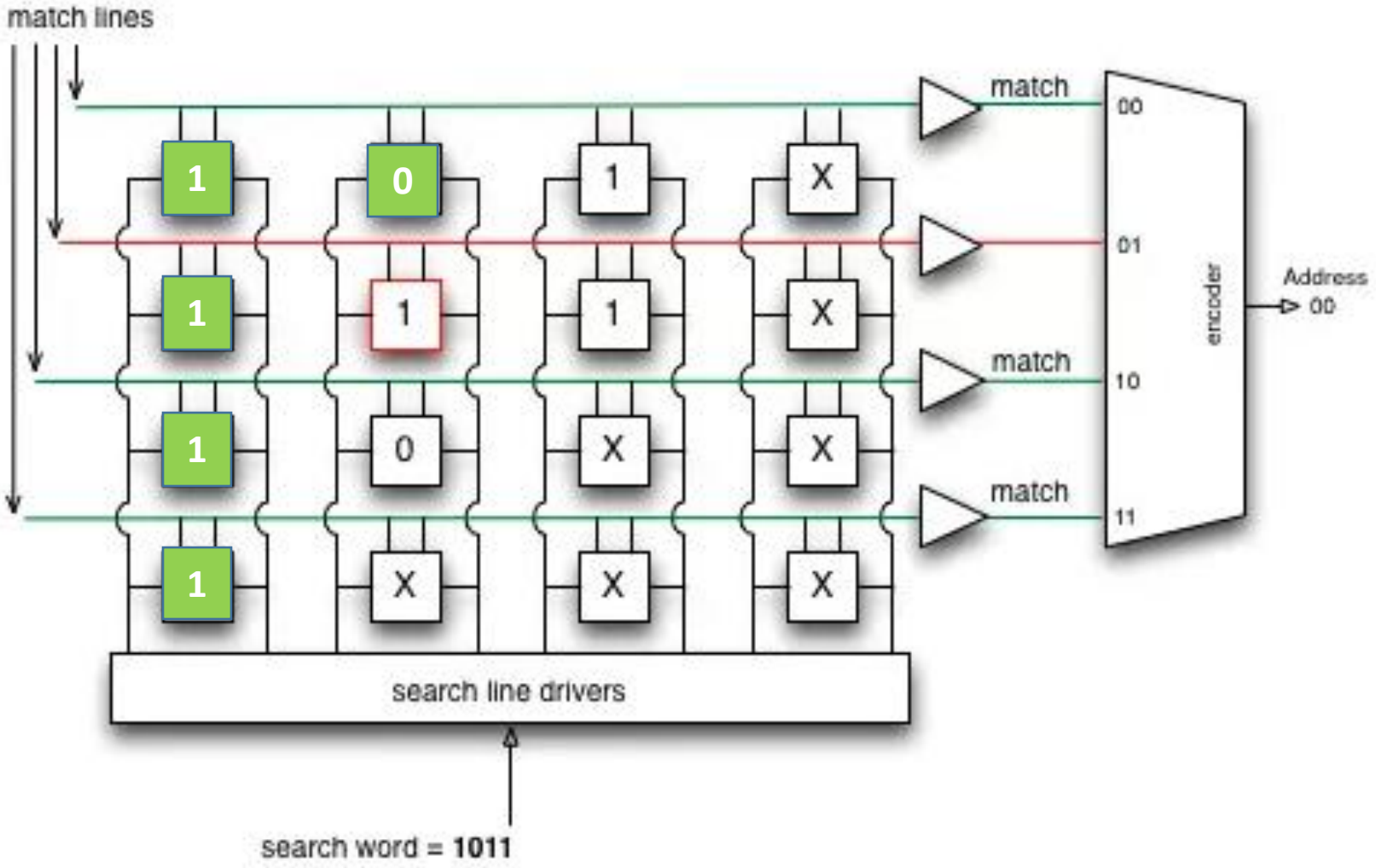


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

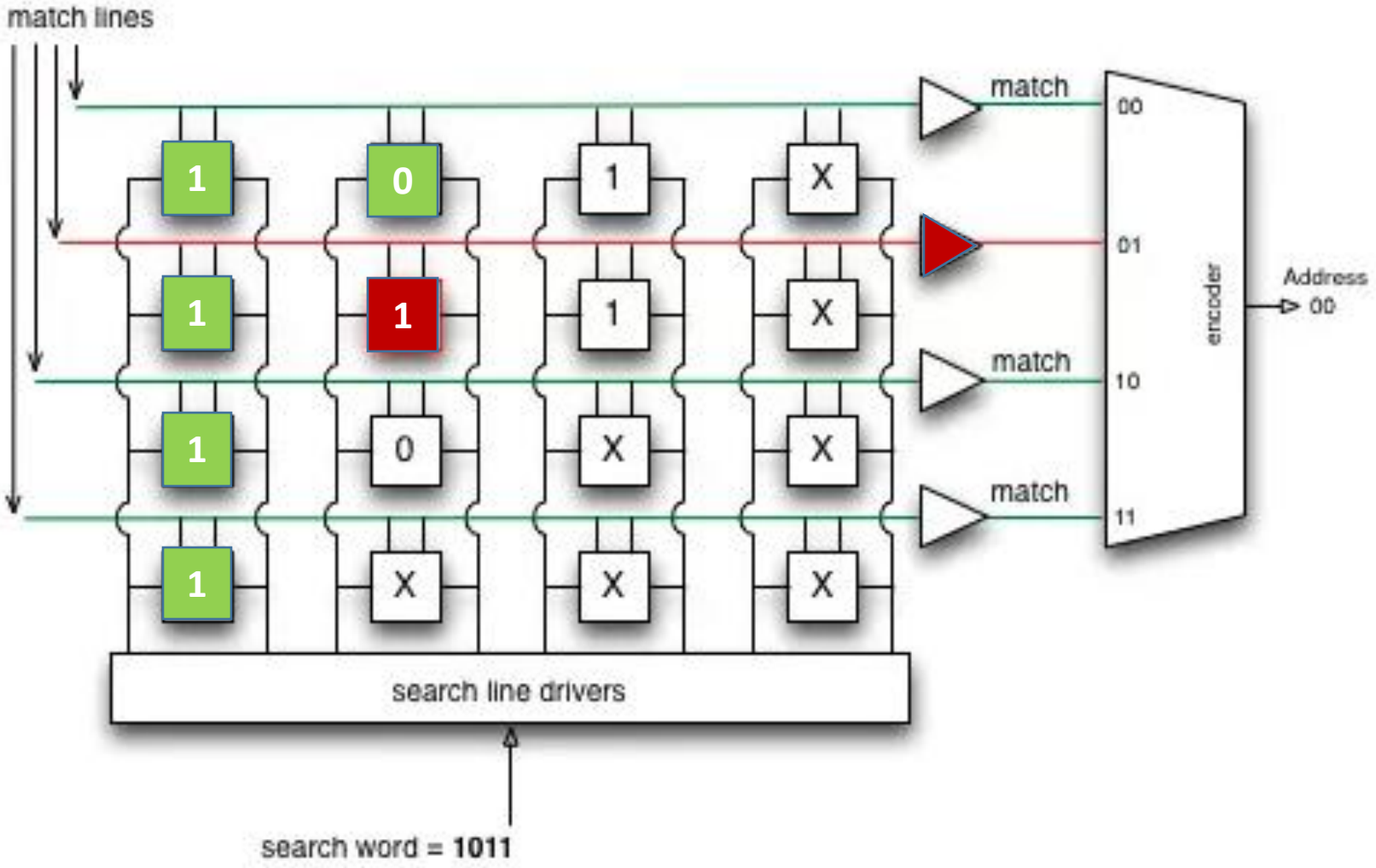


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

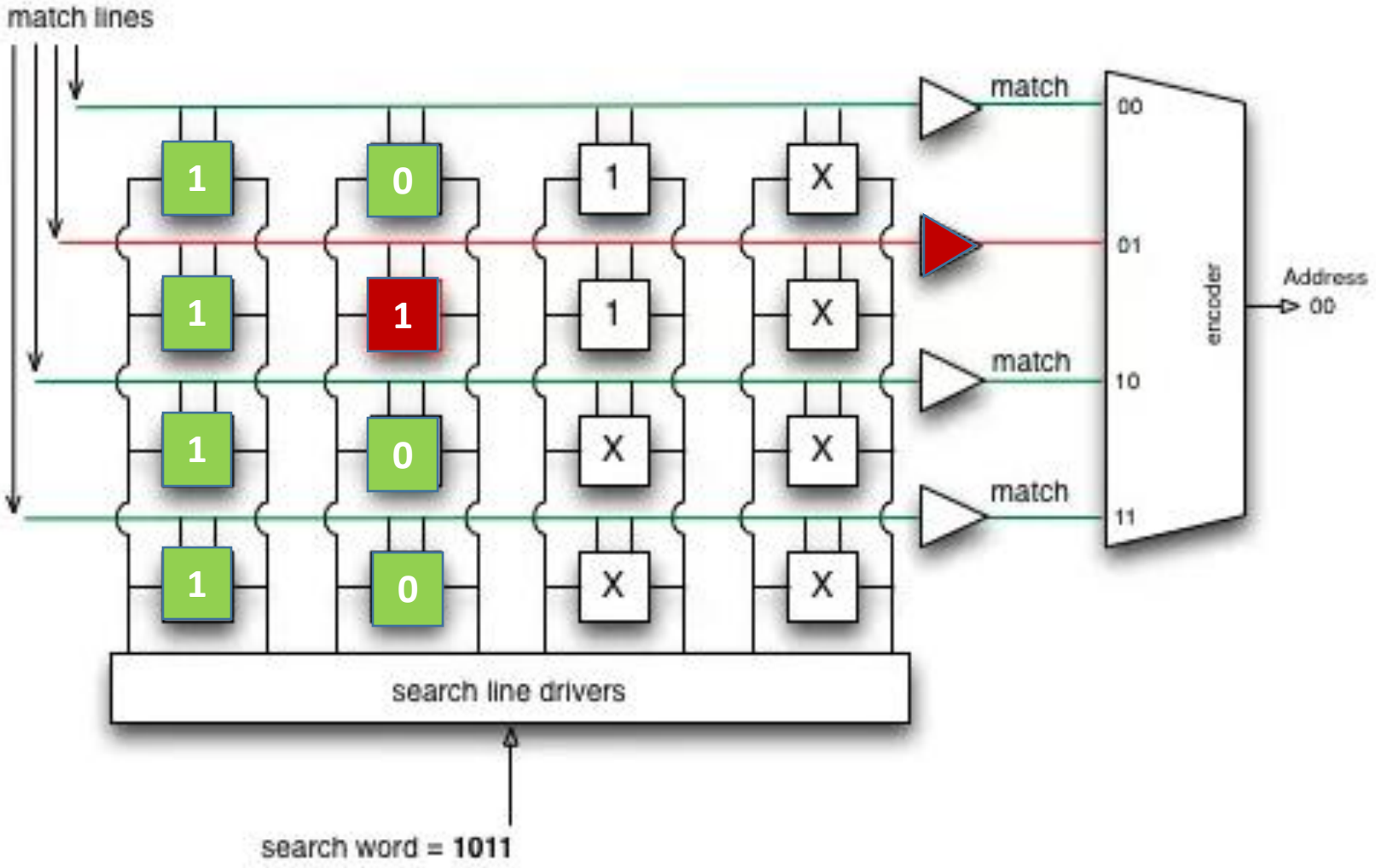


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

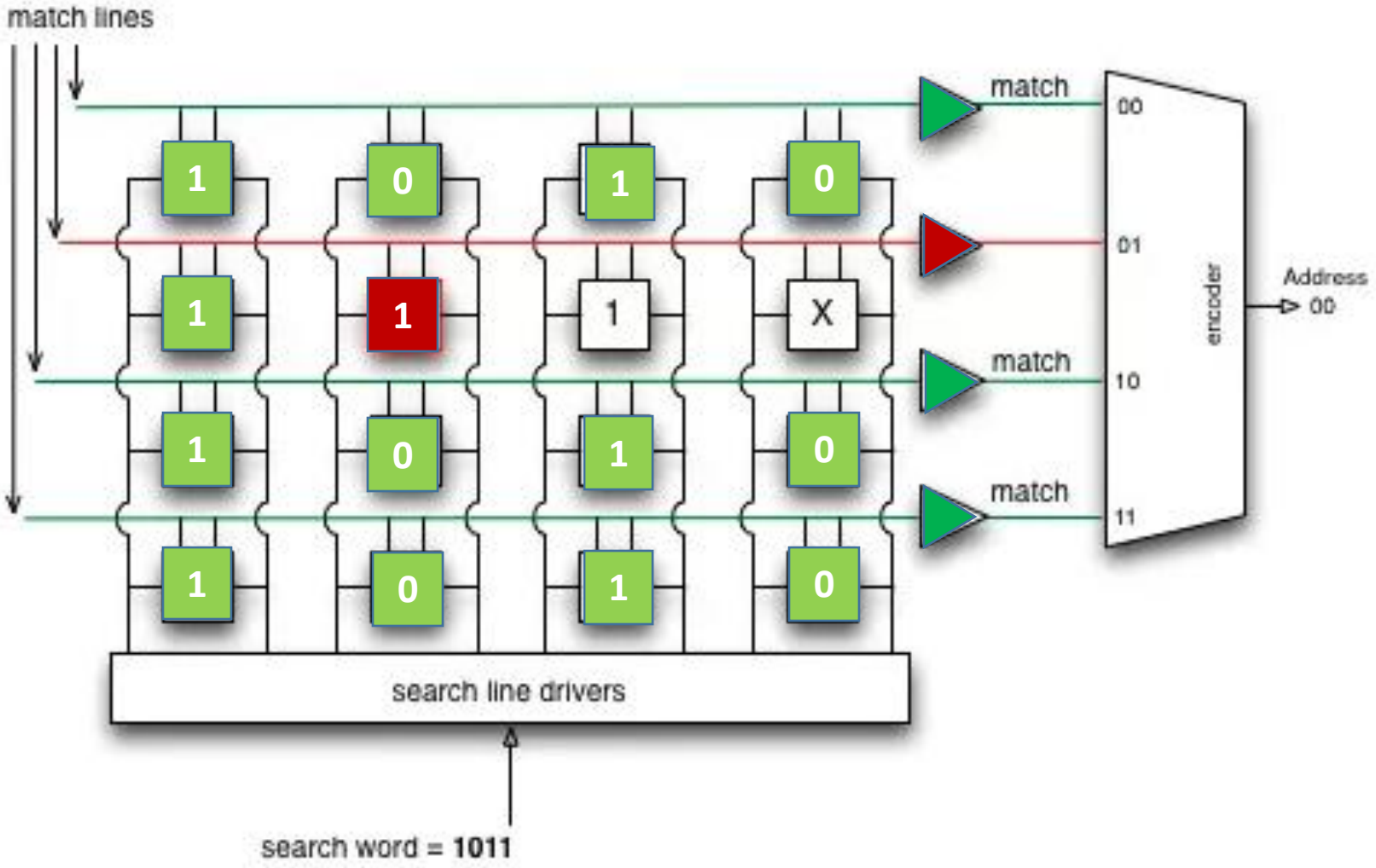


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/

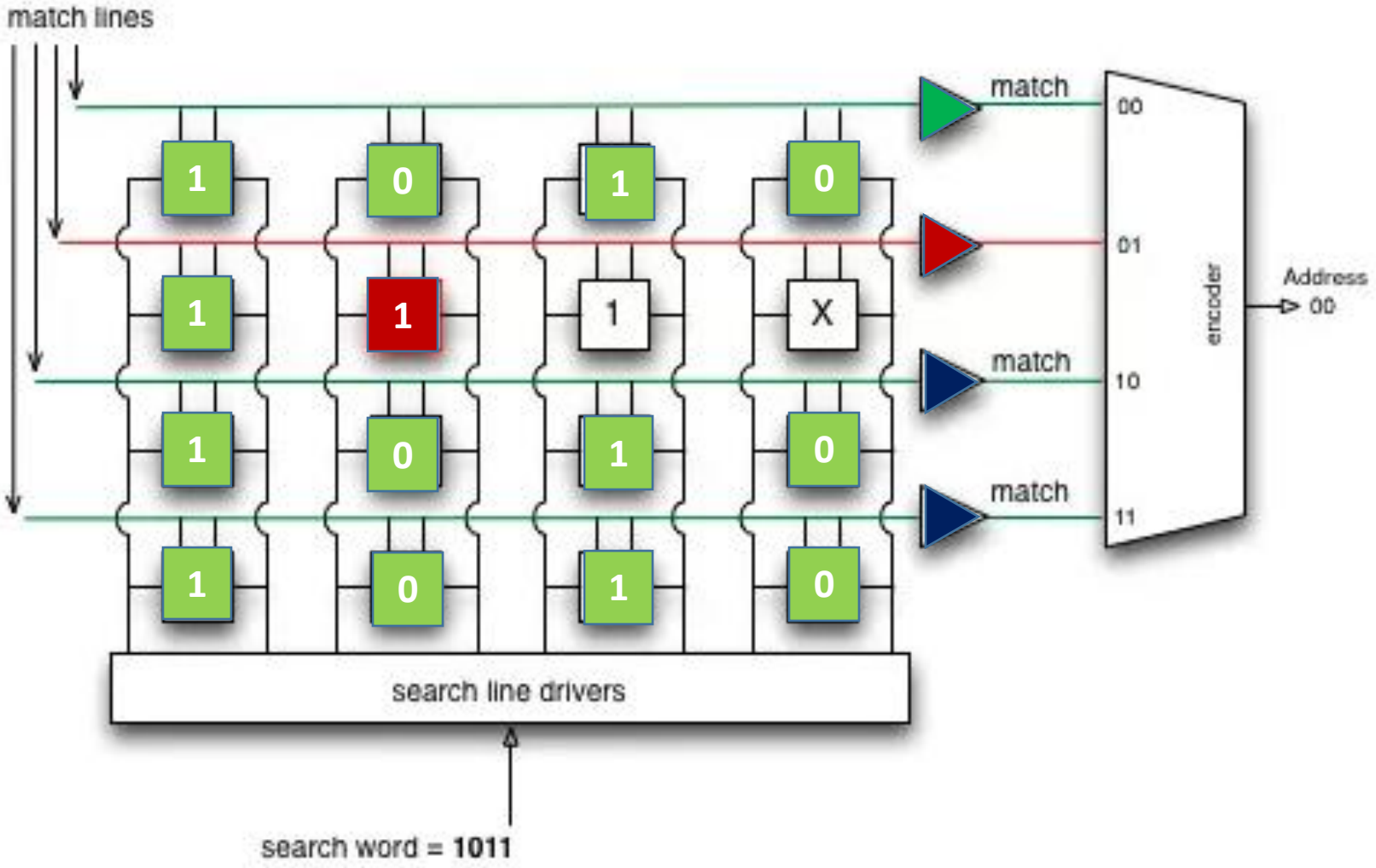


Let us try to match 1010

Basic TCAM Architecture

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

Image Source: http://thenetworksherpa.com/tcam-in-the-forwarding-engine/



Let us try to match 1010

Virtualizing Data and Control Functionalities

Part I: Network Namespace and Ethernet Bridge

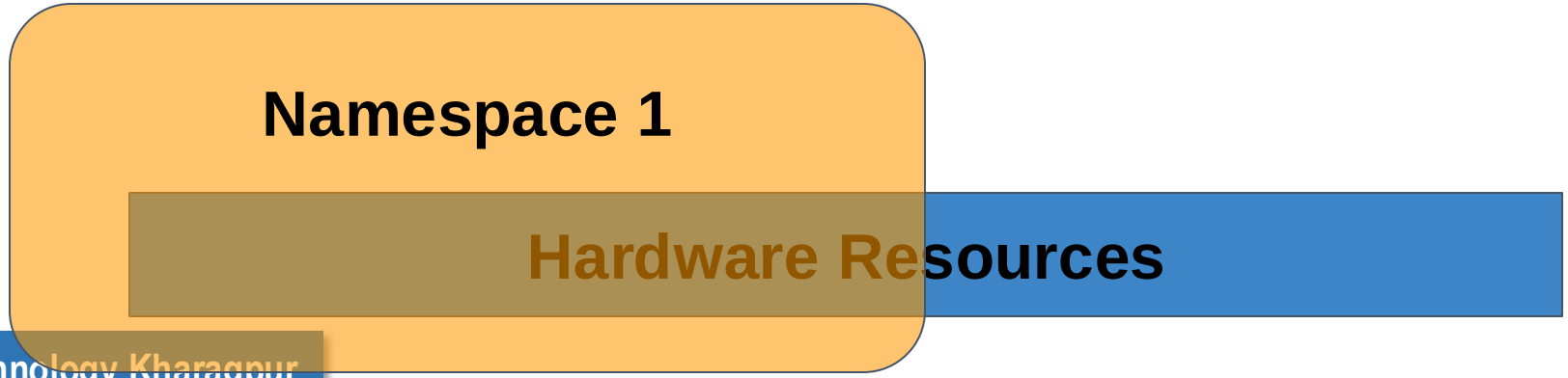
What is a Namespace?

