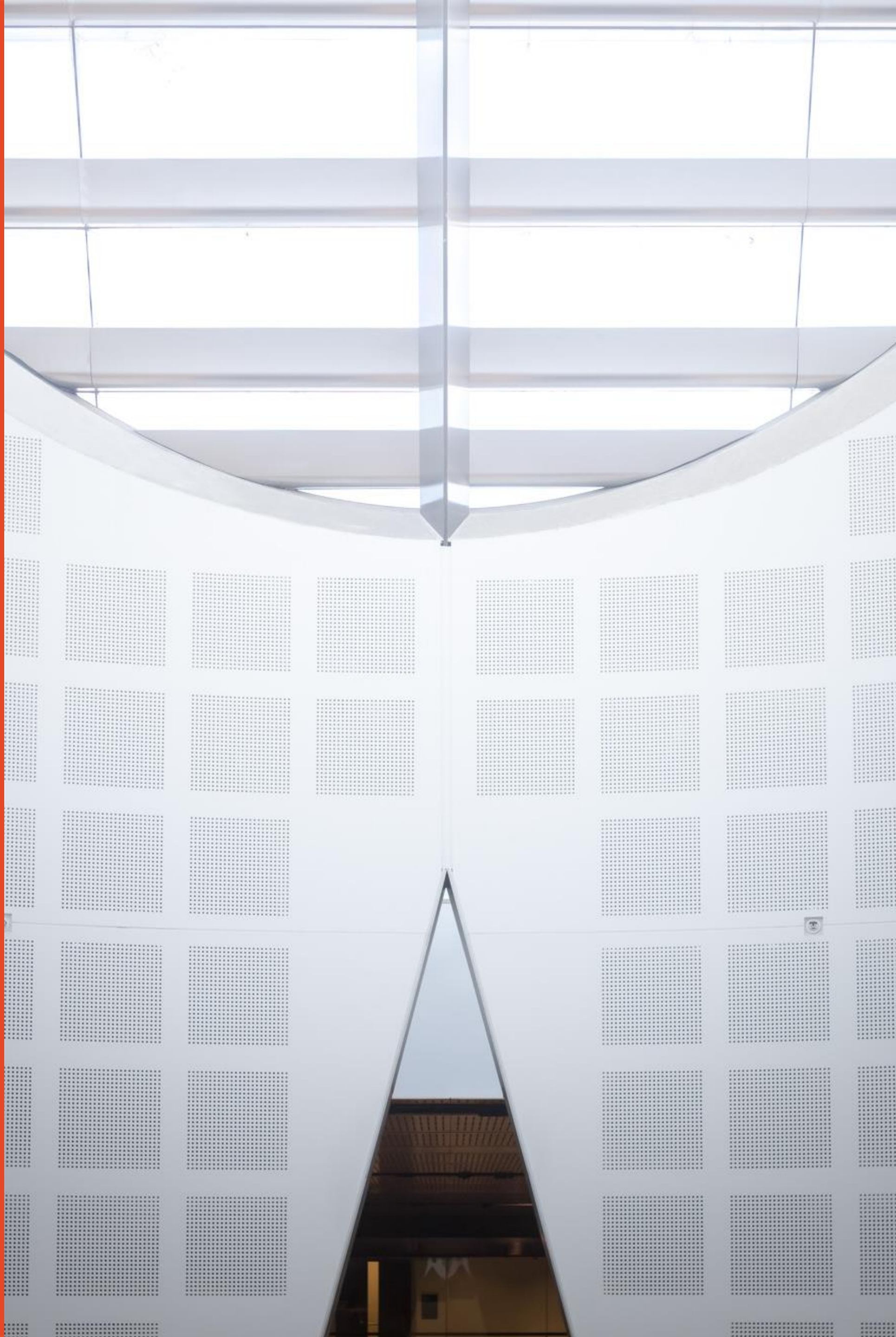
Unit Review

Presented by
Dong YUAN

School of Electrical and Computer Engineering

dong.yuan@sydney.edu.au

THE UNIVERSITY OF SYDNEY



Topics covered in this unit

- Introduction to SDN
- Networking Technologies Basics
- OpenFlow and Mininet
- Smart Switches
- Controllers Design
- ONOS Controller
- Programmable Data Planes
- Virtualisation
- Network Function Virtualisation
- SDN applications (CORD & Datacenter)
- SDN applications (SDX & SDWan)

New Internet Architecture

- Cannot make changes to the fundamental technologies of Internet, e.g., IP, routing, etc.
  - High cost to adopt new tech
  - Risks of malfunctions
  - Commercial concerns: need to see the benefit

- Redesign the Internet Architecture
  - **Stanford's Clean Slate** Initiative
  - 100s of project funded world-wide
  - WWW Conference and Journal

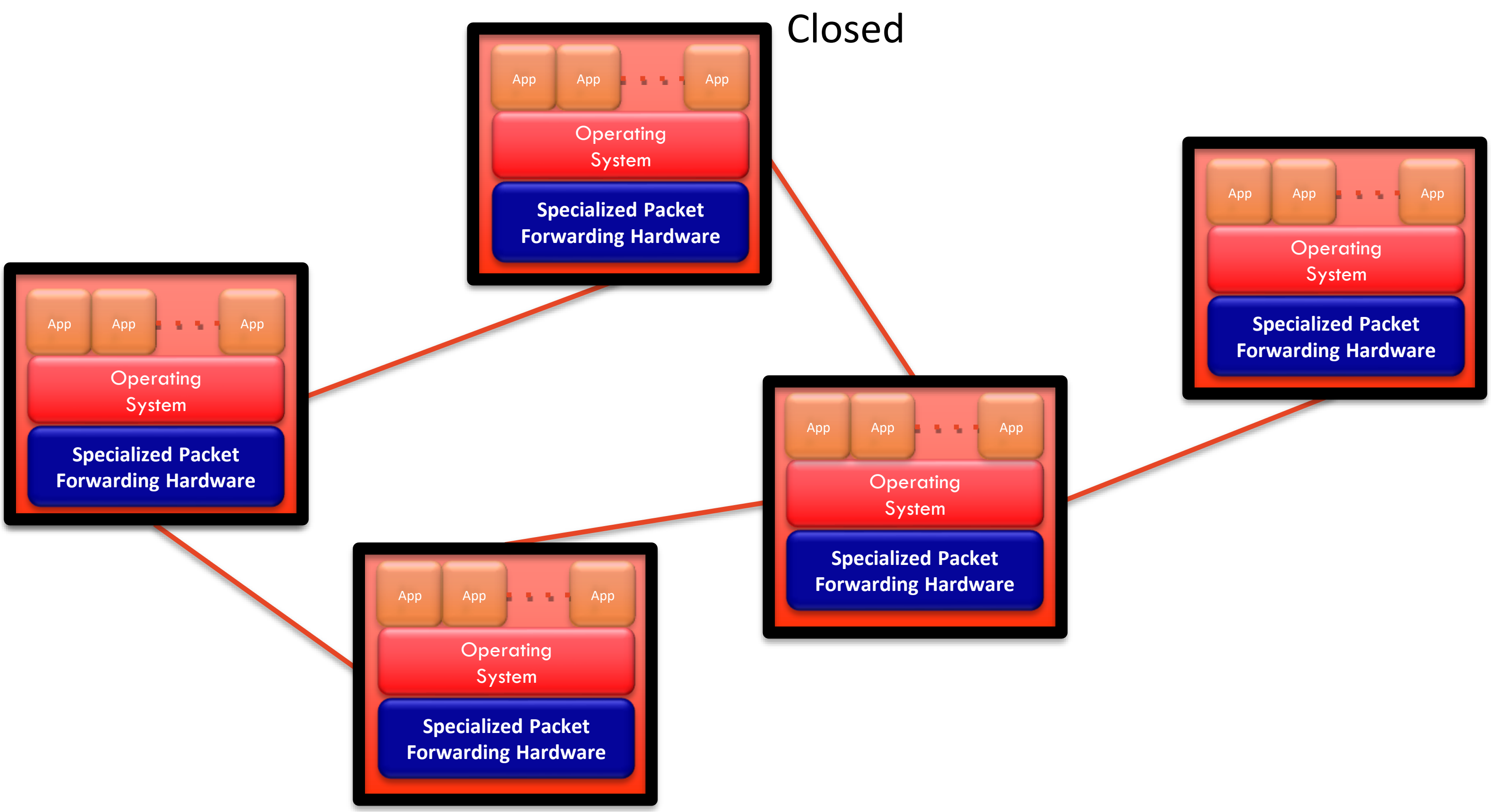
What is SDN?

