NEW YORK INSTITUTE OF TECHNOLOGY

# INCS 775
# Data Center Security

## *Software Defined Networking (SDN)*

Dr. Zakaria Alomari
zalomari@nyit.edu

# Objectives

- ❑ Introduction
  - ❑ A brief introduction
  - ❑ SDN promises and challenges

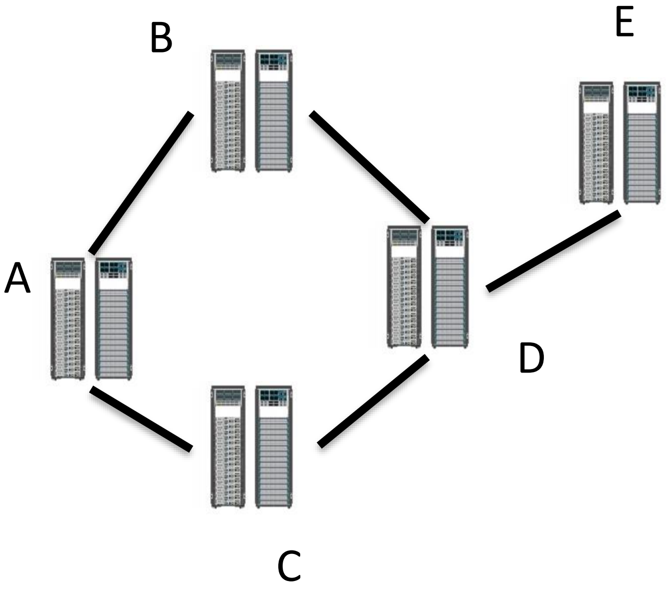
# What and Why "software defined?"

- "Software defined" becomes very popular words
  - *Software-Defined Networking*, *Software-Defined Storage*, *Software-Defined Radio*, *Software-Defined Data Center*, *Software-Defined Infrastructure* etc.
- What is it?
  - Underlying system feature is exposed to the upper layer application developer through an API.
  - System functionality is implemented over the API as an App.
- Another word for "Software defined" is "Programmable."

# Software defined networking

- **Basic Network Functions:** allowing nodes in the network to communicate with one another.
- **Network Elements** connected by links to form a topology.
- Each node runs some kind of distributed algorithm, e.g. **OSPF**, to figure out the path from A to B.
- **Network Administrator** can change network parameters to achieve certain objective: e.g. changing routes
  - ○ Limited programmability

– **Equipment vendors** provide a set of routing (network control) choices: OSPF, ISIS, BGP, etc:
  - ○ If one wants something beyond this set, **he is out of luck**.
  - ○ **This is what SDN tries to overcome: making the network control like an APP that user can develop by themselves.**
  - ○ ***SDN is to make network control programmable.***



# SDN motivation

- Why do we want to make **network control programmable**?
  - Short term:
    - Existing network control is no longer sufficient in several important areas, need **innovation** here!
      - Data centers, Wireless, network security
    - Existing network control is getting too **complicated**.
      - A **lot of different middleboxes**, each **speaks its own language**, and **interferences with one another**
        - » NAT, firewall, IDS, WAN optimizer, load balancer, traffic shapers, transparent web proxy, application accelerators etc.
      - *Would be nice to provide a **unified mechanism** to deploy and manage these middleboxes*
      - SDN promises this.
  - Long term: innovation is good for the networking industry.

# Vertical Integration vs Horizontal Integration

- **Vertical integration** is an expansion strategy where a *company takes control over one or more stages in the production or distribution of its products*.

- **Horizontal integration** is an expansion strategy that *involves the acquisition of another company in the same business line.*

# Computing systems once upon a time (Historical Context of Computing Systems)

- Vertically integrated systems
  - In the past, computing systems was often vertically integrated. This means that a single company or vendor was responsible for the entire stack of the computing system, from hardware to operating system (OS) and applications.
  - **Example**: IBM's mainframes are a classic example of vertically integrated systems where IBM provided the hardware, the OS (such as z/OS), and key applications.

# Computing systems once upon a time (Historical Context of Computing Systems)

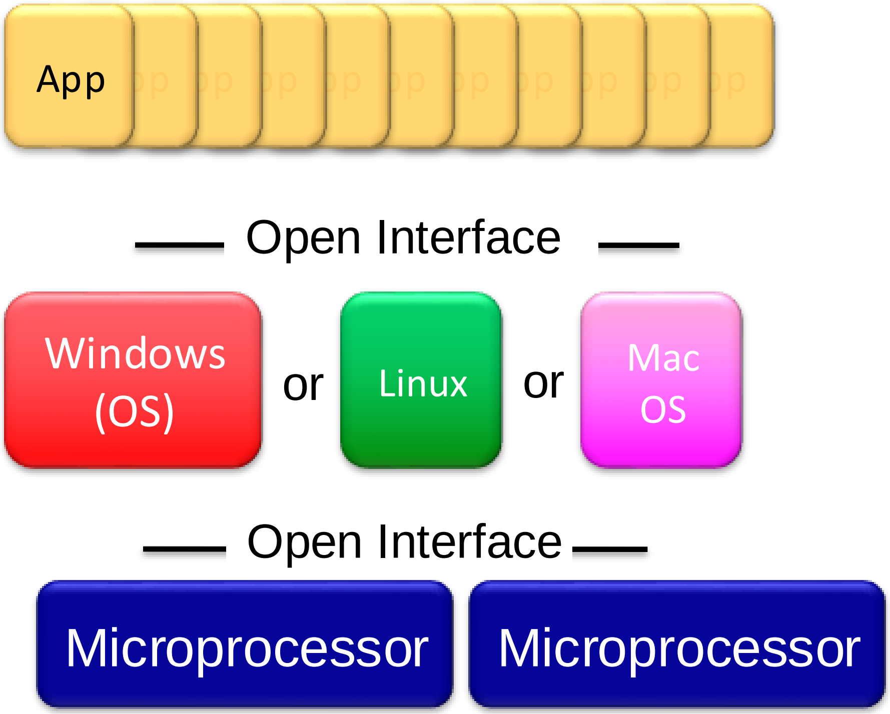
- Vertically integrated systems
  - **Proprietary** hardware
    - **Example**: Early Apple computers, such as the Macintosh, used hardware components specifically designed by Apple.
  - **Proprietary** OS
    - **Example:** The early versions of Apple's Mac OS or IBM's z/OS for mainframes.
  - **Proprietary** applications
    - **Example:** Early business software provided by IBM for its mainframe computers or the software suites provided by Digital Equipment Corporation (DEC) for its VAX systems.
  - **Highly reliable**
    - **Example:** IBM mainframes are known for their exceptional reliability and uptime, which is why they have been extensively used in critical industries like banking and finance.

# Computing systems once upon a time (Historical Context of Computing Systems)

- Can you picture **google**, **yahoo**, **facebook** on such a platform?
  - Slow software innovation
    - Proprietary development
  - Small industry

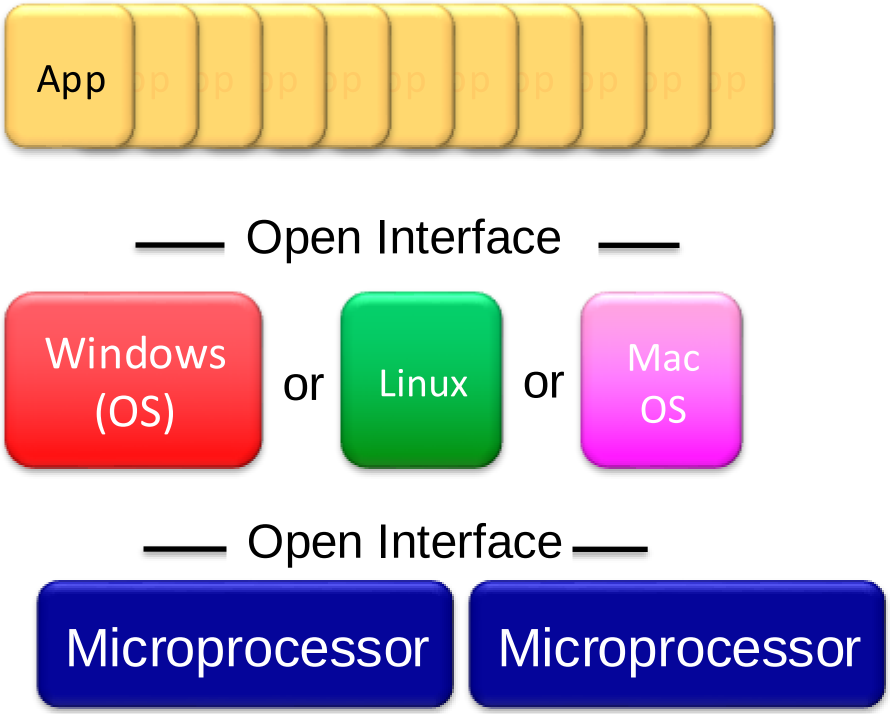
# Computing systems now (current state of computing with how it used to be)

- Open Interface
  - Modern computing systems often use open interfaces, which are standardized and publicly available specifications that allow different systems, devices, and software to communicate and work together.
  - **Example**: APIs (Application Programming Interfaces) and open-source protocols like HTTP and TCP/IP facilitate interoperability between various software and hardware from different vendors.