- In computing, a namespace is a set of signs (or names) used to uniquely identify or refer objects of various kinds. (Source: Wikipedia)
- **Linux namespace**
  - Partition kernel resources
  - One set of processes observes one set of resources, while another set of processes observes a different set of resources

Hardware Resources

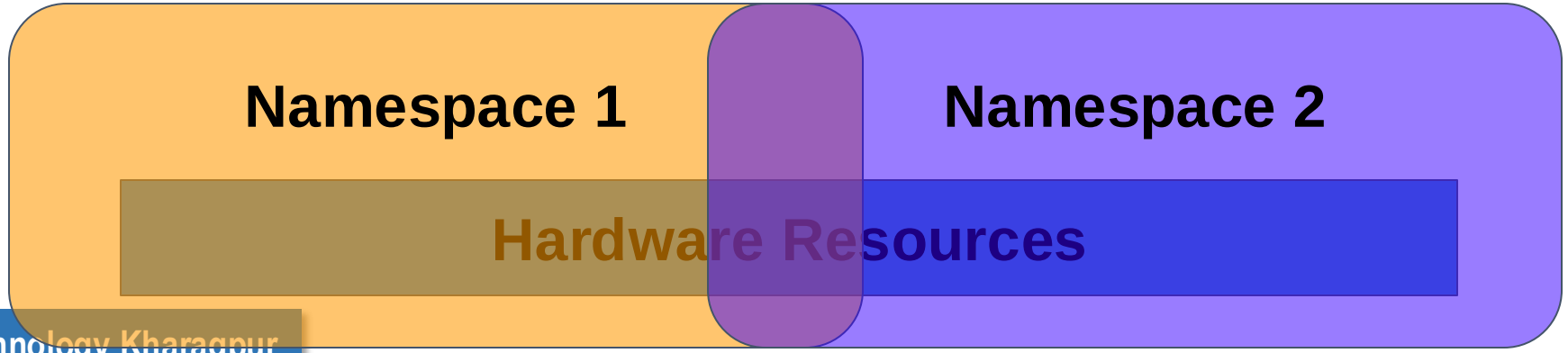
What is a Namespace?

- In computing, a namespace is a set of signs (or names) used to uniquely identify or refer objects of various kinds. (Source: Wikipedia)
- **Linux namespace**
  - Partition kernel resources
  - One set of processes observes one set of resources, while another set of processes observes a different set of resources



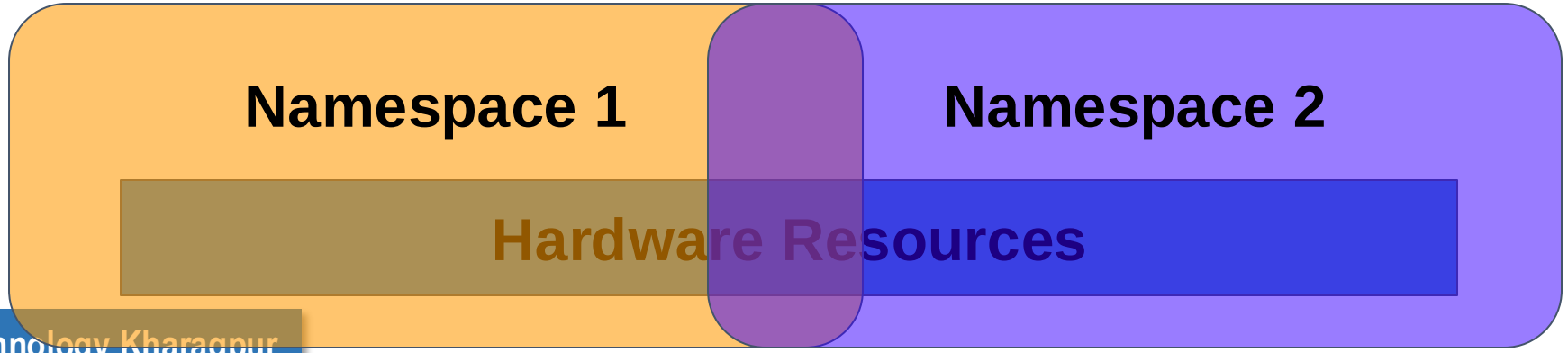
What is a Namespace?

- In computing, a namespace is a set of signs (or names) used to uniquely identify or refer objects of various kinds. (Source: Wikipedia)
- **Linux namespace**
  - Partition kernel resources
  - One set of processes observes one set of resources, while another set of processes observes a different set of resources

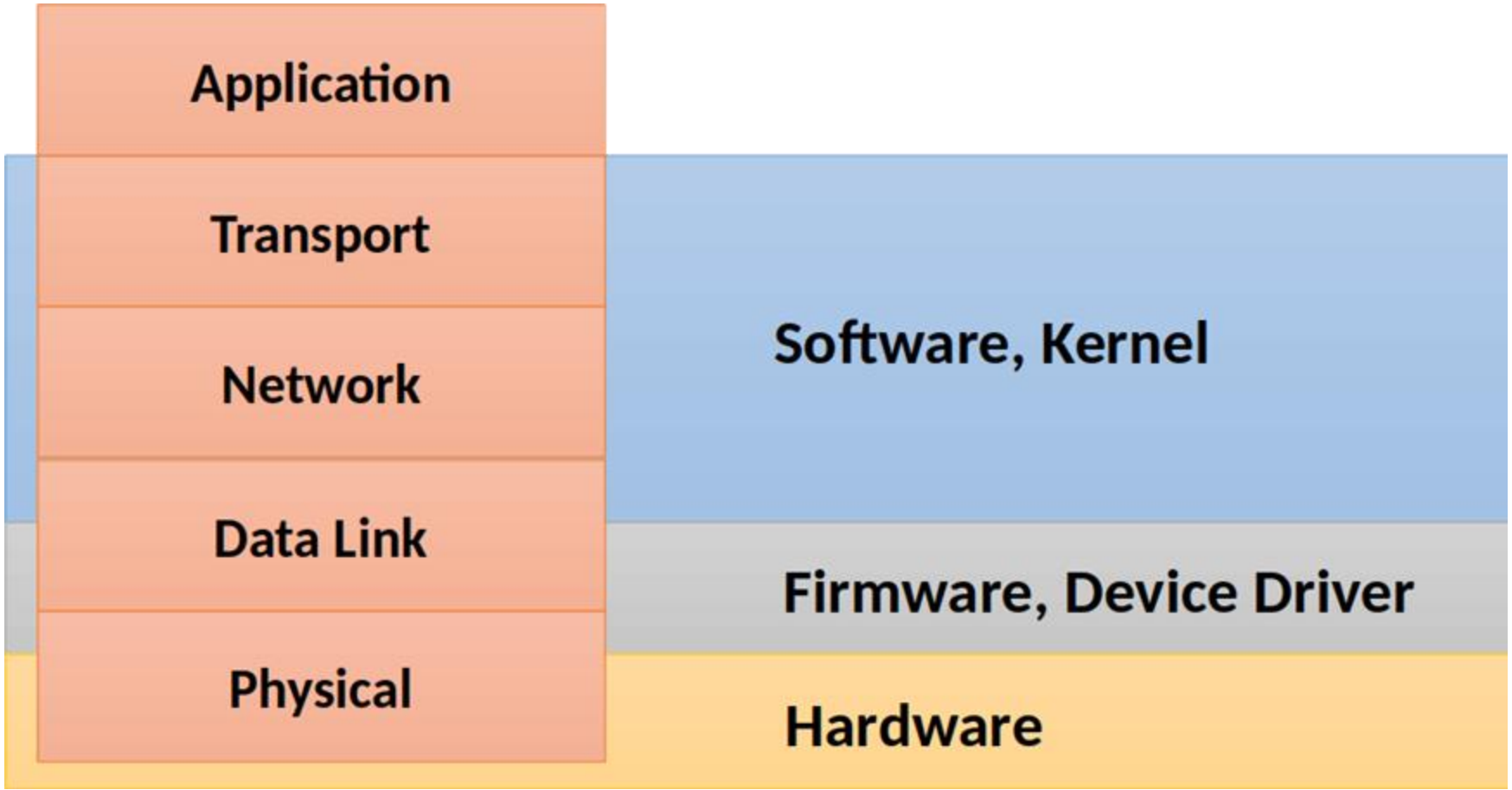


What is a Namespace?

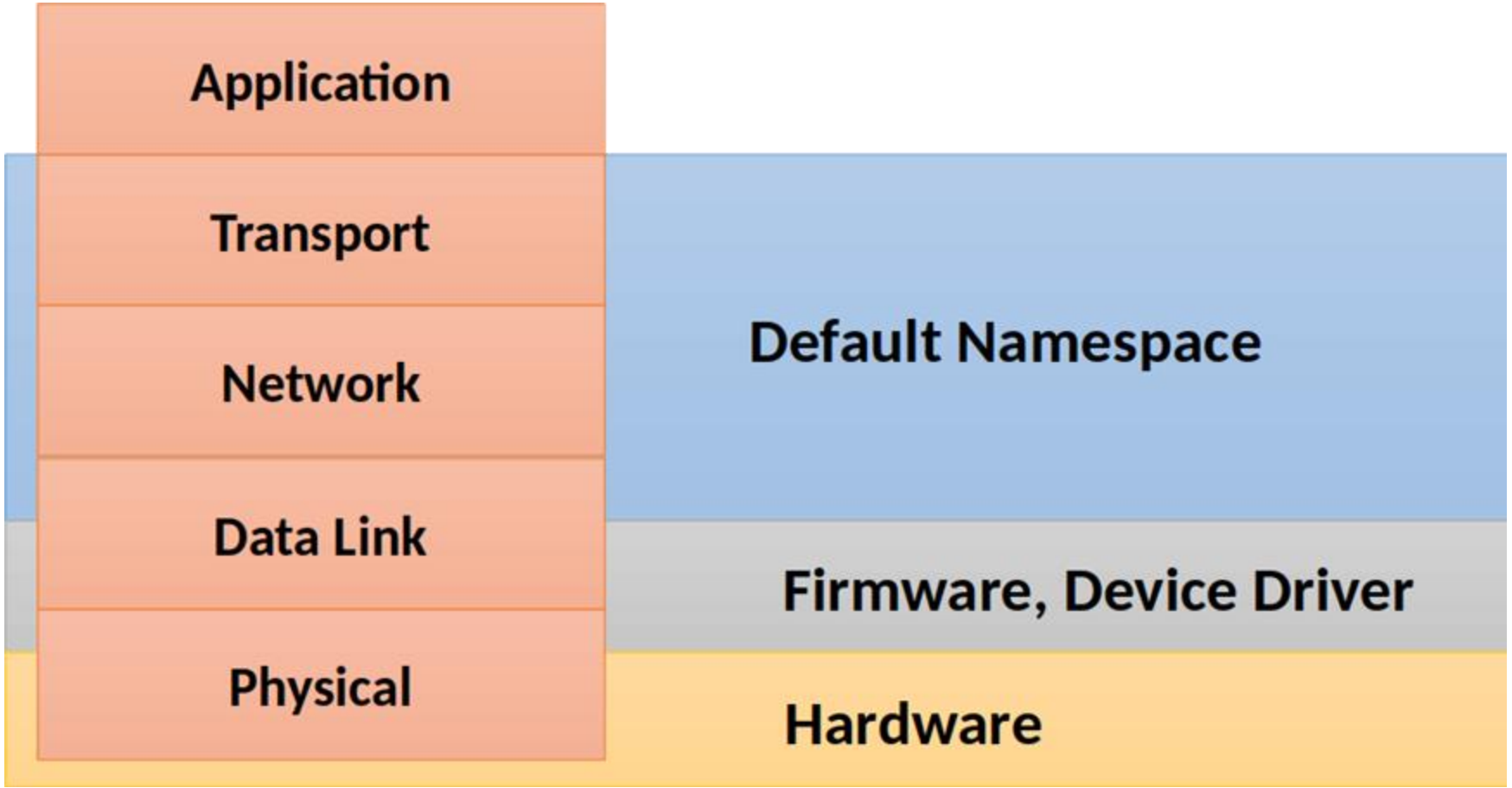
- In computing, a namespace is a set of signs (or names) used to uniquely identify or refer objects of various kinds. (Source: Wikipedia)
- **Linux namespace**
  - Partition kernel resources
  - One set of processes observes one set of resources, while another set of processes observes a different set of resources
- Are used to provide isolation or sandboxing
  - Virtualization of kernel resources (Linux containers)