- A new architecture that makes networks more programmable than in the past

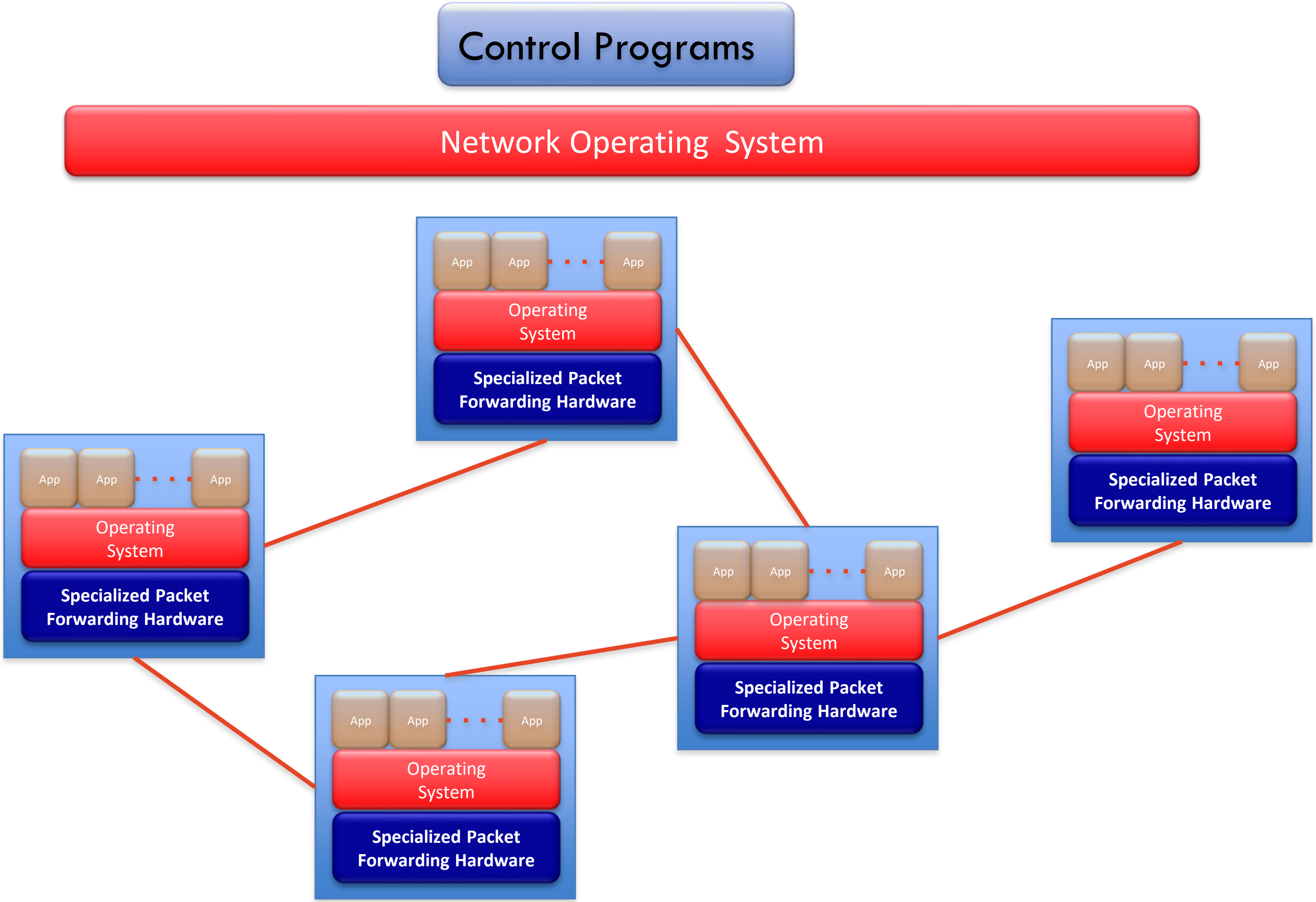
- Key principles:
  - Centralised control
  - **Open interfaces**
  - **Flow-based routing**

- **OpenFlow is the most widely adopted protocol.**
  - It is not SDN, but is one important SDN technology