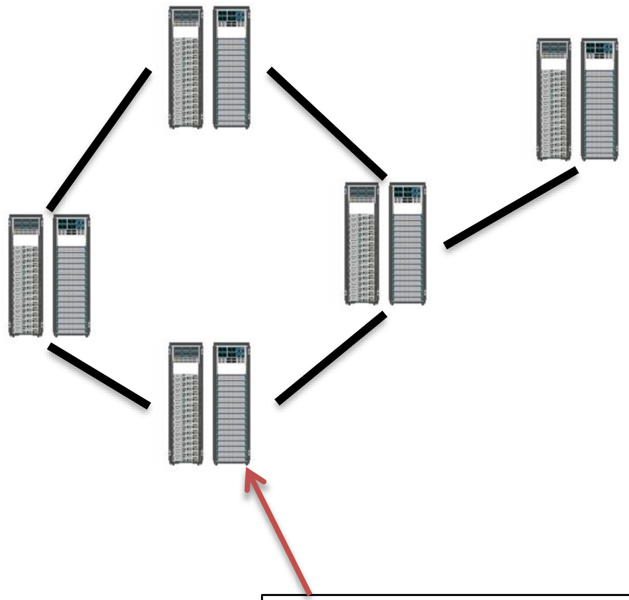


# Computing systems now (current state of computing with how it used to be)

- Fast innovation
  - Everyone can
  - participate
- Huge industry
  - Software is now part of everything.



# Conventional networking system today (before SDN)



Custom hardware
OS
Bundled applications

- Mainframe mindset: software for the control plane cannot be separated from the forwarding hardware in the data plane.
  - ***Vertically integrated, complex, closed, proprietary***
  - Innovation is only possible if one has *access to the router box.*
    - No significant innovation in the past 40 years.

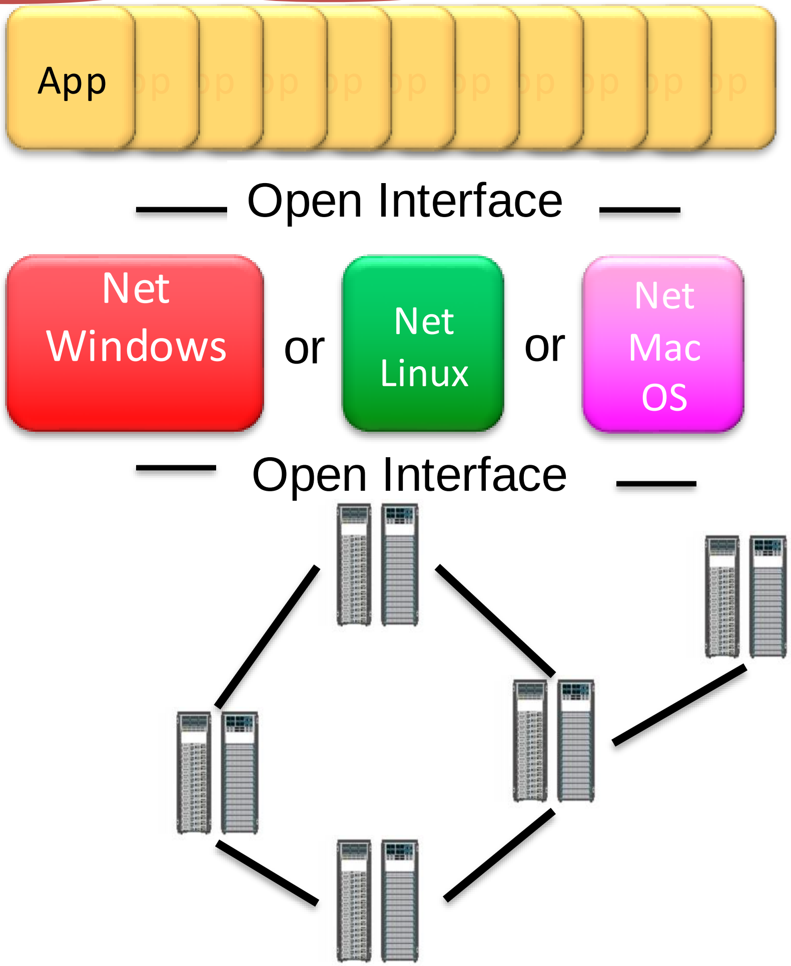
# Conventional networking system today (before SDN)

- Conventional networking systems, as they were implemented before the rise of Software-Defined Networking (SDN), are characterized by:
  1. **Hardware-Centric Approach**: Heavy reliance on specialized hardware devices.
  2. **Static Configuration:** Manual, static configuration of network devices.
  3. **Device-Specific Control:** Independent control and data planes for each device.
  4. **Complex Management:** Complicated and labor-intensive management processes.
  5. **Limited Automation:** Minimal automation, with most tasks performed manually.
  6. **Vendor Lock-In:** Dependence on a single vendor's ecosystem for compatibility and support.
  7. **Slow Adaptation to Changes:** Difficulty in quickly adapting to changing network demands.

# Ideal networking system for innovation

- An ideal networking system for innovation encompasses:
  1. **Open Standards and Interfaces:** Ensures compatibility and integration with various technologies.
  2. **Programmability**: Enables dynamic configuration and automation through software.
  3. **Virtualization**: Allows for the creation of flexible and isolated network environments.
  4. **Scalability**: Easily accommodates growth and changes in demand.
  5. **Security**: Incorporates strong security measures to protect the network and data.
  6. **Automation and Orchestration:** Reduces manual tasks and increases operational efficiency.
  7. **Support for Innovation:** Facilitates rapid development, testing, and deployment of new services.
  8. **Flexibility and Agility:** Quickly adapts to new technologies and changing requirements.

