CS3103: Computer Networks Practice

Software Defined Networking (SDN)

SDN Concept

OpenFlow Protocol/Interface

Network Virtualisation

~~SDN Lab - Overview~~

Dr. Anand Bhojan
COM3-02-49, School of Computing
banand@comp.nus.edu.sg ph: 651-67351

Source: Jennifer Rexford, Scot Shenker, Raj Jain, Bruce Maggs (Duke University), Xenofontas Dimitropoulos (ZTH), Marco Canini (UCL)

Summary

- SDN:- Separate Control plane and Data plane entities.
  - Network intelligence and state are logically centralised.
  - The underlying network infrastructure is abstracted from the applications.
- ***OpenFlow*** is communication interface/protocol between the control and data plane of an *SDN architecture*.
- Network Virtualisation - making a physical network appear as multiple logical ones (network slicing)
- ~~Overview to Lab Exercise~~
  - ~~Mininet, VirtualBox, Floodlight~~

What is Software Defined Networking (SDN)?

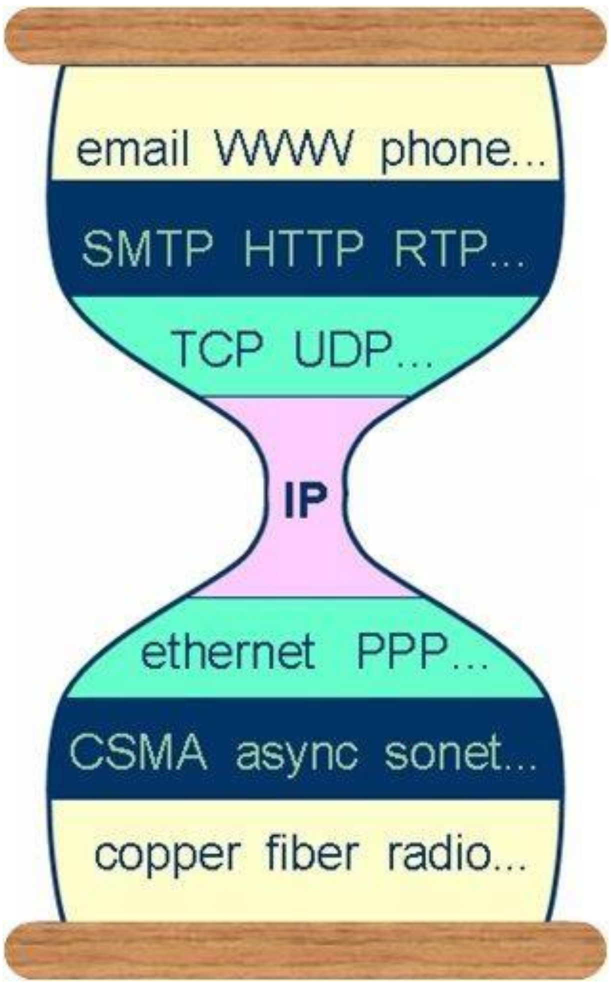
- A new approach to do networking
  - new fundamental principles
- Before knowing what it is, let us understand why we need it in the first place
  - what is wrong with the current Internet?

What is wrong with the current Internet?

The Internet: A Remarkable Story

- Tremendous success
  - From research experiment to global communications infrastructure
- The brilliance of under-specifying
  - Best-effort packet delivery service
  - Key functionality are programmable at end hosts
- Enabled massive growth and innovation
  - Ease of adding hosts and link technologies
  - Ease of adding services (Web, P2P, VoIP, …)
- But, change is easy only at the edge… ☹

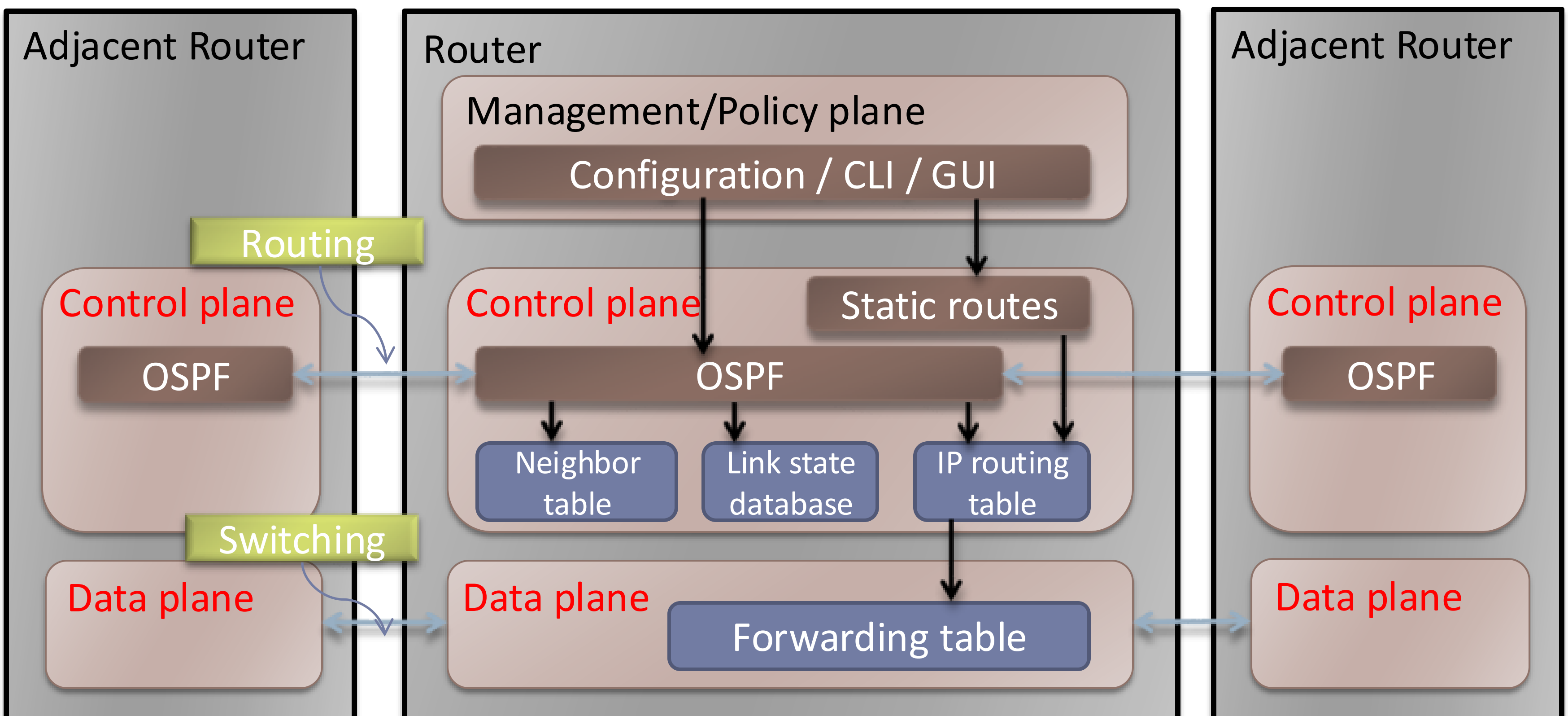
Key to the Internet's Success - Layering



- Hourglass IP model
- Layered service abstractions (why is this important?)
  - Decompose delivery into fundamental components
  - Independent, compatible
    - innovation at each layer
- But, Only for network edges?

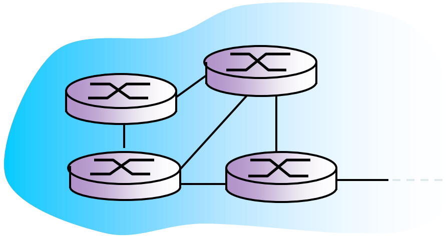
Current Internet: Complicated Router at the Core

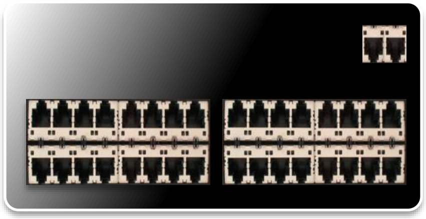
- Router can be partitioned into control and data plane
  - Management plane/ Configuration
  - Control plane / Decision: run routing algorithms/protocol (RIP, OSPF, BGP)
  - Data plane / Forwarding: forwarding datagrams from incoming to outgoing link



What is wrong with the current Internet?

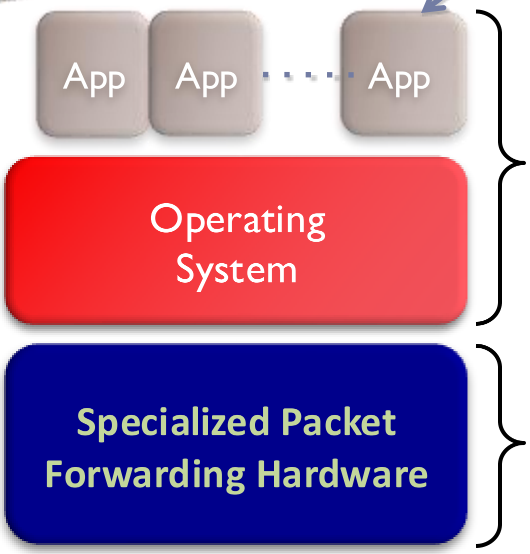
- Closed equipment
  - Software bundled with hardware.
  - Vendor-specific interfaces.
- Over specified
  - Slow protocol standardisation.
- Few people can innovate
  - Equipment vendors write the code.
  - Long delays to introduce new features.





What is wrong with the current Internet?

Routing, management, mobility management, access control, VPNs, …



Million of lines of source code

Managed by 6,000 RFCs (Over Specified)

500M gates 10Gbytes RAM

Bloated

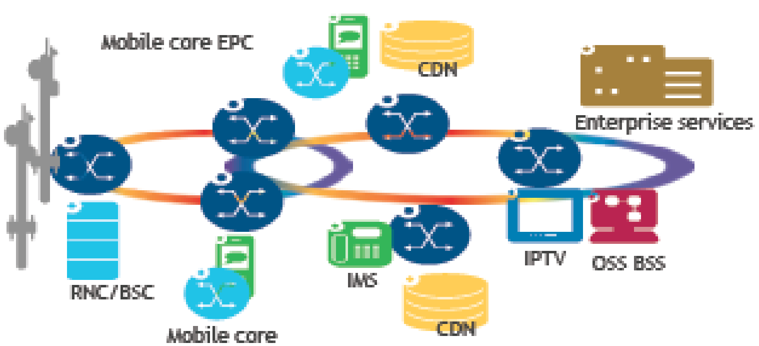
Power Hungry

- Many complex functions baked into the infrastructure
  - OSPF, BGP, multicast, differentiated services, Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …
- An industry with a "mainframe-mentality"
- Consequences: Buggy software in the equipment
  - Cascading failures, vulnerabilities, etc

What is wrong with the current Internet?