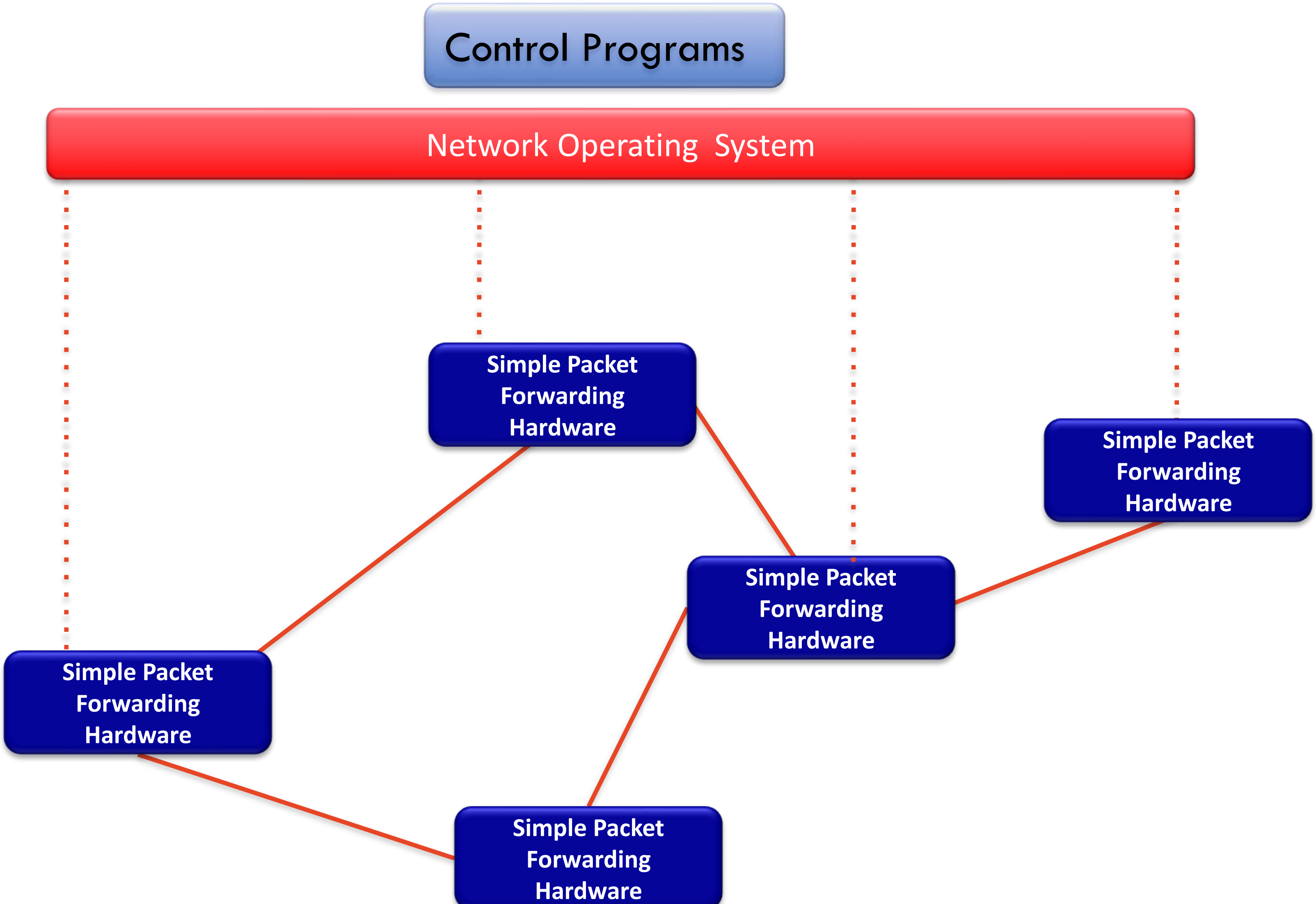
Idea: An OS for Networks



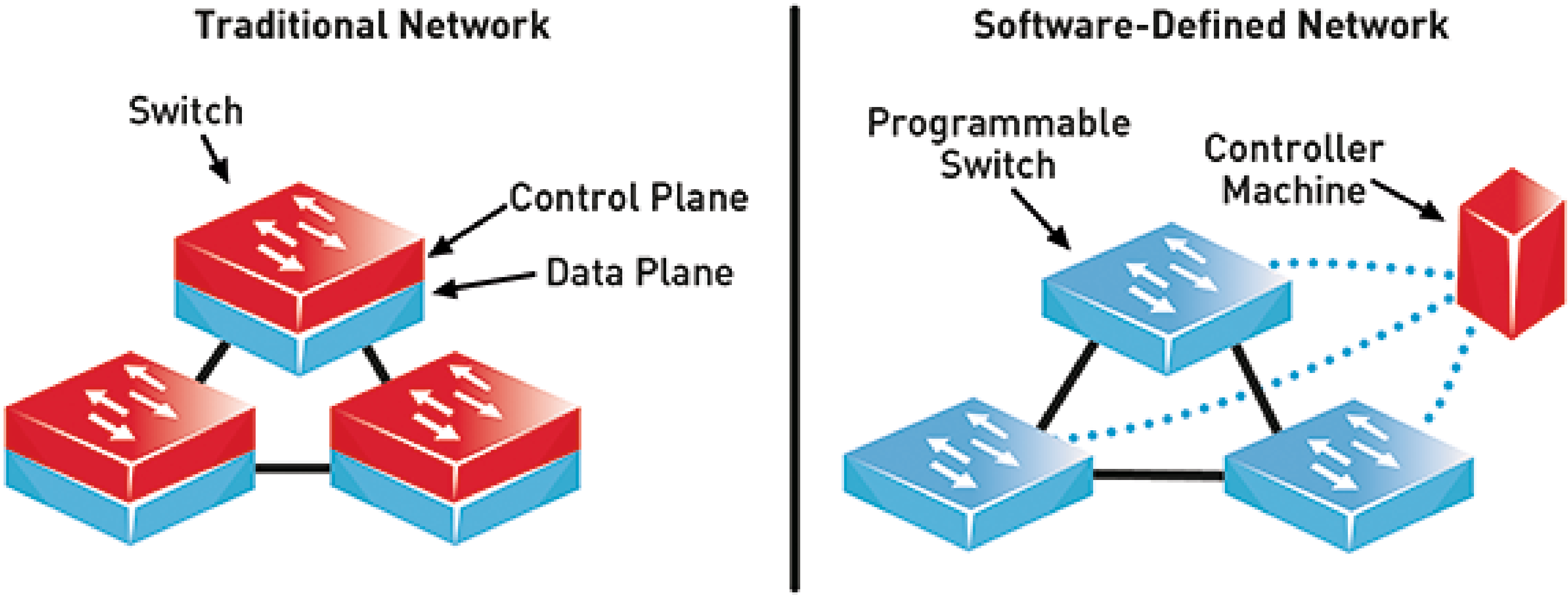
Idea: An OS for Networks



Idea: An OS for Networks



Key difference



– *Separation of data plane and control plane*

SDN Basic 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.

