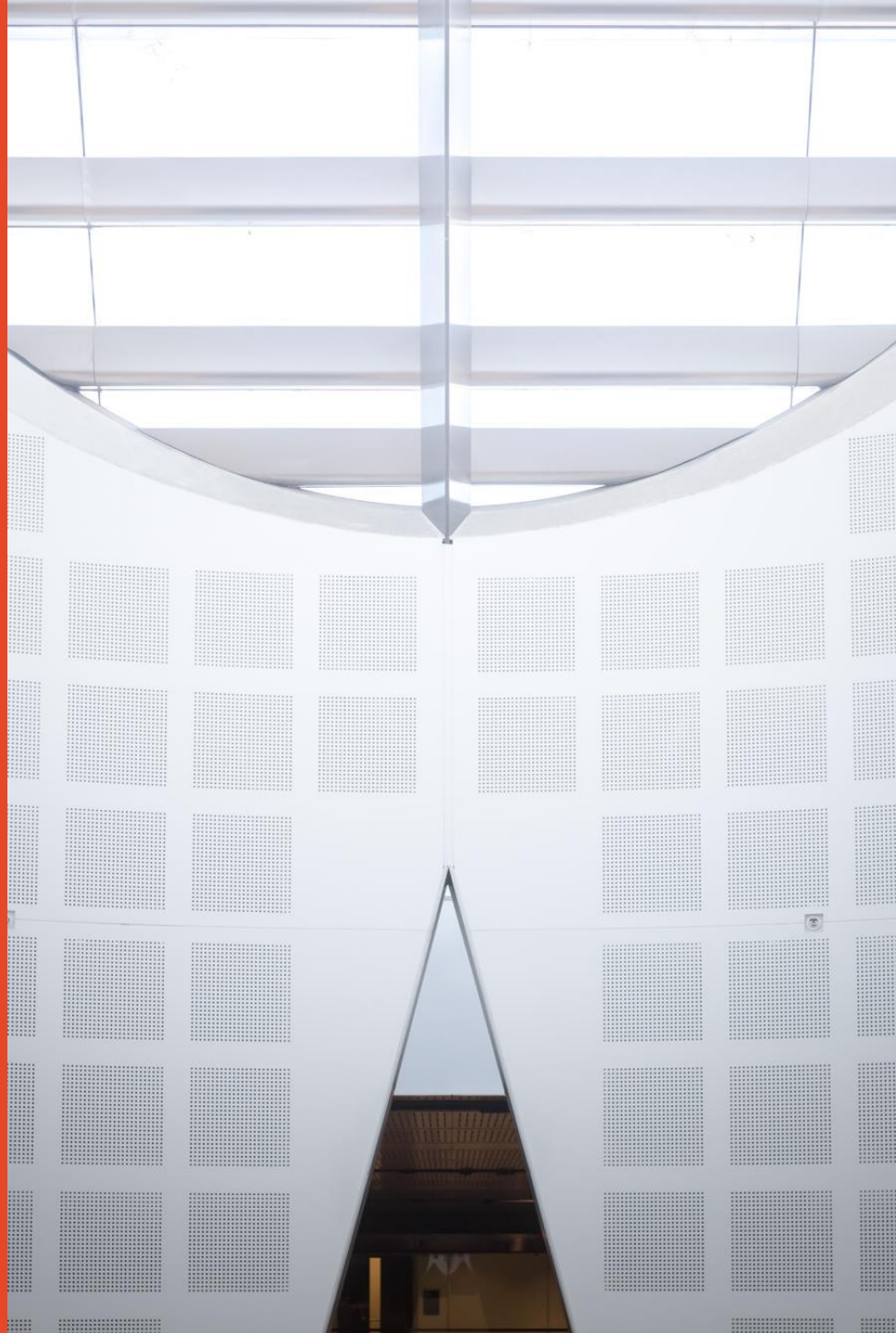
# Topic 3: Introduction to OpenFlow

**Presented by**
Dong YUAN

School of Electrical and Computer Engineering

dong.yuan@sydney.edu.au

THE UNIVERSITY OF SYDNEY

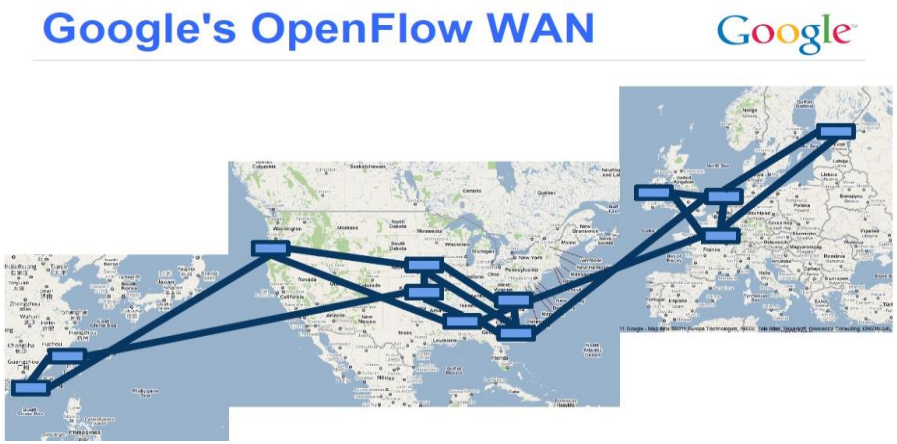


# The Origin of OpenFlow and SDN

- 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 generalizes it to all access policies (*SIGCOMM 2007*).
- The idea of *Software Defined Network* is originated from OpenFlow project (*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.

# OpenFlow Background

- Rapid Development of OpenFlow Technologies
  - 2012 ONF meeting, Google announced that…
    - Google's G-Scale network is operating using OpenFlow
    - Developed for 2 years (2010~2012.1)
    - Saved CAPEX and OPEX




  - OpenFlow was known as an open standard to test experimental protocols in the campus networks
  - OpenFlow → now evolving to Enterprise and Carrier grade SDN technologies
    - Commercial OpenFlow switches and controllers
    - NEC, NTT Data, Nicira , HP, IBM, BigSwitch, Brocade……