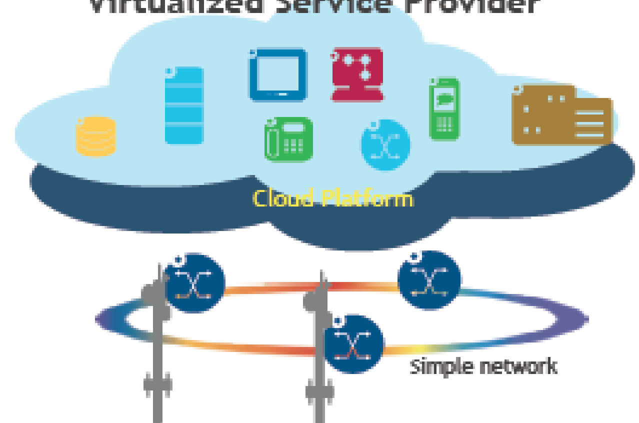
- Operating a network is expensive
  - More than half the cost of a network.
  - Yet, operator error causes most outages.

Legacy Service Provider



Each function runs on **dedicated hardware**.

Virtualized Service Provider



Network functions (firewalls, NAT, load balancers, routers) are **virtualized (NFV)** and run as **software on commodity servers**.

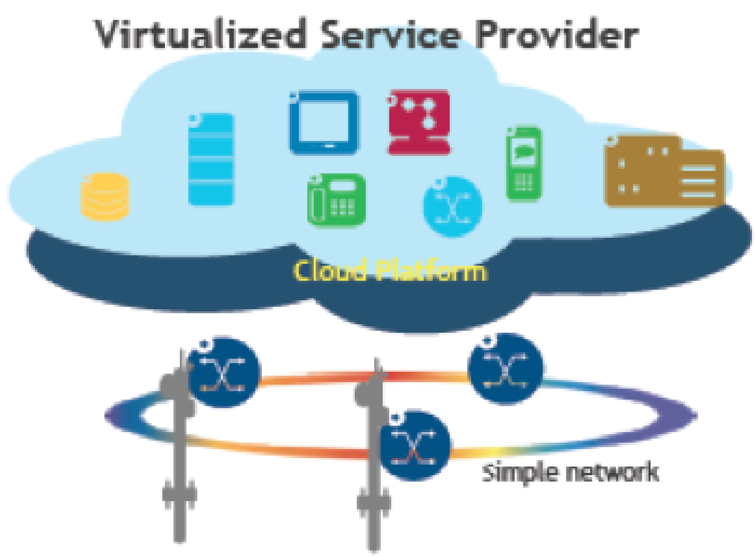
Centralized control reduces the need to log into individual devices.

Manual configuration and maintenance of many heterogeneous devices.
Vendor-specific expertise for different devices and OS versions.

Network Virtualization is one of the Applications of SDN

What is wrong with the current Internet?

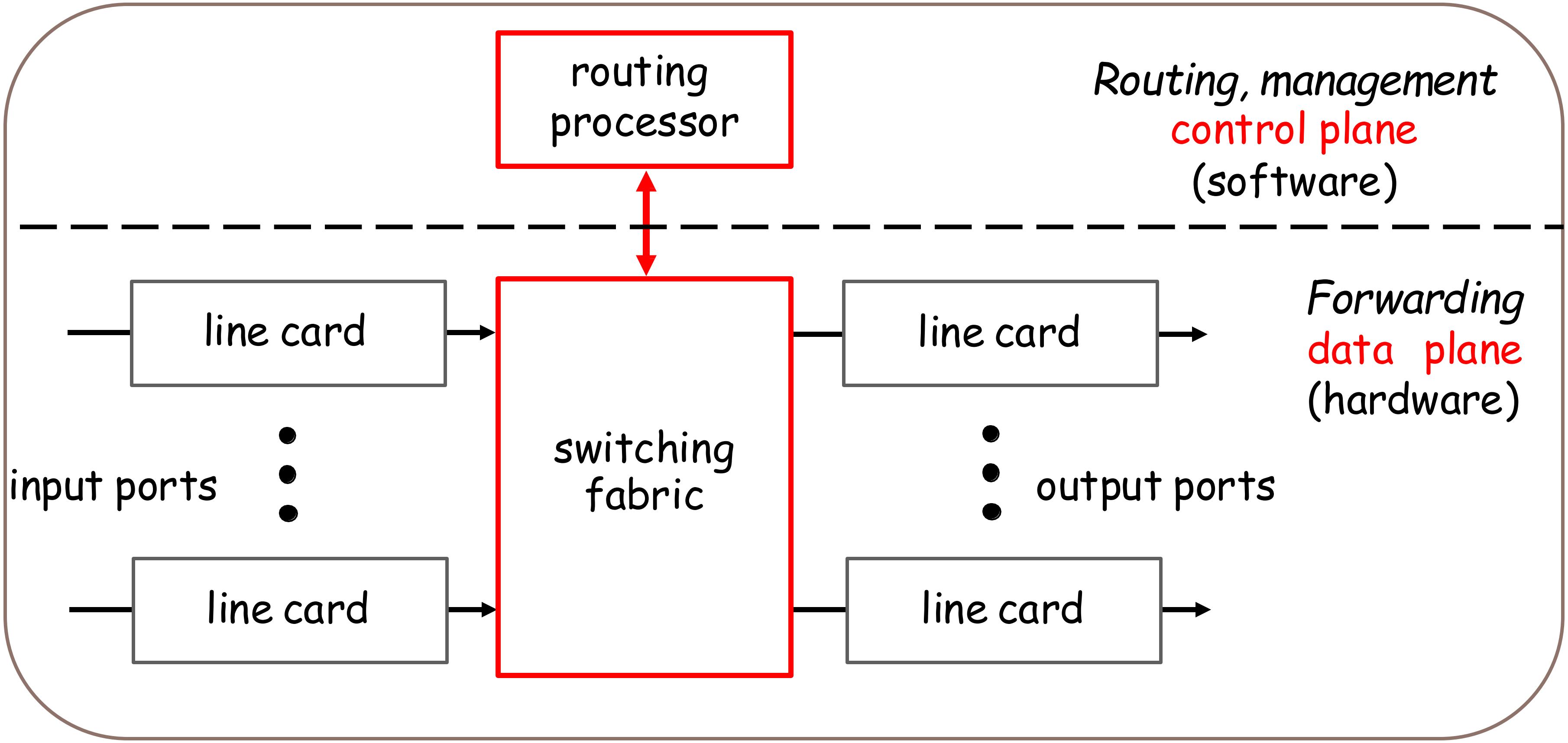
- Demand and Complexity are increasing ….
  - Major ISPs: Upgrade their internal network infrastructure (routers and switches) every 18 months to keep up with the current demands for network.
  - "AT&T Eyes Flexibility, Cost Savings With New Network Design", Wall Street journal, 2014. --- basically moving towards SDN (Network Virtualization)



What is SDN?

Traditional network Router – Simplified Look

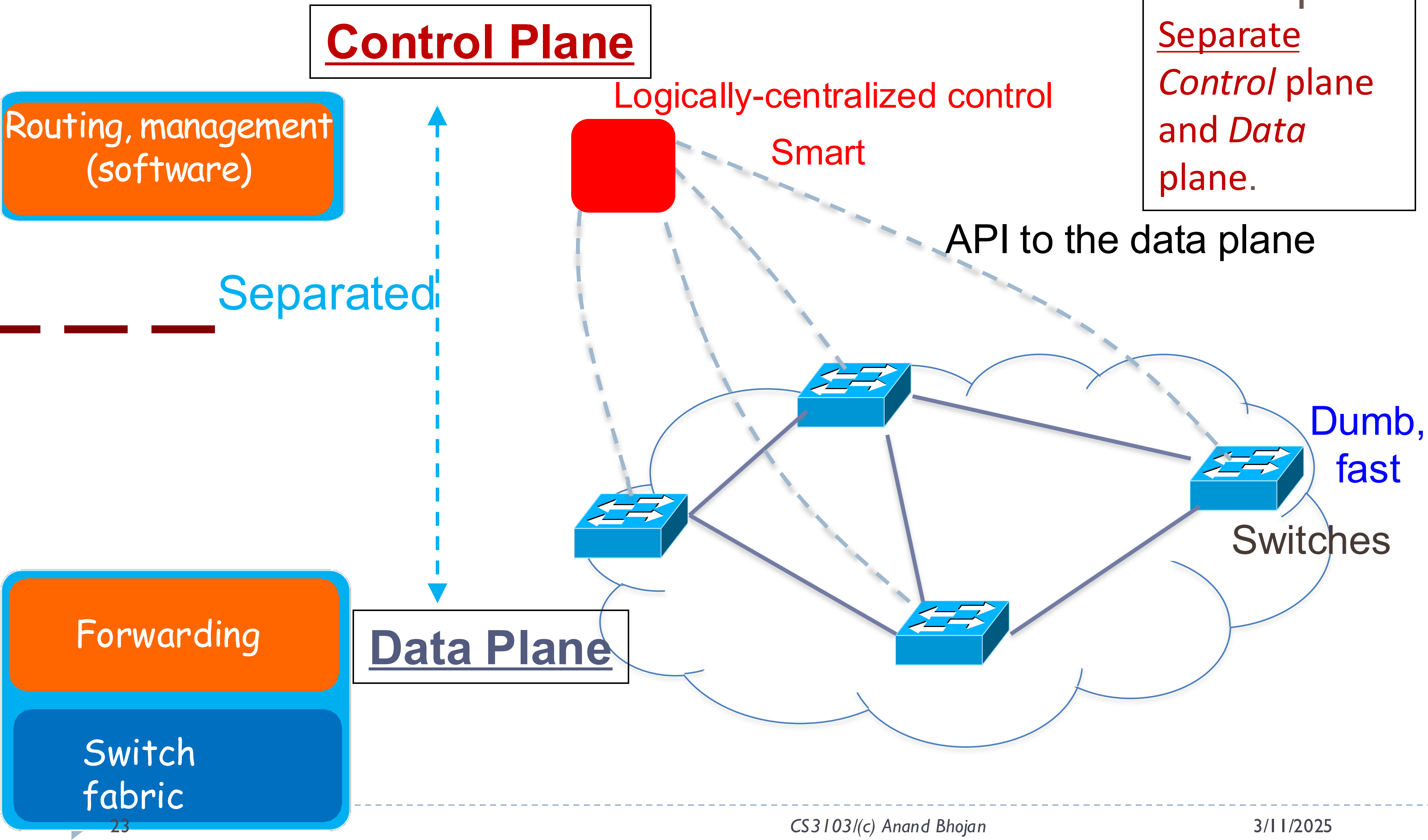
- Two key functions of a ROUTER:
  - run routing algorithms/protocol (RIP, OSPF, BGP)
  - *forwarding* datagrams from incoming to outgoing link



Q: Can we separate these two KEY functions of a Router? Any advantage?