Mininet

- Provides a simple and inexpensive network testbed for developing OpenFlow Applications
  - OpenFlow: protocol to send/receive forwarding rules from controller to switches
  - Some key ideas of OpenFlow: centralization and flow based control
- Enables complex topology testing (without need to wire up a physical network)
- Multiple concurrent developers can work independently on the same topology
- Usable out of the box without programming
- Includes a topology-aware Command Line Interface (CLI) for running or debugging networks
- Supports system-level regression tests (verifies that the previously developed and tested network still performs the same way after changes)

OpenFlow

- OpenFlow-protocol to send/receive forwarding rules from controller to switches
- Control logic is represented as a controller
- Switches perform forwarding

- 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

- Match criteria: source, destination, protocol, etc
  - Defined and added dynamically (so called SDN)
- Actions:
  - flow to which port
  - sending packet to controller
  - Drop a packet,
  - Modify packet header, etc.
- Priority: for match criteria
  - Time out period:
    - Hard TO, Idle TO

Controller

- In openflow, controller communicates with switch over a secure channel
  - Openflow protocol defines message format
  - Purpose of control channel: update flow table
  - Logic is executed at controller
  - Communication with external controllers

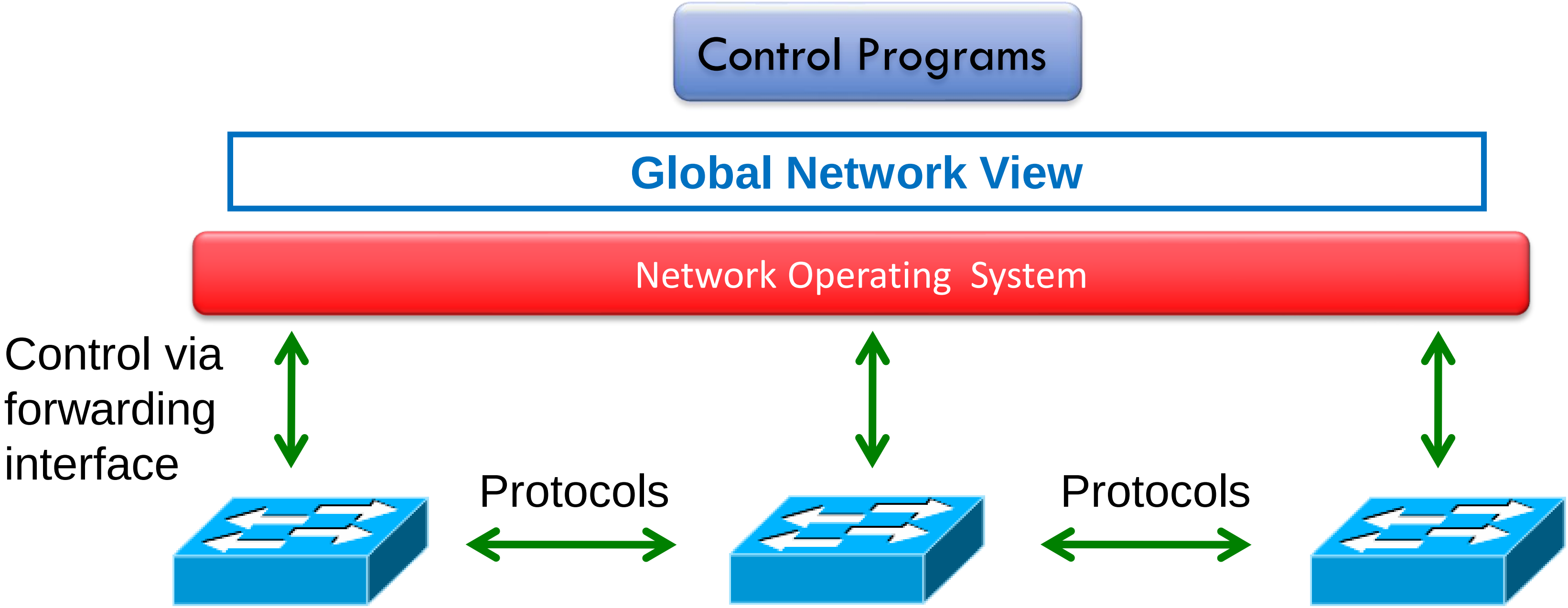
- Flow Table: Packet switching
  - All packets compare to flow table for match
    - Packet header fields matched against one of the N tables
  - Actions depend on match being found
    - Forward, Drop, modify, enqueue
  - If no match, traffic is send to controller

Openflow switches

- Whitebox Switch
  - White box switches refers to the ability to use ‘generic,’ off-the-shelf switching (or white box switching) and routing hardware, in the forwarding plane of a software-defined network (SDN).
  - They represent the foundational element of the commodity networking ecosystem
  - A common operating system for white box switches is Linux-based because of the many open and free Linux tools available that help administrators customize the devices to their needs.

Idea: An OS for Networks

Software-Defined Networking (SDN)



The Future of Networking, and the Past of Protocols, Scott Shenker, *with Martin Casado, Teemu Koponen, Nick McKeown*

Network OS

- Maintain an up-to-date view of the network state (e.g., topology, etc.)
  - Next page

- Configure network elements
  - A.k.a “south-bound” interface or API

- Provide a graph abstraction to the applications on the top
  - A.k.a “north-bound” interface or API

Network View (graph abstraction)

- In the network view:
  - Switch-level topology
  - Location of host, middlebox and other network elements
  - Location of users
  - Namespace: bindings between names and addresses
- Not in the network view:
  - Some states of network, e.g., traffic.

- Network Information Base (NIB)
  - Graph and abstraction

Challenges in building an NOS

- Scalability
- Reliability

- Good performance (fast, etc.)
- Generality and Simplicity of the “north-bound” API.
  - Easily to be used by applications

Today's Popular controllers

- ONOS and OpenDayLight (ODL)
- ONOS (2014)
  - From Open Networking Foundation
  - Previously ON.LAB funded by Stanford and Berkeley
- ODL (2013)
  - From Linux Foundation

- Both ONOS and ODL are written in Java and designed for modular use with a customizable infrastructure
- Both support OpenStack
- Every ONOS partner is also an ODL member