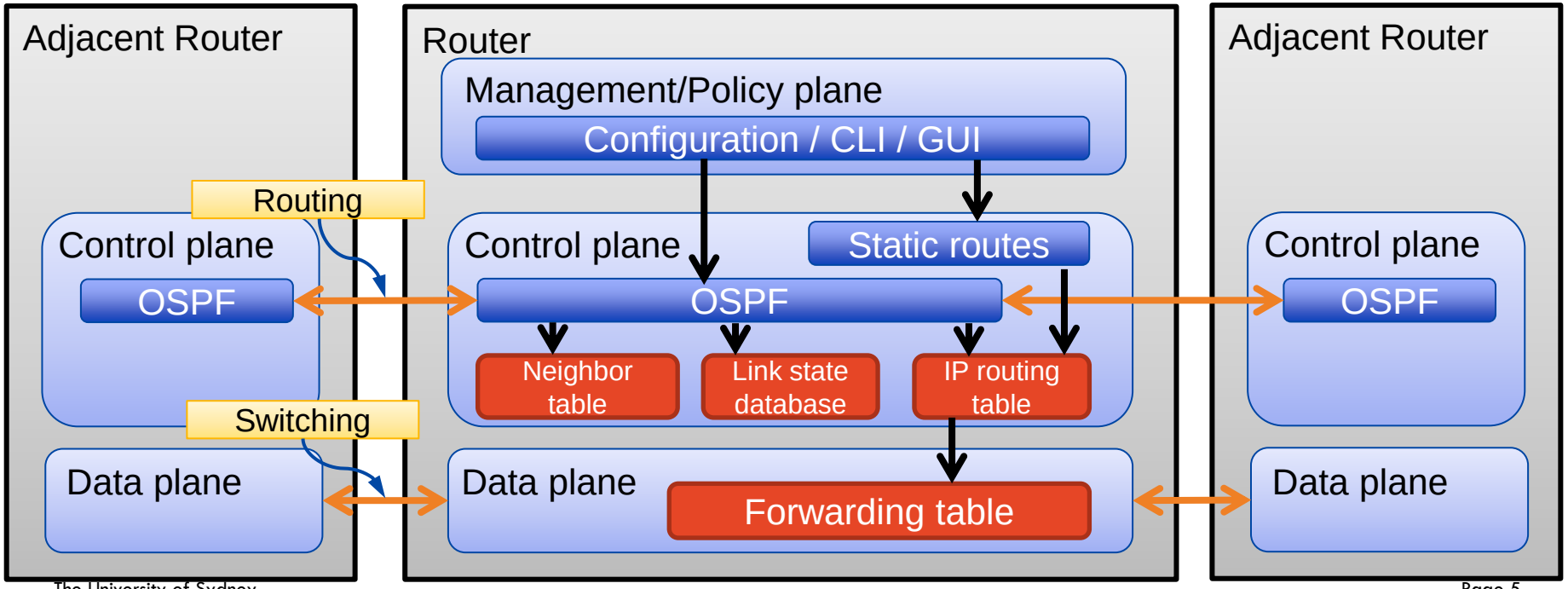
# What is OpenFlow?

- Allow separation of control and data planes.
- Centralization of control.
- Flow based control.
- Takes advantage routing tables in Ethernet switches and routers.

- SDN and 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.

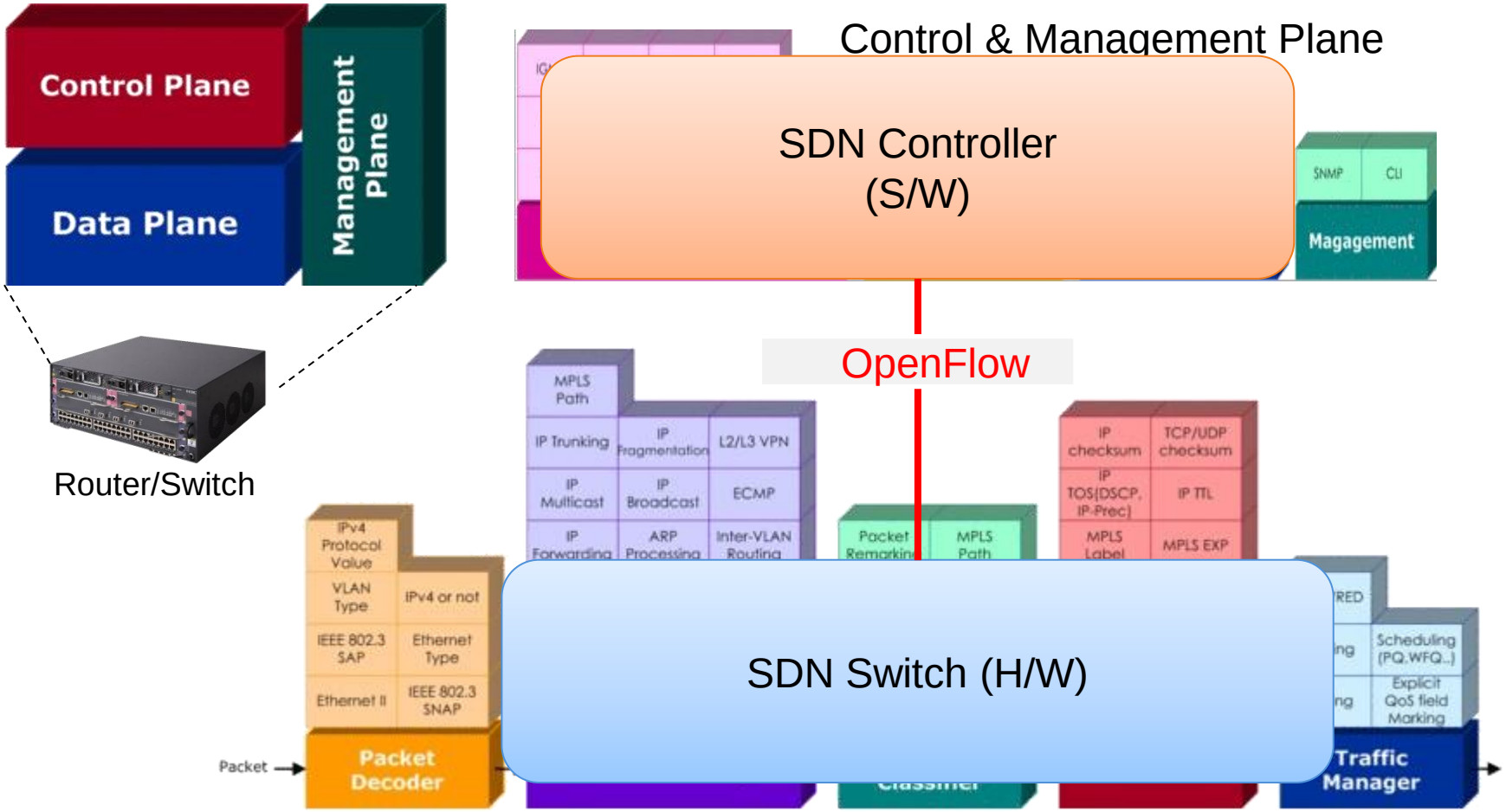
# Traditional Network Node

- Router
  - Router can be partitioned into three planes
    1. Management plane → configuration
    2. Control plane → make decision for the route
    3. Data plane → data forwarding