# Ideal networking system for innovation



Network apps

API of Net OS

Network Operating Systems

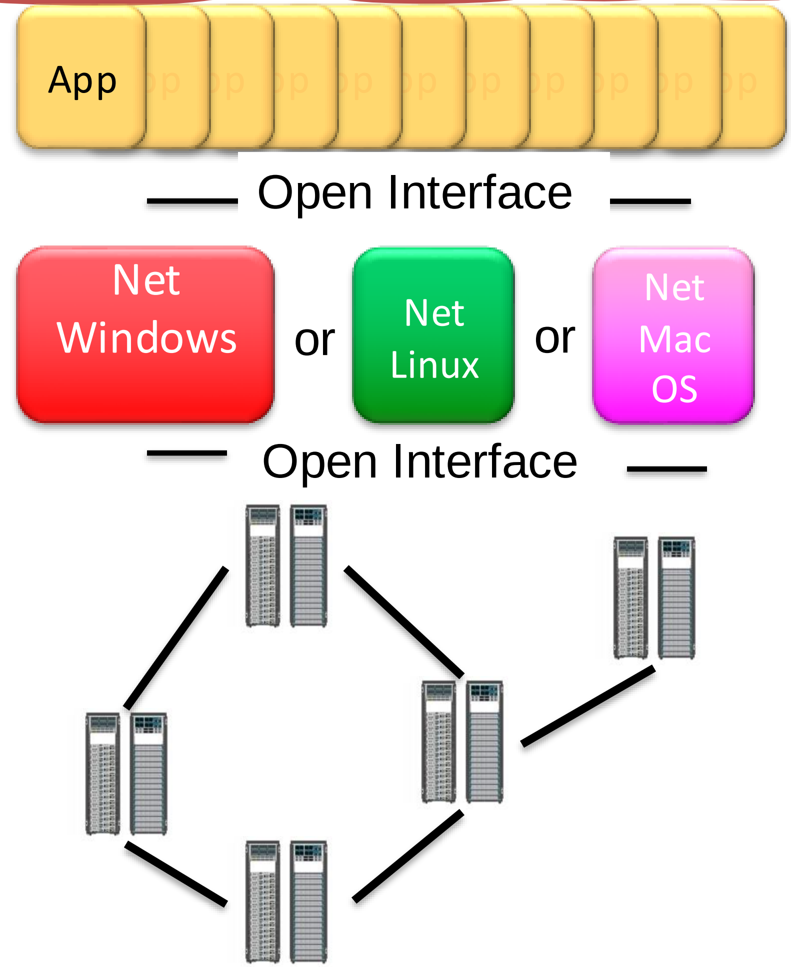
API for controlling
Network hardware

Network hardware

# Control Plane and Data Plane

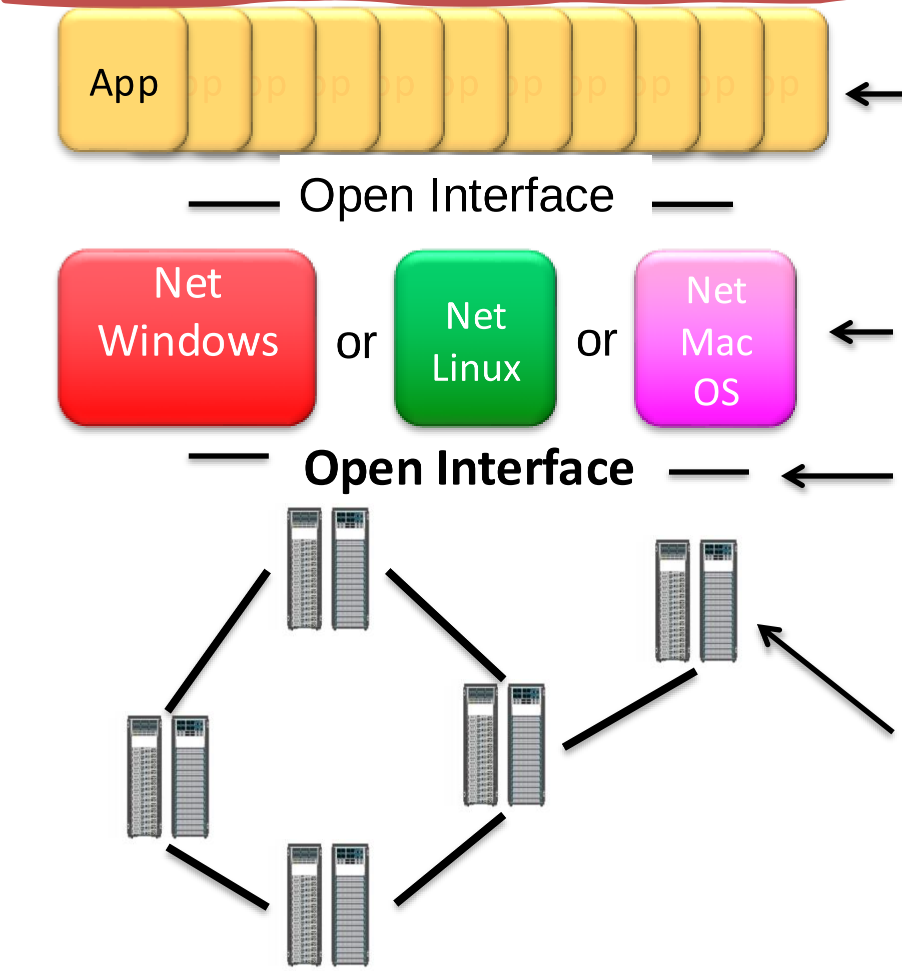
- **Control plane of a network**
  - The functions of a network that control the behavior of the network
    - E.g.: Which path to take for a packet? Which port to forward a packet? Should the packet be dropped?
  - Control plane functions are typically realized by software such as routing protocols, firewall code, etc.
- **Data plane of a network**
  - The functions of a network that actually forward or drop packets.
  - Data plane functions are typically realized by hardware
- Control plane and data plane are *vertically integrated in traditional networking equipment*
  - Separating software from hardware → separating control plane from data plane.

# Ideal networking system for innovation



- Separate *software* from *hardware*
- Standardize the interface
  - *Each layer provides an abstraction*
- Innovation is possible for anyone just like software development for a computing system.
- This is the vision of SDN/OpenFlow.

# SDN now: separate forwarding hardware from controlling software



4. Firewall, virtual network, TE, etc

3. SDN controllers (floodlight, nox, etc)

1. OpenFlow: standardized for Ethernet/IP/TCP

2. OpenFlow enabled switches/routers
   simple hardware doing forwarding only
   forwarding table can be set by other entity through OpenFlow

# Contrast between SDN and Conventional network

| SDN | Conventional |
|---|---|
| Controller may not be in the same box as the forwarding hardware | Forwarding hardware and its control are in the same box |
| Centralized routing algorithm with logically global view | Distributed routing algorithm |
| Network functions are realized with a global view | Network functions must be realized in a distributed manner, error-prone |
| New abstraction must be developed for the centralized view | Network abstraction is embedded in the distributed algorithms |

# Major paradigm shift with SDN

- No longer use distributed control protocols
  - *Design one distributed system (NOS) with the global view of the network*
  - *Use for all control functions*
- Now just defining a centralized control ***function***

**Configuration = Function(global view)**

- **This may look easier, but this is not how it used to work, everything is new – innovation at all levels for this to happen.**
  - High level programming languages to describe network configuration
  - Compiling and runtime system to realize the program efficiently, correctly, and safely.
  - Abstraction design
  - Debugging infrastructure
  - network OS design
  - etc.

# How an SDN operates?

- Network applications specification the network functions (not the detailed implementation on the physical devices):
  - Access control: who can talk to who
  - Isolation: who can hear my broadcasts
  - Routing: only specify routing to the degree you care
    - Some flows over satellite, others over landline
  - TE: specify in terms of quality of service, not routes
- Network OS (or something like a compiler) compiles the network application and computes the configurations on physical devices based on the global view
- Network OS distributes the configuration to physical devices through OpenFlow.

# SDN promises

- A lower-entry point for innovation in the network control.
- Solve the issues in the current network configuration challenges.
  - Data plane interacts with many control entities
  - Configure locally to achieve a global network function.

# Some SDN issues

- Abstraction
  - A new programming system to specify network functions (programming SDN)
  - An API that provides network abstraction to network application (SDN controller design)
- Performance (scalability)
  - Controller
  - Communication between controller and devices
  - Forwarding
- Correctness and debugging – A SDN program has a higher bar than a typical program, multiple levels
- Security

# Objectives

- ❑ Introduction
  - ❑ A brief introduction
  - ❑ SDN promises and challenges
- ❑ Networks without SDN
- ❑ Challenges facing network managers

# Routing and Switch are complicated

- Multiple routing and switching protocols
  - Each router has its own control plane
  - Protocols are designed to make routers understand network topology
  - No one protocol that can handle all the network demands
    - BGP, OSPF, EIGRP, STP, PIM
  - IP header examine and routing table look up often slow things down
  - Various technology invented to speed up
    - MPLS, FIB, QoS, NP on each module, VSS
      - MPLS: is a networking technology that routes traffic using the shortest path based on "labels", rather than network addresses, to handle forwarding over private wide are network,

# Routing and Switch are complicated

- VPN, GRE tunnels, DMVPN
  - GRE requires statically defined endpoints, which makes dynamic IP's impossible.
  - DMVPN allows up to configure a single tunnel at the hub site, Each spoke site can then simply join the DMVPN network, with minimal effort, and optionally with dynamic IP address
- Other network devices like firewalls, IPS, wireless
- Branches, remote campuses, DR (Disaster Recovery) sites, Campus dorms
- Demand for research fast lane
- Tens of thousands lines of router and firewall configures

# Network operations are expensive

- The demand for wire speed routing and high capacity resulting very expensive network operations
  - The two ISP grade routers FSU use cost $1.4 million
    - Florida State University (FSU) has invested $1.4 million in purchasing two high-end routers from an Internet Service Provider (ISP).
  - Firewalls also cost half million
  - IPS devices add another half million
  - Yearly maintenance fee over quarter million

# Difficult to Operate

- Difficult to troubleshoot
  - There are so many layers, protocols, and extremely detailed configurations and settings
- Difficult to make modification
  - Once a protocol is in place, it’s very hard to change it
- Difficult to adapt dynamic demands
  - Network design principle is to make network predictable
  - Demands are dynamic and sometime unpredictable

# Current Solutions

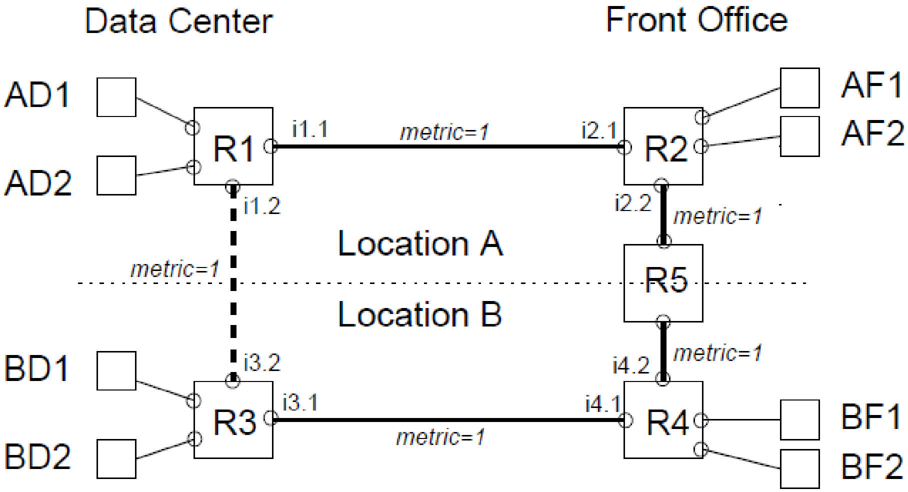
- Build a network that with high speed 10g, 40g, and hope network will not outgrow it in five years
- Define network flows to limit cross network traffics
- Rely on vendors to provide solutions
    - Wireless system performance
    - Data Center traffic congestions