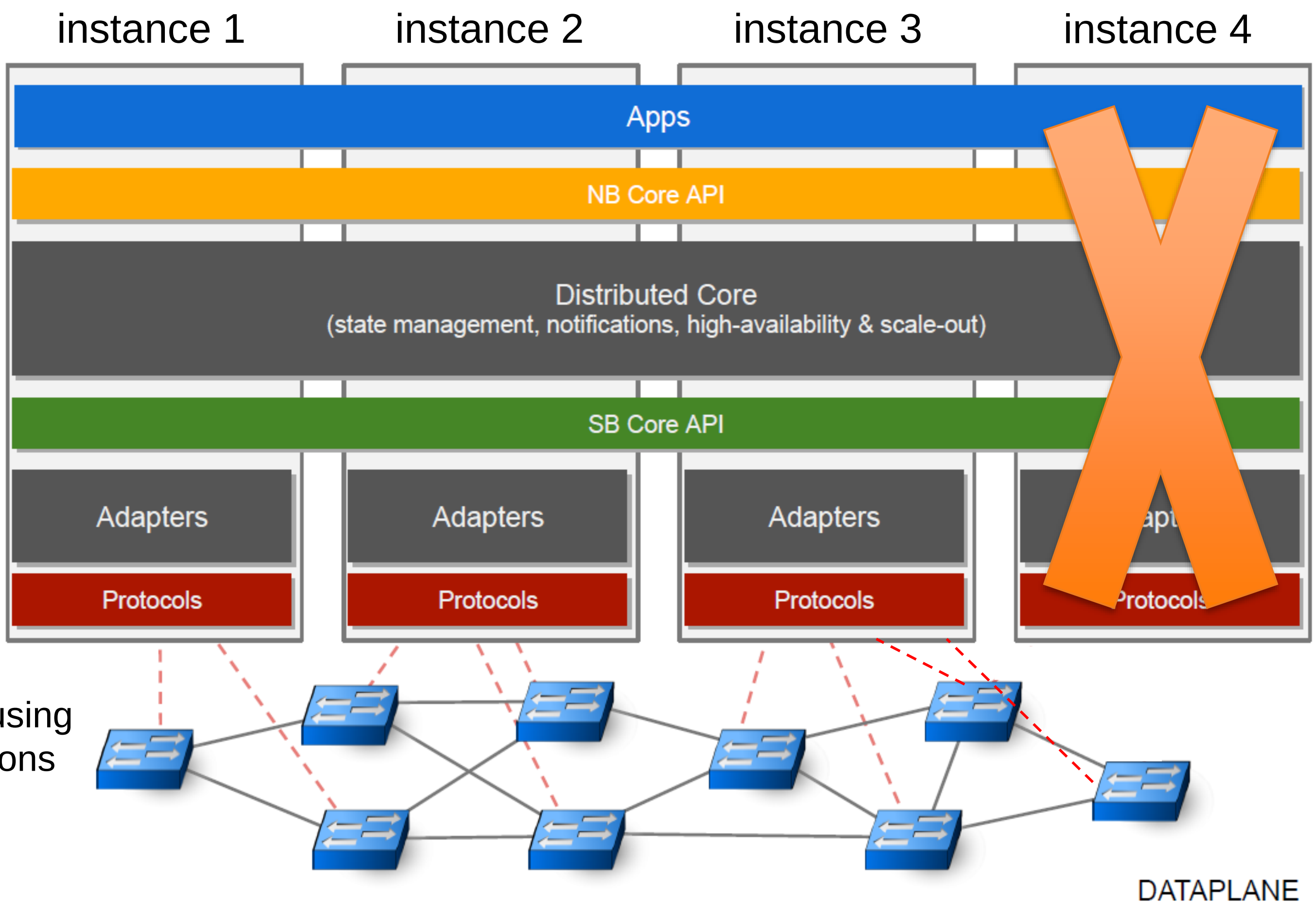
Differences

- ONOS vs. ODL
  - Carrier-grade networks vs. Cloud provider
  - Pure SDN vs. Legacy
  - Academic initiated vs. Corporate initiated

ONOS Tiers and Distributed Architecture

- Distributed Architecture
  - Six tiered architecture
  - Each ONOS instance is equipped with the same software stack

- Northbound Abstraction
  - Network graph
  - Application intents
- Core
  - Distributed
  - Protocol independent
- Southbound Abstraction
  - Generalized OpenFlow
  - Pluggable & extensive
- Adapters
  - Multiple southbound protocol enabling layer
- Protocols
  - Self-defined protocols using generalized SDN functions



Consistency Definition

- Strong Consistency: Upon an update to the network state by an instance, all subsequent reads by any instance returns the last updated value.

- Strong consistency adds complexity and latency to distributed data management.

- Eventual consistency is slight relaxation – allowing readers to be behind for a short period of time.

ONOS Distributed Core

- Distributed Core
  - Responsible for all state management concerns
  - Organized as a collection of “STORES”
    - E.g., topology, links, link resources and etc.
  - State management choices (ACID vs. BASE)
    - ACID (Atomicity, Consistency, Isolation, Durability)
    - BASE (Basically Available, Soft state, Eventually consistency)
- State and Properties

State	Properties
Network Topology	Eventually consistent, low latency access
Flow Rules, Flow Stats	Eventually consistent, shardable, soft state
Switch – Controller Mapping Distributed Locks	Strongly consistent, slow changing
Application Intents Resource Allocations	Strongly consistent, durable

Overview of data plane

- Wide range of functions
  - Forwarding
  - Access control
  - Mapping header fields
  - Traffic monitoring
  - Buffering and marking
  - Shaping and scheduling
  - Deep packet inspection
- Data plane design goals
  - Flexible
  - Extensible
  - Clean interfaces

Motivation

- SDN protocols require data-plane changes
- Performance requirement
  - Protocols must forward packets at acceptable speeds.
- Support different protocols
  - Run in parallel with existing protocols

- Requirement of SDN data plane – a platform that
  - Forwards packets at high speed
  - Run multiple protocols in parallel

Existing Approaches

- Develop Custom Software
  - Flexible, easy to program
  - Slow forwarding speeds

- Develop Custom Hardware
  - Excellent performance
  - Long development cycles, rigid

- Develop programmable hardware
  - Flexible and fast
  - Programming is difficult

Network Assembly Language

- Openflow's design was motivated by the underlying device layout
  - Controller is limited in supporting new functions not supported by Openflow
- New Chipsets are adding data plane functions.
- New languages are specifying data plane at a high level – people like to use

- What's in between?

- http://netasm.cs.princeton.edu/

Need for network assembly

- A low-level programming language for programmable network devices
- Provides a one to one correspondence with the underlying hardware
- Uses well-defined constructs to define low-level packet operations
- Enables writing highly optimized network programs

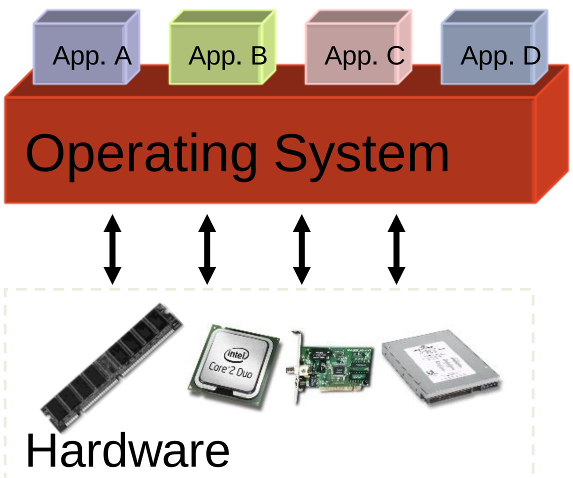
P4: Programming protocol-independent packet processors

- P4 is a high-level language for programming protocol-independent packet processors.
- P4 works in conjunction with SDN control protocols like OpenFlow.
  - OpenFlow explicitly specifies protocol headers on which it operates. This set has grown from 12 to 41 fields in a few years, increasing the complexity of the specification.
- P4 propose how OpenFlow should evolve in the future.