Network Protocol Stack: Namespace for Networks

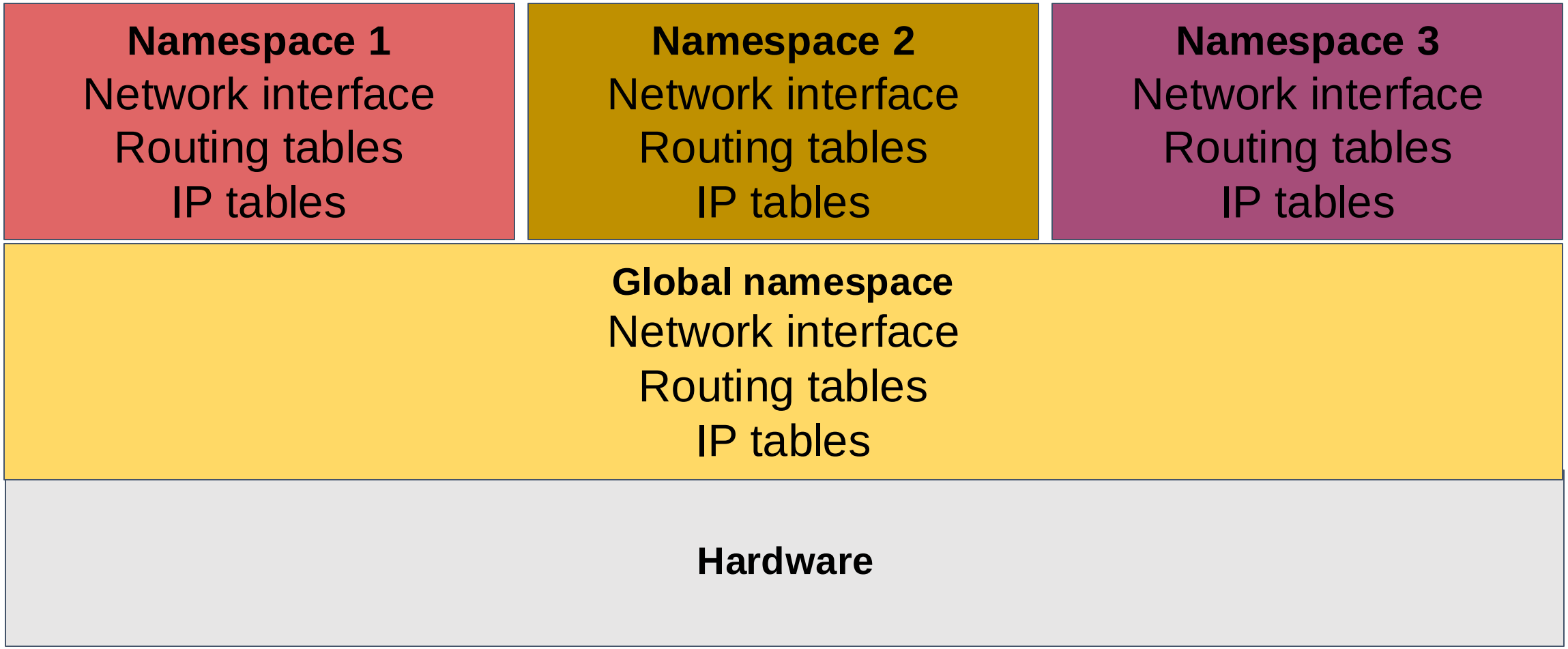


Network Protocol Stack: Namespace for Networks



Network Protocol Stack: Namespace for Networks

Network Protocol Stack: Namespace for Networks



Create a Private Network using Network Namespace

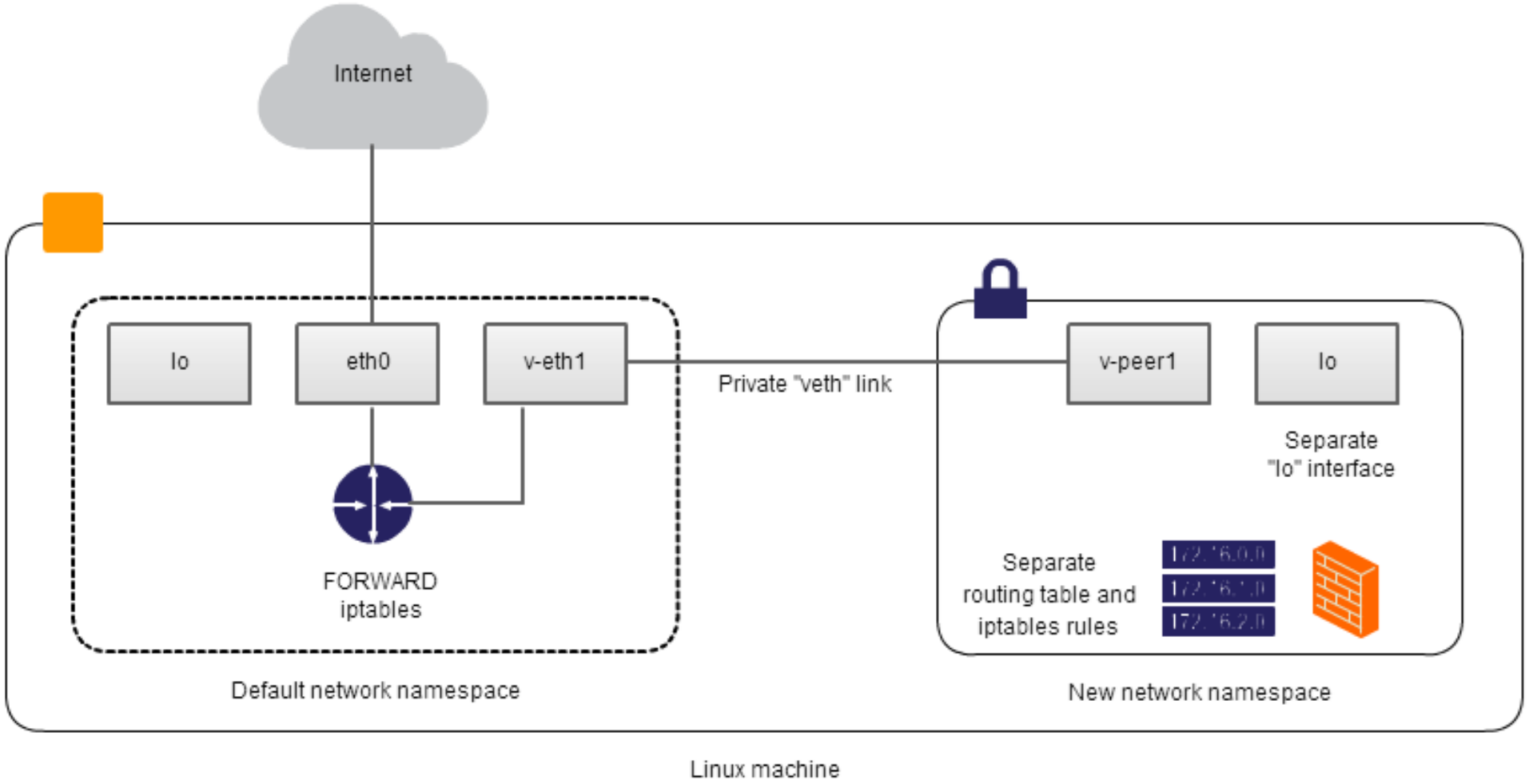
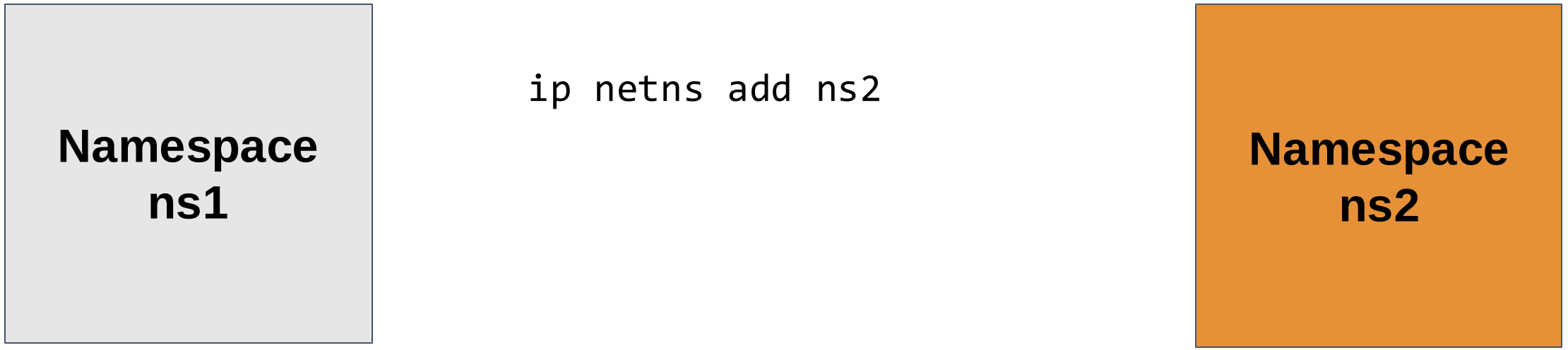


Image source: https://blog.famzah.net/2014/06/05/private-networking-per-process-in-linux/

Interconnecting Network Namespace through veth


```
ip netns add ns1
```

Interconnecting Network Namespace through veth

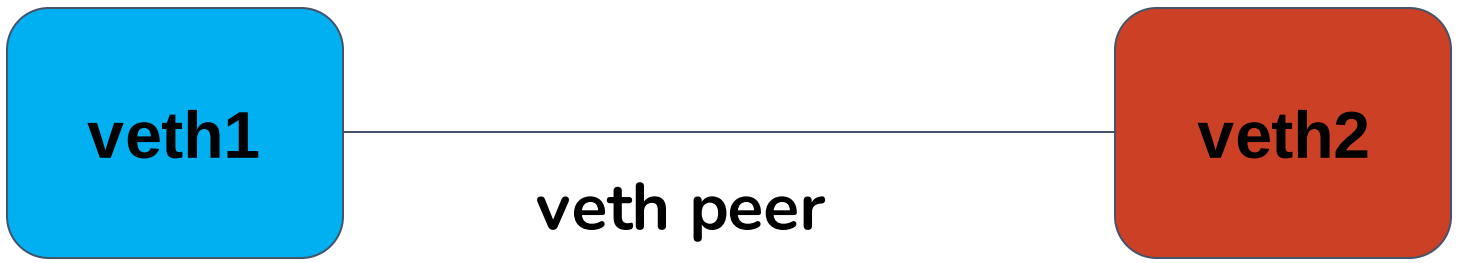


Interconnecting Network Namespace through veth

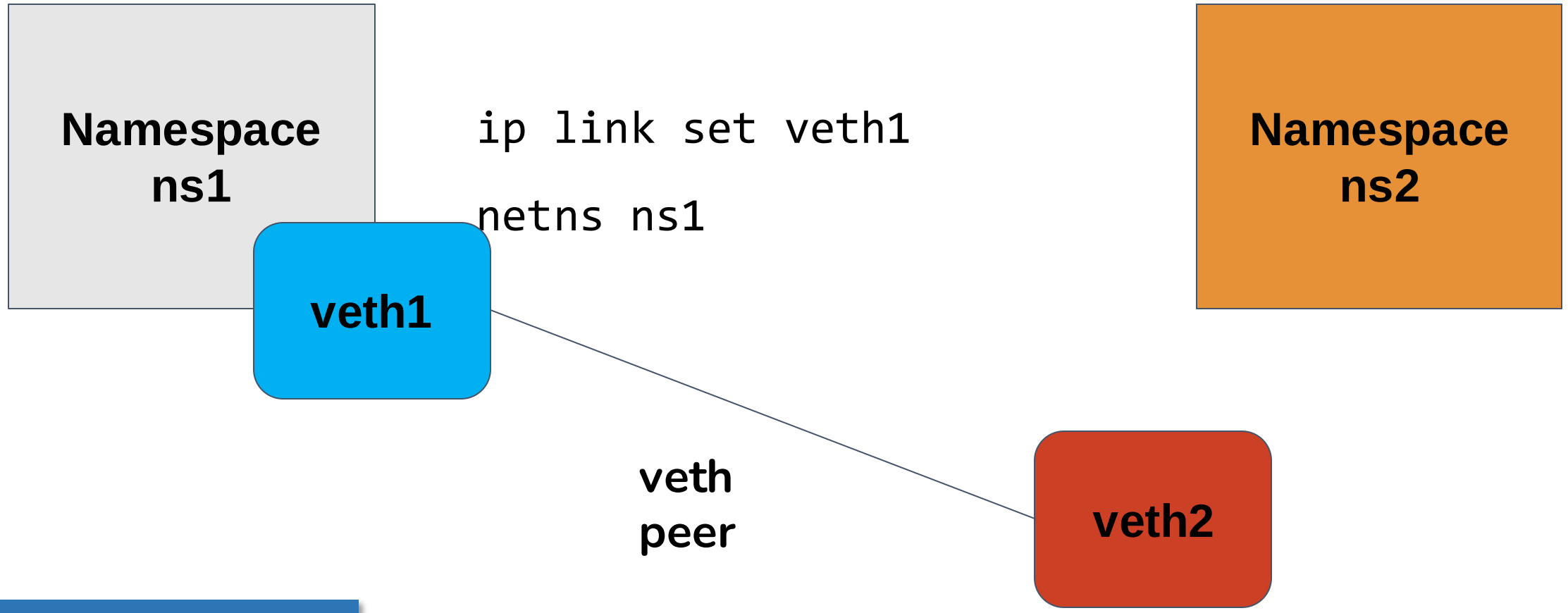
Namespace ns1

```
ip link add veth1 type
veth peer name veth2
```

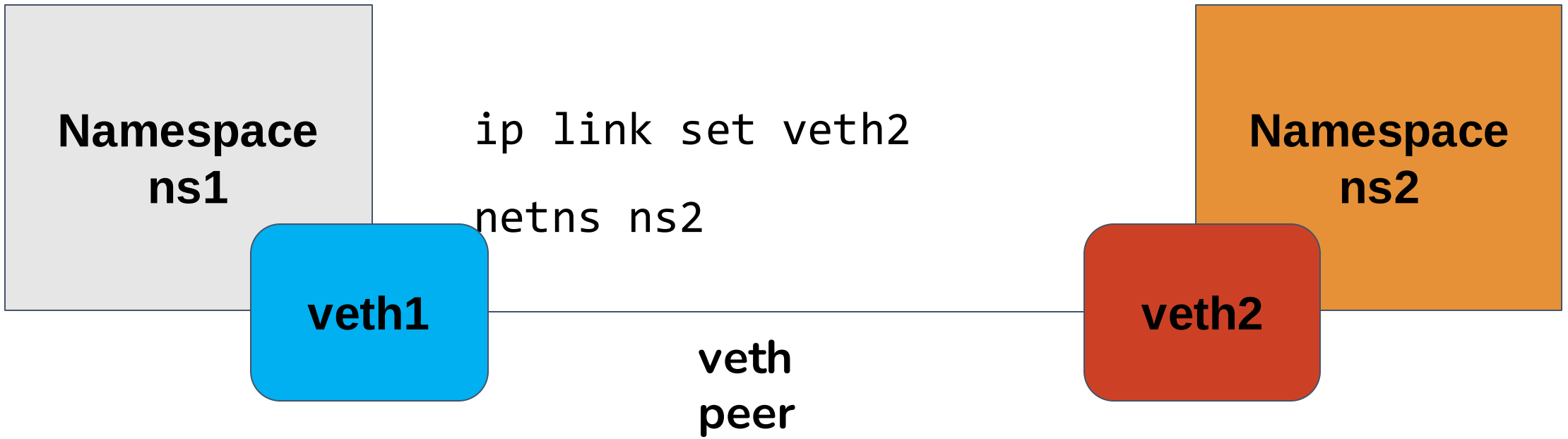
Namespace ns2



Interconnecting Network Namespace through veth



Interconnecting Network Namespace through veth



Next, you have to configure the IP addresses to the interfaces

Linux Ethernet Bridge

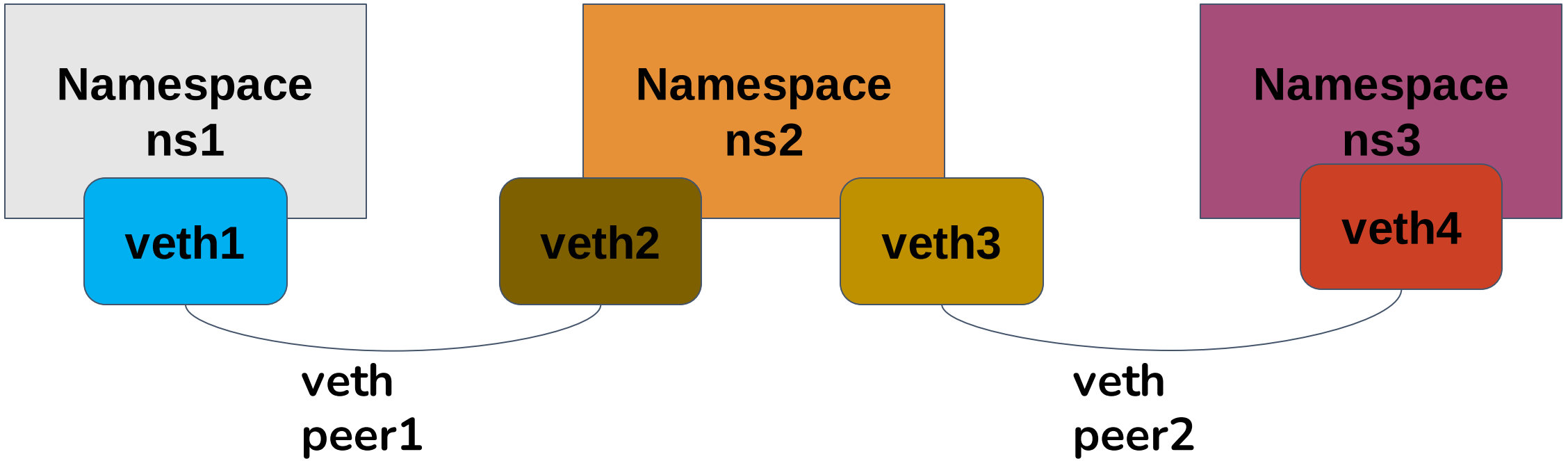
Namespace ns1

Namespace ns2

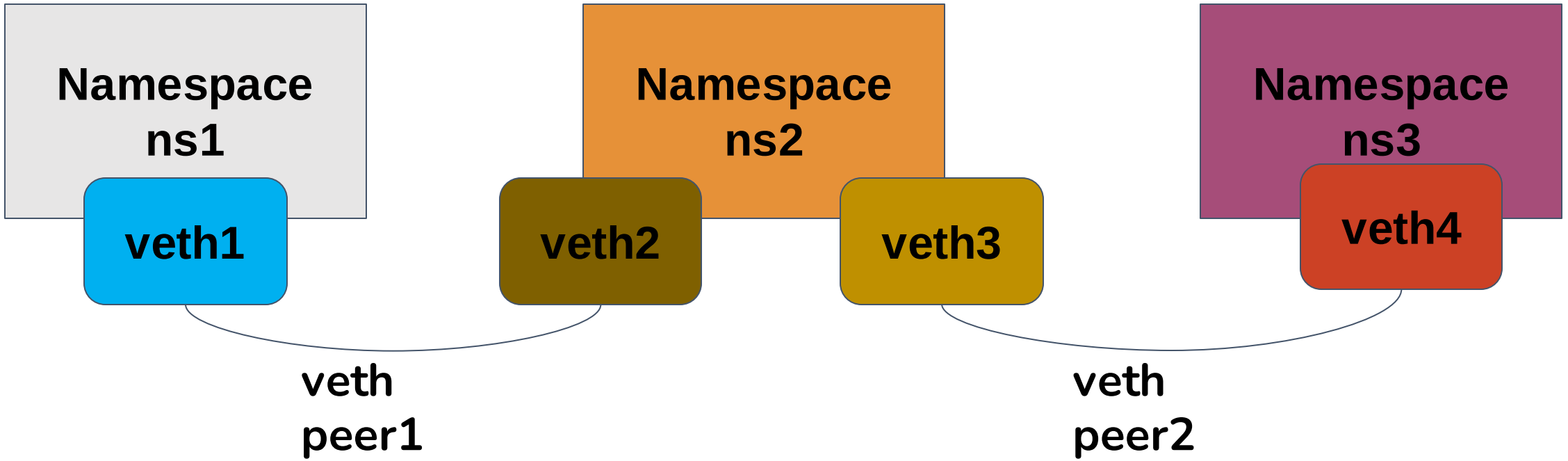
Namespace ns3

How can you connect these three namespaces?

Linux Ethernet Bridge

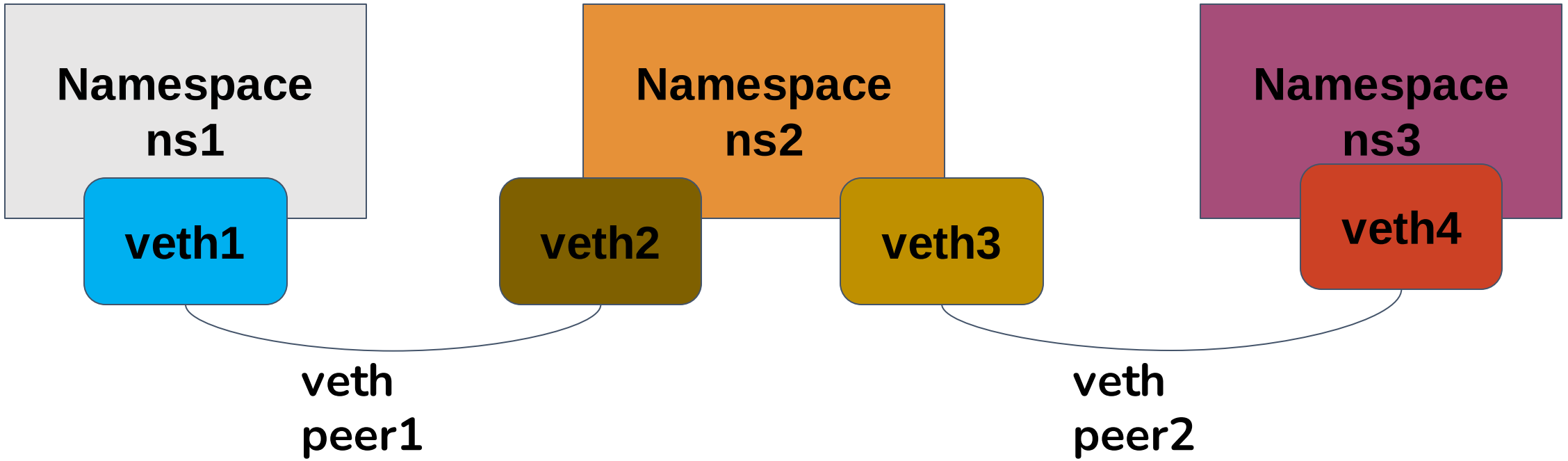


Linux Ethernet Bridge



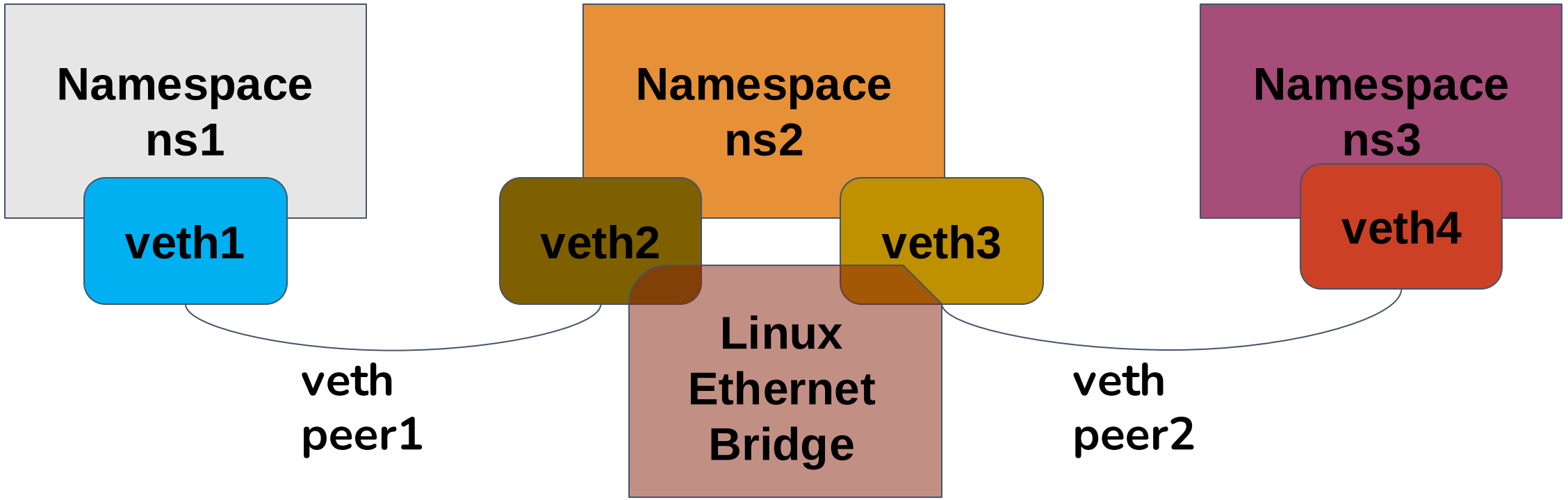
Is this complete?