Imagine **IF** The Network is……..!!!

SDN Concept: Separate *Control* plane and *Data* plane.



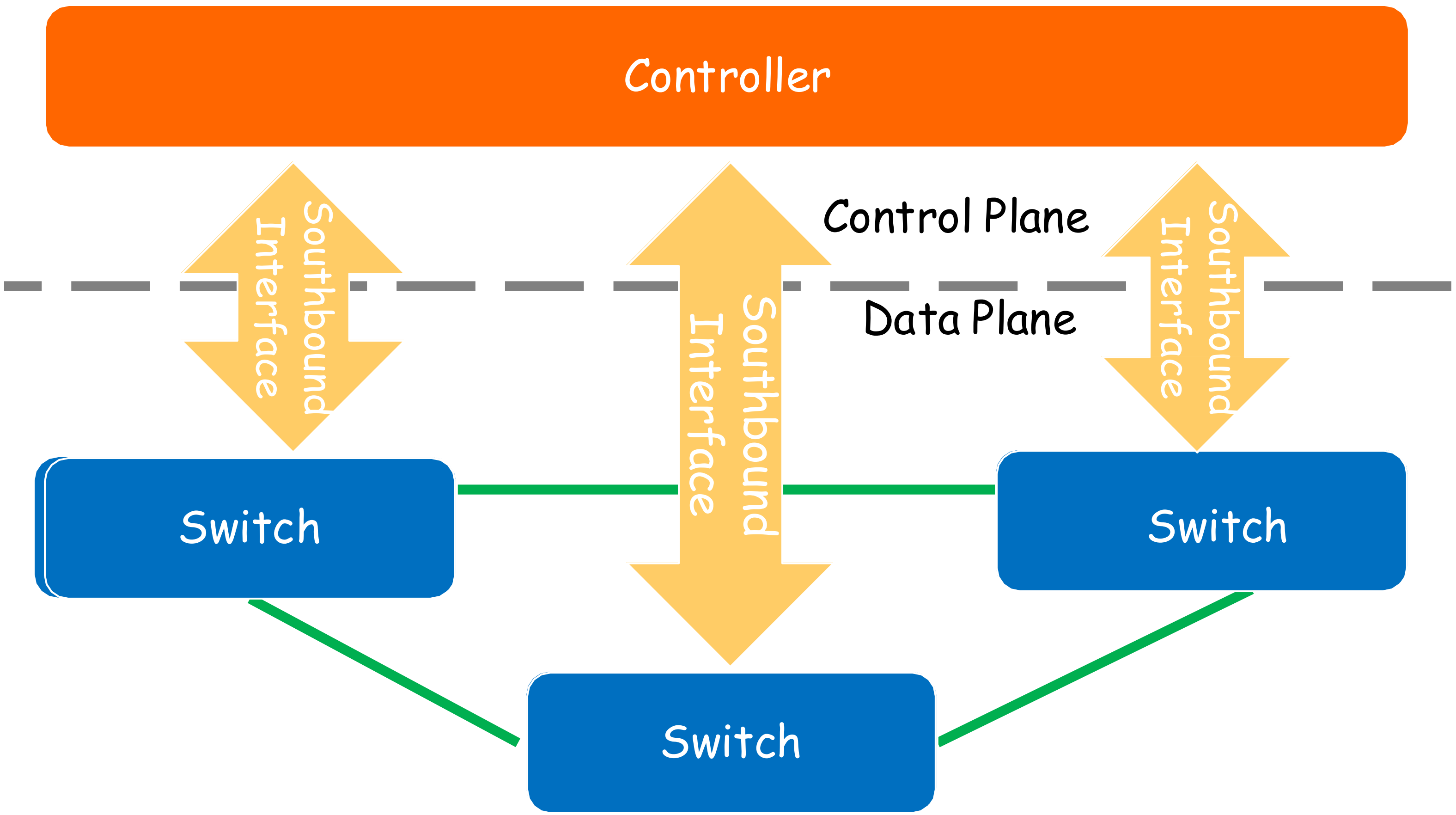
Benefits of Separation

- Independent evolution and development
  - The software control of the network can evolve independently of the hardware.
- Control from high-level software program
  - Control behavior using higher-order programs
  - Debug/check behavior more easily

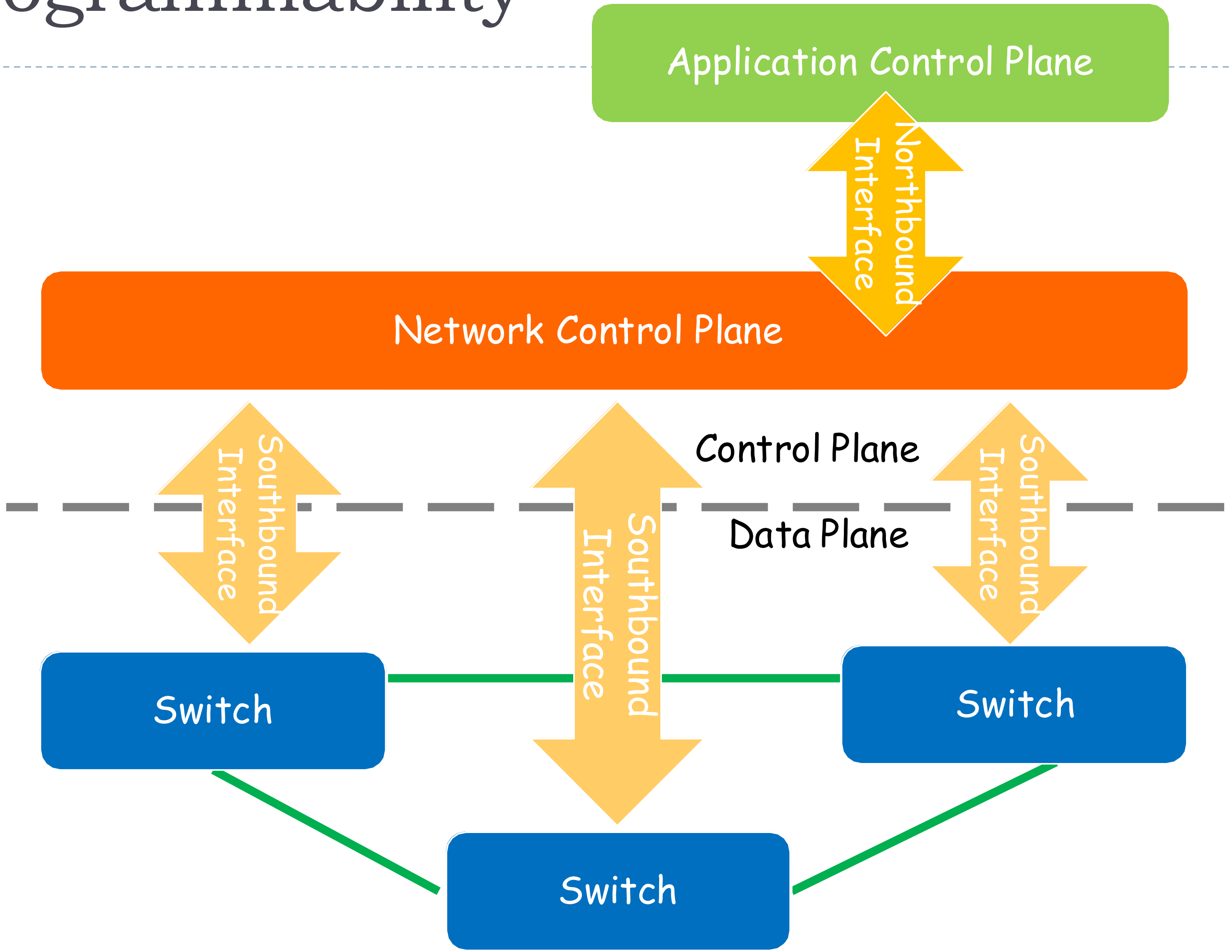
Benefits of Centralization

- Centralised decisions are easier to make
  - E.g., OSPF (RFC 2328) 244 pages
  - Distributed system part (builds consistent network 100 pages)
  - Routing algorithm (Dijkstra's algorithm 4 pages)
- Logically vs. physically centralised
  - Issues with of a physically centralised controller?
  - How to implement a logically centralised one?

Open Interfaces



Programmability



Benefits of Open Interfaces and Programmability

- Enable competitive technologies
  - Independent developments
  - Rapid innovation and fast evolution
  - Cheap and better networks
- Make network management much easier
  - Management goals are expressed as policies
  - New control/services for network providers
  - Detailed configuration are done by controller

Summary (What we Studied): SDN Concept

- Separate Control plane and Data plane entities.
  - Network intelligence and state are logically centralized.
  - The underlying network infrastructure is abstracted from the applications.
- Execute or run Control plane software on general purpose hardware.
  - Decouple from specific networking hardware.
  - Use commodity servers and switches.
- Have programmable data planes.
  - Maintain, control and program data plane state from a central entity.
- An architecture to control not just a networking device but an entire network.

The Real-World Birth of SDN: Google's Story

SDN in Real World – Google's Story

- The industries were skeptical whether SDN was possible.
- Google had big problems **(2009–2011)**
  - By 2009, Google's internal network — connecting its global data centers — was facing critical challenges:
  - **Traffic growth** due to replication and synchronization between data centers.
  - **Rigid control plane**: Each router made local decisions (distributed protocols like OSPF/BGP).
  - **Low link utilization**: They had to keep ~30–40% of bandwidth unused as a safety margin.
  - **Slow reconfiguration**: Network policy updates were error-prone and slow.
- Google went a head and implemented SDN.
  - Built their hardware and wrote their own software for their internal datacenters.
  - Industries surprised when Google announced first production grade SDN: **B4**, a **software-defined WAN** connecting its global data centers.
- How did they do it?
  - Read "*B4: Experience with a Globally-Deployed Software Defined WAN*", ACM Sigcomm 2013.