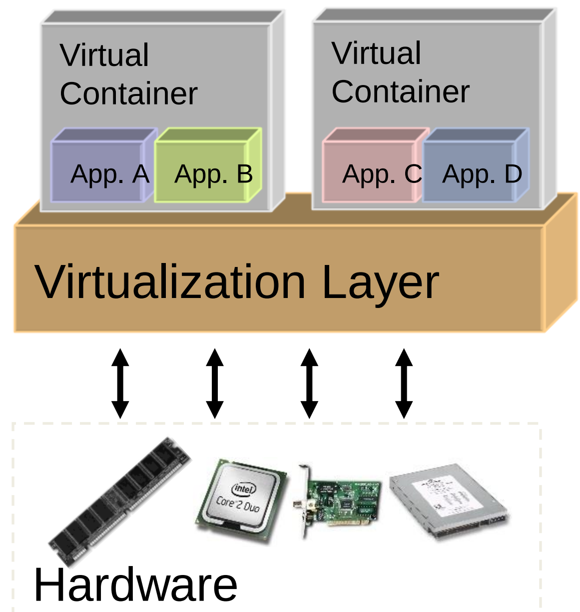
What is virtualization?

Virtualization is a broad term (virtual memory, storage, network, etc)

Virtualization basically allows one computer to do the job of multiple computers, by sharing the resources of a single hardware across multiple environments



'Nonvirtualized' system
A single OS controls all hardware platform resources



Virtualized system
It makes it possible to run multiple Virtual Containers on a single physical platform

What is network virtualization?

Network Virtualization is the process of combining hardware and software network resources and network functionality into a single, software-based administrative entity, a virtual network. - In computing

- Two categories :
  - External network virtualization
    - Combining many networks, or parts of networks, into a virtual unit.
  - Internal network virtualization
    - Providing network-like functionality to the software containers on a single system.
    - This was before SDN and NFV

NFV

- Combining forwarding devices and middleboxes into a common control framework
- NFV enables network operators to implement network policies without worrying about:
  - Placement: Where to place the functions (middleboxes) in the network.
  - Steering: How to route traffic through these functions.

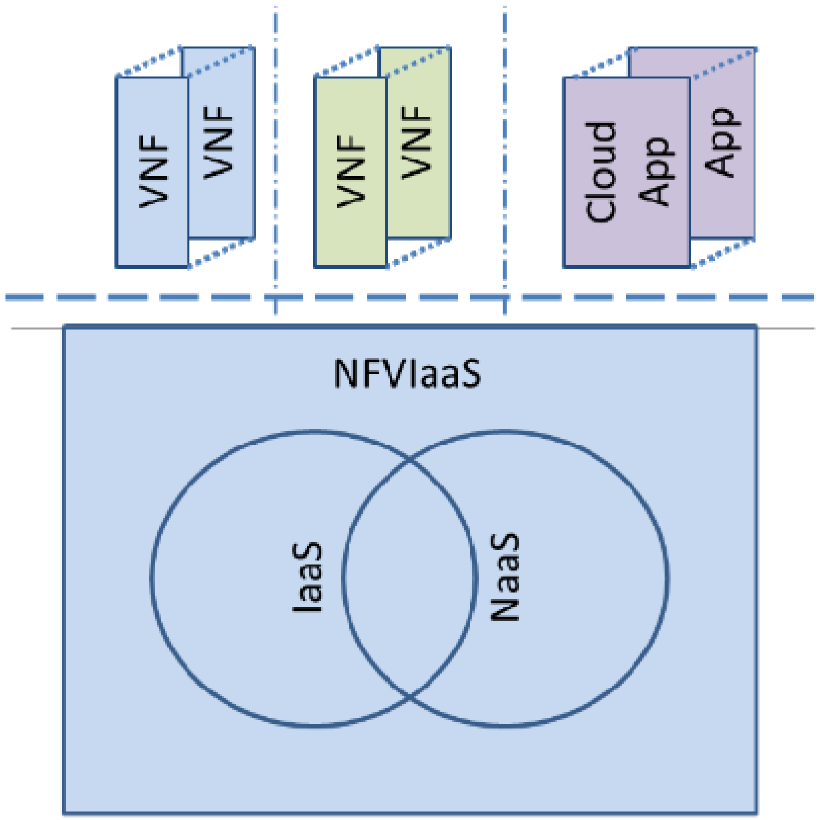
  - Placement and steering are two difficult problems for the traditional network with middleboxes.

NFV Infrastructure as a Service (NFVIaaS)

NFV Infrastructure :

- provide the capability or functionality of providing an environment in which Virtualized network functions (VNF) can execute

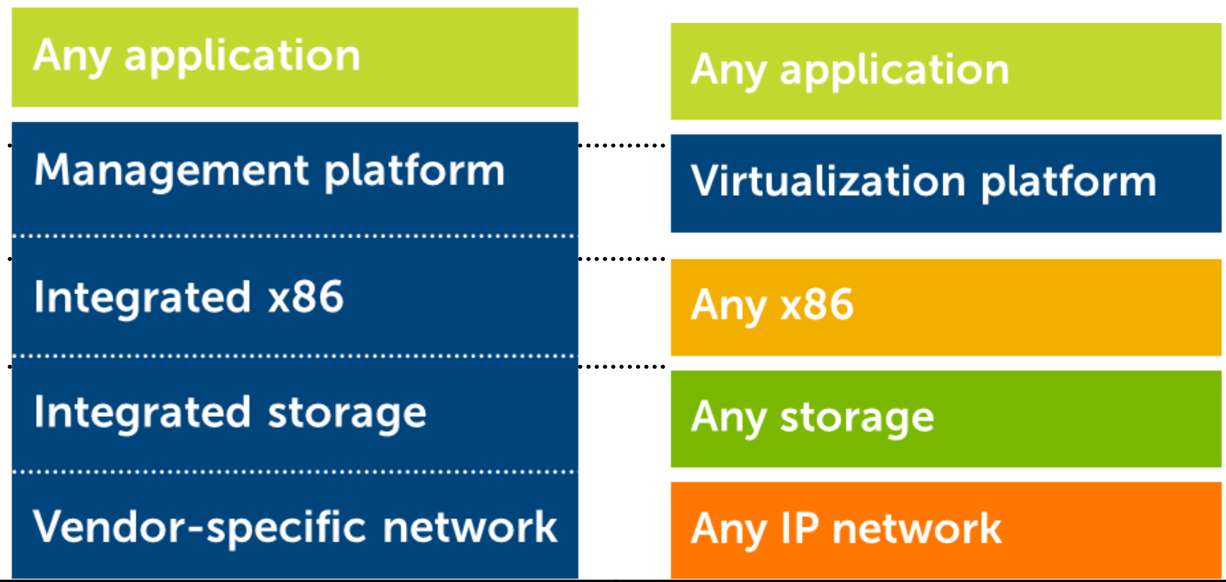
- **NFVIaaS** provides compute capabilities comparable to an **IaaS cloud computing service** as a run time execution environment **as well as support the dynamic network connectivity services** that may be considered as comparable to **NaaS**