# Why Adopting SDN is Slow

- Most network admins cannot program
- Most programmers lack understanding of networks
- Vendors are concerned
  - Routers/switches become commodity goods
- SDN is not mature
- Security concerns
- Performance concerns
- Reliability concerns
- Programmer/netadmin turnovers
- Someone to blame

# Example
# Local configuration global effect

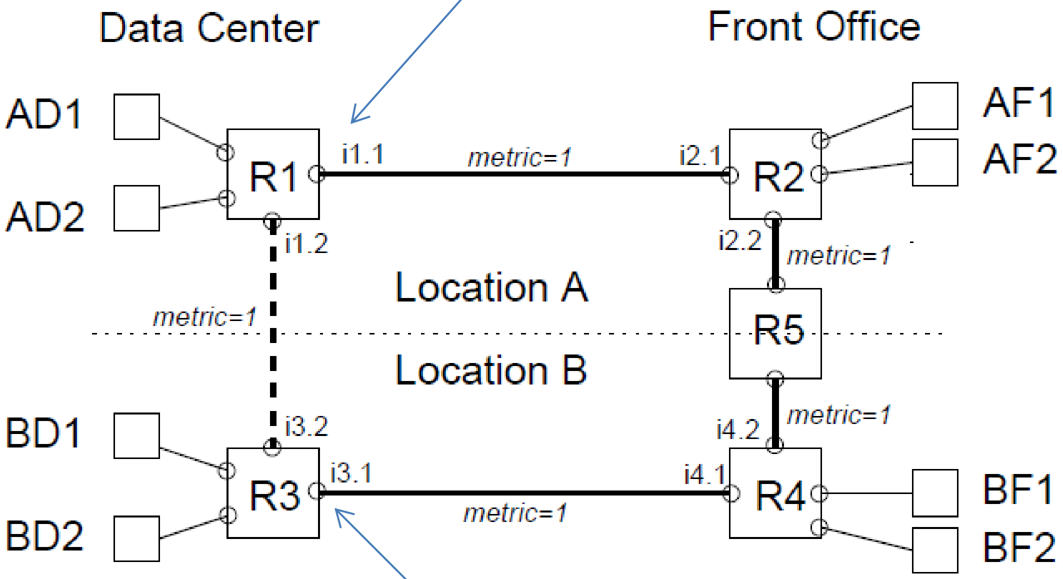
- A company has two locations: A and B
- Both A and B have front offices and data centers
- Security policy: front office computers can talk to front office computers and its own data centers, but not other office's data center; data center computers can talk to each other.



# Example
# Local configuration global effect

- Security policy implementation

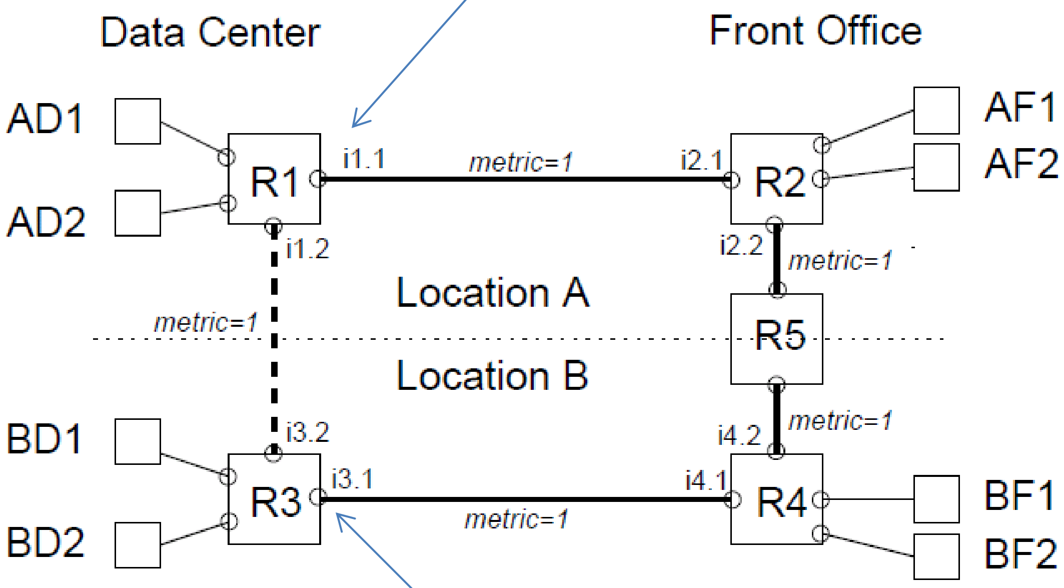
Drop all packets
From BF subnet



Drop all packets
From AF subnet

# Example
# Local configuration global effect

- The two datacenter wants to be the backup of each other, add a link. What happen next?



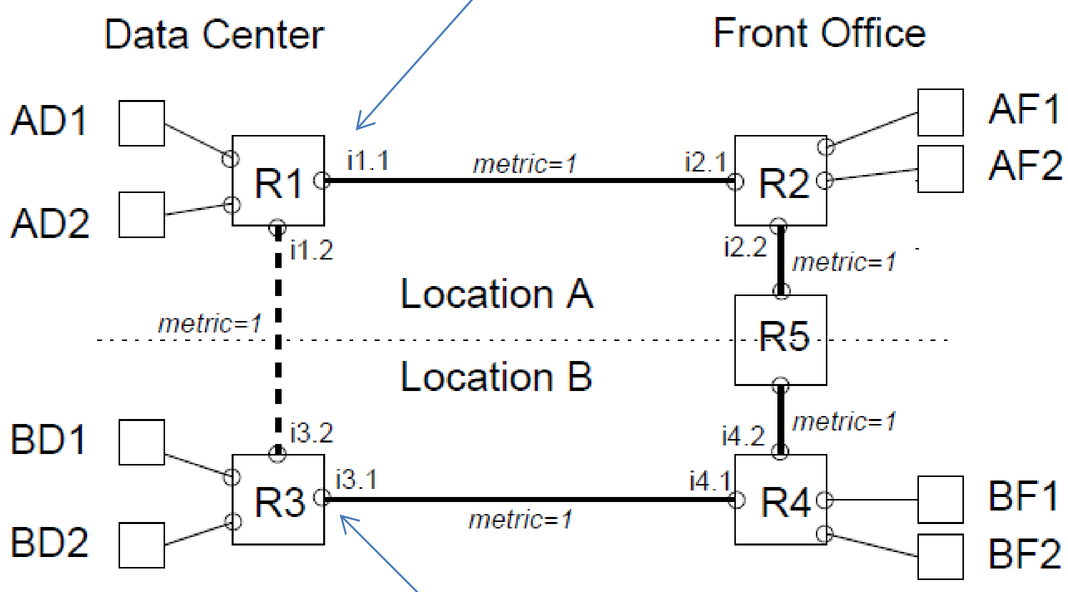
Drop all packets From BF subnet

Drop all packets From AF subnet

# Example
# Local configuration global effect

- The security policy is broken: it requires routing and packet filtering to work in concert; but the routing automatically adapts to topology change
  - Much nicer to use global network view to do the function

Drop all packets
From BF subnet



Drop all packets
From AF subnet

# Objectives

- ❑ Introduction
  - ❑ A brief introduction
  - ❑ SDN promises and challenges
- ❑ Networks without SDN
- ❑ Challenges facing network managers
- ❑ Traditional Network Architecture
- ❑ SDN Architecture
- ❑ SDN Controllers

# Traditional IP networks

- Traditional IP networks are **Complex** and **hard to manage**
- Network operator needs to **configure each individual network device separately using low-level and often vendor-specific commands**
- The **control plane** and the **data plane** are bundled inside the **networking devices**

# Network Planes

- **Data Plane**
  - responsible for ***forwarding packets***
- **Control Plane**
  - responsible for ***defining the behavior of the network***
- **Service Plane**
  - Responsible for ***providing services beyond packet forwarding, such as firewall services or quality of service (QoS) policies.***
- **Management Plane**
  - Responsible for ***configuring routing protocol, monitors network devices and handle anomalies***

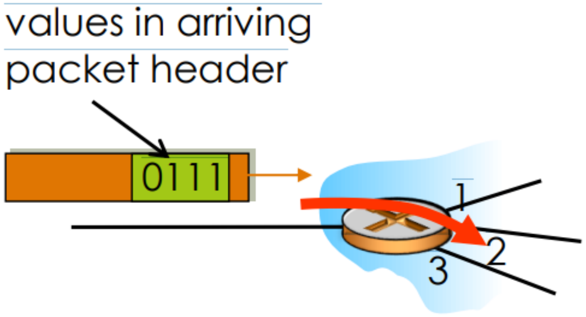
# Network Planes

| | |
|---|---|
| **Control Plane** | Its is responsible for making decisions about how data should be forwarded through a network, such as routing decisions in a router. |
| **Data Plane** | It is responsible for actually forwarding the data packets based on the decisions made by the control plane. |
| **Service Plane** | It provides specific services or functionalities, such as firewall services or quality of service (QoS) policies. |
| **Management Plane** | The is responsible for managing and configuring the devices in the network, including tasks like device provisioning, monitoring, and configuration management. |

Each plane plays a distinct role in network architecture and operations.