B4, the first global production grade SDN

- **Architecture:**
  - Built on **OpenFlow** (the first open SDN **protocol**).
  - **Commodity switches** (white-box) with programmable forwarding tables.
  - A **central SDN controller** (cluster of servers) that had a **global view** of the entire network.
  - **Centralized traffic engineering**: dynamically allocates bandwidth based on real-time demand from applications.
- **Benefits:**
  - **>90% link utilization** (vs ~30% before SDN).
  - **Rapid reconfiguration** in seconds instead of hours/days.
  - **Lower cost** — using commodity hardware.
  - **Automated fault recovery** and rerouting.
- **Evolution – From B4 to Jupiter**
  - Google extended the SDN design to its **data center networks** under the project **Jupiter** (launched ~2015).
  - **Jupiter** scaled SDN principles to **hundreds of thousands of servers** and **millions of virtual machines**.
  - It used **centralized management**, **programmable switches**, and **SDN controllers** to dynamically optimize internal and external traffic.

The Origin of SDN

- 2006: Martin Casado, a PhD student at Stanford and team propose a clean-slate security architecture (SANE) which defines a centralized control of security (in stead of at the edge as normally done). Ethane generalises it to all access policies.
- The idea of *Software Defined Network* is originated from OpenFlow project (*ACM SIGCOMM 2008*).
- 2009: Stanford publishes OpenFlow V1.0.0 specs.
- June 2009: Martin Casado co-founds Nicira.
- March 2011: **Open Networking Foundation** is formed.
- Oct 2011: First Open Networking Summit. Many Industries (Juniper, Cisco) announced to incorporate.
- July 2012: VMware buys Nicira for **$1.26B**.
- Lesson Learned: Imagination is the key to <u>*unlock*</u> the power of possibilities.

In 4 years!

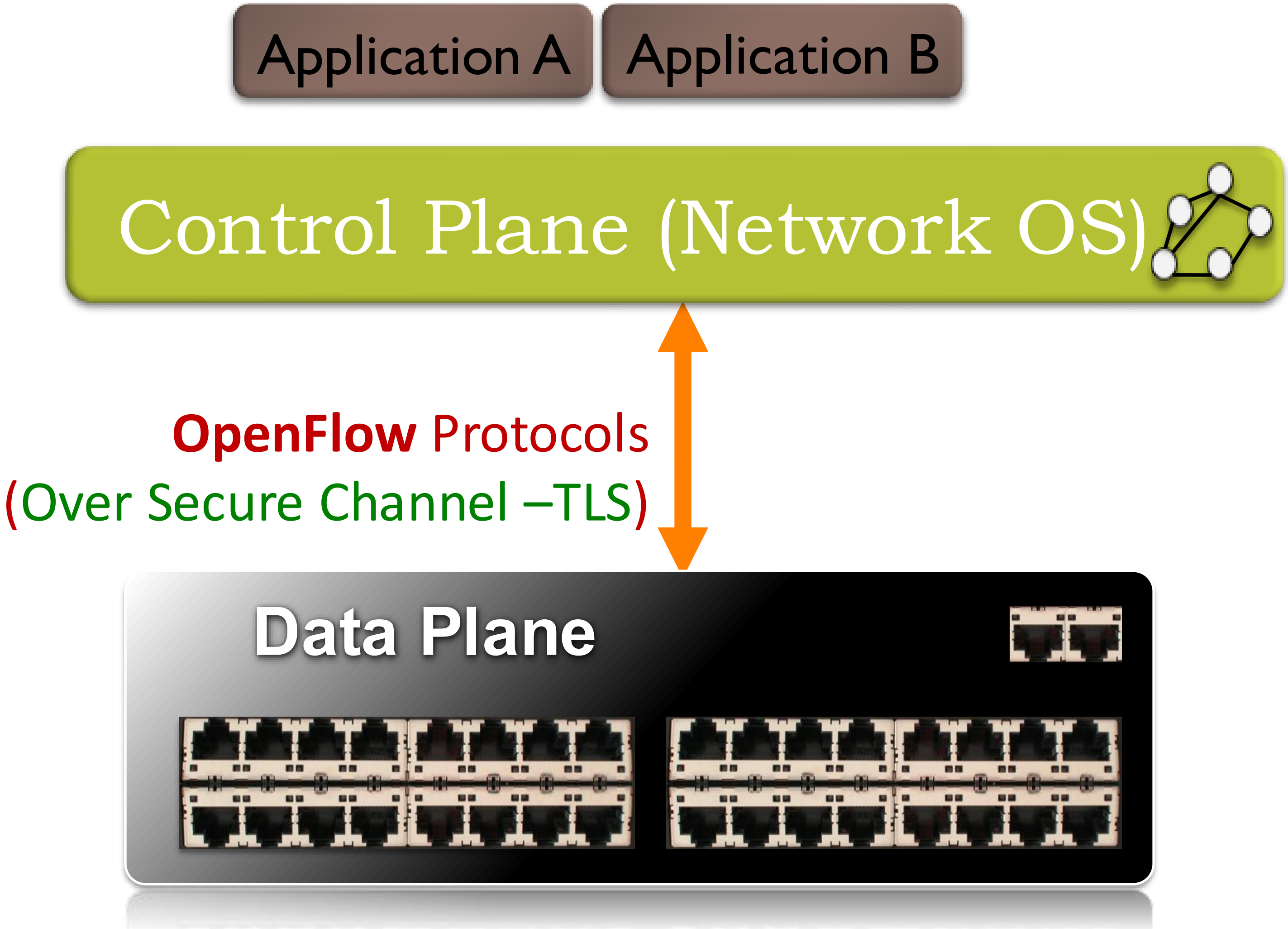
Martin Casado

OpenFlow Protocol (Switch Spec Ver 1.5.1, Year 2015)

Managed by: Open Networking Foundation (ONF – opennetworking.org).

Specs: https://opennetworking.org/software-defined-standards/specifications/

What is OpenFlow?



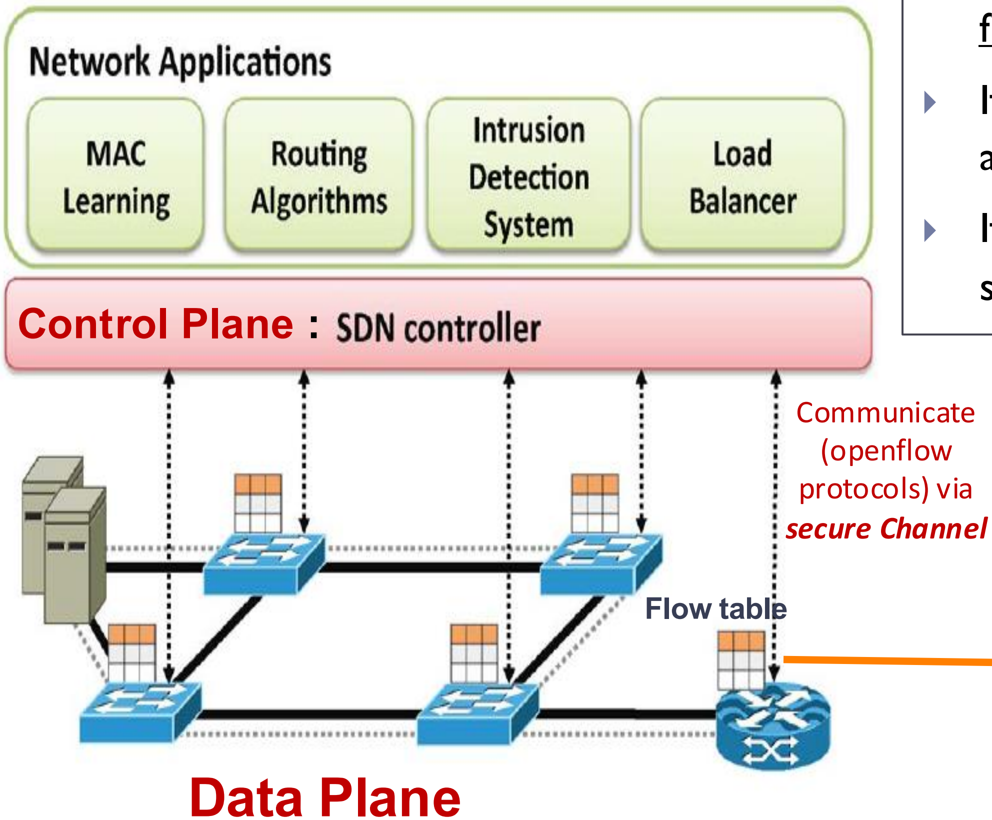
What is OpenFlow?

- OpenFlow is a **standard communication protocol** that allows a **SDN controller** to communicate with **network devices** (switches, routers) to manage packet forwarding.
  - Allow separation of control and data planes.
  - Centralisation of control.
  - Flow based control. (uses Flow table, its entries are controlled by Control Plane)
  - Takes advantage routing tables in Ethernet switches and routers (L2/L3).

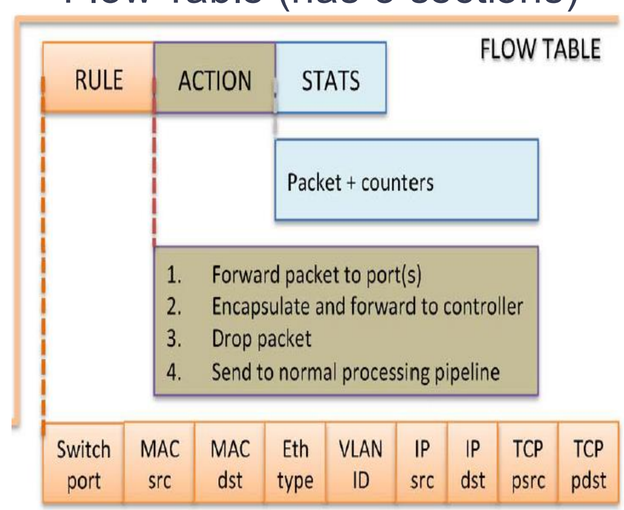
- SDN is not OpenFlow.
  - **SDN** is a concept of the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices.
  - ***OpenFlow*** is communication interface between the control and data plane of an *SDN architecture*.
    - Allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual.
    - Think of as a protocol used in switching devices and controllers interface.

Basic OpenFlow: How Does it Work?

- Controller manages the traffic (network flows) by manipulating the flow table at switches.
  - Instructions are stored in flow tables.
- When packet arrives at switch, match the header fields with flow entries in a flow table.
- If any entry matches, performs indicated actions and update the counters.
- If it does not match, Switch asks controller by sending a message with the packet header.