Linux Ethernet Bridge

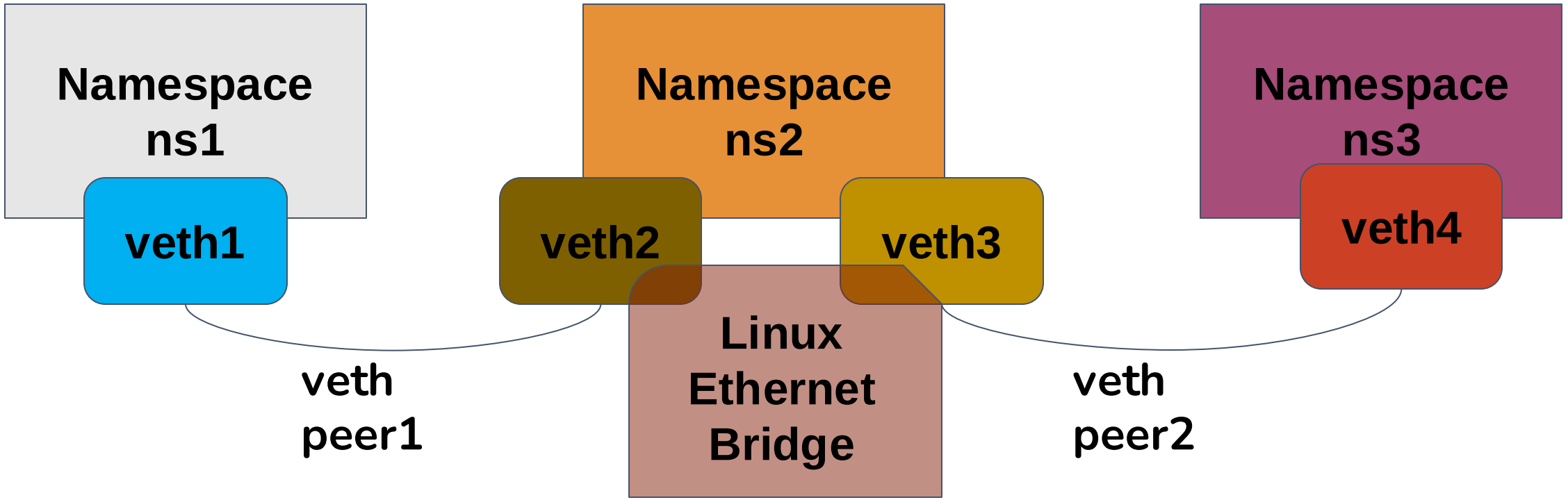


How will you forward the packets from veth2 to veth3?

Linux Ethernet Bridge



Linux Ethernet Bridge

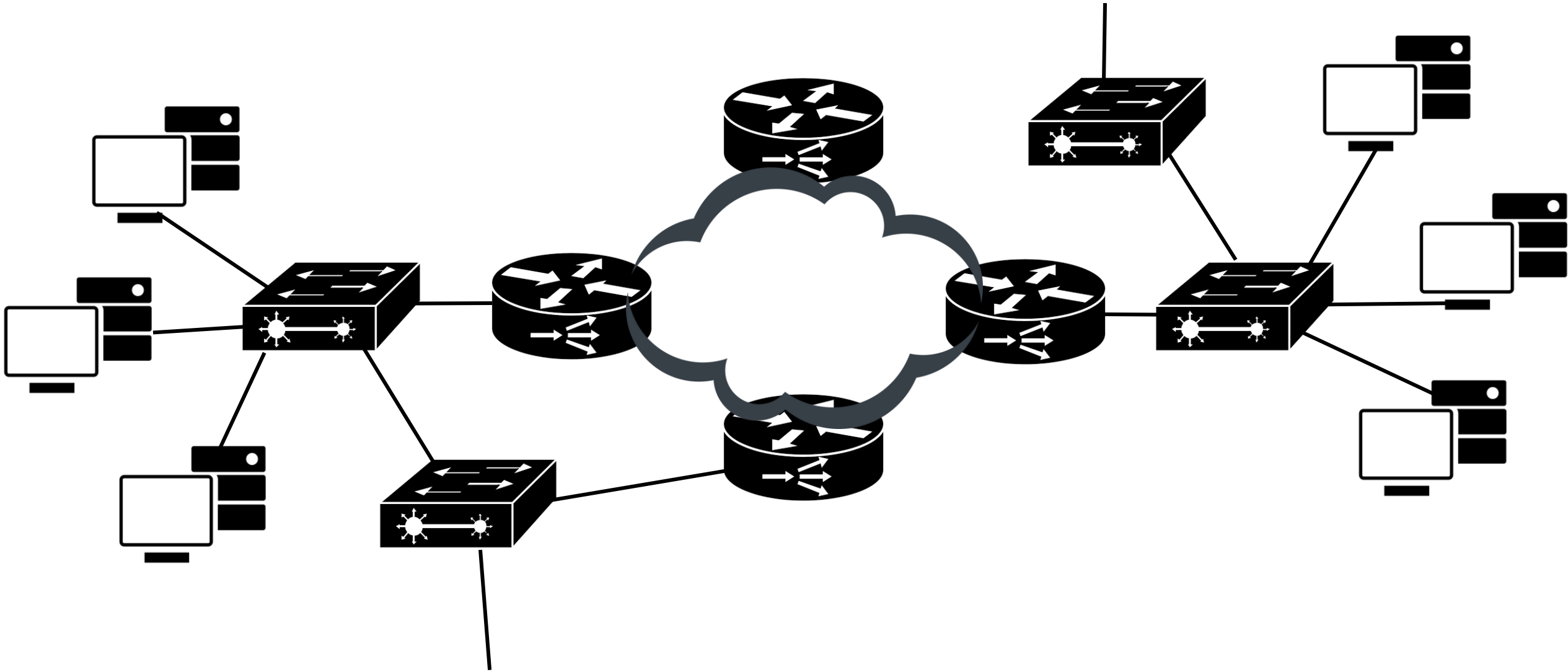


Check `brctl`

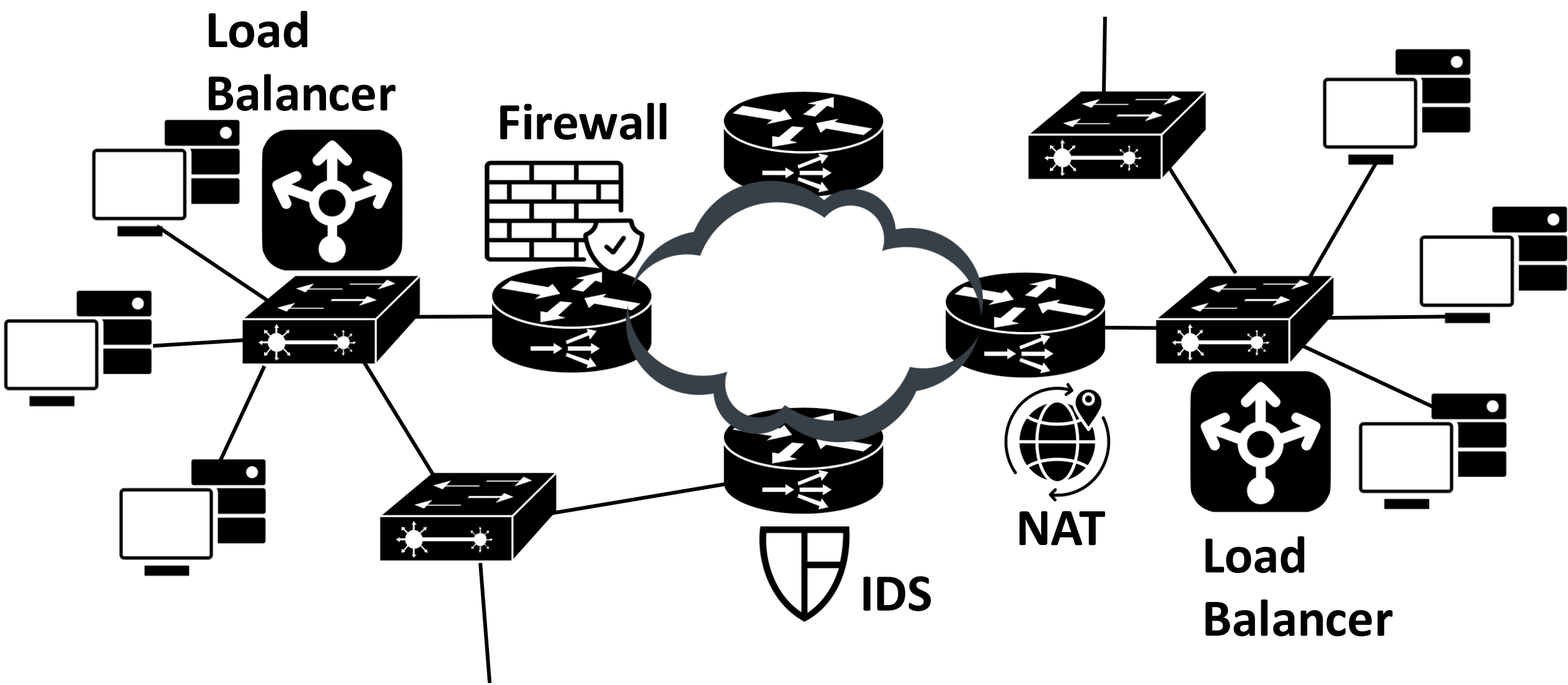
Virtualizing Data and Control Functionalities

How Do We Virtualize Network Functions?

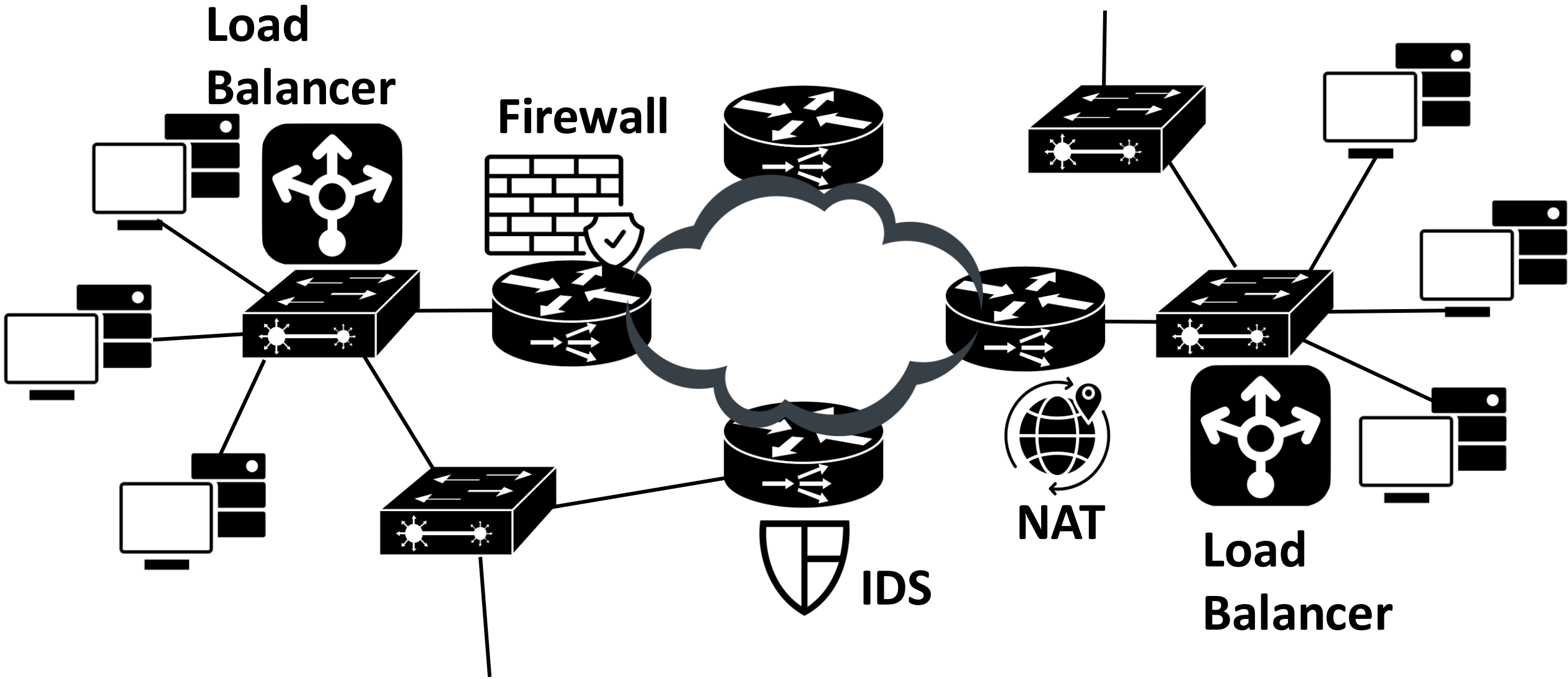
The Network that We have Studied in CS31204



The Network in Reality



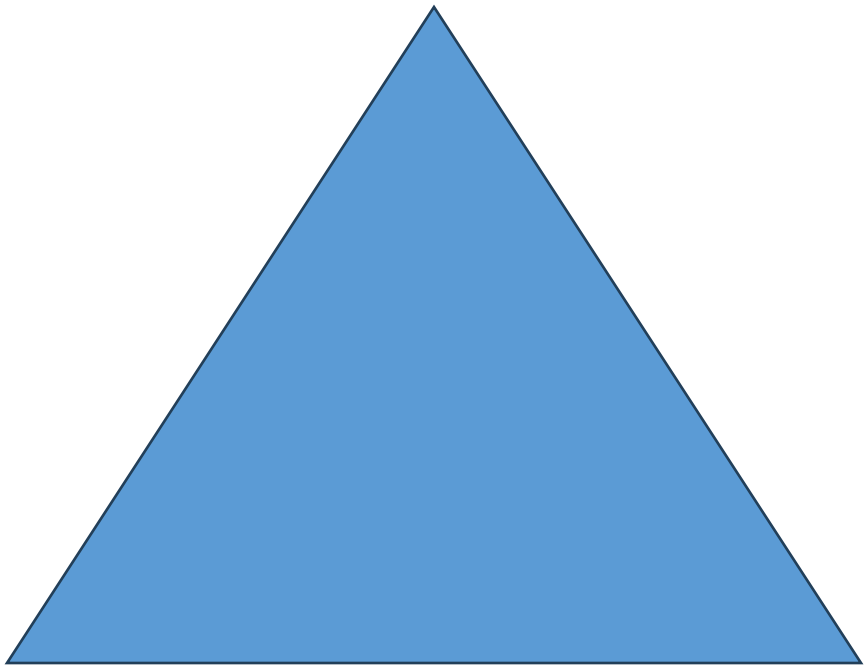
The Network in Reality



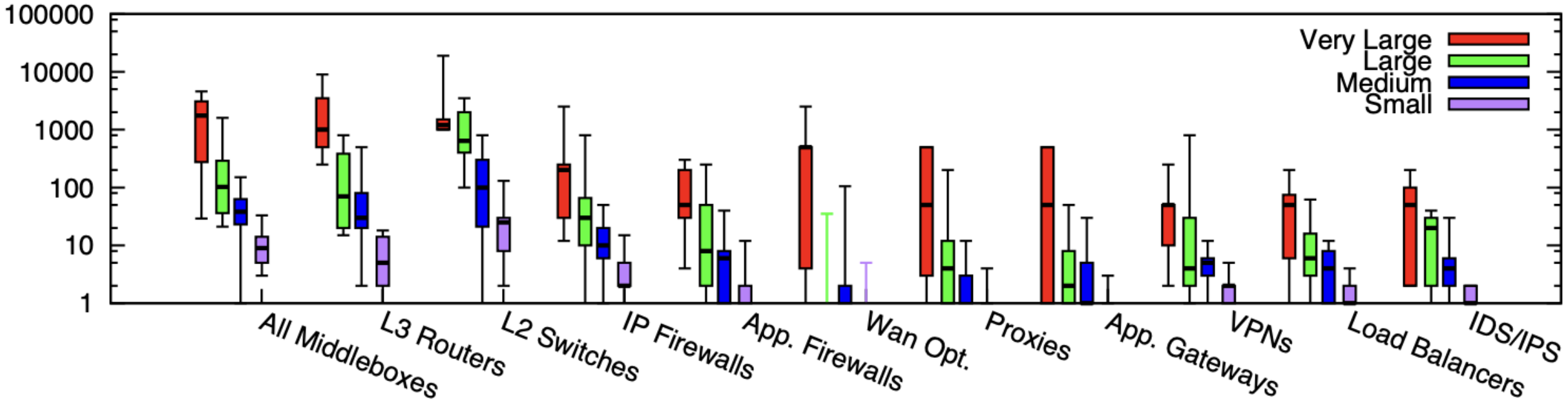
Lots of in-network processing: Middleboxes

Network Management has become Complicated over Time …

- New applications
- New devices
- New policies
- New threats



Middleboxes Kills Network Performance!



Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale

Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service

Justine Sherry
UC Berkeley

Shaddi Hasan
UC Berkeley

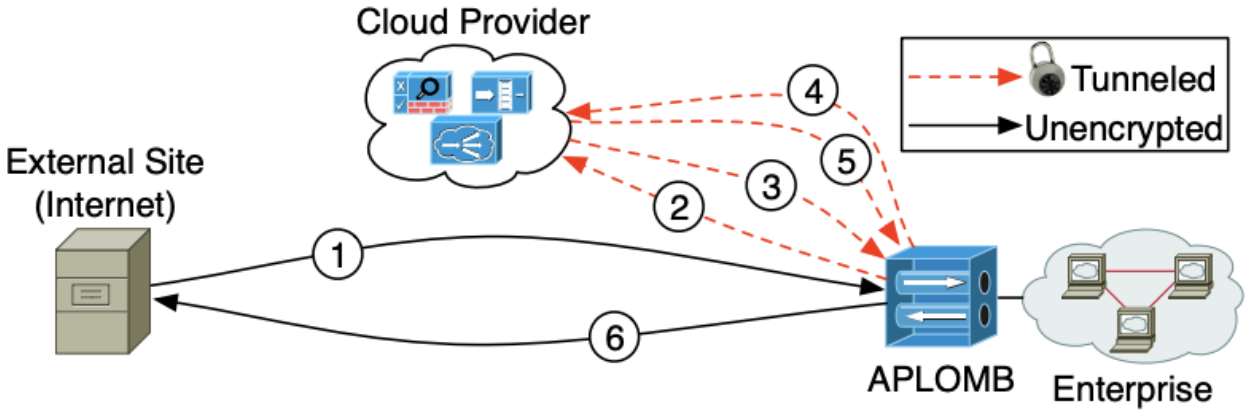
Colin Scott
UC Berkeley

Arvind Krishnamurthy
University of Washington

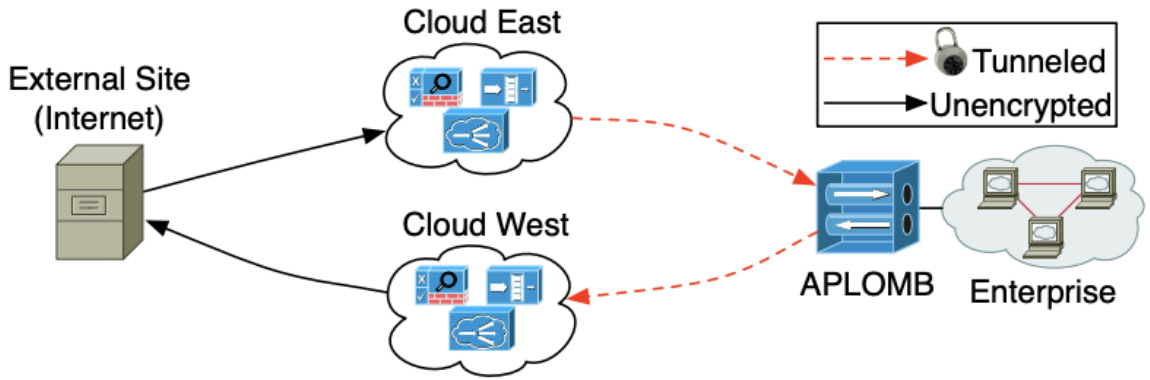
Sylvia Ratnasamy
UC Berkeley

Vyas Sekar
Intel Labs

Middleboxes Kills Network Performance!