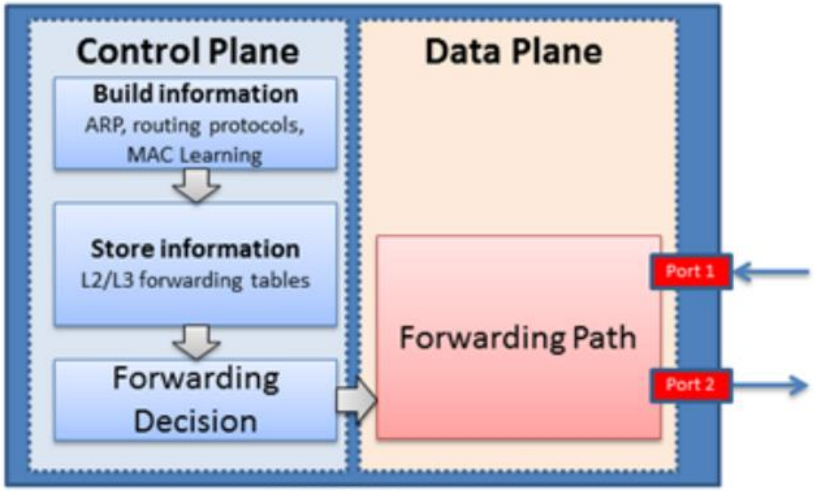
# Data plane

- **Responsible for moving packets**
  - local, per-router function
  - ***Determines how packet arriving on router input port is forwarded to router output port***
  - forwarding function
- **Other data plane functions:**
  - Packet scheduling and buffering
  - Packet fragmentation
  - Bandwidth metering
  - Access control
  - Packet cloning for multicasting
  - Traffic monitoring



# Control plane

- **Defines the network behavior**
  - Network-wide logic
  - ***Determines how packet is routed among routers along end-to-end path from source host to destination host***
  - Route computation function



- **Other control plane functions:**
  - FIB installation in routers
  - Topology discovery
  - Access control list generation
  - QoS enforcement

# Service plane

- The service plane in a network refers to the layer that offers advanced services beyond basic packet forwarding.

- These services are designed to enhance the functionality, security, and performance of the network. Example of such services include:
  - **Security Firewalls**: Devices or software that monitor and control incoming and outgoing network traffic based on predetermined security rules. They act as a barrier between a trusted internal network and untrusted external networks to block malicious traffic.
  - **WAN Acceleration:** Techniques and technologies used to improve the speed and efficiency of data transfer over wide area networks (WANs). This includes data compression, traffic optimization, and protocol acceleration to enhance performance, especially for remote offices and cloud-based applications.

# Service plane

- Services are typically realized by middleboxes
  - ✓ Load balancers, Web proxy, Web cache
  - ✓ Firewall
  - ✓ Intrusion Detection System (IDS)
  - ✓ Intrusion Protection System (IPS)
  - ✓ WAN Optimizers, etc

# Management plane

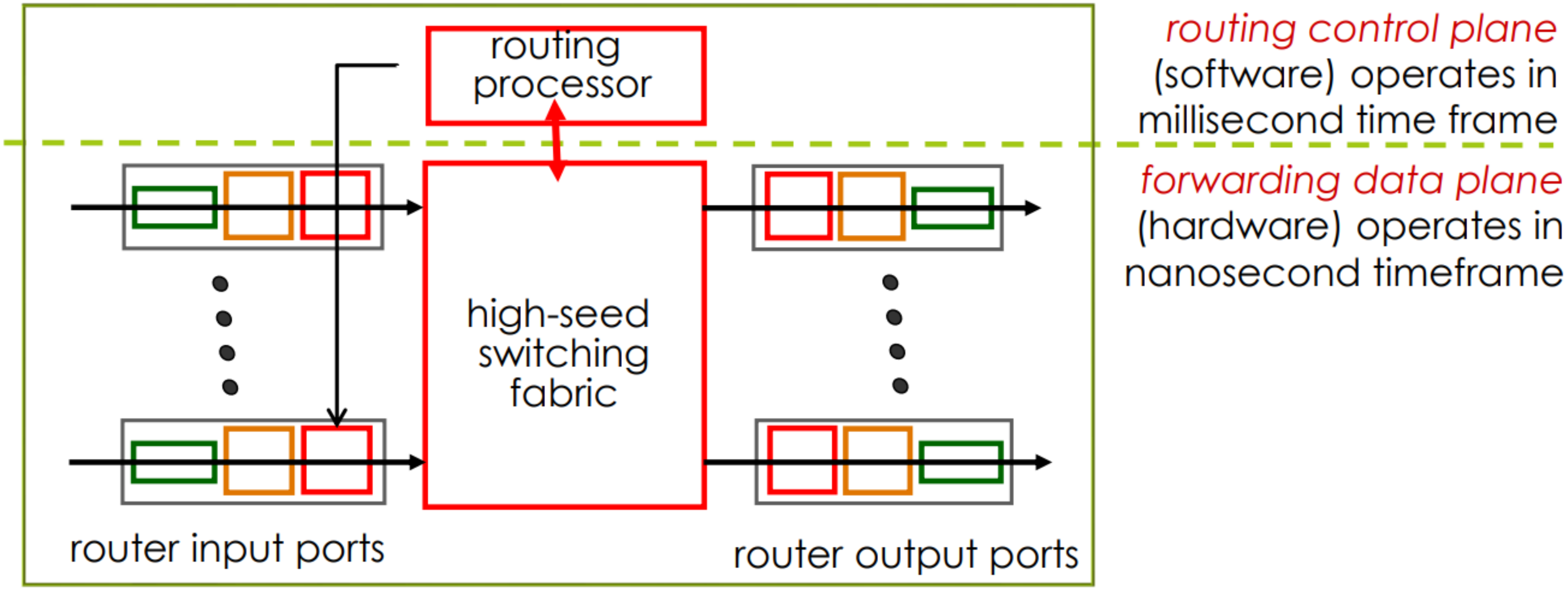
- The management plane in a network is responsible for the administrative tasks that keep the network operating smoothly and efficiently. Specifically it:
  - **Configures Routing Protocols:** Sets up and manages the protocols that determine how data is routed across the network. This involves configuring settings, updating protocol parameters, and ensuring that routing protocols operate correctly to maintain optimal data paths.
  - **Monitors Network Devices:** Continuously oversees the status and performance of network devices such as routers, switches, and firewalls. This involves collecting data on device health, traffic loads, and performance metrics to ensure the network operates within desired parameters.
  - **Reacts to Anomalies:** Detects and responds to unusual or abnormal behavior in the network**. This includes** identifying issues like device failures**,** security breaches**, or** performance degradation **and** taking corrective actions to resolve them**, such as** rerouting traffic**,** applying patches**, or** alerting network administrators**.**

# Management plane

- Management Plane **configures routing protocol**, **monitors network devices and reacts to anomalies**.
  - Responsible for managing FCAPS
    - **F**ault
    - **C**onfiguration
    - **A**ccounting
    - **P**erformance
    - **S**ecurity

# Traditional Networking

- Control plane is vertically integrated with data plane
  - Each router runs its own control plane
- Generic router architecture

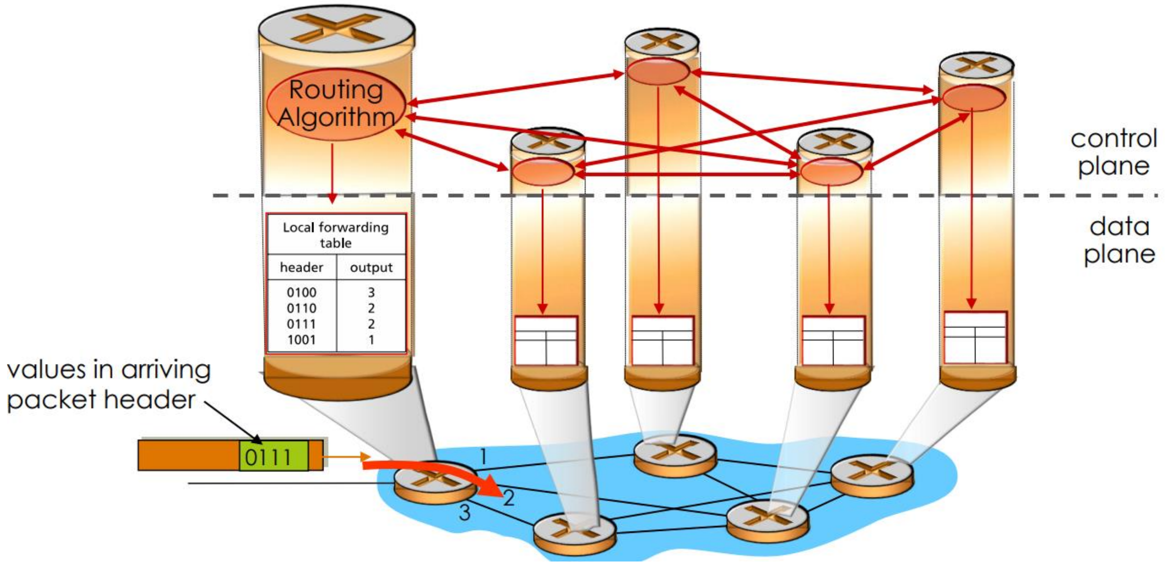


*routing control plane* (software) operates in millisecond time frame

*forwarding data plane* (hardware) operates in nanosecond timeframe

# Traditional Routing

- Traditional routing protocols are distributed
  - Routers exchange connectivity info. With each other
  - Each router builds its own routing table
  - Forwarding tables (FIB) are populated based on information available in the routing table