# SDN Concept

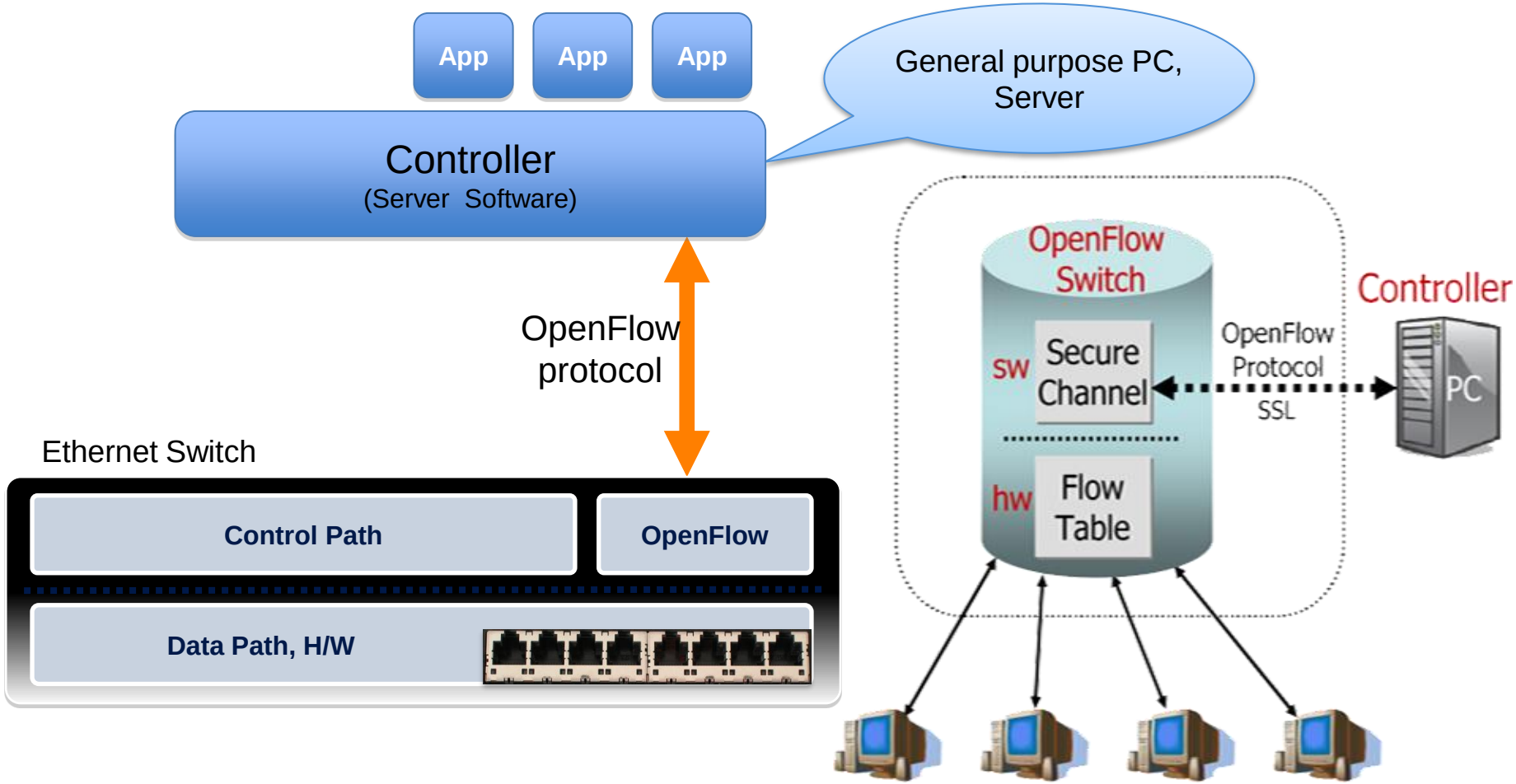
– SDN separates Control and Data plane functions