NFV vs SDN

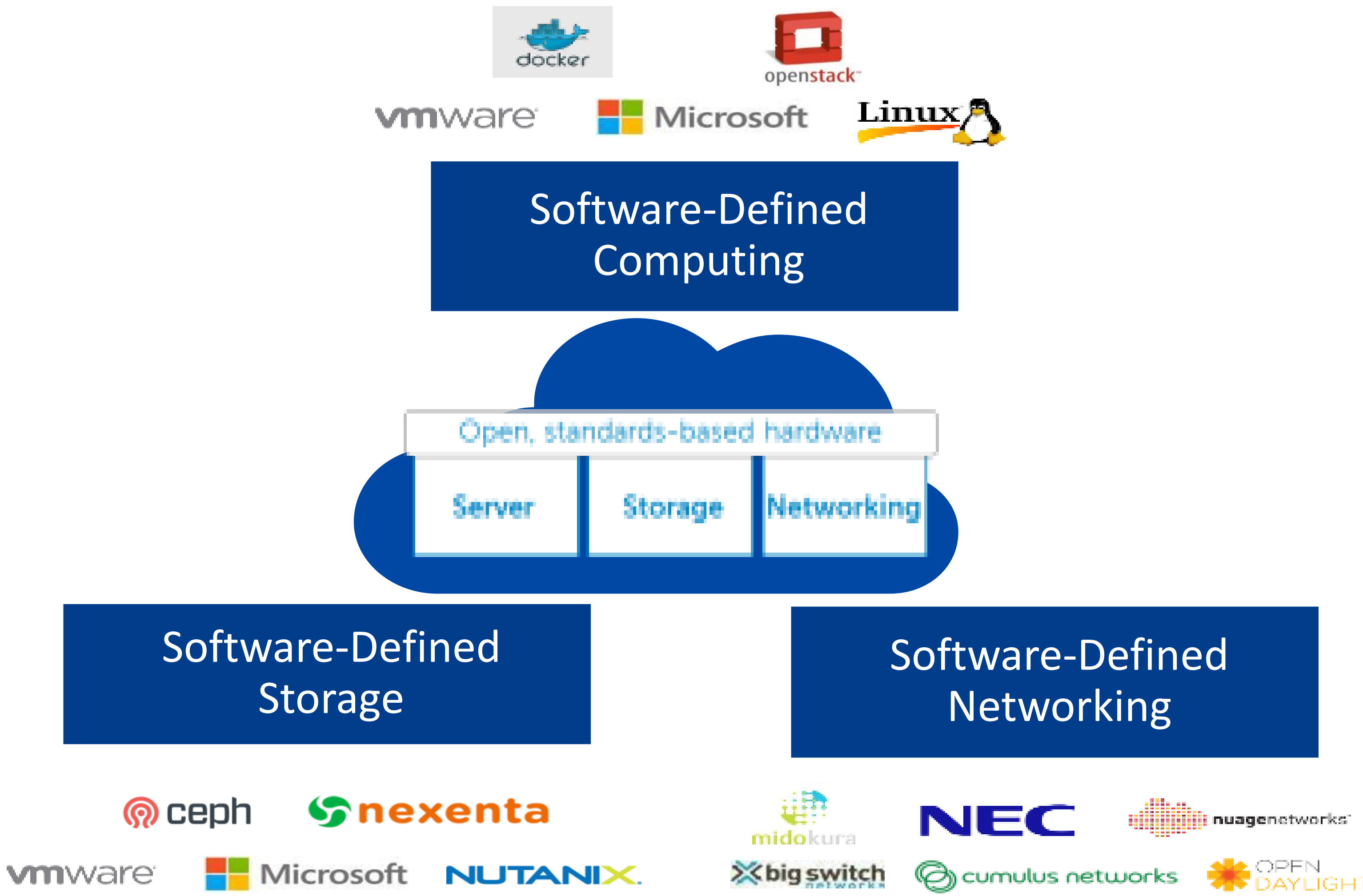
- NFV: re-definition of **network equipment architecture**
- NFV was born to meet Service Provider (SP) needs:
  - Lower CAPEX by reducing/eliminating proprietary hardware
  - Consolidate multiple network functions onto industry standard platforms
- SDN: re-definition of **network architecture**
- SDN comes from the IT world:
  - Separate the data and control layers, while centralizing the control
  - Deliver the ability to program network behavior using well-defined interfaces

SDDC delivers needed agility and efficiency



Benefit	Hardware-defined (HDDC)	Software-defined (SDDC)
Innovation	**Slow** Long hardware/ASIC cycles	**Fast** Rapid software innovation
Flexibility	**No** Lock-in	**Yes** Choice of infrastructure
Ease of insertion/ deployment	**Low** Requires forklift upgrade	**High** Non-disruptive