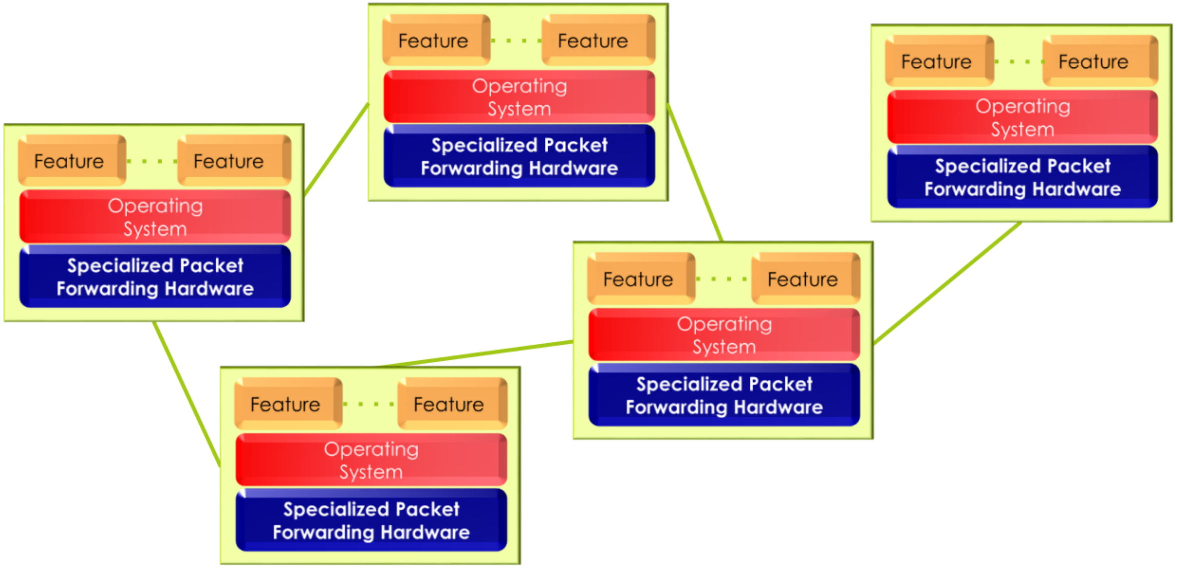


Local forwarding table

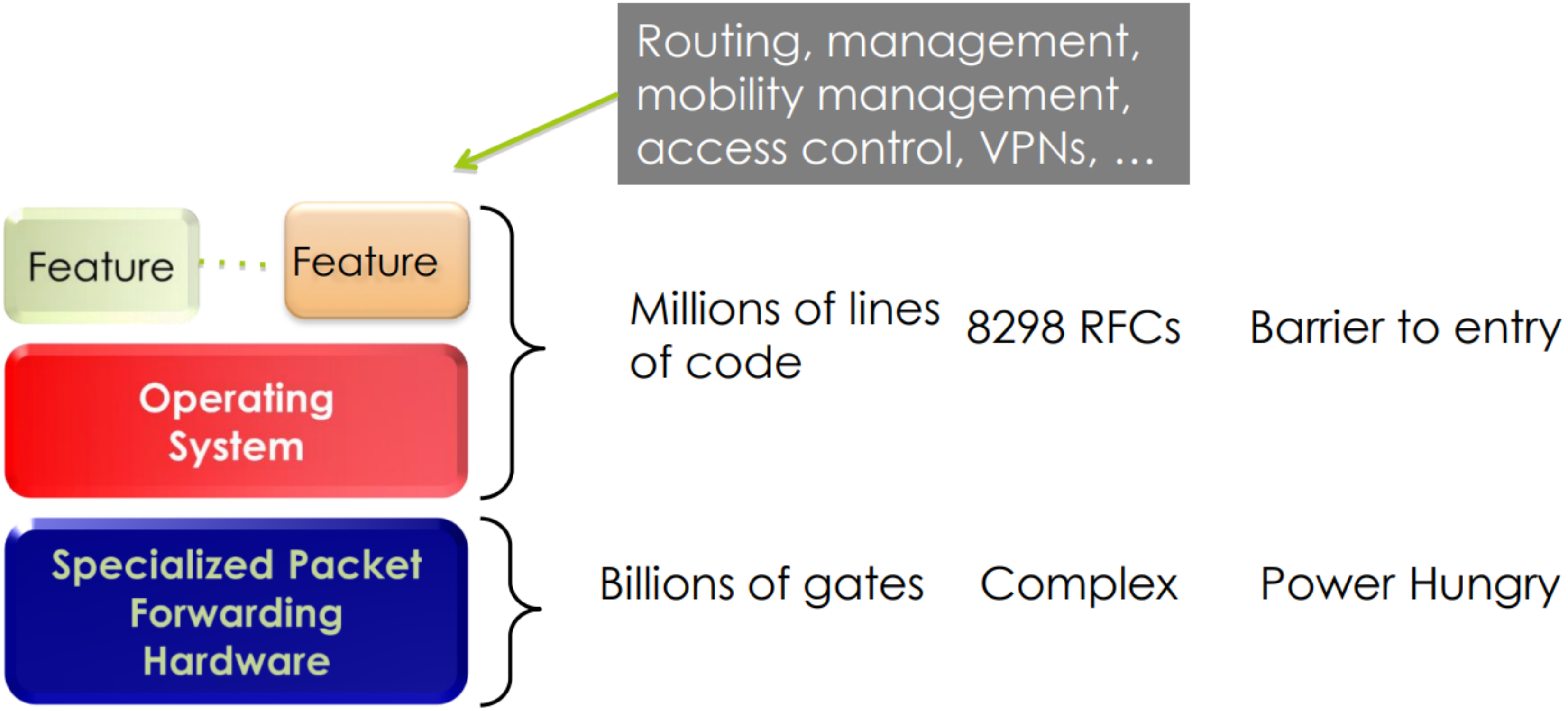
| header | output |
|---|---|
| 0100 | 3 |
| 0110 | 2 |
| 0111 | 2 |
| 1001 | 1 |

# Distributed Control Plane

- Distributed per-router control plane
  - A vertically integrated router is a networking device that includes both switching hardware and proprietary software for implementing Internet standard protocols such as IP, RIP, ISIS, OSPF, and BGP.
  - These protocols are utilized within the router's proprietary operating system, such as Cisco IOS, to facilitate communication and routing within networks.