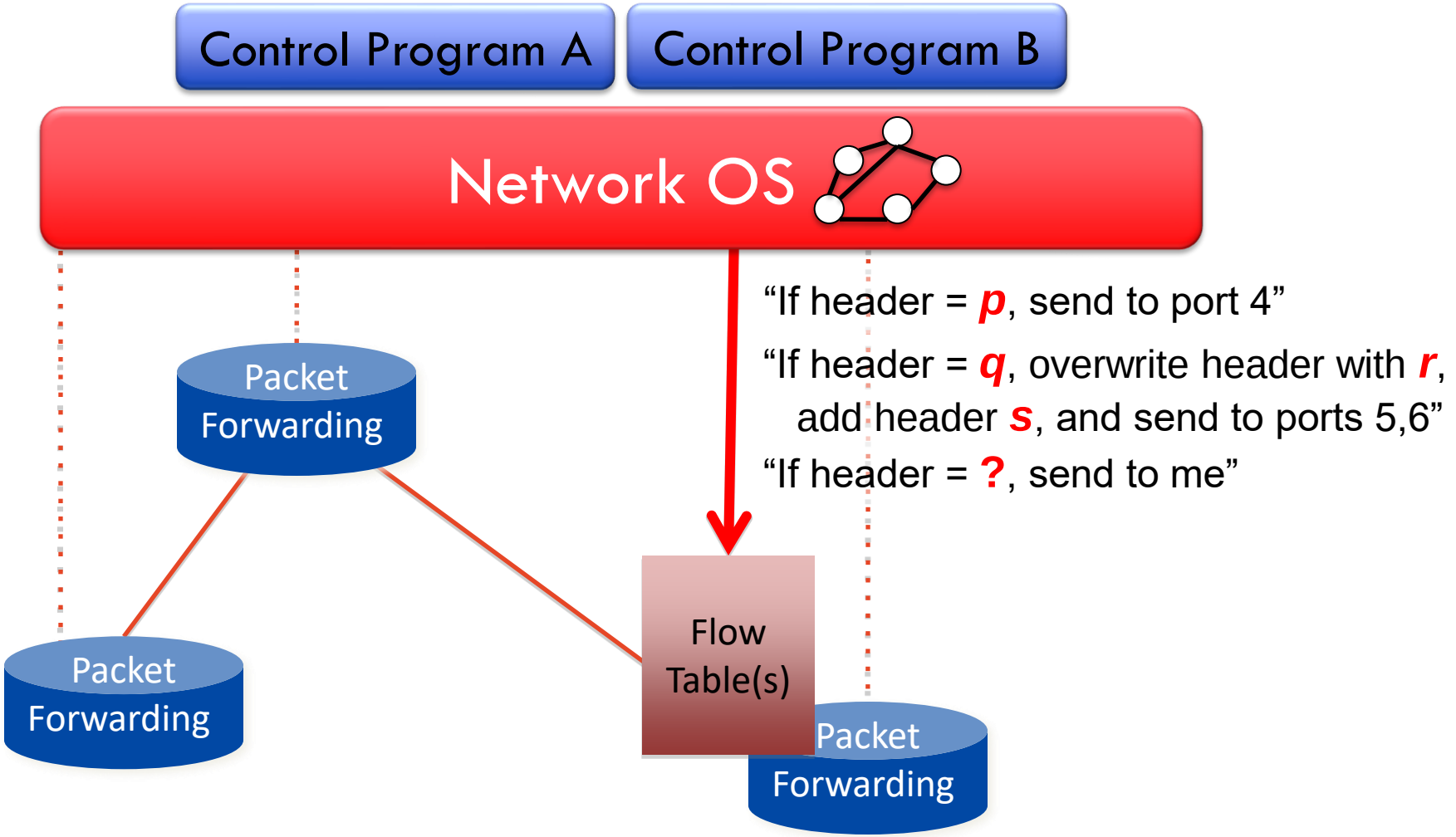

(source "Understanding L3 Switch", Netmanias Talk, 2011/11/09)

# How Does OpenFlow Work (in general)?

– OpenFlow Switch and Tables



# OpenFlow Basics

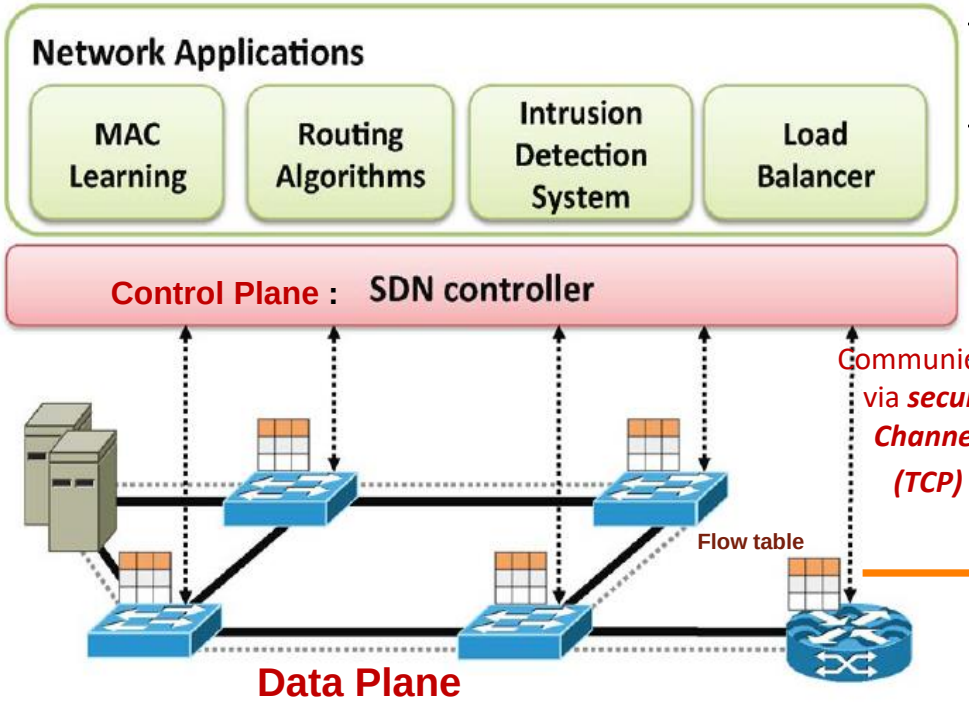


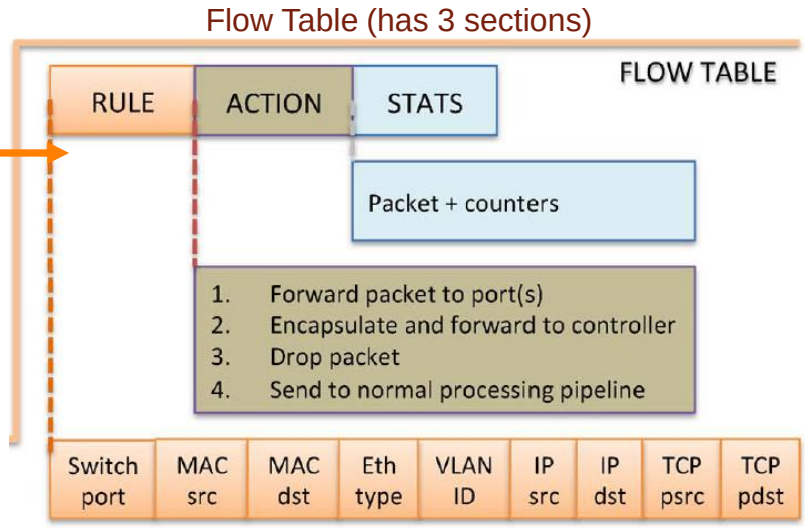
# OpenFlow Basics: Planes of Networking

- Data Plane: all tasks associated with data packets sent by an end user. Examples include forwarding, fragmentation and reassembly etc.
- Control Plane: all tasks that do not involve end-user data packets. These tasks are necessary to perform data plane tasks. Examples include creating routing tables, handling security policies etc.
- Management Plane: all tasks associated with provisioning and monitoring of the networks. Examples include representing new devices and protocols, fault, security, configuration etc.
- Services Plane: services associated with improving performance, security, accounting etc.

# 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 Does not match, Switch asks controller by sending a message with the packet header.