(a) "Bounce" redirection inflates latency.



(b) Direct IP redirection in multi-PoP deployments cannot ensure that bidirectional traffic traverses the same PoP.

Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service

Justine Sherry
UC Berkeley

Shaddi Hasan
UC Berkeley

Colin Scott
UC Berkeley

Arvind Krishnamurthy
University of Washington

Sylvia Ratnasamy
UC Berkeley

Vyas Sekar
Intel Labs

Pain-points for the Network Administrators

- Middlebox management is hard – increases both capex and opex

	Misconfig.	Overload	Physical/Electric
Firewalls	67.3%	16.3%	16.3%
Proxies	63.2%	15.7%	21.1%
IDS	54.5%	11.4%	34%

Fraction of network administrators who estimated misconfiguration, overload, or physical/electrical failure as the most common cause of middlebox failure.

Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service

Justine Sherry
UC Berkeley

Shaddi Hasan
UC Berkeley

Colin Scott
UC Berkeley

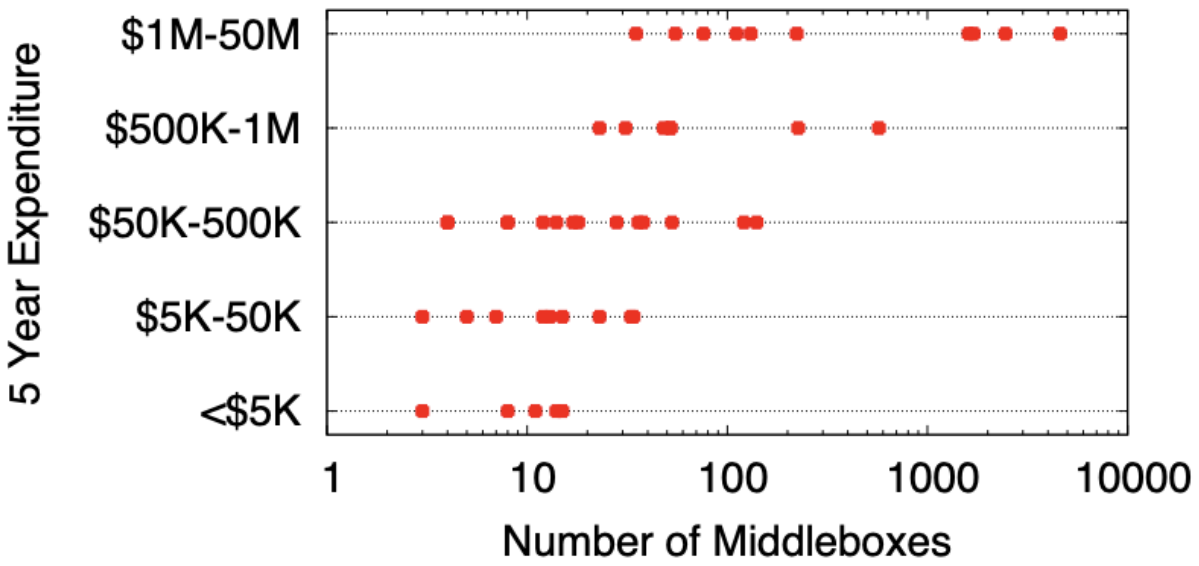
Arvind Krishnamurthy
University of Washington

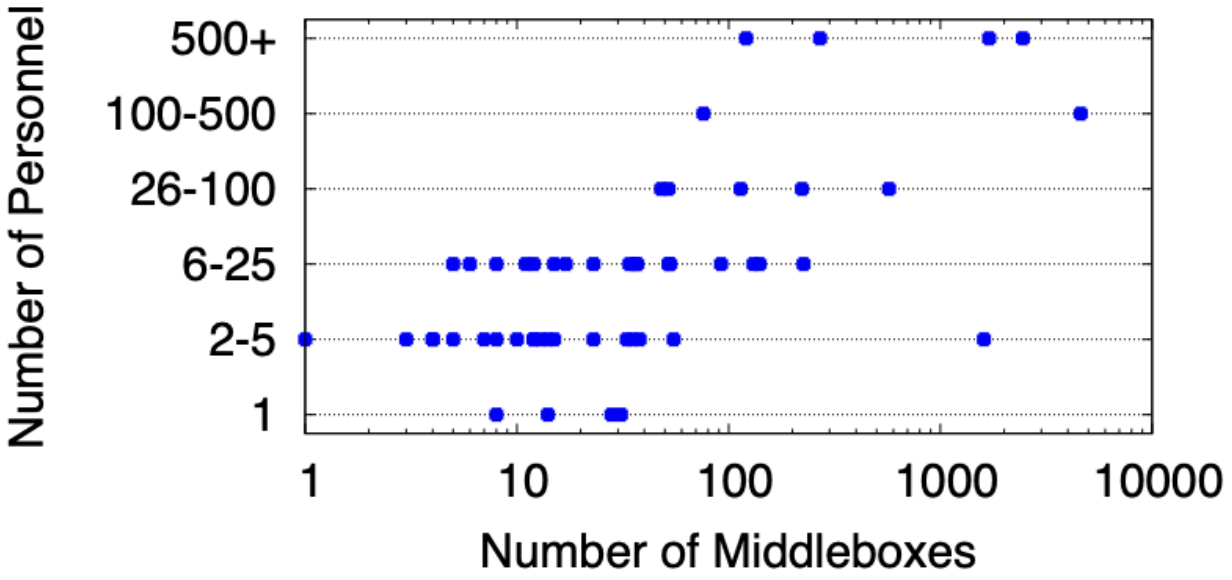
Sylvia Ratnasamy
UC Berkeley

Vyas Sekar
Intel Labs

Pain-points for the Network Administrators

- Middlebox management is hard – increases both capex and opex





Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service

Justine Sherry
UC Berkeley

Shaddi Hasan
UC Berkeley

Colin Scott
UC Berkeley

Arvind Krishnamurthy
University of Washington

Sylvia Ratnasamy
UC Berkeley

Vyas Sekar
Intel Labs

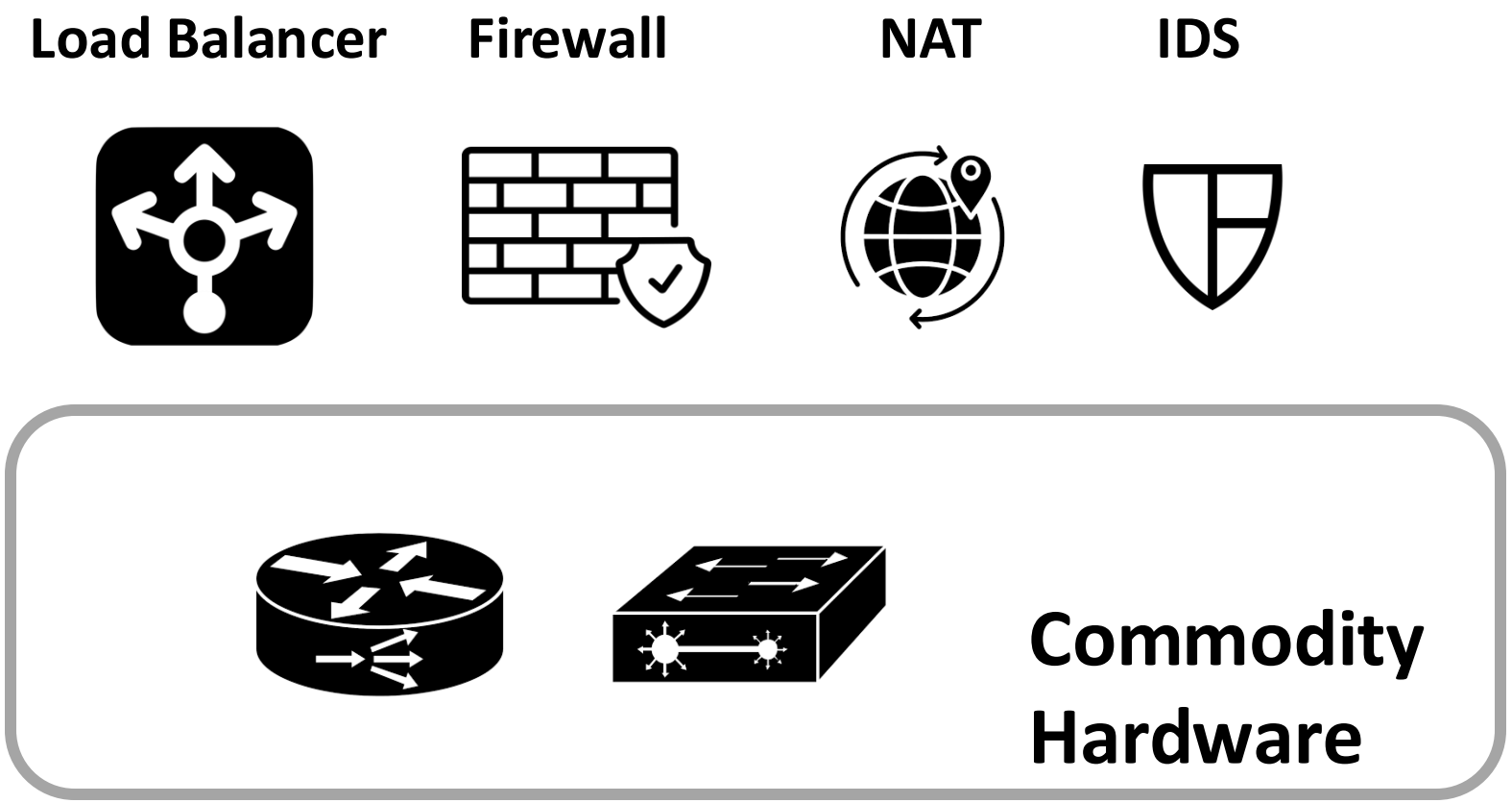
Network Virtualization: Core Idea

- Networking to get the same benefit as of cloud/IT world
  - Virtualization: Use the same hardware for multiple purposes – reduces capex
  - Consolidation: A single point of management (think about the cloud service providers managing all your computing resources) -- reduces opex

- Network-wide controller to control the management functionalities
  - Software-defined Networking (SDN) -- *we'll see this later in details*

Hradware/Software Abstraction

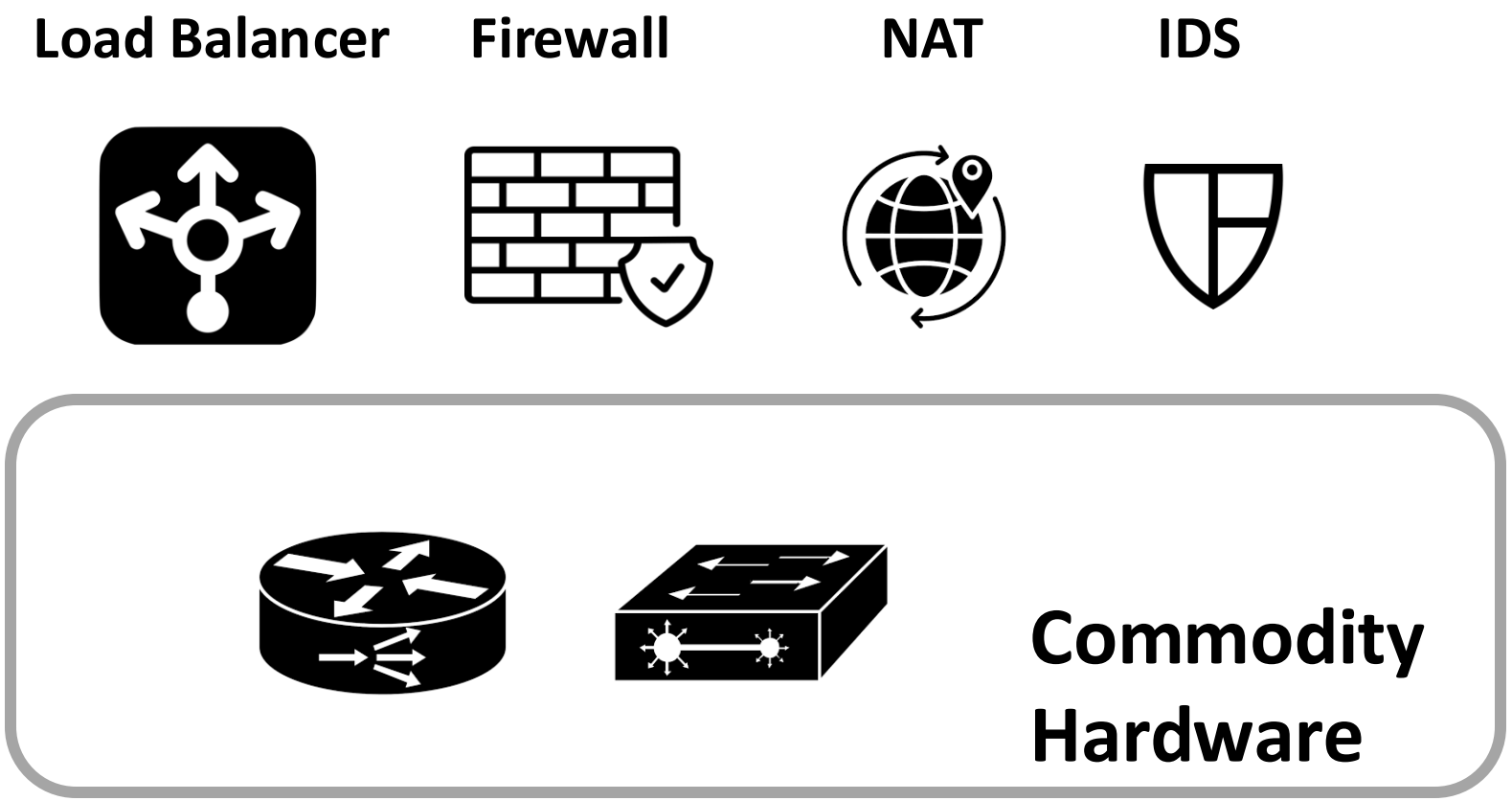
- **Decouple** hardware and software rather than today's specialized boxes for each of the separate functions



Hradware/Software Abstraction

- **Decouple** hardware and software rather than today's specialized boxes for each of the separate functions

Reduces CAPEX

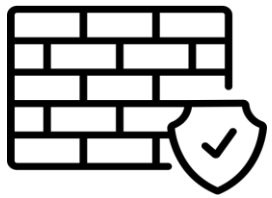


Hradware/Software Abstraction

- **Decouple** hardware and software rather than today's specialized boxes for each of the separate functions

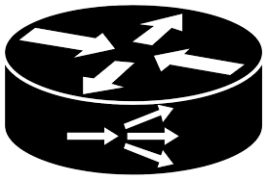
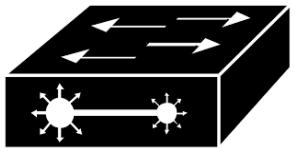
App Filter | Load Balancer | Firewall | NAT | IDS





Reduces CAPEX

Enables extensibility

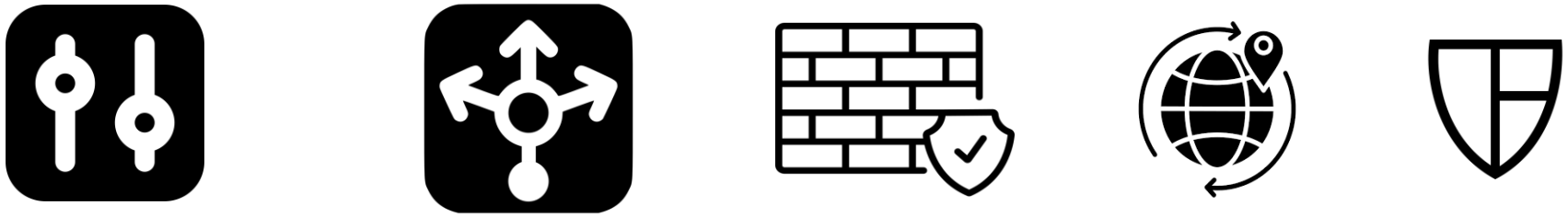



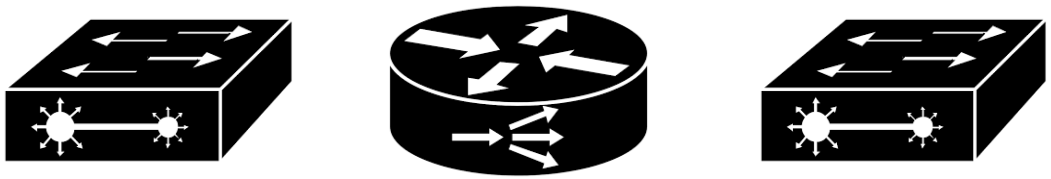
Commodity Hardware

Hradware/Software Abstraction

- **Decouple** hardware and software rather than today's specialized boxes for each of the separate functions

App Filter | Load Balancer | Firewall | NAT | IDS





Commodity Hardware

Reduces CAPEX

Enables extensibility

Flexible resource allocation

Network Function Virtualization

- Virtual machines (VMs) to implement various network functions
  - Application functionalities (NATs, Firewalls, IDS, Proxies, …)
  - Network functionalities (Routing algorithms, Forwarding policies, Security functionalities – authentication, authorization, access control)

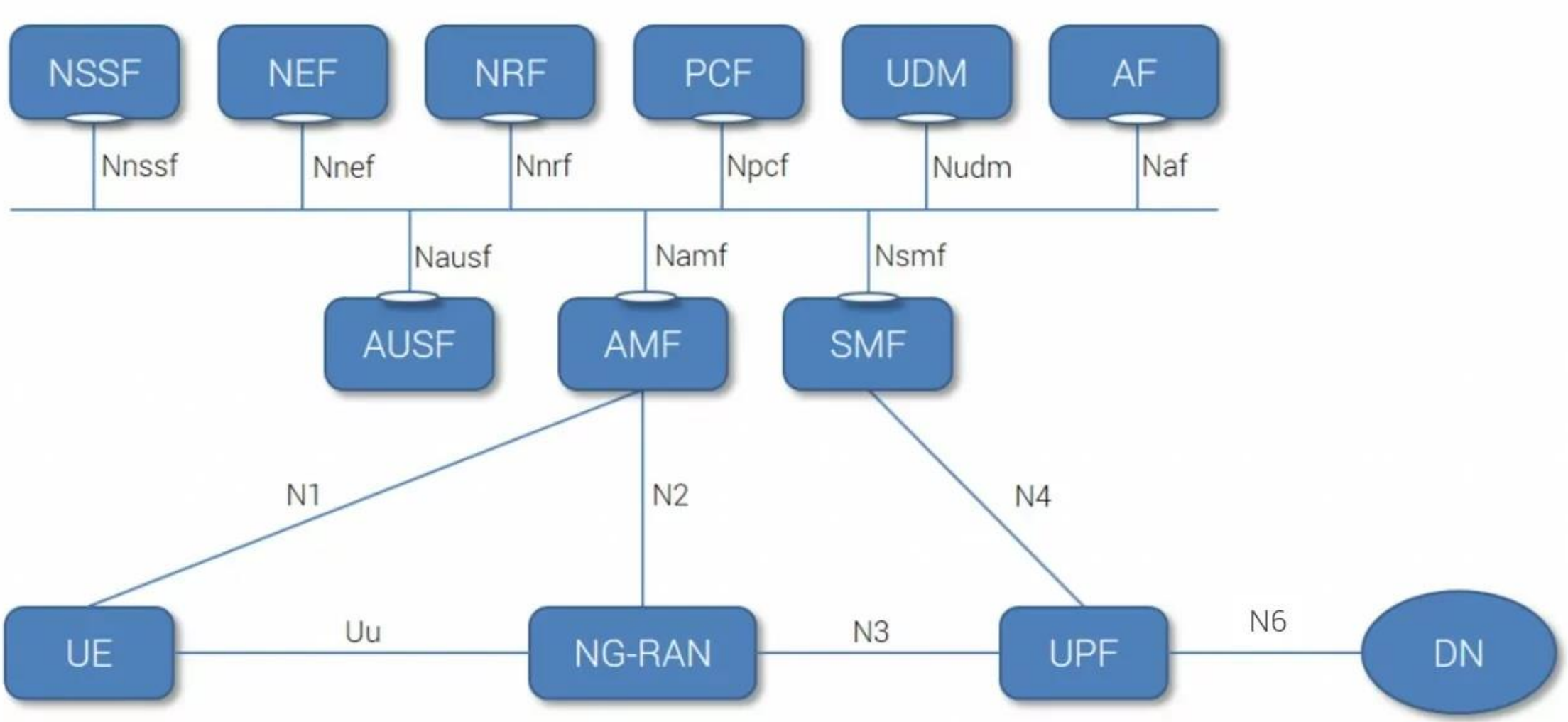
- Even lightweight containers (like dockers, Kubernetes, etc.) can be used to spawn network functions
  - All advantages of virtualization (quick provisioning, scalability, reduced capex and opex, mobility, etc.)