# Traditional Network Element



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

Feature · · · · Feature

Operating System

Specialized Packet Forwarding Hardware

Millions of lines of code

8298 RFCs

Barrier to entry

Billions of gates

Complex

Power Hungry

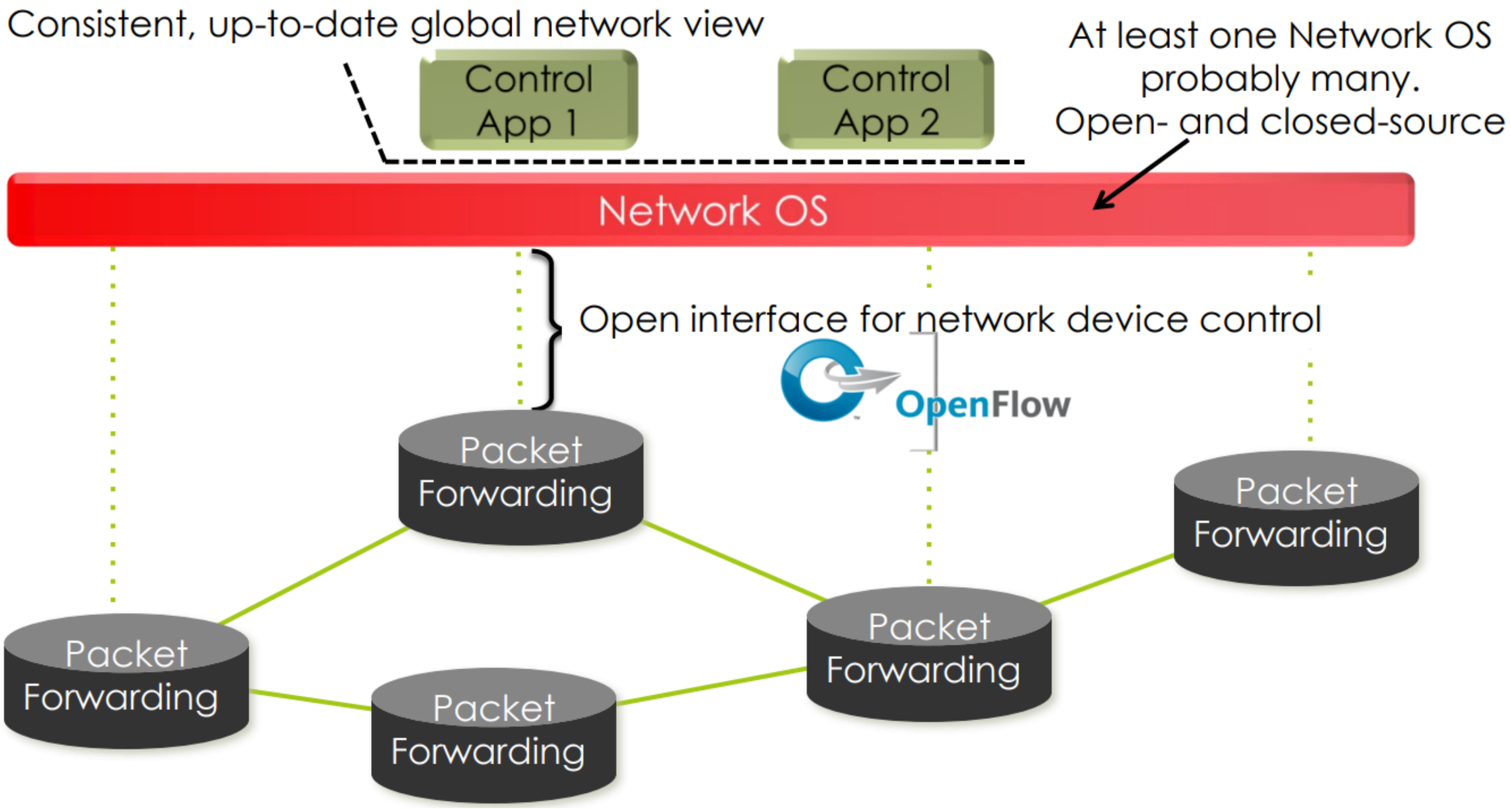
# What is Software Defined Networking

- Software-Defined Networking (SDN) ***is an emerging networking paradigm that gives hope to change the limitation of current network infrastructure.***
  - **First**, it breaks the vertical integration by separating the network's control logic (**the control plane**) from the underlying routers and switches that forward the traffic (**the data plane**).
  - **Second**, with the separation of the control and data planes, **network switches become simple forwarding devices** and the **control logic is implemented in logically centralized controller** (or network operating system), **simplifying policy enforcement and network configuration and evolution**.

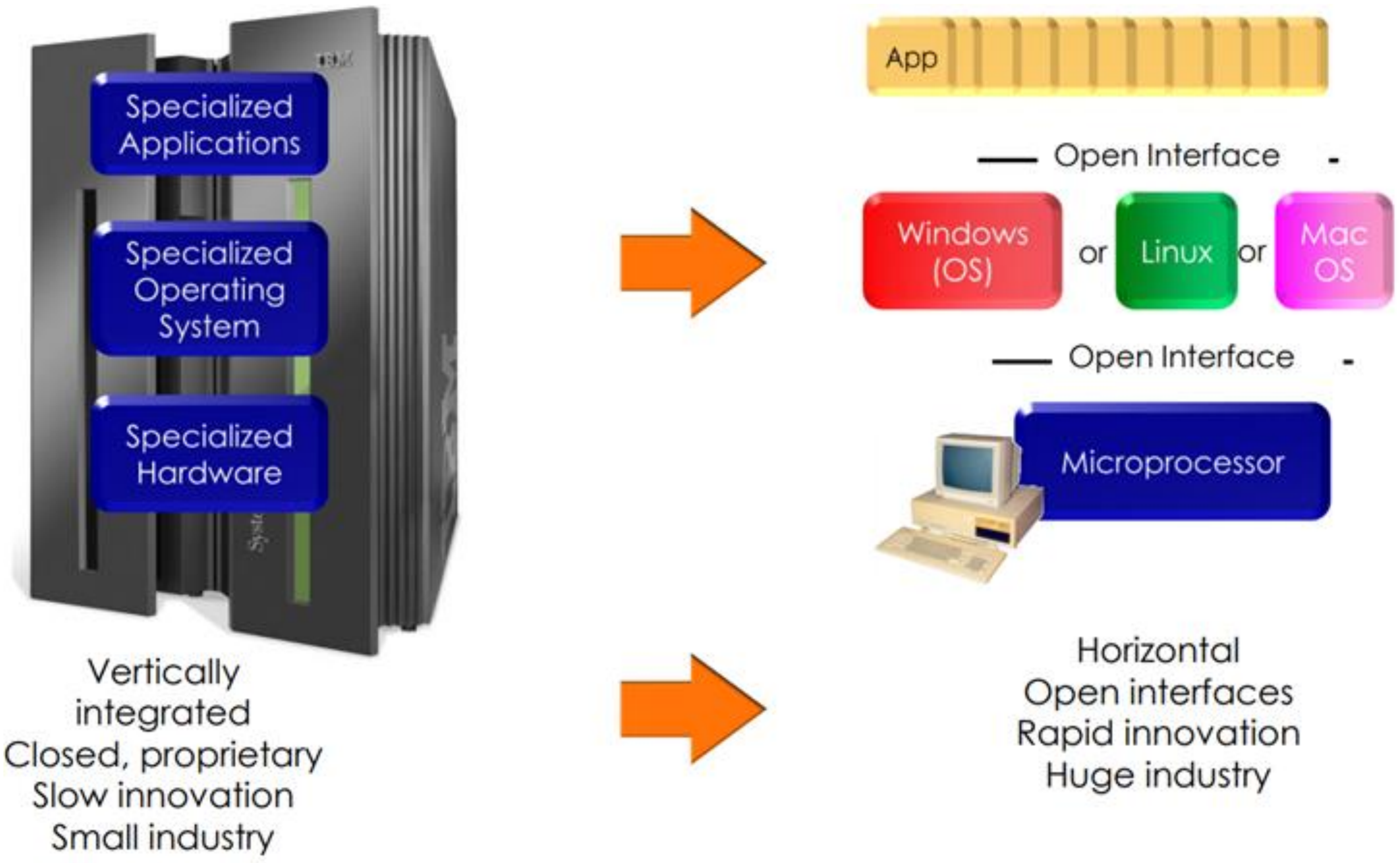
# The advent of SDN

- 2005: renewed interest in rethinking network control plane
- Software Defined Networking (SDN) emerged as a new networking architecture that
  - *Separates control plane from data plane*
  - *Software running on a logically centralized server remotely controls network hardware*
  - *Open standards (vendor neutral)*
  - *Directly programmable*
  - *Facilitates automation*
  - *programmability*

# SDN architecture


Consistent, up-to-date global network view
Control App 1
Control App 2
At least one Network OS probably many. Open- and closed-source
Network OS
Open interface for network device control
OpenFlow
Packet Forwarding
Packet Forwarding
Packet Forwarding
Packet Forwarding
Packet Forwarding