# OpenFlow Protocol Format

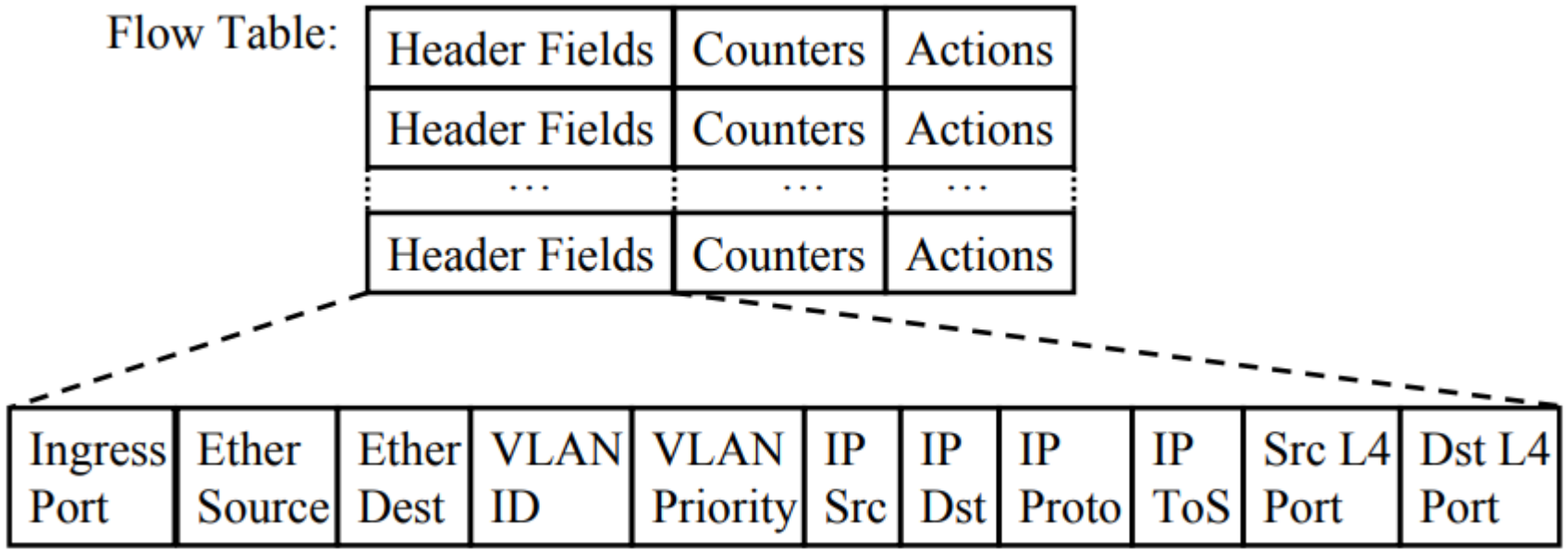
- Protocol Layer
  - OpenFlow control message relies on TCP protocol
  - Controllers listen on TCP port 6633/6653 to setup conn. with switch
  - OpenFlow message structure
    - Version - Indicates the version of OpenFlow which this message belongs
    - Type - Indicates what type of message is present and how to interpret the payload (version dependent)
    - Message length - Indicates where this message will be end, starting from the first byte of header
    - Transaction ID (xid) - A unique value used to match requests to response

**OpenFlow Message Structure**

| Bit Offset | 0 ~ 7 | 8 ~ 15 | 16 ~ 23 | 24 ~ 31 |
|---|---|---|---|---|
| 0 ~ 31 | Version | Type | Message Length | |
| 32 ~ 63 | Transaction ID | | | |
| 64 ~ ? | Payload | | | |

# OpenFlow V1.0

– Packet Arrival: Once packet arrives, the header fields are matched with flow entries in a table, if any entry matches, the counters indicated in that entry are updated and indicated actions are performed.



# Flow Table and Flow Entry Timer

- Example of Flow Table

| Port | Src MAC | Dst MAC | VLAN ID | Priority | EtherType | Src IP | Dst IP | IP Proto | IP ToS | 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 |

- Flow Entry Timer:
  - Idle timeout: Remove entry if no packets received for this time
  - Hard timeout: Remove entry after this time
  - If both are set, the entry is removed if either one expires.
  - Purge Flow timer: Number of seconds after which an invalid OpenFlow entry is deleted from the flow table

# OpenFlow Table: Basic Stats

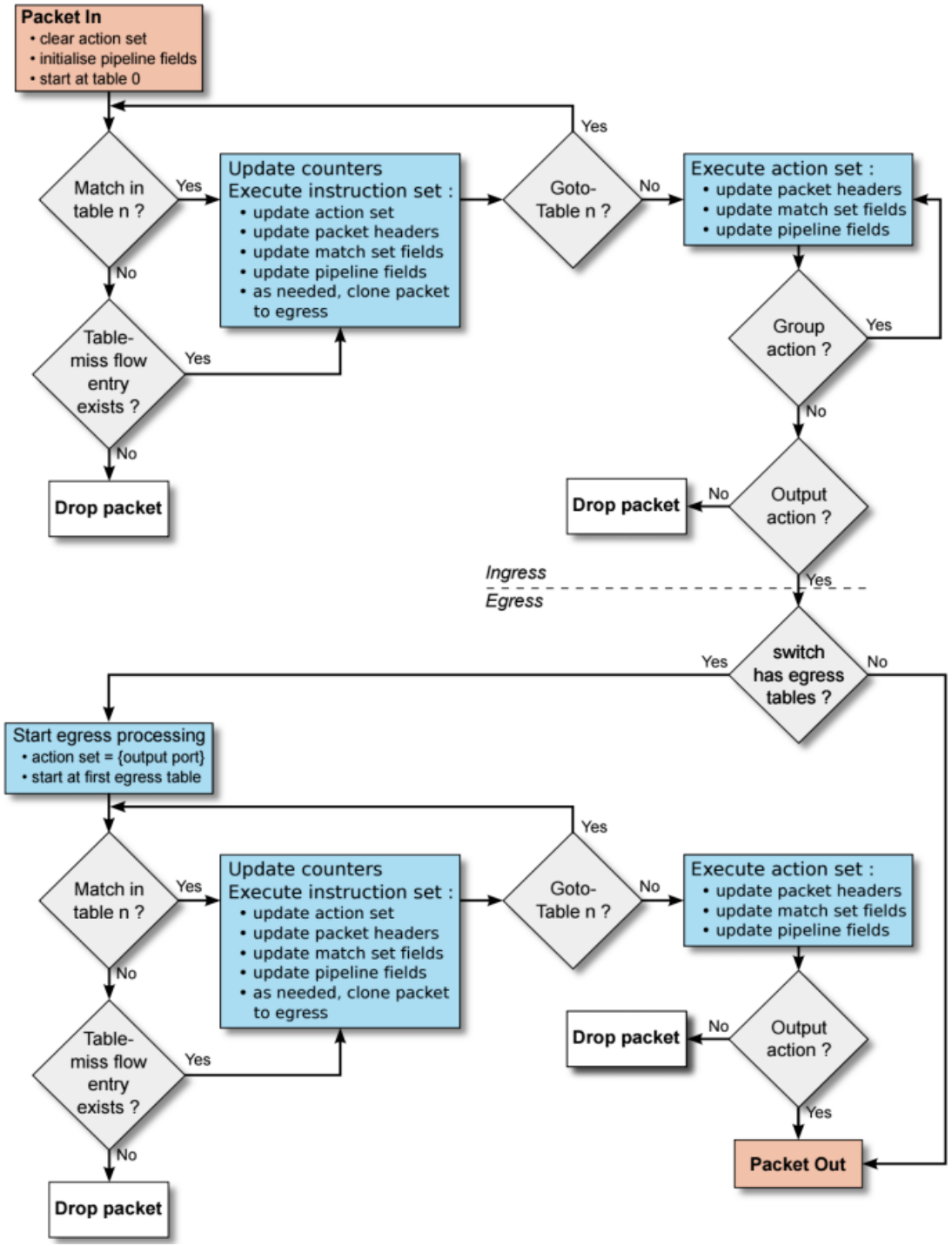
| Per Table | Per Flow | Per Port | Per Queue |
|---|---|---|---|
| Active Entries | Received Packets | Received Packets | Transmit Packets |
| Packet Lookups | Received Bytes | Transmitted Packets | Transmit Bytes |
| Packet Matches | Duration (Secs) | Received Bytes | Transmit overrun errors |
| | Duration (nanosecs) | Transmitted Bytes | |
| | | Receive Drops | |
| | | Transmit Drops | |
| | | Receive Errors | |
| | | Transmit Errors | |
| | | Receive Frame Alignment Errors | |
| | | Receive Overrun erorrs | |
| | | Receive CRC Errors | |
| | | Collisions | |