Flow Table (has 3 sections)



Match the packet header

The Actual Flow Table Looks Like

Port	Src MAC	Dst MAC	VLAN ID	Priority	EtherType	Src IP	Dst IP	IP Proto Protocol	IP ToS QoS	Src L4 Port ICMP Type	Dst L4 Port ICMP Code	Action	Counter
*	*	0A:C8:*	*	*	*	*	*	*	*	*	*	Port 1	102
*	*	*	*	*	*	*	192.168.*.*	*	*	*	*	Port 2	202
*	*	*	*	*	*	*	*	*	*	21	21	Drop	420
*	*	*	*	*	*	*	*	0x806	*	*	*	Local	444
*	*	*	*	*	*	*	*	0x1*	*	*	*	Controller	1

Match

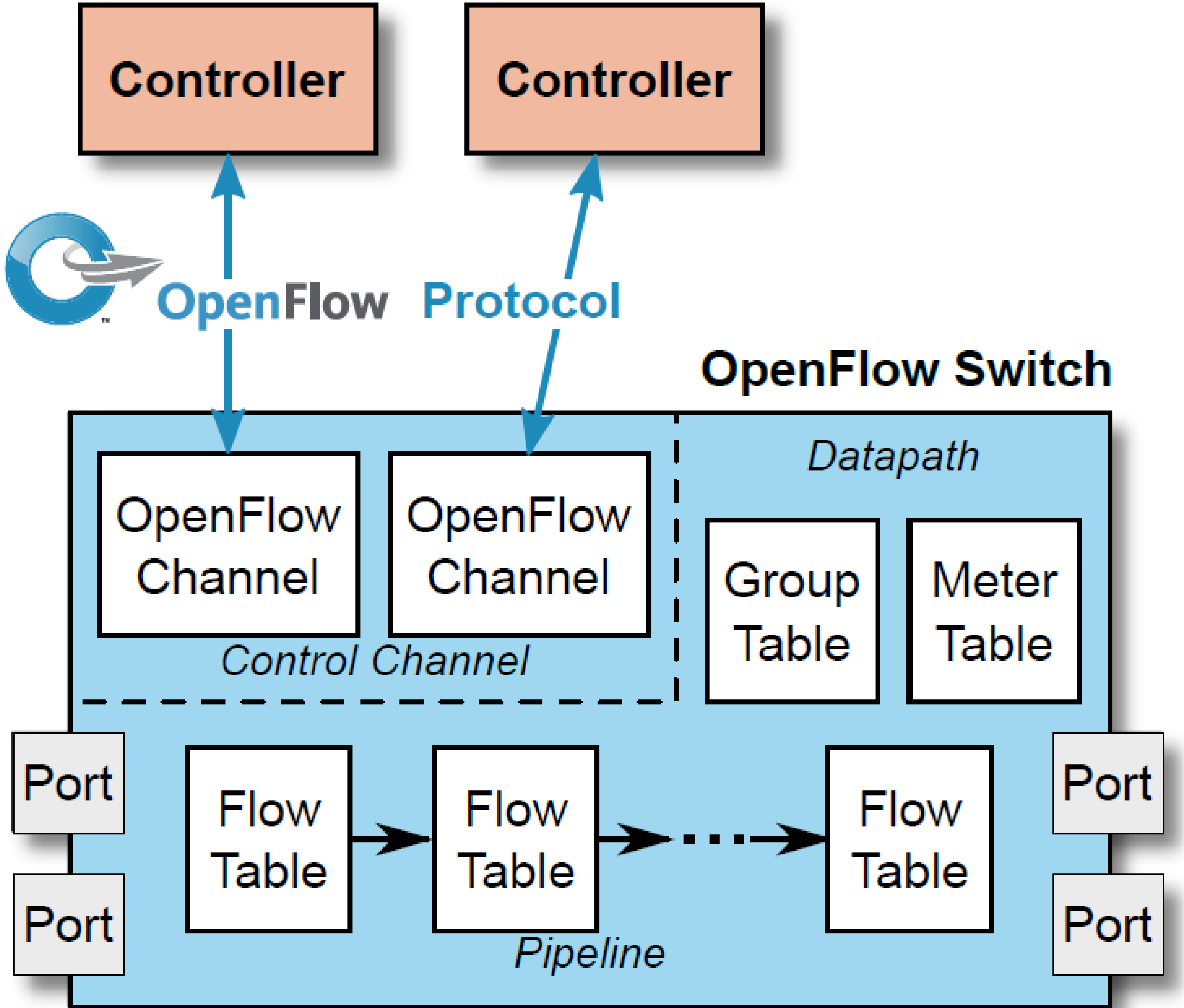
Action

Stats

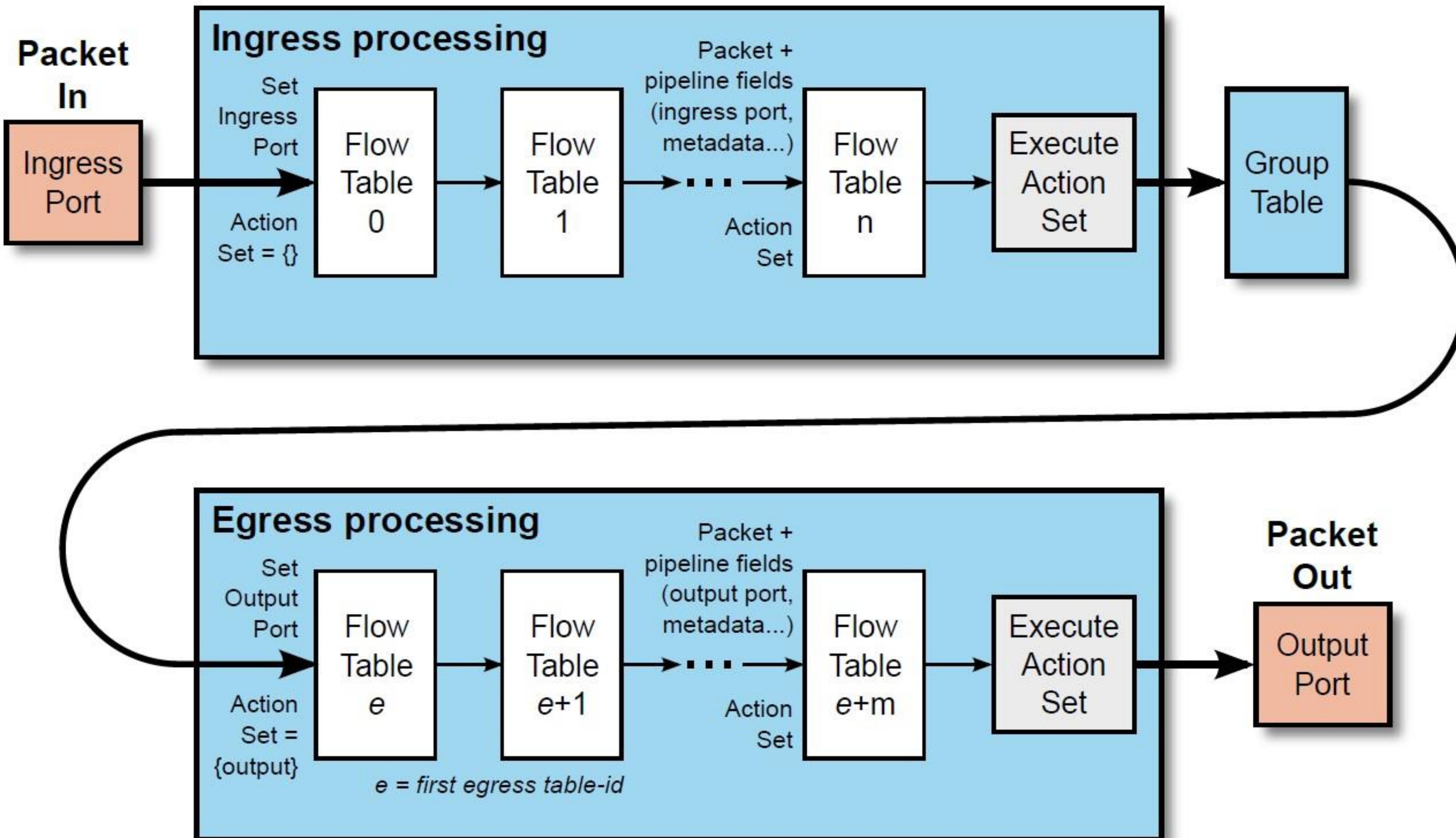
OpenFlow Table: Basic Actions

FYORP

- All: To all interfaces except incoming interface.
- Controller: Encapsulate and send to controller.
- Local: send to its local networking stack.
- Table: Perform actions in the <u>next</u> flow table (table chaining or multiple table instructions).
- In_port: Send back to input port.
- Normal: Forward using traditional Ethernet.
- Flood: Send along minimum spanning tree except the incoming interface.