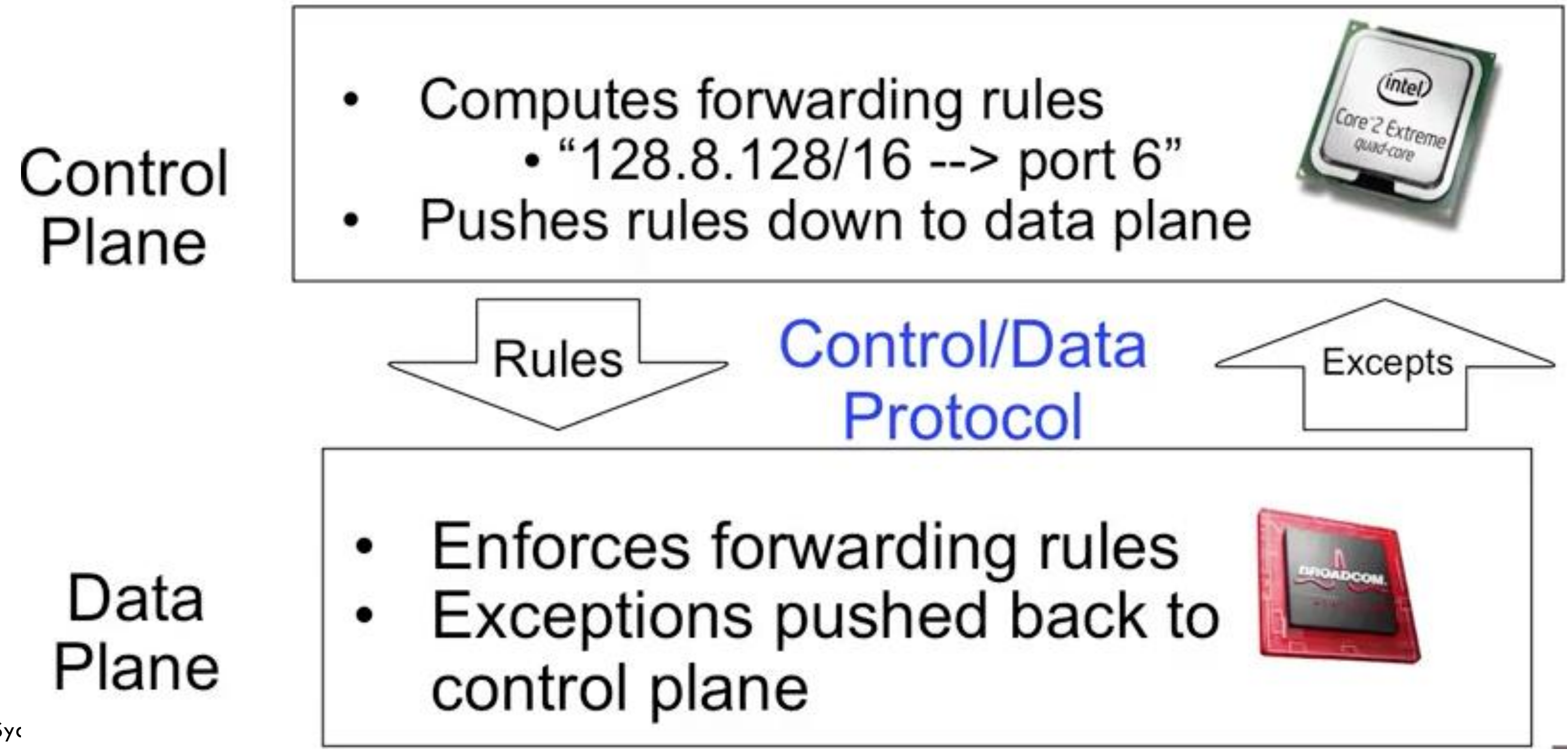
Enabling the Future Ready Enterprise



Network Slicing

- One of the enabling technologies for SDN in Data Center

- Traditional Network Device Control



How to facilitate network slicing

- Data plane unmodified
  - No performance penalty

- Control Policy: Specify resource limits fro each slice
  - Link bandwidth
  - Maximum number of forwarding rules
  - Topology
  - Fraction of switch/router CPU

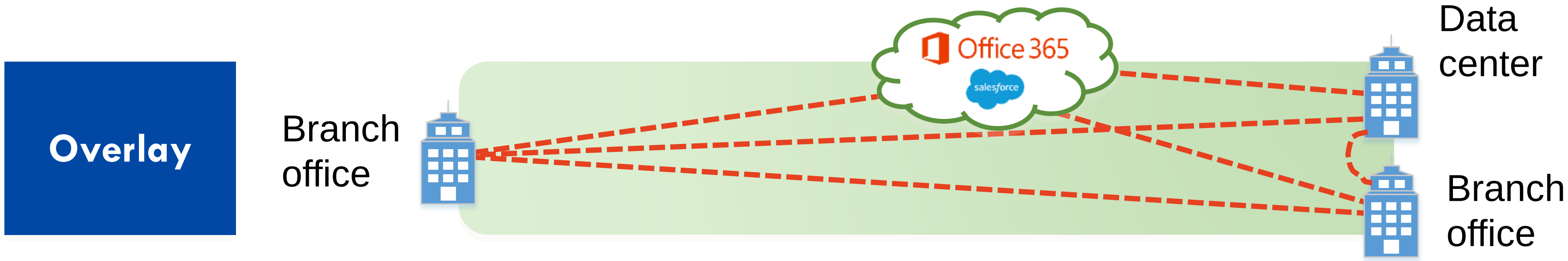
SDN for Interdomain Routing - SDX

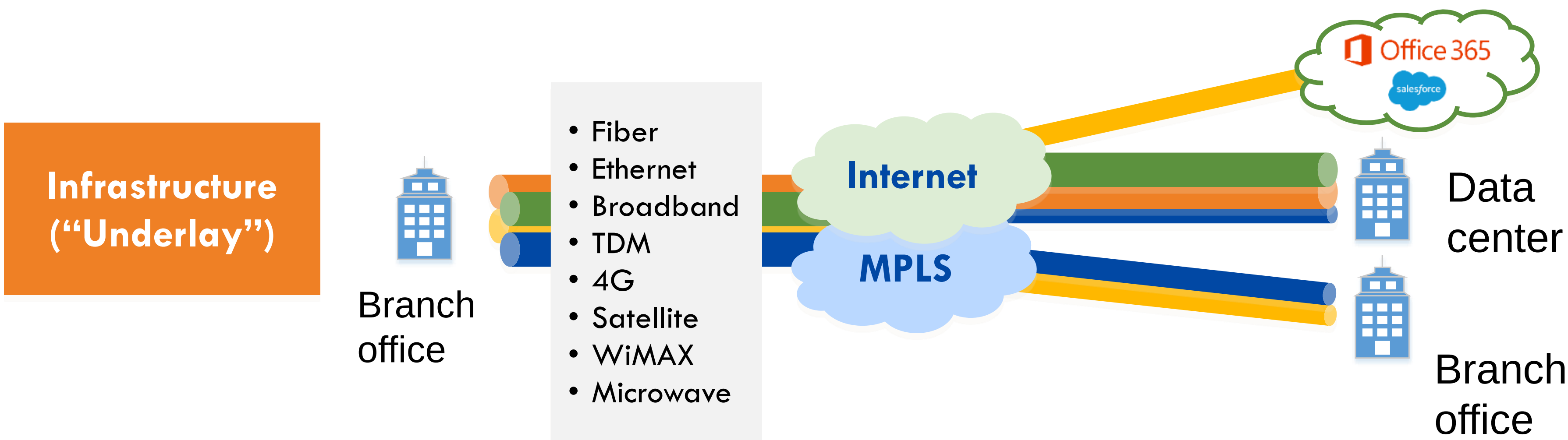
- Forwarding on **multiple header fields** (not just destination IP prefixes)

- Ability to **control entire networks** with a single software program (not just immediate neighbors)

- **Direct control** over data-plane forwarding (not indirect control via control-plane arcana)

SD-WAN





Source: coevolve.com

B4: Google's Software Defined WAN

- Google's private WAN connecting its data centers
  - Elastic bandwidth demands
    - Can tolerate periodic failures with temporary BW reduction
  - Small number of sites
    - Allows special optimization
  - Complete control of end application
    - Application priorities and control bursts
  - Cost Sensitivity
    - Unsustainable cost projection with traditional approach (2-3x cost of a fully utilized WAN).

Good Luck!

Now this is not the end.

It is not even the beginning of the end.

But it is, **perhaps**, the end of the beginning.

-Winston Churchill

USS feedback

- I need at least 67% response!
- http://sydney.edu.au/itl/surveys/complete/
- Your participation is appreciated!
- Lottery – iPad, gift card, etc.