- Provide counter for incoming flows or packets.
- Information on counter can be retrieved to control plane.
- Can be used to monitor network traffic.

# Additional Feature to Rules and Stats

| OpenFlow Version | Match fields | Statistics | # Matches | | # Instructions | | # Actions | | # Ports | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Req | Opt | Req | Opt | Req | Opt | Req | Opt |
| v 1.0 | Ingress Port | Per table statistics | 18 | 2 | 1 | 0 | 2 | 11 | 6 | 2 |
| | Ethernet: src, dst, type, VLAN | Per flow statistics | | | | | | | | |
| | IPv4: src, dst, proto, ToS | Per port statistics | | | | | | | | |
| | TCP/UDP: src port, dst port | Per queue statistics | | | | | | | | |
| v 1.1 | Metadata, SCTP, VLAN tagging | Group statistics | 23 | 2 | 0 | 0 | 3 | 28 | 5 | 3 |
| | MPLS: label, traffic class | Action bucket statistics | | | | | | | | |
| v 1.2 | OpenFlow Extensible Match (OXM) | | 14 | 18 | 2 | 3 | 2 | 49 | 5 | 3 |
| | IPv6: src, dst, flow label, ICMPv6 | | | | | | | | | |
| v 1.3 | PBB, IPv6 Extension Headers | Per-flow meter | 14 | 26 | 2 | 4 | 2 | 56 | 5 | 3 |
| | | Per-flow meter band | | | | | | | | |
| v 1.4 | — | — | **14** | **27** | **2** | **4** | **2** | **57** | **5** | **3** |
| | | Optical port properties | | | | | | | | |

# OpenFlow Matching



# OpenFlow: Actions

- Drop
- Modify Field
- Masking: this allows matching only selected fields. Examples include Dest. IP, Dest. MAC etc.
- Forward: forward to physical port or to virtual port. Examples include
  - All: to all interface except incoming interface
  - Controller: encapsulate and send to controller
  - Table: Perform actions in the flow table
  - Etc.
- Secure Channel: Between controller and the switch using TLS
- Flow tables have already been implemented in modern switches using Ternary Content Addressable Memories (TCAMs)
- Flow table entries can be sent by controller beforehand or on-demand.
- Etc.

# Models can be perfect and clean, reality is dirty!

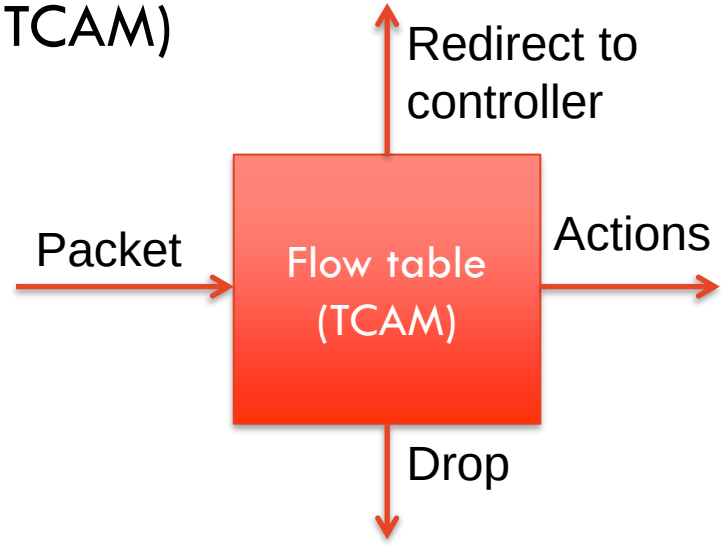
- The match/action model can ideally be used to program any network behavior and to get rid of protocol limitations at any level
- But unfortunately, with OF:
  - Matches can be done only on a set of predefined header fields (Ethernet, IPv4, MPLS, VLAN tag, etc.)
  - Actions are limited to a rather small set
  - Header manipulation (like adding label/tags, rewriting of fields, etc.) is limited to standard schemes
- As a result, OF is not really protocol independent and standards (including OF standards) are still necessary

# Where do OF limitations come from?

- OpenFlow has been designed having in mind current specialized HW architecture for switches

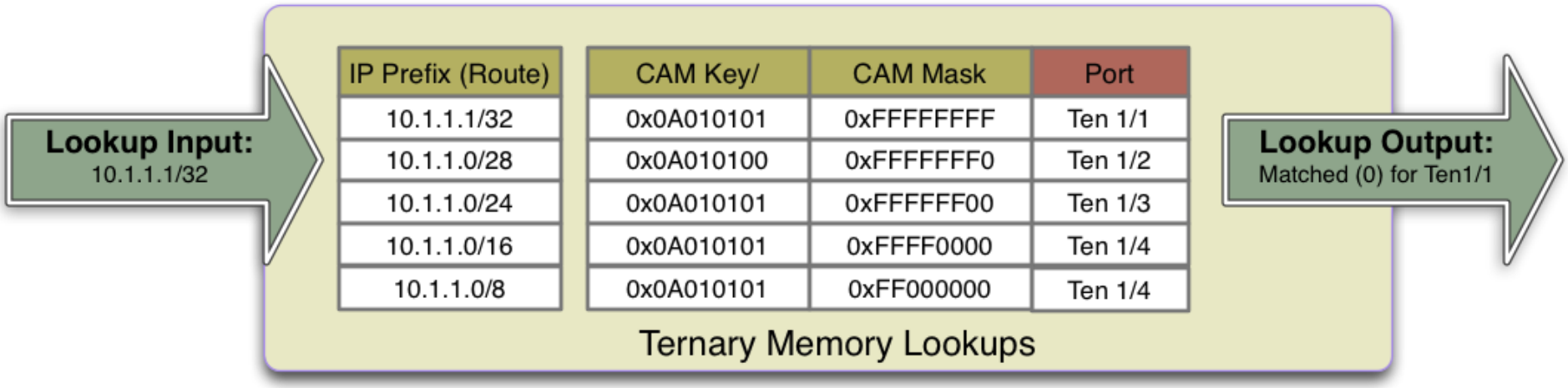
- Specialized HW is still fundamental in networking
  - General purpose HW (CPU) and soft-switches are still 2 order of magnitude slower
  - Architectures based network processors are also at least 1 order of magnitude slower