- Standardization of APIs for communication across the VNFs and across VNFs and hosts (ETSI NFV Release 6, Started 2023)

Example: Mobile Network Functions

- User plane function (UPF) to support forwarding and routing
  - Switches (OpenvSwitch -- https://www.openvswitch.org/)
  - Routers (Click -- https://github.com/kohler/click)
- Access and Mobility Management Function (AMF)
- Session Management Function (SMF)
- Policy Control Function (PCF)
- Authentication Server Function (AUSF)
- Unified Data Management (UDM) - Authentication and Key Agreement (AKA)
- Network Exposure Function (NEF)
- Network Slice Selection Function (NSSF)
- ...

Example: Service-based Architecture of 5G System



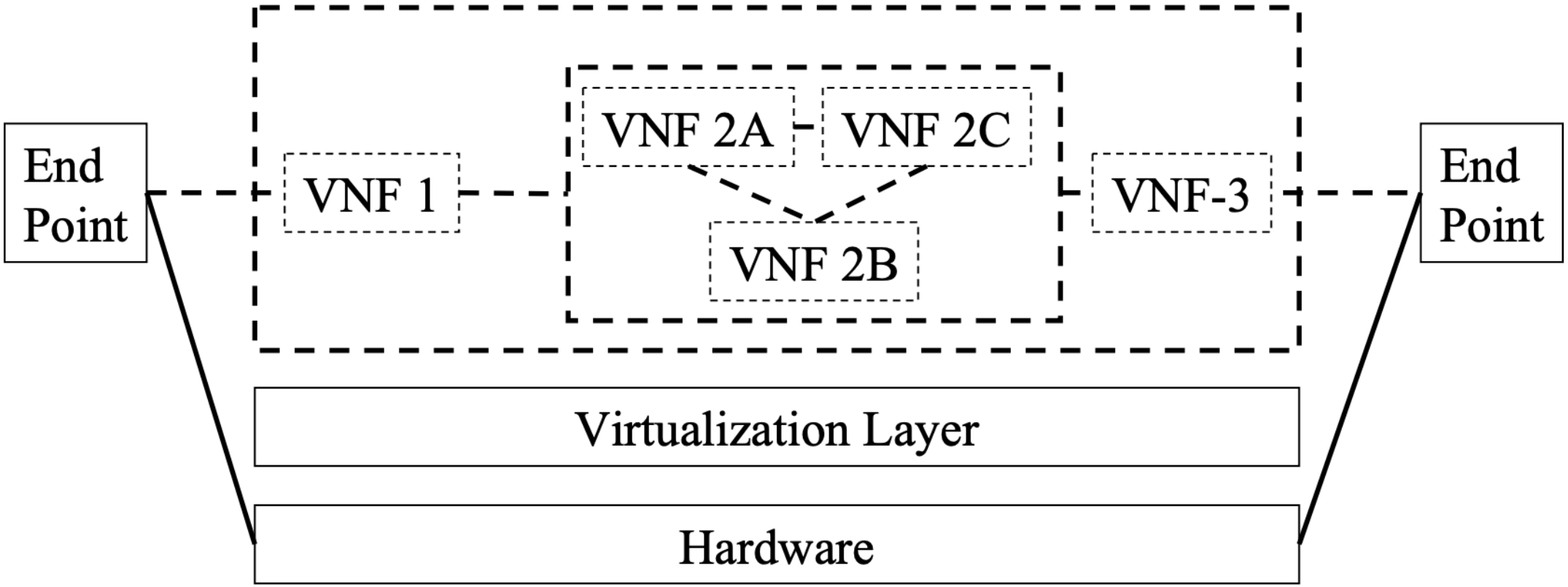
3GPP TS 23.501 V15.0.0 (2017-12) *System Architecture for the 5G System (Stage 2)*

Key NFV Concepts

- **Network Function (NF):** Functional building block with a well-defined interfaces and well-defined functional behavior
- **Virtualized Network Function (VNF):** Software implementation of NF that can be deployed in a virtualized infrastructure
- **VNF Set:** Connectivity between VNFs is not specified, e.g., residential gateways
- **VNF Forwarding Graph:** Service chain when network connectivity order is important, e.g., firewall, NAT, load balancer
- **NFV Infrastructure:** Hardware and software required to deploy, mange and execute VNFs including computation, networking, and storage.

Reference: ETSI, "Architectural Framework", http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

Network Forwarding Graph



Reference: ETSI, "Architectural Framework", http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

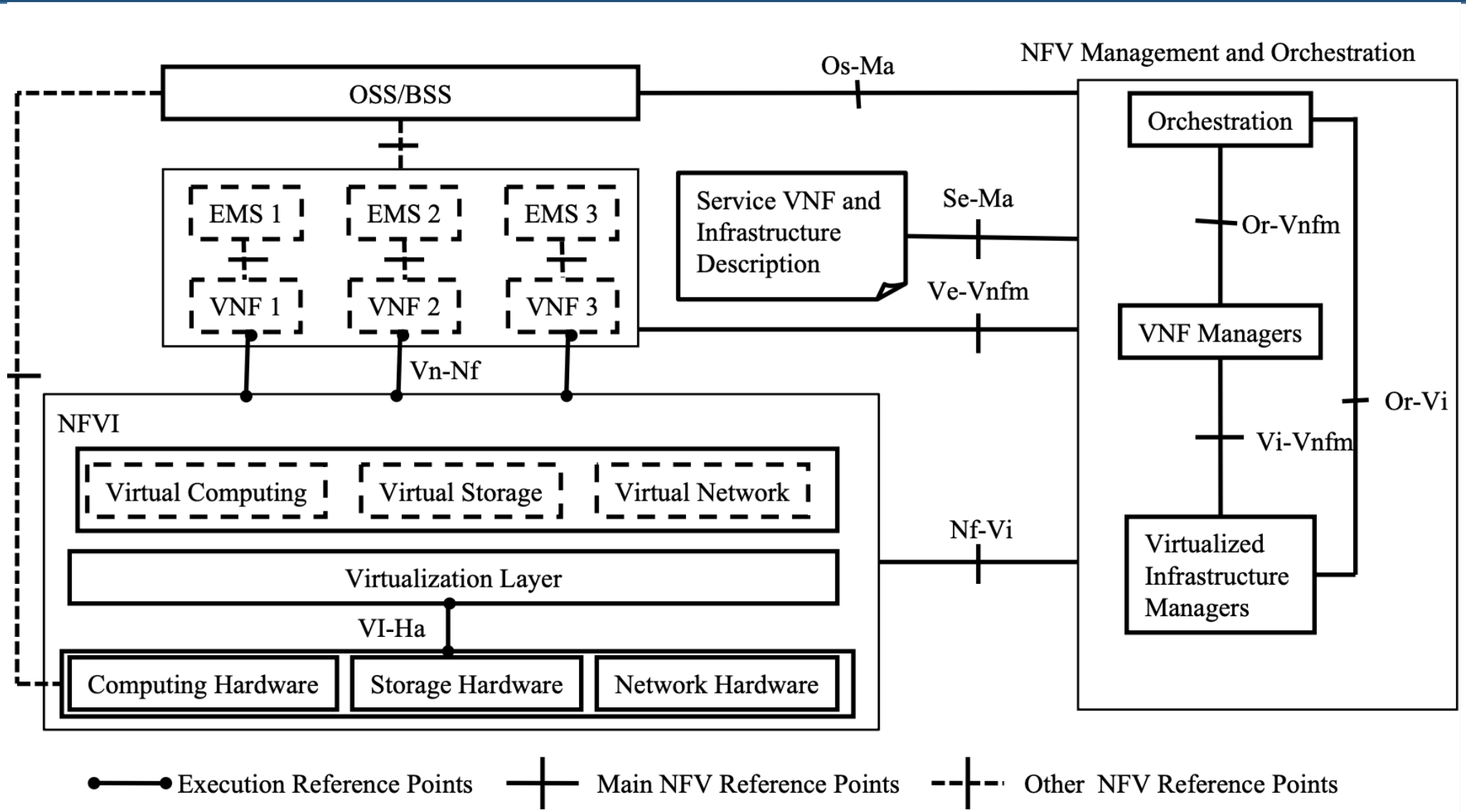
Key NFV Concepts

- **NFVI Point of Presence (PoP)**: Location of NFVI
- **NFVI-PoP Network**: Internal network
- **Transport Network**: Network connecting a PoP to other PoPs or external networks
- **VNF Manager**: VNF lifecycle management e.g., instantiation, update, scaling, query, monitoring, fault diagnosis, healing, termination
- **Virtualized Infrastructure Manager**: Management of computing, storage, network, software resources
- **Network Service**: A composition of network functions and defined by its functional and behavioral specification
- **NFV Service**: A network services using NFs with at least one VNF.

Key NFV Concepts

- **User Service**: Services offered to end users/customers/subscribers.
- **Deployment Behavior**: NFVI resources that a VNF requires, e.g., Number of VMs, memory, disk, images, bandwidth, latency
- **Operational Behavior**: VNF instance topology and lifecycle operations, e.g., start, stop, pause, migration, etc.
- **VNF Descriptor**: Deployment behavior + Operational behavior
- **NFV Orchestrator**: Automates the deployment, operation, management, coordination of VNFs and NFVI.
- **VNF Forwarding Graph**: Connection topology of various NFs of which at least one is a VNF

NFV Architecture



Reference: ETSI, "Architectural Framework", http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

NFV Reference Points (ETSI)

- Virtualization Layer-Hardware Resources (**VI-Ha**)
- VNF – NFVI (**Vn-Nf**)
- Orchestrator – VNF Manager (**Or-Vnfm**)
- Virtualized Infrastructure Manager – VNF Manager (**Vi-Vnfm**)
- Orchestrator – Virtualized Infrastructure Manager (**Or-Vi**)
- NFVI-Virtualized Infrastructure Manager (**Nf-Vi**)
- Operation Support System (OSS)/Business Support Systems (BSS) – NFV Management and Orchestration (**Os-Ma**)
- VNF/ Element Management System (EMS) – VNF Manager (**Ve-Vnfm**)
- Service, VNF and Infrastructure Description – NFV Management and Orchestration (**Se-Ma**): VNF Deployment template, VNF Forwarding Graph, service-related information, NFV infrastructure information

NFV: Summary

- In-network packet processing is expensive, particularly for middleboxes
  - However, middleboxes are the key for network innovations
- Virtualization at the network core makes management flexible, reducing capex and opex
  - NFV is the key for 5G/6G network core
  - Flexible APIs, Network slicing (creating multiple virtual networks on top of a shared physical infrastructure), App stack
- Triggers innovations and management optimizations over the classical protocol stack

Software Defined Networking (SDN)

The Networking Stack

- **Data Plane:** All activities involving network packets
  - Forwarding
  - Fragmentation and reassembly
  - Multicast and broadcast services – packet replication

- **Control Plane:** All activities necessary to perform data plane functionalities, but do not involve the network packets
  - Routing table construction
  - Compliance to packet handling policies
  - Service availability beacons

The Networking Stack – Management and Service

- **Management Plane**: All activities related to monitoring the networks
  - Fault, Configuration, Accounting, Performance and Security (FCAPS)
  - Device initialization

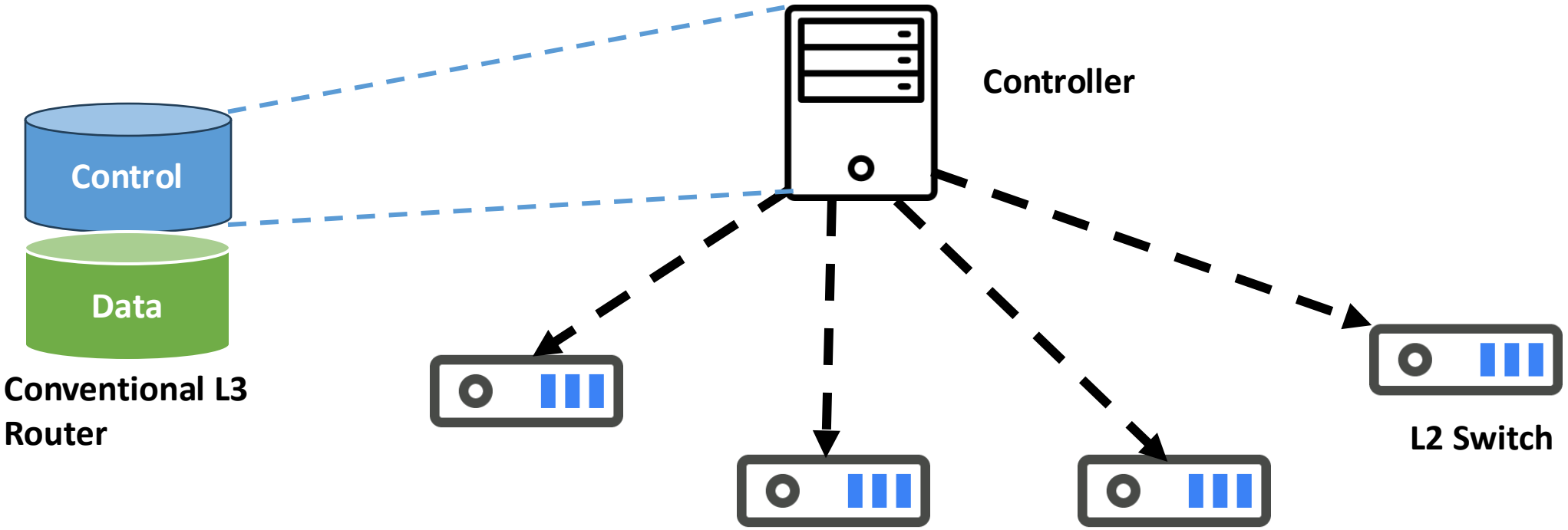
- **Service Plane:** Middlebox services to handle scalability, performance and security activities
  - Firewall, proxy, load balancers, IDS, …

Data vs Control Logic in the Network Stack

- Data plane activities need to run at the *line rate*
  - For a 100 Gbps Ethernet, packets need to be processed at that rate
  - Specialized hardware, like TCAMs, are used
  - Some data plane activities, like broadcast, involves CPU
- All control activities are handled by the CPU
- In the conventional networking stack, both the CPU and the specialized data plane hardware are integrated, making interdependent functionalities
  - Increases CAPEX and OPEX – the reasons that routers are expensive
  - Price of a L3 router (few Lakhs) vs L2 switch (few thousands)
  - Makes management hard

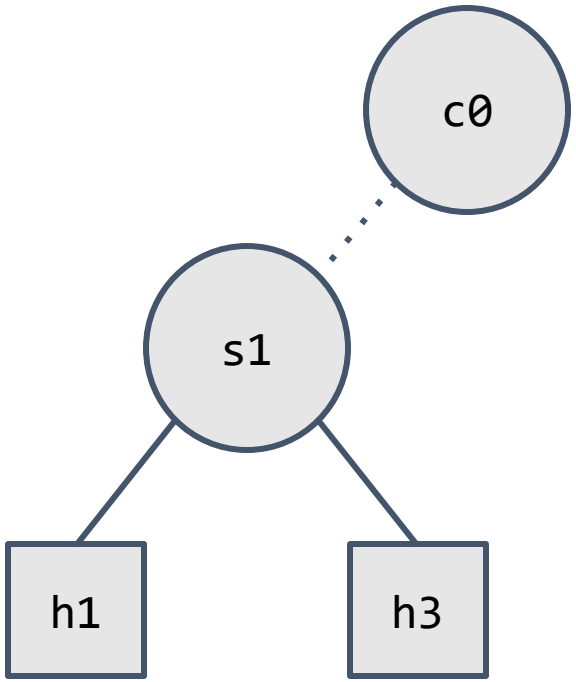
SDN: Key Idea

- Physical separation of the control plane from data (forwarding) plane
  - A central controller (CPU) controls multiple data plane devices
  - The control logic is taken out of the routers and is placed on a central controller



Control Logic: A Simple Example

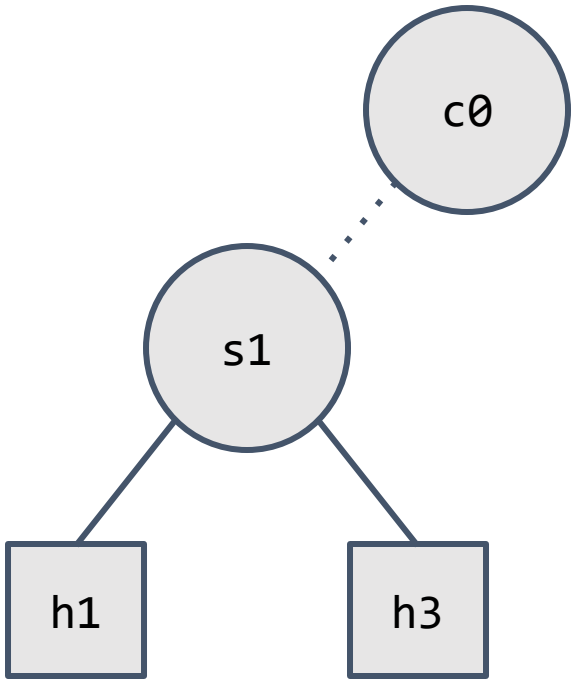
- Controller works as the "brain" of the network
- All policies, routing logic, etc., are placed in the controller
- Dynamically decides the forwarding logic and update the same at the switches