Main components of an OpenFlow switch

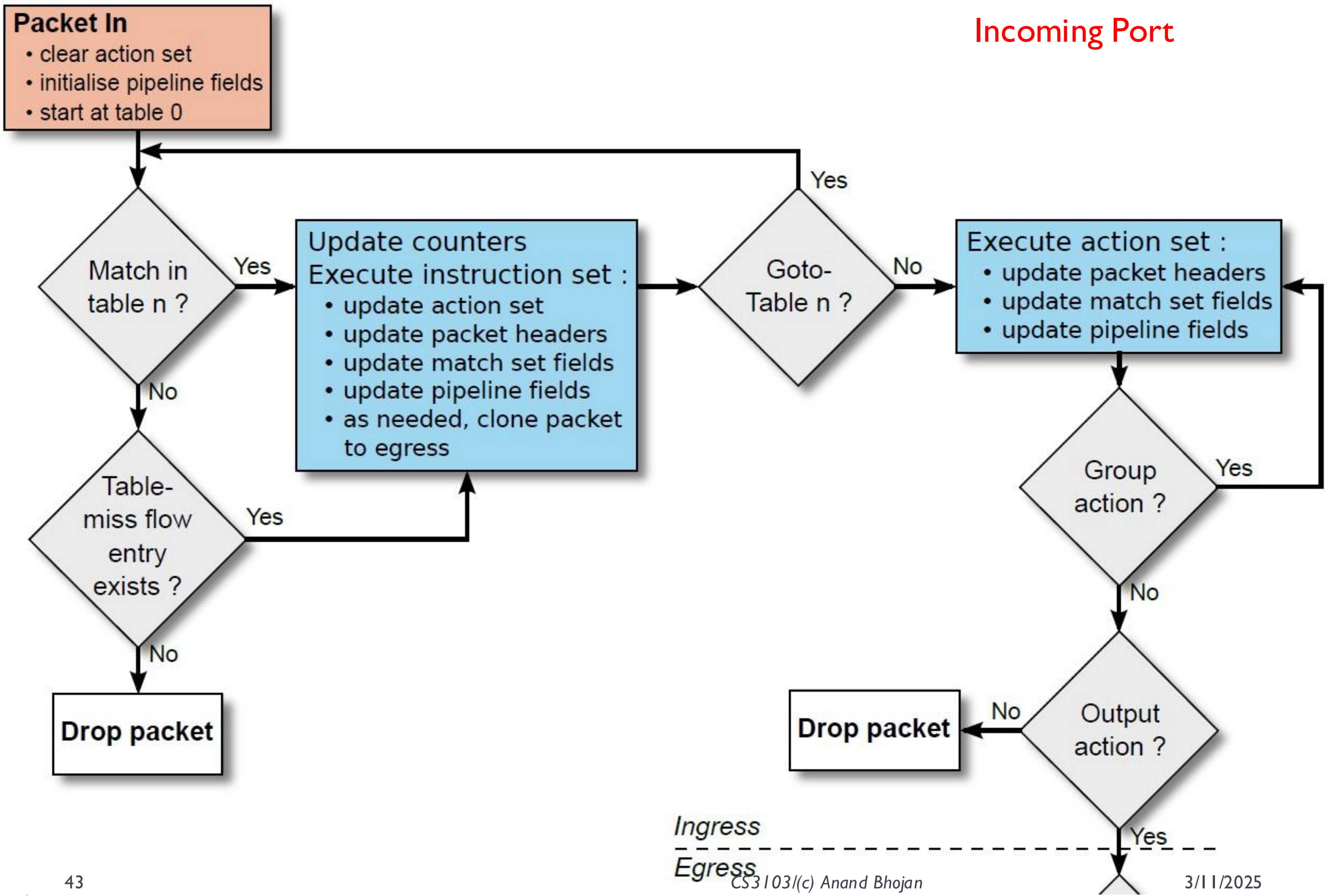


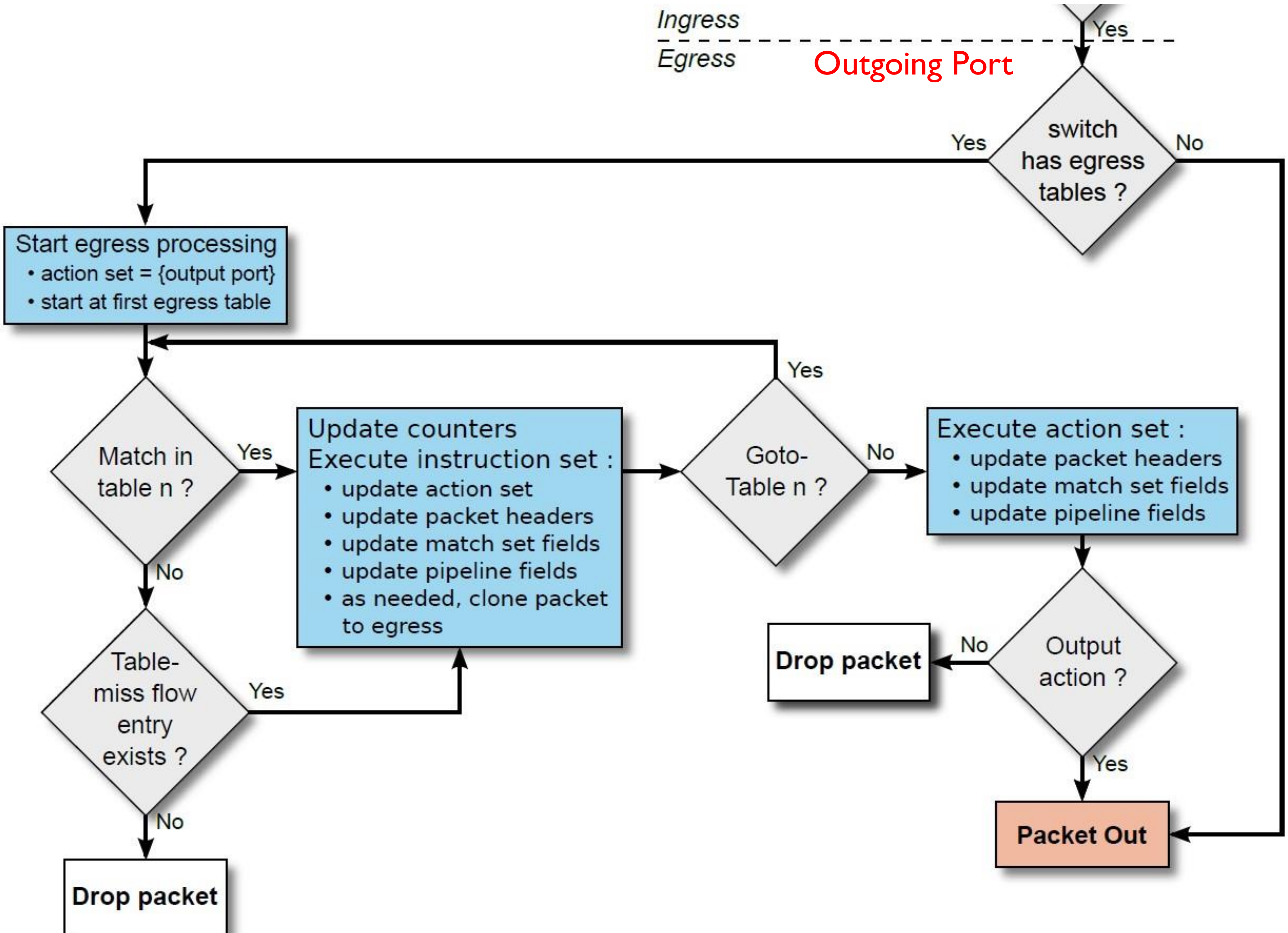
Packet flow through the processing pipeline

OpenFlow terminology associated with packets

- For each packet from a packet flow
  - Header and header field
  - Pipeline fields
    - values attached to the packet during pipeline processing, e.g., ingress port and metadata
  - Action: an operation that acts on a packet
    - e.g., drop, forward to a port, modify (decreasing TTL)
  - Action set
    - accumulated while processed by flow tables
    - executed at then end of pipeline processing

Packet flow through an OpenFlow switch




Ingress
Egress
Outgoing Port
Yes
switch has egress tables ?
Yes
No
Start egress processing
• action set = {output port}
• start at first egress table
Yes
Match in table n ?
Yes
Update counters
Execute instruction set :
• update action set
• update packet headers
• update match set fields
• update pipeline fields
• as needed, clone packet to egress
Goto-Table n ?
No
Execute action set :
• update packet headers
• update match set fields
• update pipeline fields
No
Table-miss flow entry exists ?
Yes
Drop packet
No
Output action ?
Yes
No
Drop packet
Packet Out

Summary - OpenFlow

- ***OpenFlow*** is communication interface between the control and data plane of an *SDN architecture*.
- Openflow Switch Components
  - Flow table
- A packet through OpenFlow Switch

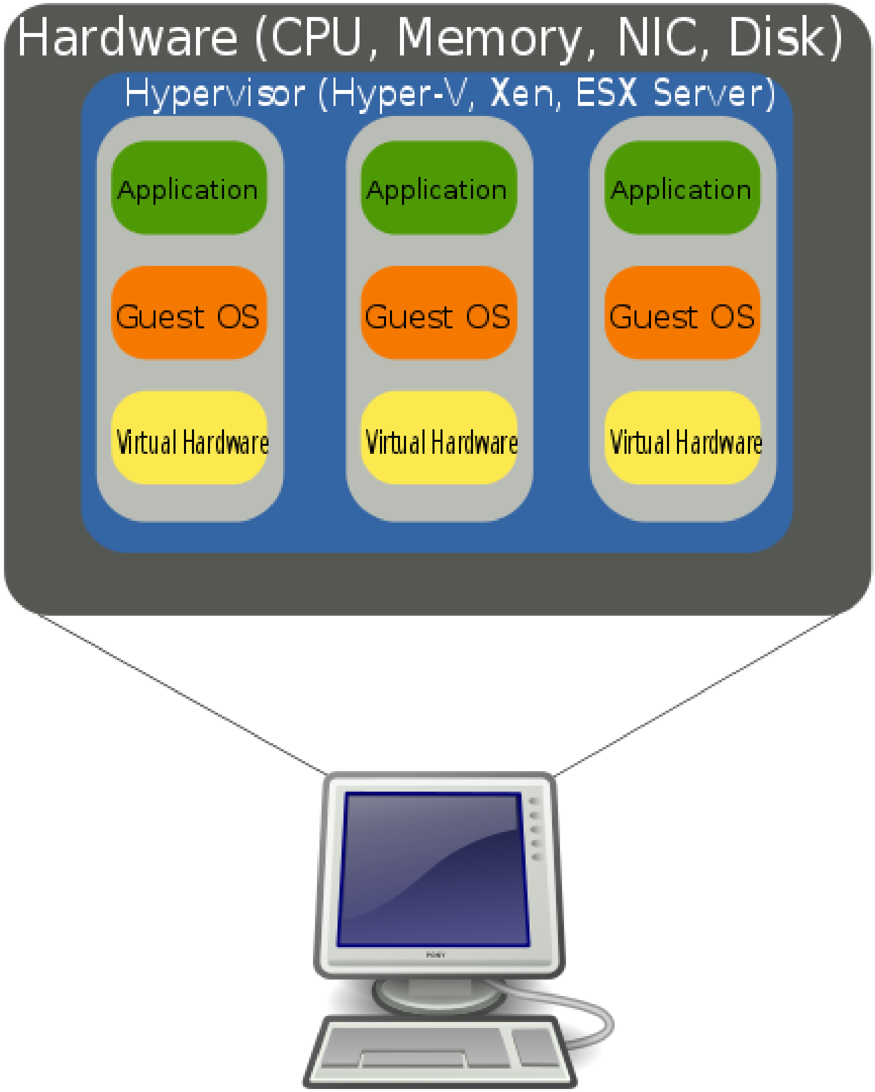
What is Network Virtualisation?

Network Virtualisation

- An application of SDN
- Virtualization
  - Abstraction between the physical resources and their logical representation
  - Can be implemented in various layers of a computer system or network
    - Storage Virtualization
    - Server Virtualization
    - Network Virtualization

Server Virtualisation

- Server virtualization refers to the partitioning of the resources of a single physical machine into multiple execution environments each of which can host a different server.