- Ternary Content Address Memory (TCAM)



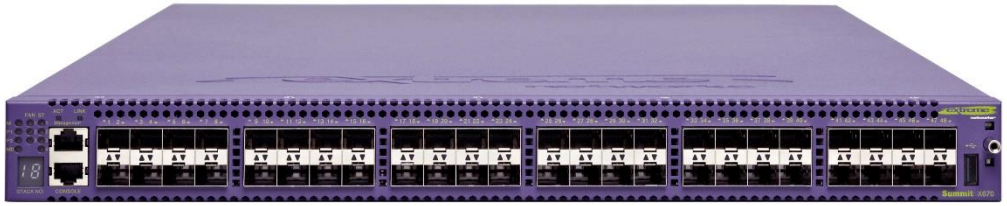
# Where do OF limitations come from?

- The reference HW model for OF flow tables is TCAM (Ternary Content Addressable Memory)
  - TCAM (ternary content-addressable memory) is a specialized type of high-speed memory that searches its entire contents in a single clock cycle.
  - A Ternary CAM (TCAM) stores 0, 1, and "don't care". The "Don't Care" at an additional cost over binary CAM since the internal memory cell must now encode three possible states. This is usually implemented by adding a mask bit ("care" or "don't care" bit) to every memory cell.



# Where do OF limitations come from?

- TCAMs however are typically expensive components that are used by manufacturers only when strictly necessary
- Less expensive memory components based on predefined search keys are often used for most of the common functions of a switch
- OF success depends on its "vendor neutral" approach where implementations issues are completely opaque (including reuse of standard modules for e.g. MAC and IP forwarding)
- Specialized ASICs are typically complex with a number of hard limitations on table types, sizes, and match depth



# Software and Hardware OpenFlow Switches

- **Software**
  - XORPlus: Open source switching software
  - Pantou: commercial wireless router/access point to an OpenFlow enabled switch
  - LINC: Open source implementation that runs on Linux, Solaris, Windows, MACOS and FreeBSD
  - Indigo: Open source implementation that runs on physical switches
  - Open vSwitch
- **Hardware**
  - IBM 8264
  - HP 3500, 5400zl, 6600 etc.
  - NetGear 7328SO, NetGear 7352SO
  - Juniper MX, Juniper EX
  - Arista 7050
  - Brocade MLXe, Brocade CER, etc.

# Open vSwitch

- Nicira concept
- Open Source Virtual Switch
- Default switch in XenServer 6.0, Xen Cloud Platform and supports VirtualBox, Xen KVM etc.
- Integrated into many cloud management systems including OpenStack, OpenQRM etc.
- Distributed with Ubuntu, Fedora Linux , Debian, FreeBSD
- An accelerated version of Open vSwitch in its own Data Plane Development Kit (DPDK) is available with Intel

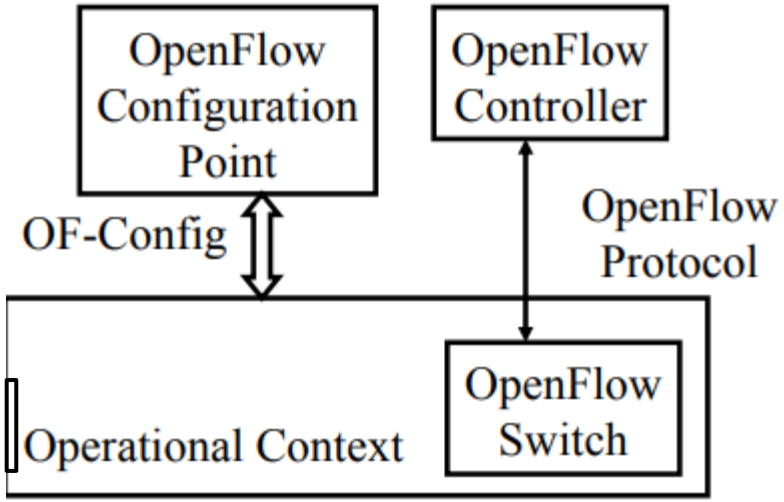
# Bootstrapping

- Switches require initial configuration: Switch IP address, Controller IP address, Default gateway
- Switches connect to the controller
- Switches provides configuration information about ports
- Controller installs a rule to forward Link Layer Discovery Protocol (LLDP) packets to controller then sends, one after the other, LLDP packets to be sent out to port i (i=1,2,…,n) which are forwarded to respective neighbors. The neighbors send the packets back to controller
- LLDP is a one-way protocol to advertise the capabilities at fixed intervals
- Controller determines the topology from LLDP packets