Control Logic: A Simple Example

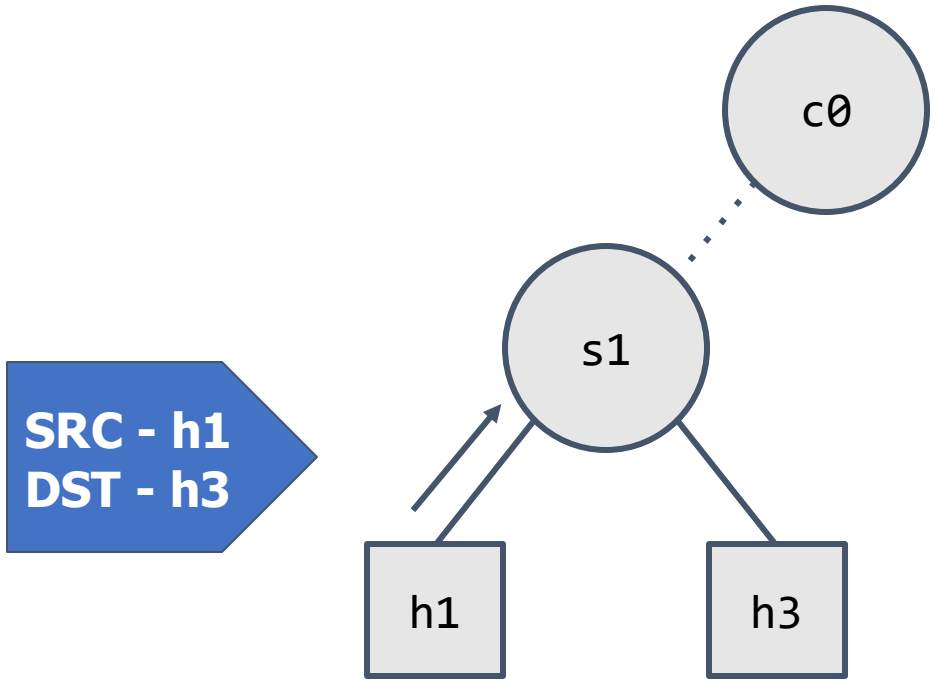
- Controller works as the "brain" of the network
- All policies, routing logic, etc., are placed in the controller
- Dynamically decides the forwarding logic and update the same at the switches

a packet wants to go from h1 to h3, the switch initially does not know how to forward the packet



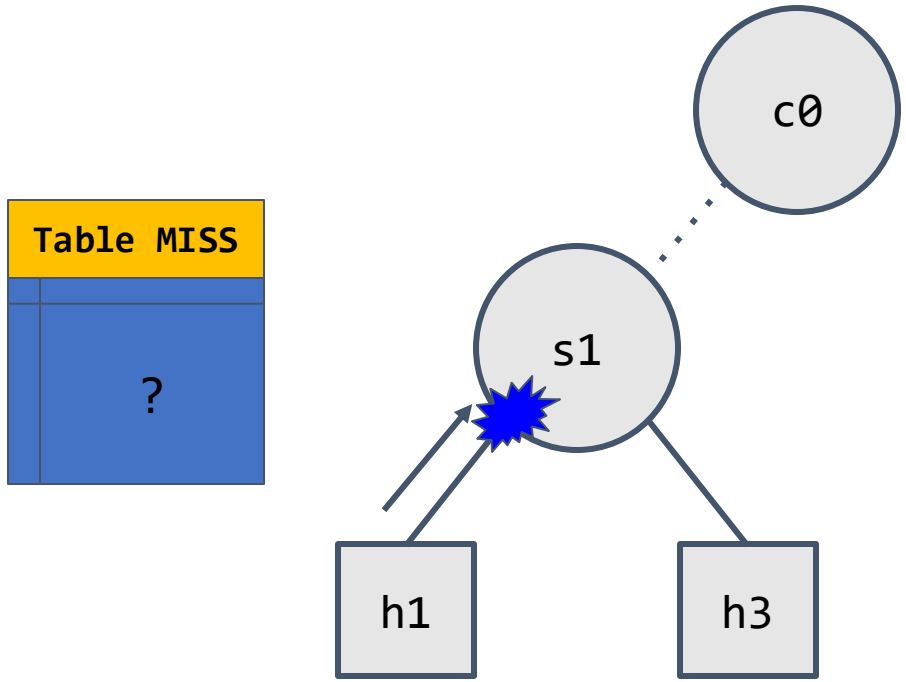
Control Logic: A Simple Example

- Controller works as the "brain" of the network
- All policies, routing logic, etc., are placed in the controller
- Dynamically decides the forwarding logic and update the same at the switches



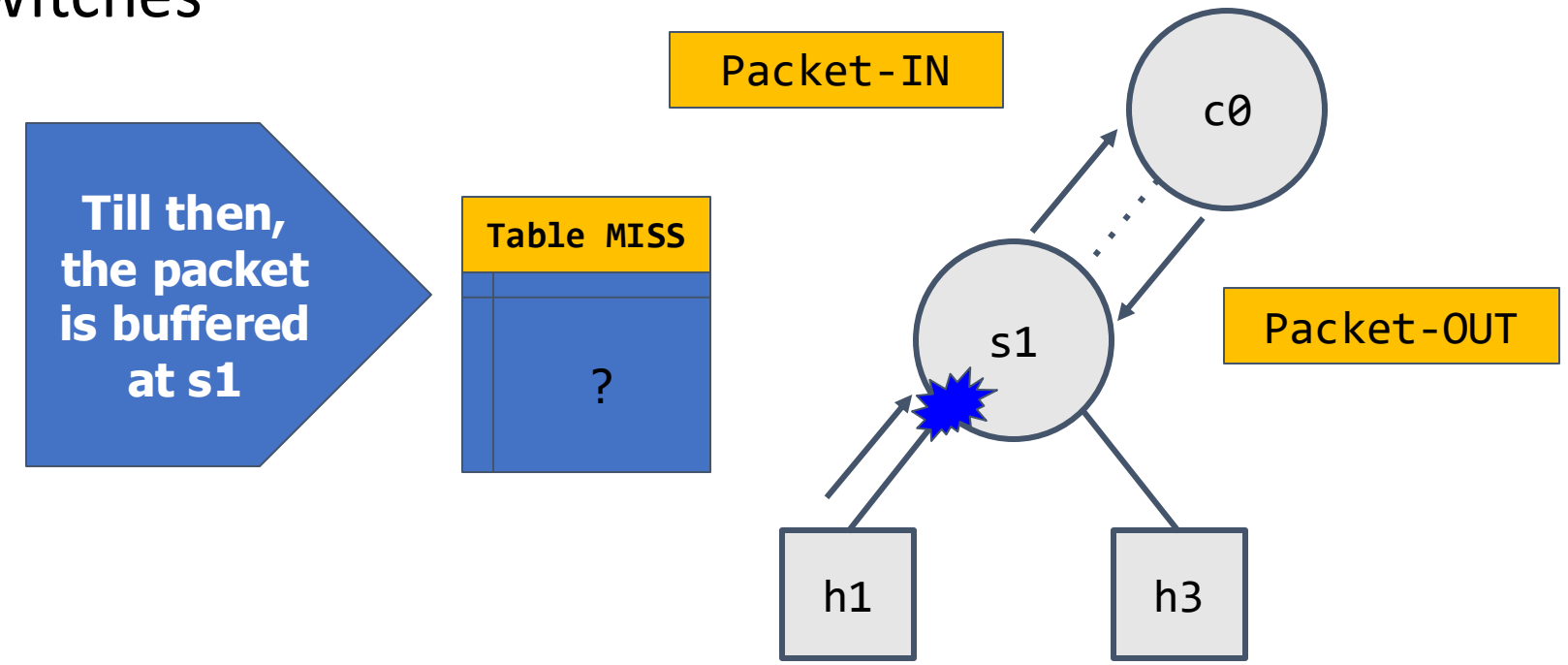
Control Logic: A Simple Example

- Controller works as the "brain" of the network
- All policies, routing logic, etc., are placed in the controller
- Dynamically decides the forwarding logic and update the same at the switches



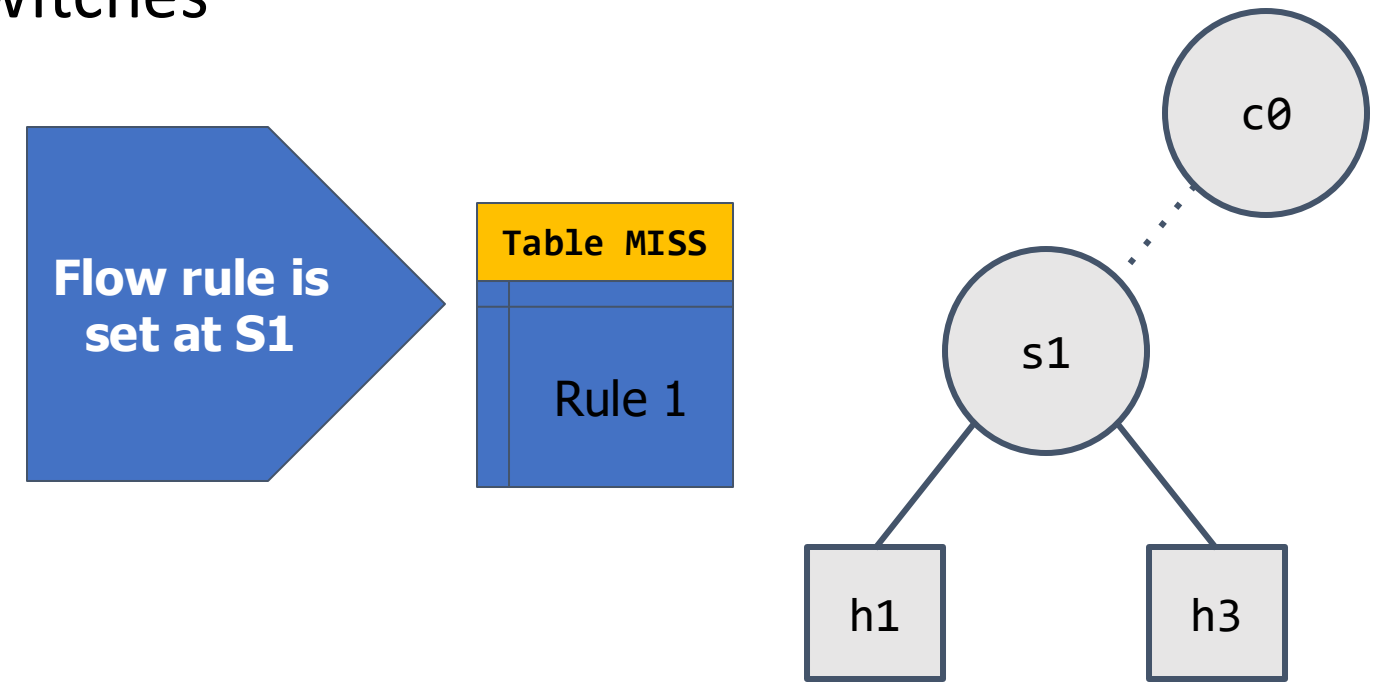
Control Logic: A Simple Example

- Controller works as the "brain" of the network
- All policies, routing logic, etc., are placed in the controller
- Dynamically decides the forwarding logic and update the same at the switches



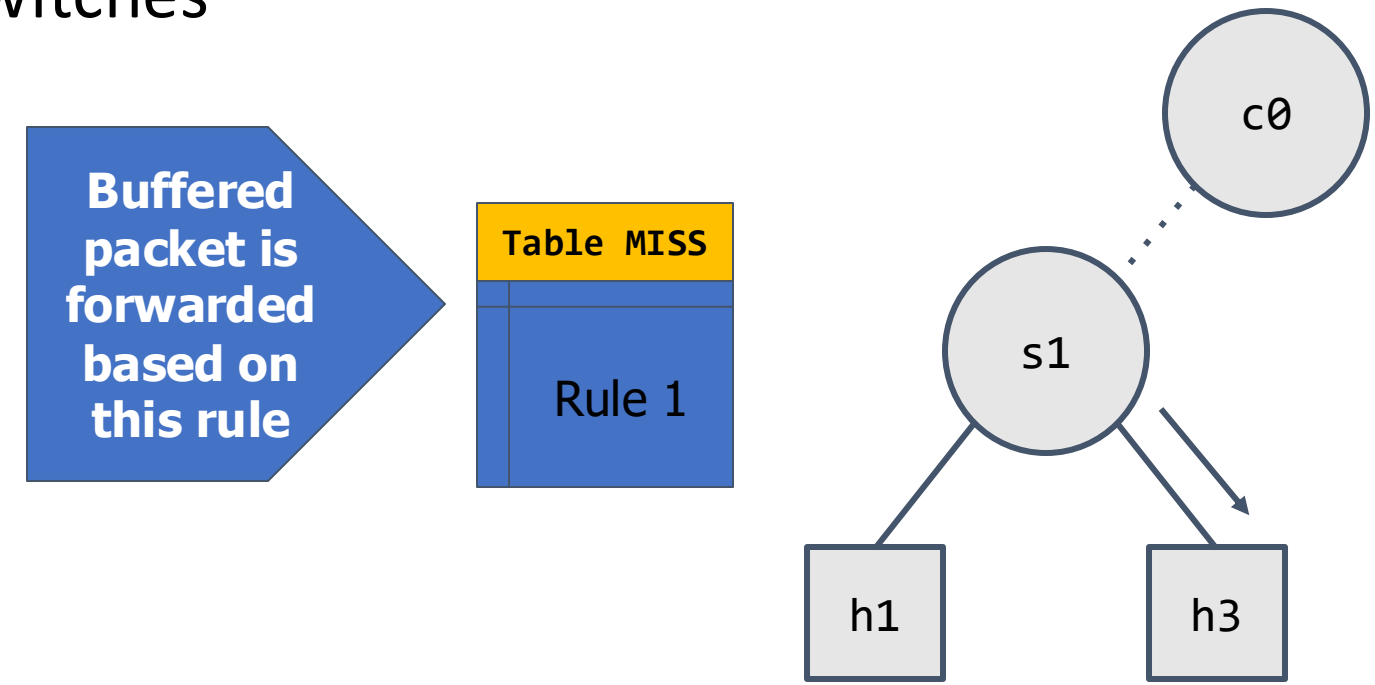
Control Logic: A Simple Example

- Controller works as the "brain" of the network
- All policies, routing logic, etc., are placed in the controller
- Dynamically decides the forwarding logic and update the same at the switches



Control Logic: A Simple Example

- Controller works as the "brain" of the network
- All policies, routing logic, etc., are placed in the controller
- Dynamically decides the forwarding logic and update the same at the switches

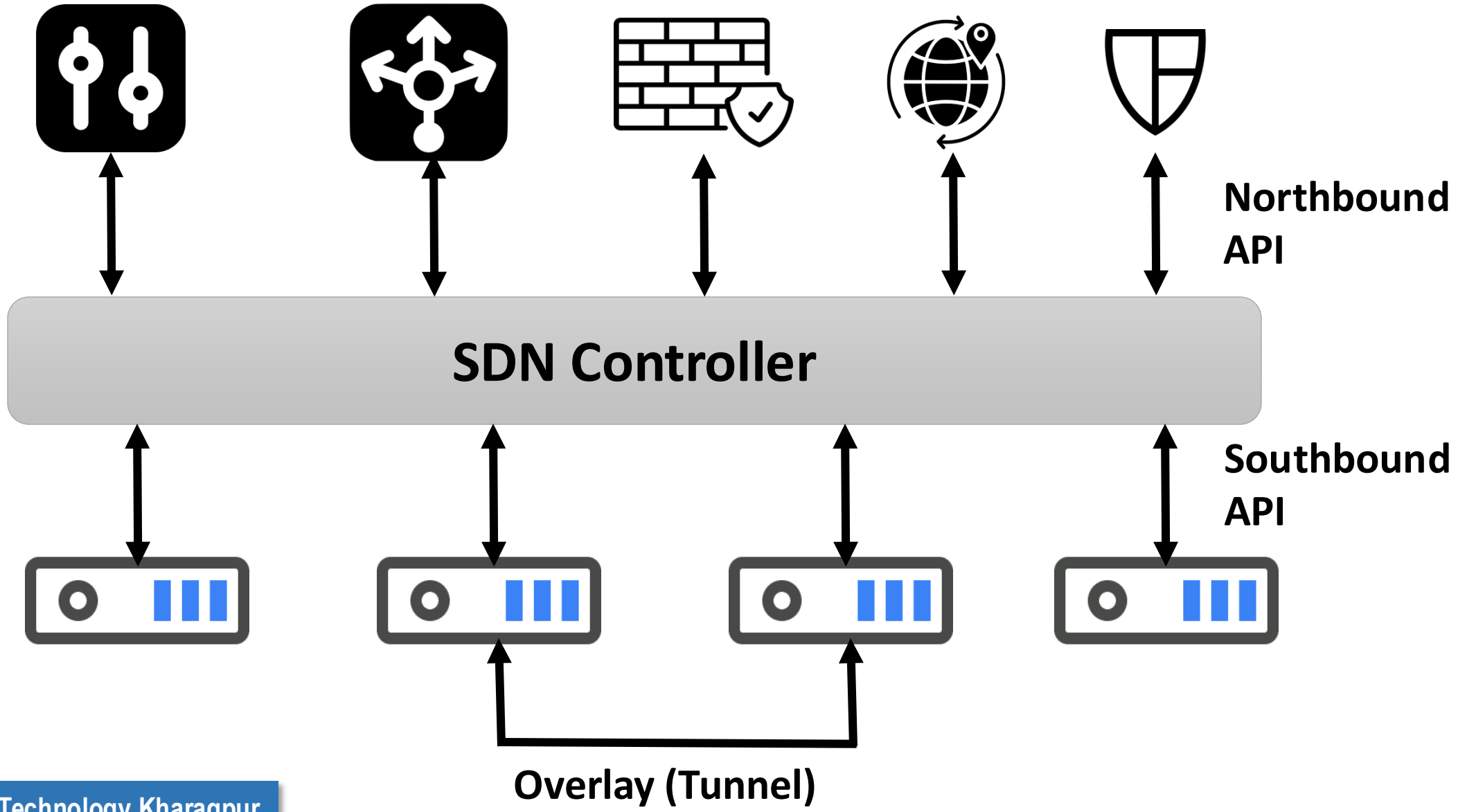


SDN Can Augment NFV

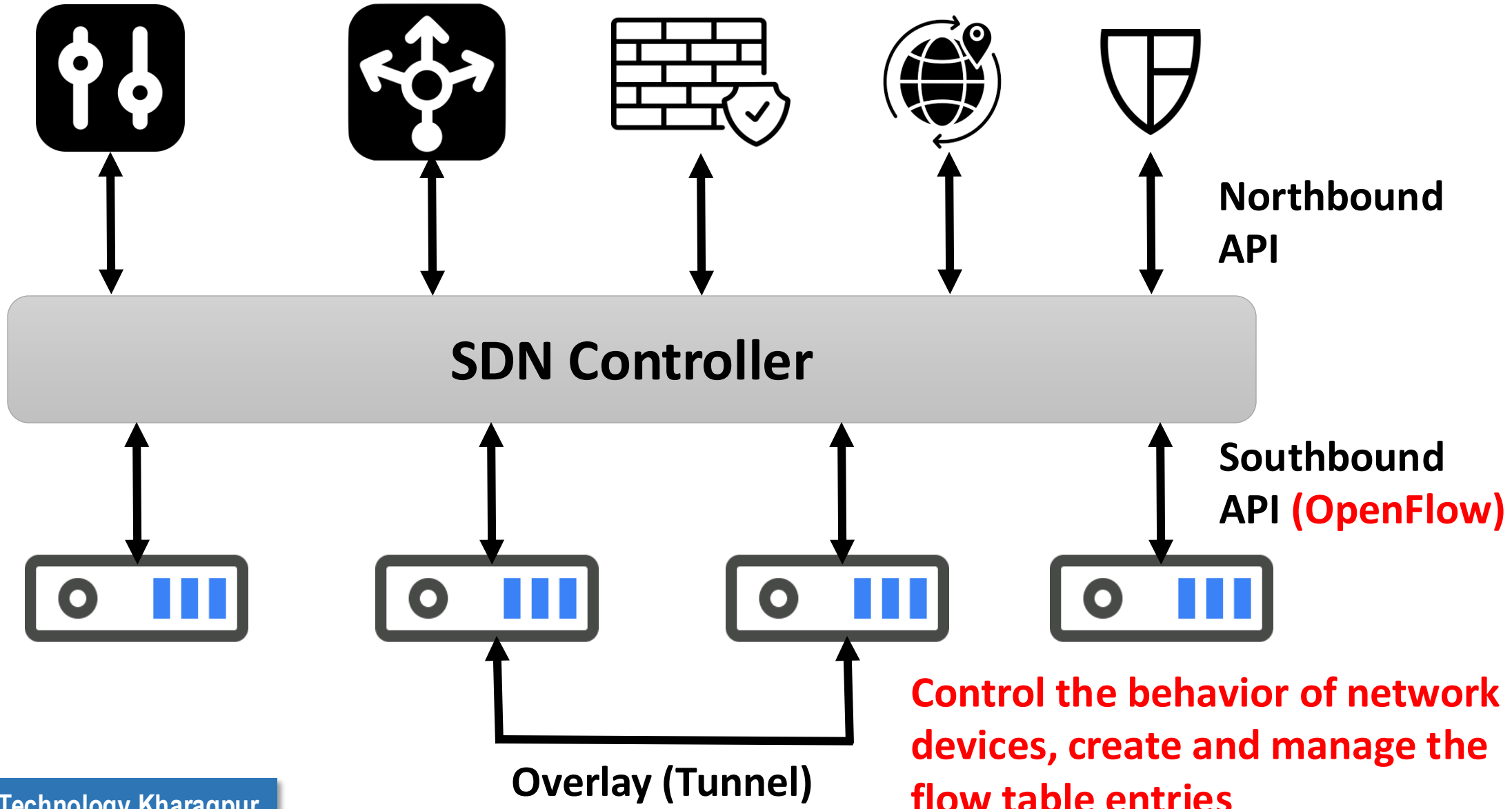
- Use network resources without worrying about,
  - Where it is physically located
  - How much resource is available
  - How the resource is organized