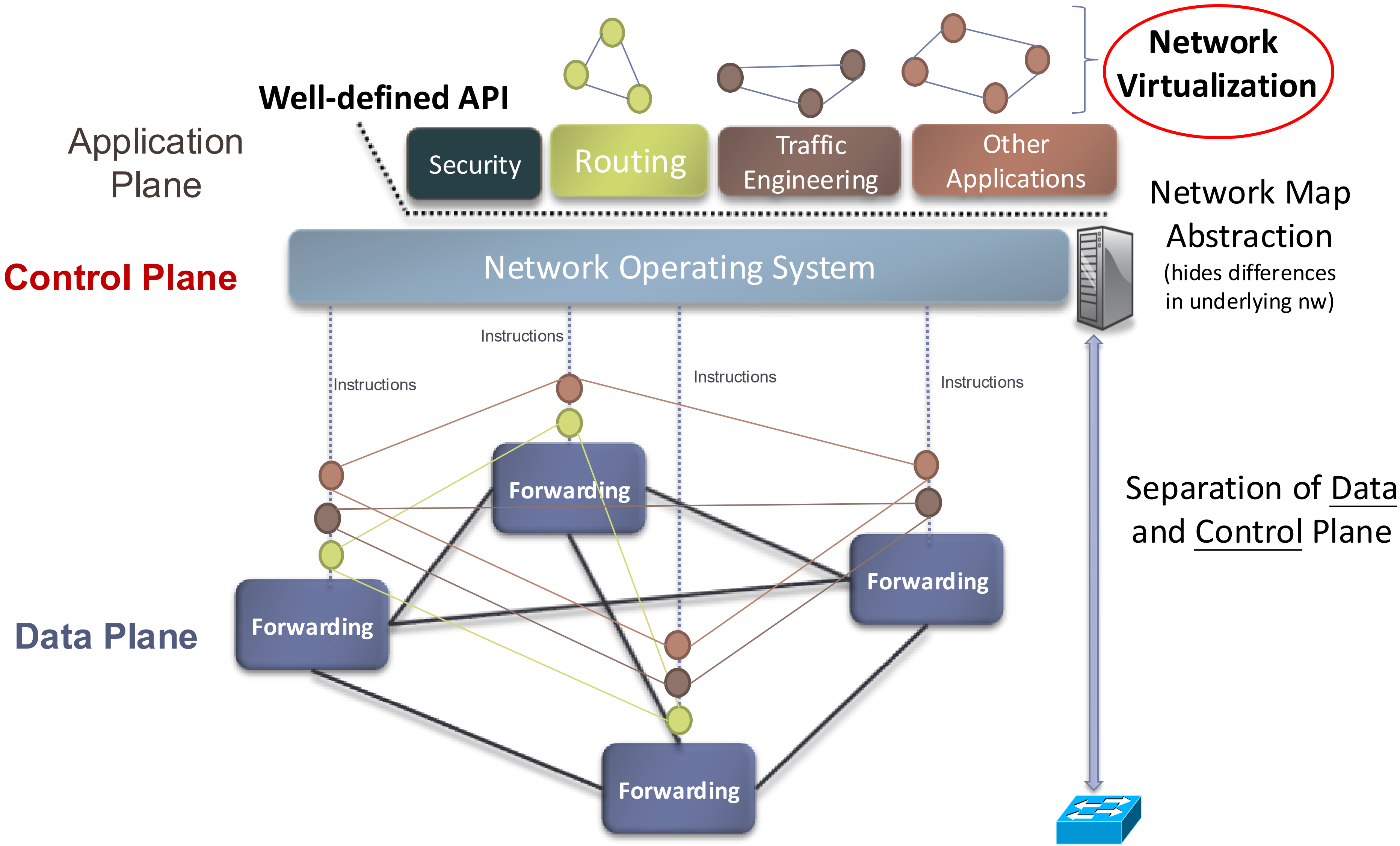


Img src: Infusion Inc

Network Virtualisation

- Allows heterogeneous virtual networks that are isolated, independently managed to coexist over a shared physical network infrastructure.
  - making a physical network appear as multiple logical ones (network slicing)

Software-Defined Network with key Abstractions



Summary – Network Virtualisation

- making a physical network appear as multiple logical ones (network slicing)

Self-Learning Activity:

- *How the following tools help in Network Virtualisation?*
  - *Vswitch*
  - *FlowVisor (Hypervisor for Networks)*
- *Compare the tools available for Network Virtualisation.*

P4

P4 – Language for Programmable NW devices

- P4 = *Programming Protocol-independent Packet Processors*
- High-level language for **programmable network devices** (switches, NICs, routers).
- Allows defining **how packets are parsed, processed, and forwarded**.

Key Features:

- **Protocol-independent** — write your own packet headers, parsers, and actions.
- **Target-independent** — works on programmable ASICs, FPGAs, and software switches.
- **Flexible packet processing** — tables, match-action pipelines, counters, meters.
- **Control over data-plane behavior** — beyond what static hardware allows.

```
Packet In
    |
    ▼
+------------------+
|     Parser       |  <- Extract Ethernet & IPv4 headers
+------------------+
    |
    ▼
+------------------+
|  Match-Action    |  <- ipv4_lpm table
|      Table       |      - Matches dst IP
|                  |      - Actions: forward / drop
+------------------+
    |
    ▼
+------------------+
|    Action        |  <- forward or drop
+------------------+
    |
    ▼
+------------------+
|   Deparser       |  <- Emit headers back to packet
+------------------+
    |
    ▼
Packet Out
```

```
// Match-Action Table
table ipv4_lpm {
    key = { hdr.ipv4.dstAddr: lpm; }
    actions = { forward; drop; }
    default_action = drop();
}

// Forward Action
action forward(bit<48> dst_mac, bit<9> port) {
    hdr.ethernet.dstAddr = dst_mac;
    standard_metadata.egress_spec = port;
}

// Drop Action
action drop() {
    mark_to_drop();
}
```

Note: LPM = Longest Prefix Match. When a packet arrives, the switch/router looks for LPM of the **destination IP** (hdr.ipv4.dstAddr) in the table.

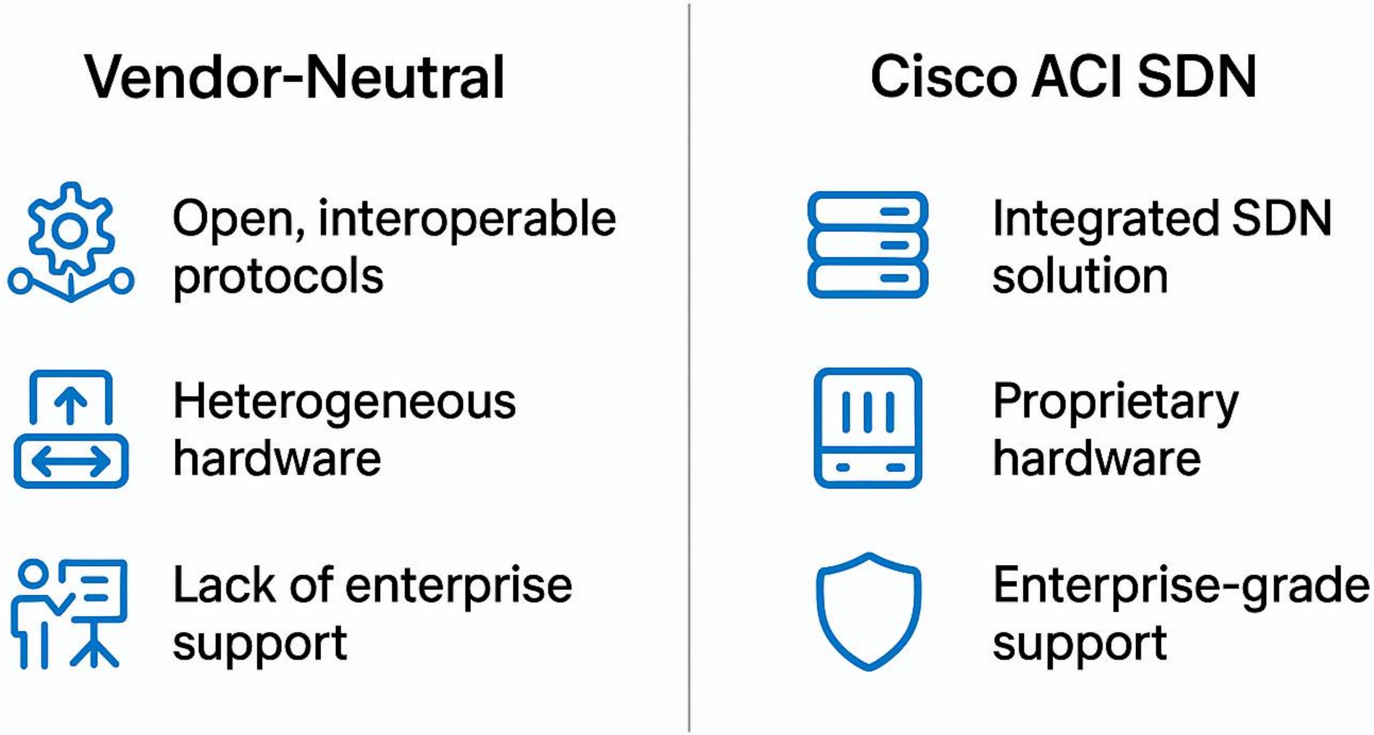
Note: dst_mac is a **parameter passed to the action** (eg. taken from matching LPM entry in the flow table). Set the packet's Ethernet destination MAC address to the value dst_mac.

Career

Vendor specific SDN solutions

- SDN is **conceptually vendor-neutral**, but in practice, **Cisco dominates** because of:
  - Existing installed base
  - Integrated SDN solutions (Cisco ACI - Application Centric Infrastructure)
  - Strong enterprise support, tools, and SLAs
  - Risk aversion and deployment simplicity

Vendor-Neutral SDN vs Cisco ACI



Certifications are still important:-

- Vendor Neutral: ONF Certified SDN **Engineer (OCSE)** or **(Professional) OCSP**.
- Cisco Certifications with DataCentre networking focus are in high demand. For example… **300-620 DCACI exam** to earn the Data Centre ACI Implementation Specialist credential. **Advanced version (DCACIA)** to demonstrate advanced ACI skills.

Salary Range (Data Centre Systems Engineer)

- **Junior to mid-level (2-5 years experience, with certifications but not yet senior): ~** S$60,000-S$90,000/year.
- **Mid to senior level (5-10 years experience, strong certifications + multiple technologies, possibly supervisory/lead role): ~** S$90,000-S$130,000/year.
- **Senior/lead/architect level (10+ years, deep data-centre & vendor stack expertise, maybe team lead or multi-site responsibility): ~** S$130,000-S$180,000+ per year (and possibly more depending on bonus/allowances).