# OpenFlow Configuration Protocol (OF-Config)

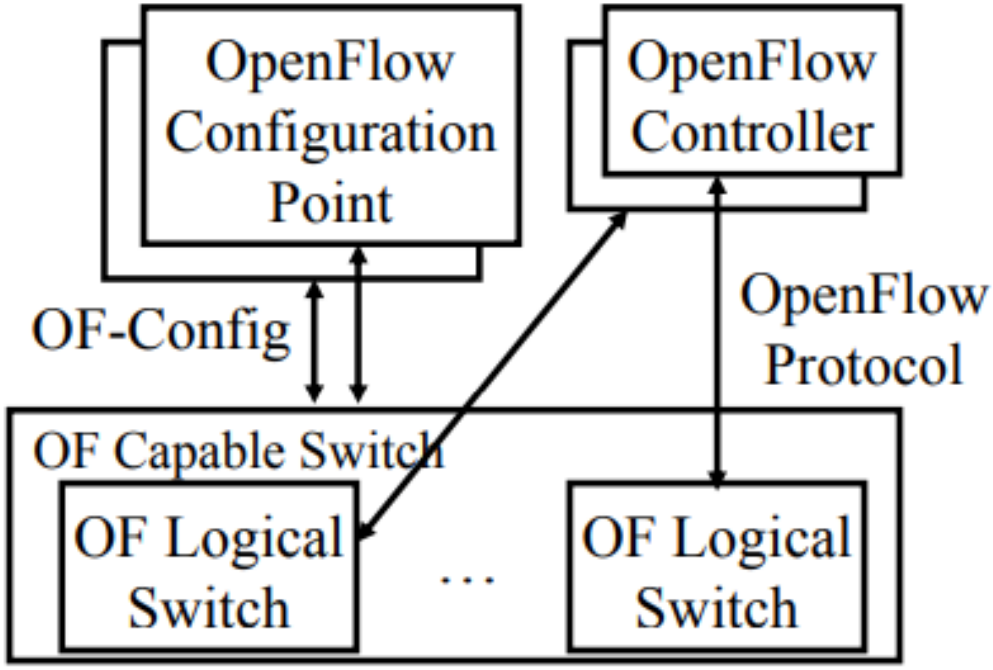
- OpenFlow Control Point: Entity that configures OpenFlow switches

- Protocol used for configuration and management of OpenFlow Switches, assignment of OF controllers so that switches can initiate connections to them using IP address of controller, port number of controller, TCP etc.



# OF-Config contd.

- A physical switch = one or more logical switches each controlled by an OF controller
- Allows configuration of logical switches



# Openflow advantages

- Ease of configuration
- Ease of network management
- Security
  - DDoS attack
- Availability
  - Load balancing
  - Adding new resources
  - Fault tolerance

# Applications

- Network virtualization and NFV
- Data center network
- Wide area network and SD-Wan
- Wireless network and operators' network

# Challenges

- Reliability
  - Single point failure
- Scalability
  - Support large scale network
- CAPEX and OPEX
  - Equipment upgrade and compatibility
  - New requirements for network engineers
- Security

# Security challenges

- Man-in-the-middle Attacks
  - TLS is difficulty to implement
  - Listener Mode
- Switch Authentication
- Flow Table Verification
- Denial of Service Risks
  - For the controllers

# Questions

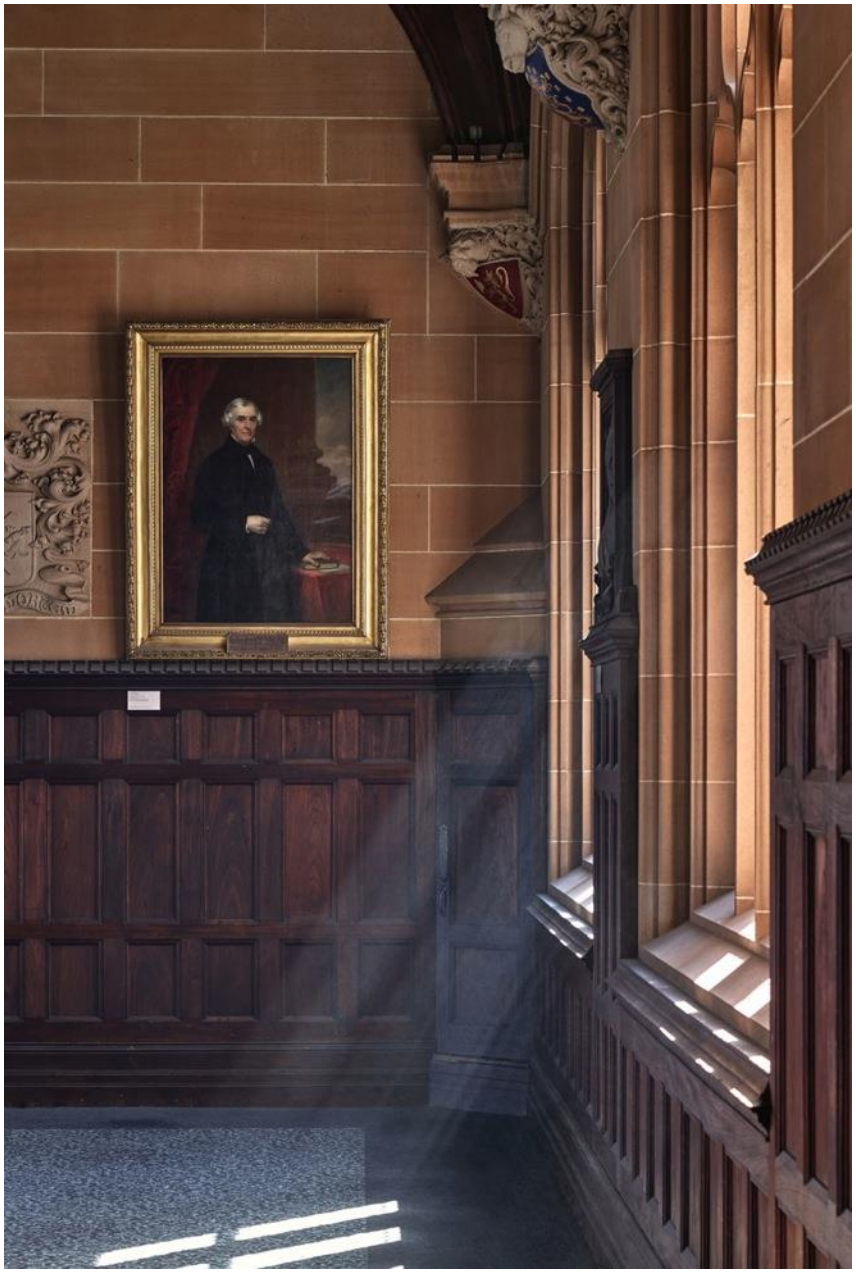
- What's the difference between flow tables in SDN and routing tables in traditional network?
  - Difference between a switch and a router
- How are routing algorithms implemented with Openflow?
  - Comparing to traditional network
- Referring to 7 Layers OSI model, which is Openflow's layer(s)?

# Reading references


- Casado, Martin, et al. "Ethane: Taking control of the enterprise." *ACM SIGCOMM computer communication review* 37.4 (2007): 1-12.
- McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." *ACM SIGCOMM Computer Communication Review* 38.2 (2008): 69-74.
- Lara, Adrian, Anisha Kolasani, and Byrav Ramamurthy. "Network innovation using openflow: A survey." *IEEE communications surveys & tutorials* 16, no. 1 (2013): 493-512.
- Benton, Kevin, L. Jean Camp, and Chris Small. "OpenFlow vulnerability assessment." *ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 151-152. 2013.

# Thank you!

End



THE UNIVERSITY OF SYDNEY