- The controller can log the virtualized resources and allocate them to the applications/ redirect packets dynamically towards the hosted services
  - Provides flexibility in resource allocation and monitoring
  - Administrators do not need to configure each and every router, a policy update at the controller will suffice

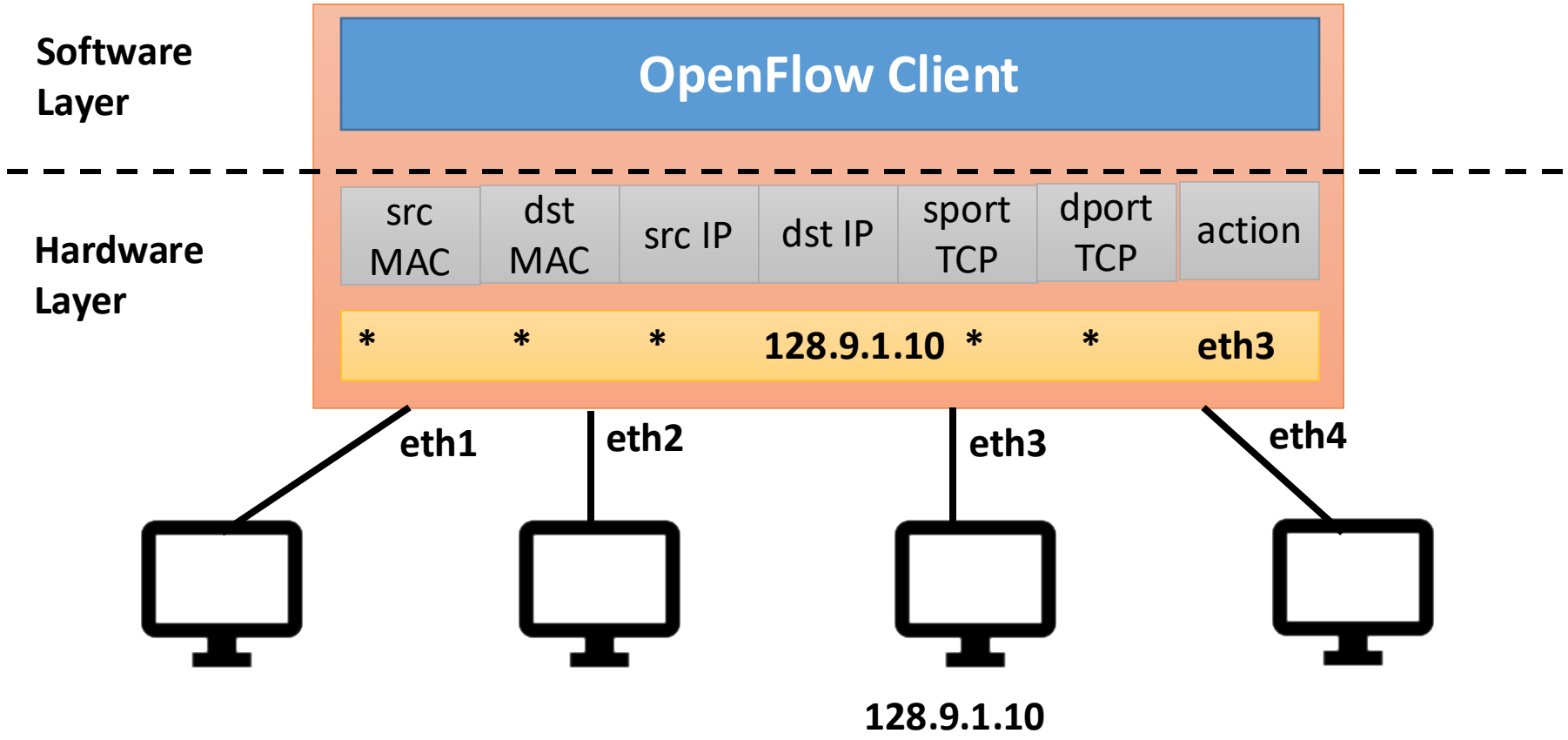
The SDN Architecture



The SDN Architecture: OpenFlow

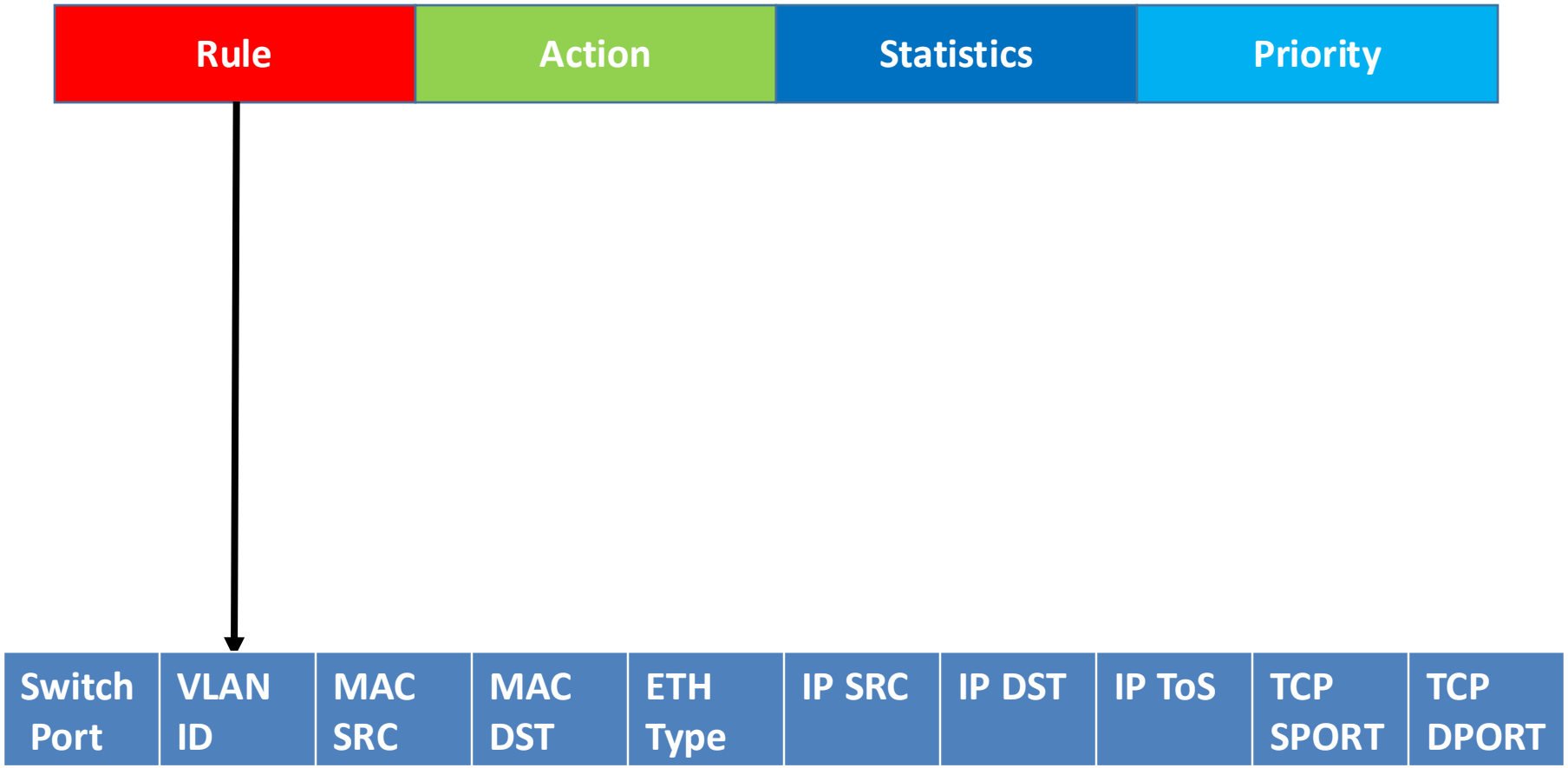


OpenFlow Example

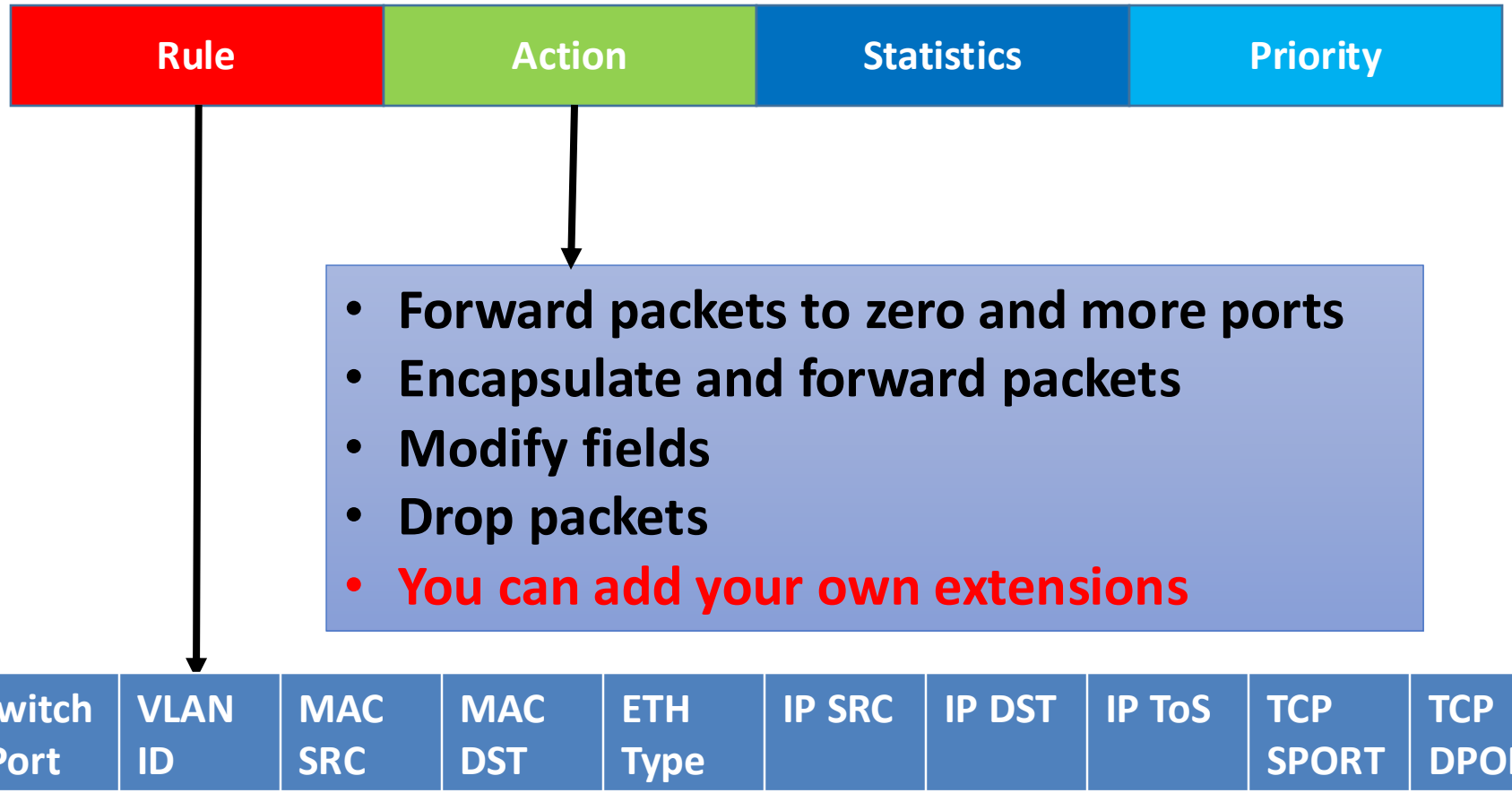


OpenFlow Flow Table

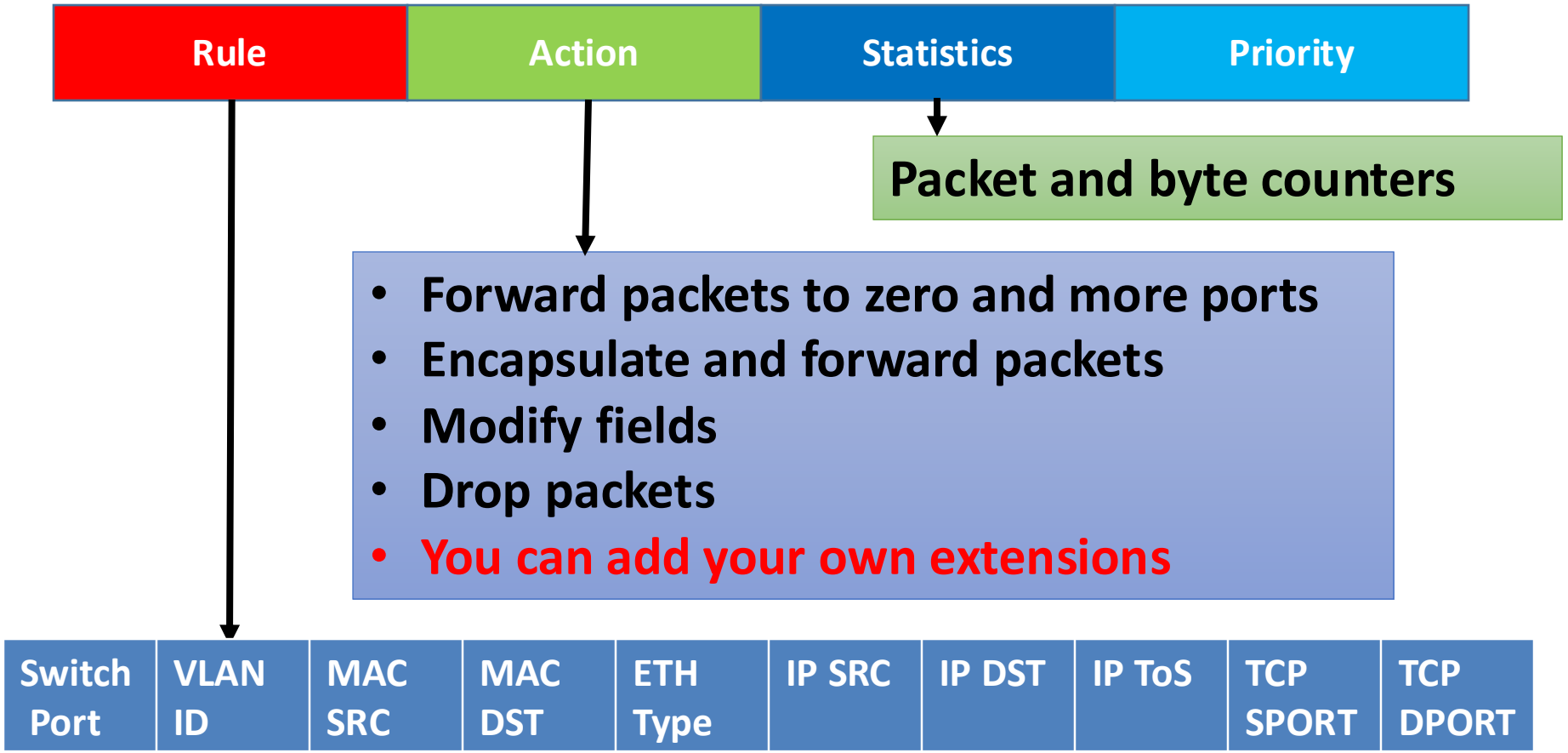
OpenFlow Flow Table



OpenFlow Flow Table



OpenFlow Flow Table



Examples of OpenFlow Flow Table

- **Switching**

Switch Port	VLAN ID	MAC SRC	MAC DST	ETH Type	IP SRC	IP DST	IP ToS	TCP SPORT	TCP DPORT	Action
*	*	*	12:3F:.	*	*	*	*	*	*	eth2

- **Firewall**

Switch Port	VLAN ID	MAC SRC	MAC DST	ETH Type	IP SRC	IP DST	IP ToS	TCP SPORT	TCP DPORT	Action
*	*	*	*	*	*	*	*	*	22	drop

- **Forwarding**

Switch Port	VLAN ID	MAC SRC	MAC DST	ETH Type	IP SRC	IP DST	IP ToS	TCP SPORT	TCP DPORT	Action
*	*	*	*	*	*	202.2.*.*	*	*	*	eth2

Examples of OpenFlow Flow Table

- **Flow Switching**

Switch Port	VLAN ID	MAC SRC	MAC DST	ETH Type	IP SRC	IP DST	IP ToS	TCP SPORT	TCP DPORT	Action
*	*	00:1F: …	14:B2:..	0800	202.1.*.*	212.19.*.*	*	80	8080	eth2

- **Source Routing**

Switch Port	VLAN ID	MAC SRC	MAC DST	ETH Type	IP SRC	IP DST	IP ToS	TCP SPORT	TCP DPORT	Action
*	*	*	*	*	16.2.3.*	202.2.*.*	*	*	*	eth2

- **VLAN Switching**

Switch Port	VLAN ID	MAC SRC	MAC DST	ETH Type	IP SRC	IP DST	IP ToS	TCP SPORT	TCP DPORT	Action
*	2	*	14:B2:..	*	*	*	*	*	*	eth2, eth3

In Summary

- The core network stack is primarily responsible for data transmission across two hosts, but the actual network contains several additional control and management functionalities
  - Makes the network complex, performance becomes a bottleneck

- Virtualization introduces flexibility and openness in network innovation
  - The community realized the issues with TCP much after its innovation, but then ensuring compatibility becomes a challenge with the newer innovations on transport protocols (example. QUIC) -- Simulation failed almost in every cases!
  - Virtualization brings up this flexibility – you can deploy and test your own protocol on top of a running network


Happy
Learning!
Some resources
related to this topic