Lab

(Not included for this sem)
(You may try if you like…)

Lab Experiment – Tools (Extension of Lab 2 on MiniNET)

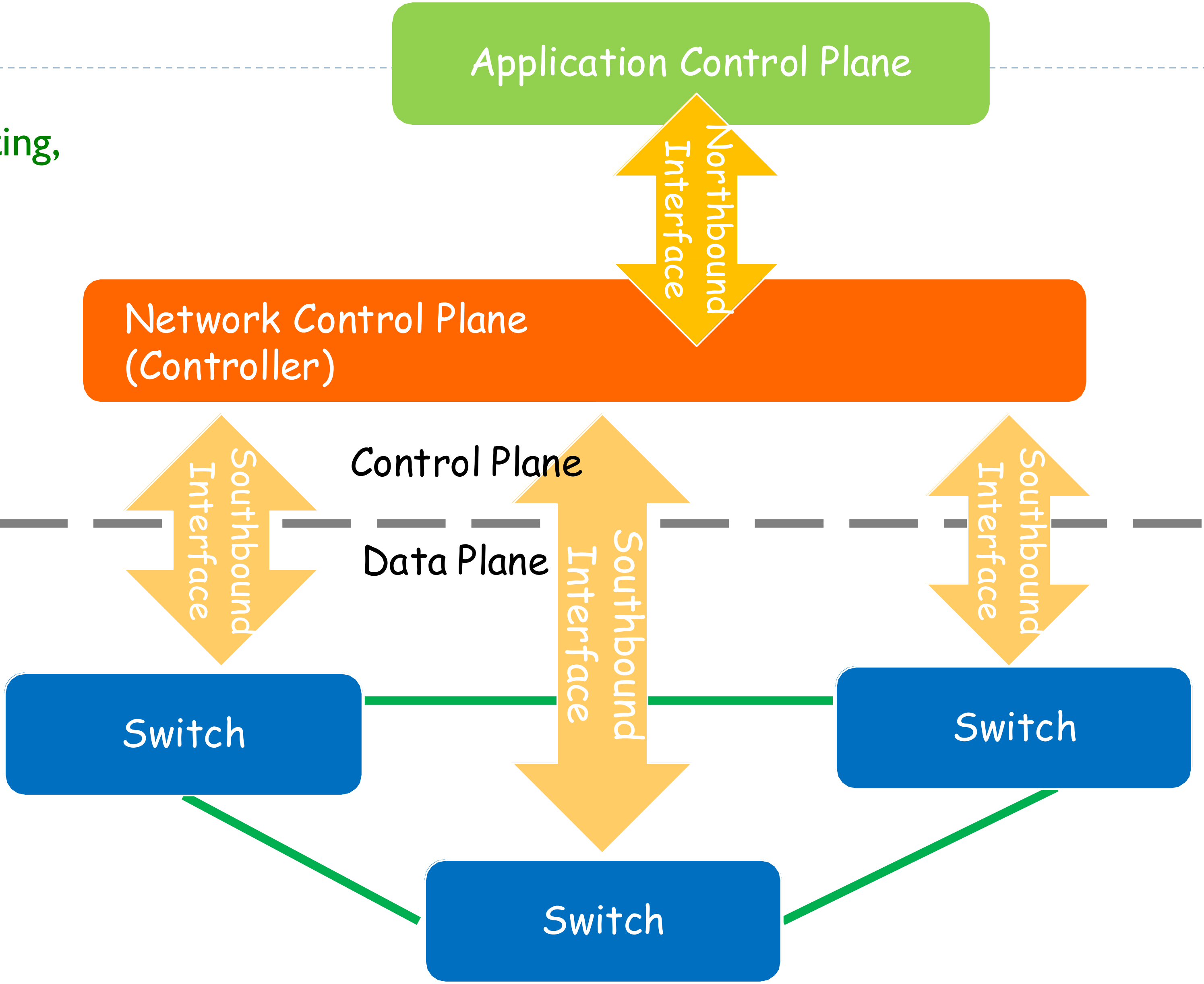
- Mininet (Stanford Univ.)
  - Mininet is a Network Emulator. Mininet creates a Realistic (Realtime) virtual OpenFlow network - controller, switches, hosts, and links - on a single real or virtual machine.
  - Mininet VM – **pre-packaged Mininet/Ubuntu VM**. This VM includes Mininet itself, all OpenFlow binaries and tools pre-installed, and tweaks to the kernel configuration to support larger Mininet networks.
- Virtalbox - Virtualization System (Oracle)
  - VirtualBox is a cross-platform virtualization application. What does that mean? For one thing, it installs on your existing Intel or AMD-based computers, whether they are running Windows, Mac, Linux or Solaris operating systems. Secondly, it extends the capabilities of your existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time.
- Floodlight- SDN Controller
  - Floodlight is written in Java and thus runs within a JVM.

Lab Experiment - Tools

Write your N/W applications (routing, firewall, etc)

Floodlight- SDN Controller

Mininet VM – Provides the Data Plane

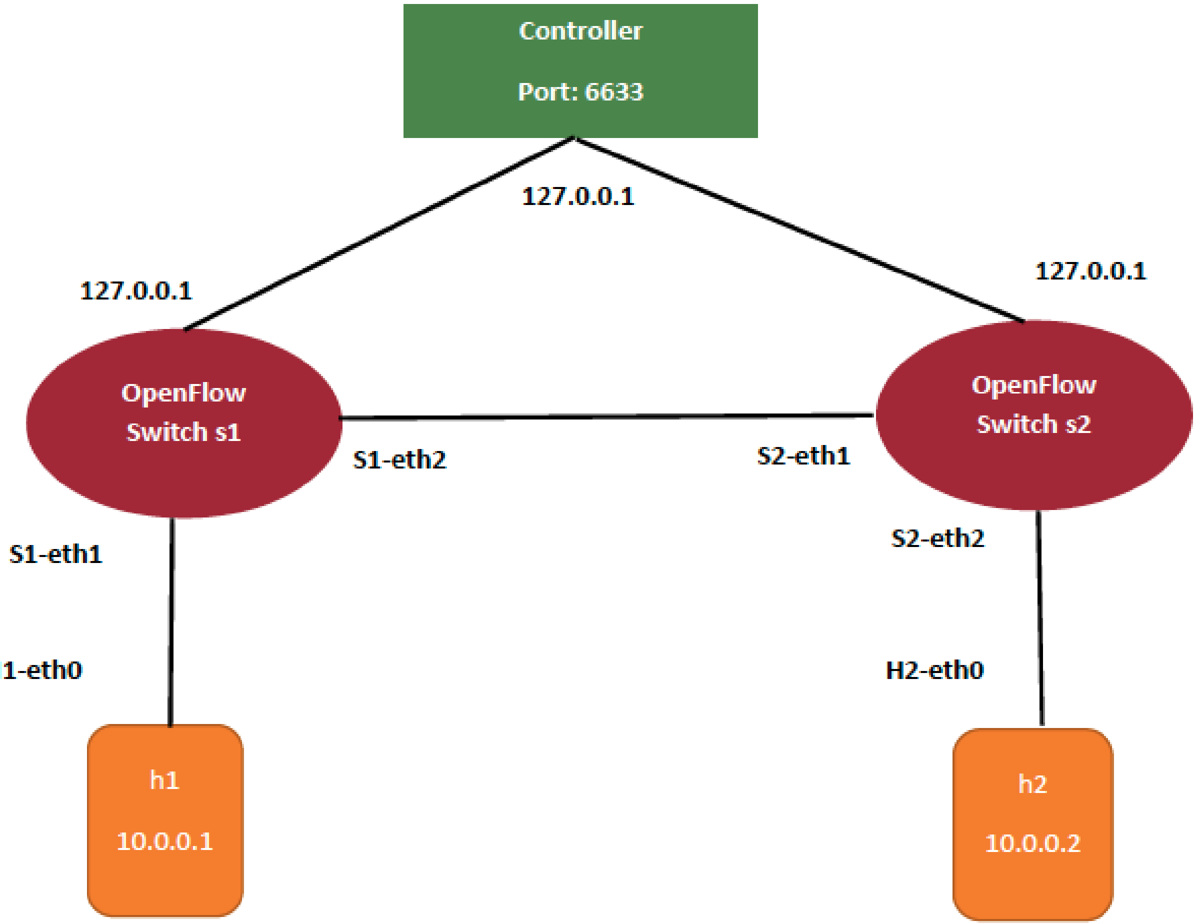


Lab Experiment – Procedure

- Run Virtualbox
- Start VM
  - Command: **startvm**
  - Default User id and password: **mininet**
  - Optional: Use sudo for root access when needed
  - Optional: Use linux 'xterm' command to access any host and run s/w when needed
  - Optional: Run wireshark and capure "**Loopback : Lo" to capture traffic in loopback address of VM.**
- Run Floodlight [when SDN controller is needed)
  - **java -jar target/floodlight.jar**
  - Connect mininet to SDN controller.
  - Mininet contineously communicates with controller over secure connection(TLS) to update topology, receive flowtable updates, etc
  - sudo mn --controller=remote,ip=controller ip,port=6653
    - *Eg. sudo mn --custom LinearTopo.py --topo mytopo --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13*

Lab Experiment – PART A

- A: Create a Virtual Topology



Topology can be created in Python.
For this lab, Pyhton code is provided. (Simple to Understand)

Lab Experiment – PART B & C

- B. Add constraints to Virtual Topology
  - Add bandwidth and delay constraints
- C. Start SDN Controller (Floodlight)
  - Define flow based VLANs

References:

- Sources:
  - “Software-Defined Networking: A Comprehensive Survey”, D. Kreutz, F. Ramos, et el. 2015.
  - “Survey on Software-Defined Networking”, W. Xia, Y. Wen, et el. 2015.
  - Lecture notes : Jennifer Rexford, Scot Shenker, Raj Jain, Bruce Maggs (Duke University), Xenofontas Dimitropoulos (ZTH), Marco Canini (UCL), and unknown Taiwanese scholar.

- Supplement Documents:
  - “Software-Defined Networking: State of the Art and Research Challenges”, M. Jammal, T. Singh, et el.
  - “The Road to SDN: An Intellectual History of Programmable Networks”, N. Feamster, Jenniger Rexford, E. Zegura.
  - “A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Network”, B. Astuto, et el.

Summary

- SDN:- Concept that Separates Control plane and Data plane entities.
  - Network intelligence and state are logically centralised.
  - The underlying network infrastructure is abstracted from the applications.
- ***OpenFlow*** is a communication protocol between the control and data plane of an *SDN architecture*.
- Network Virtualisation - making a physical network appear as multiple logical ones (network slicing)
- ~~Overview to Lab Exercise~~
  - ~~Mininet, VirtualBox, Floodlight~~

End-of-Module Quiz/Test

- **Time:** During Week 13 Lecture Hour [12th Nov 2-3.30pm]
- **Coverage:** All LECTURE contents from week 1 to week 12. [Mostly testing high level concepts. Lab commands, lab settings are not included].
- **Format:** The quiz will be similar to your Weekly lab quiz, there will be MCQs with Single and Multiple Responses.
- **Mode:** CLOSED Book [You can access only Canvas QUIZ->Final Quiz, nothing else. **One cheat sheet, written/typed both sides**). **Bring your laptop fully charged.**
- **Venue:** Same as Lecture Venue.

END!
THANK YOU!