# Analogy: Mainframe to PC evolution



Specialized Applications

Specialized Operating System

Specialized Hardware

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

App

Open Interface

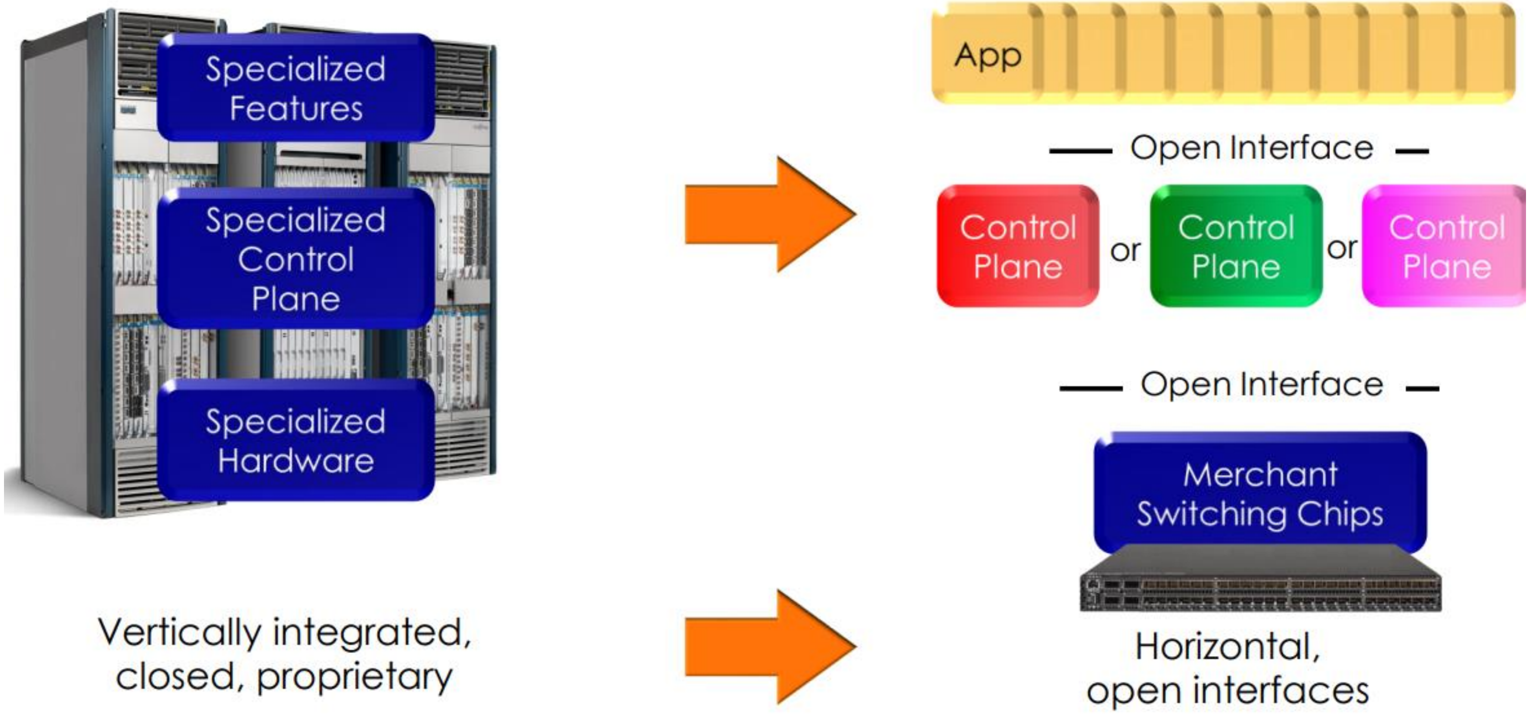
Windows (OS) or Linux or Mac OS

Open Interface

Microprocessor

Horizontal
Open interfaces
Rapid innovation
Huge industry

# Similar network evolution with SDN



Specialized Features

Specialized Control Plane

Specialized Hardware

App

— Open Interface —

Control Plane or Control Plane or Control Plane

— Open Interface —

Merchant Switching Chips

Vertically integrated, closed, proprietary

Slow innovation

Horizontal, open interfaces

Rapid innovation

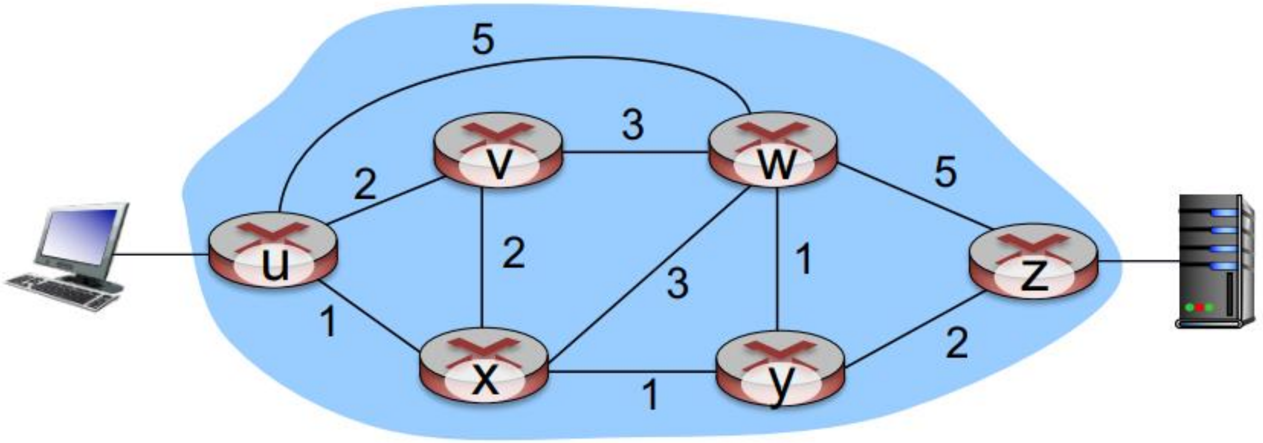
# Why SDN for future networks?

- Separation of control and data planes
  - *Enables independent development of new features without hardware modification*
- Global network view at control plane
  - *Network-wide view allows easier application of policies*
- Management Automation
  - *Minimize human intervention*
  - *Lower operational cost*
- Monitoring
  - *On-demand monitoring of resources*
  - *Dynamically change what to monitor*
  - *Application performance can be improved by advanced monitoring features*

# Why SDN for future networks?

- Elasticity
  - *Resources can be scaled up or down on-demand*
- More Flexible/Easier Performance Management
  - *Traffic engineering*
  - *Capacity optimization*
  - *Load balancing*
  - *Faster failure recovery*
- Network Function Integration
  - *On-demand provisioning of network functions*
    - *Firewalls, IDS, IPS, etc.*

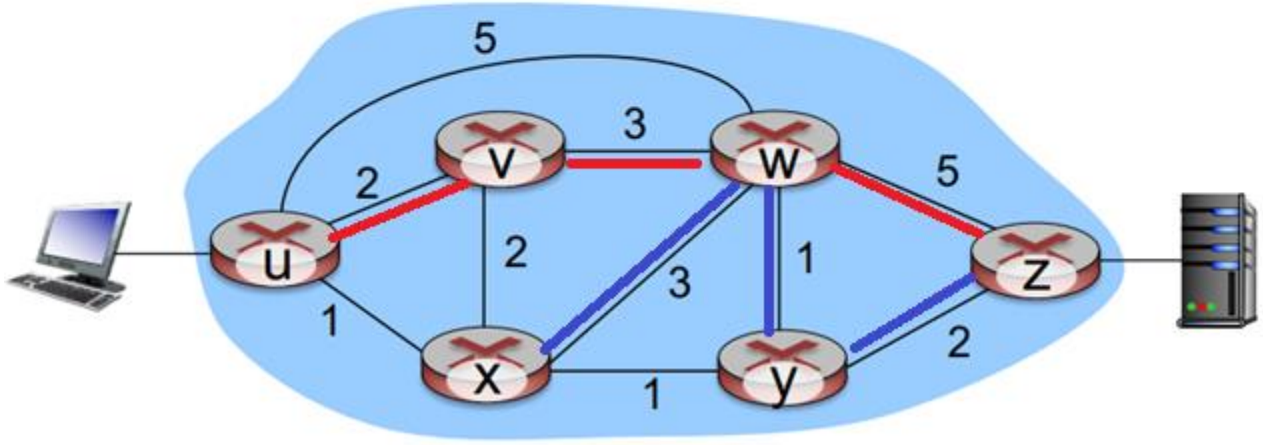
# Traffic engineering: difficult traditional routing



Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

A: need to define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

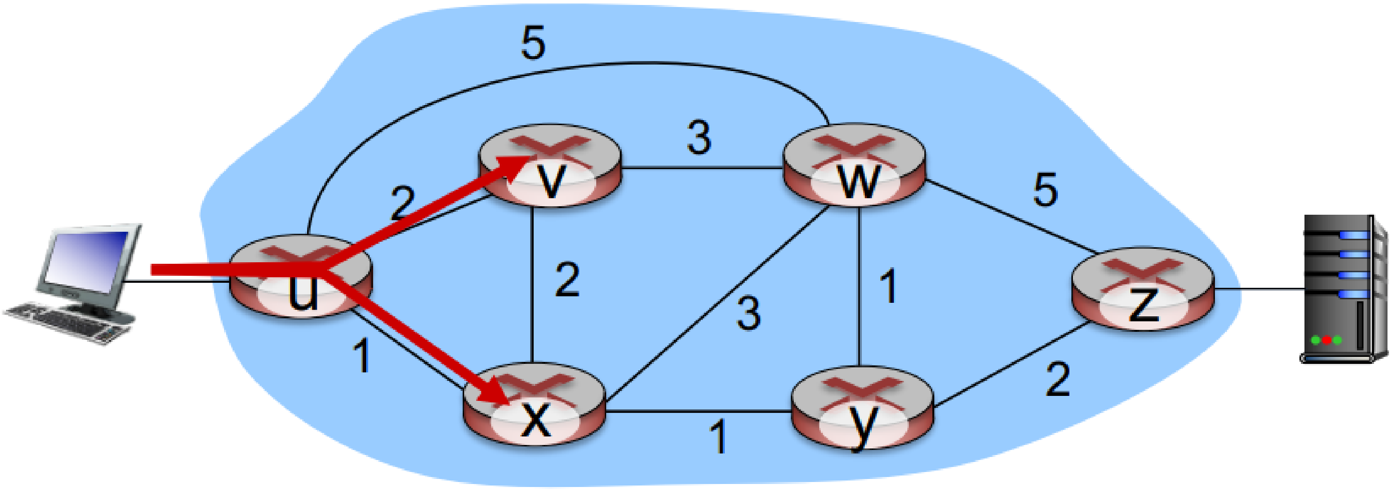
# Traffic engineering: difficult traditional routing



Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

A: need to define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

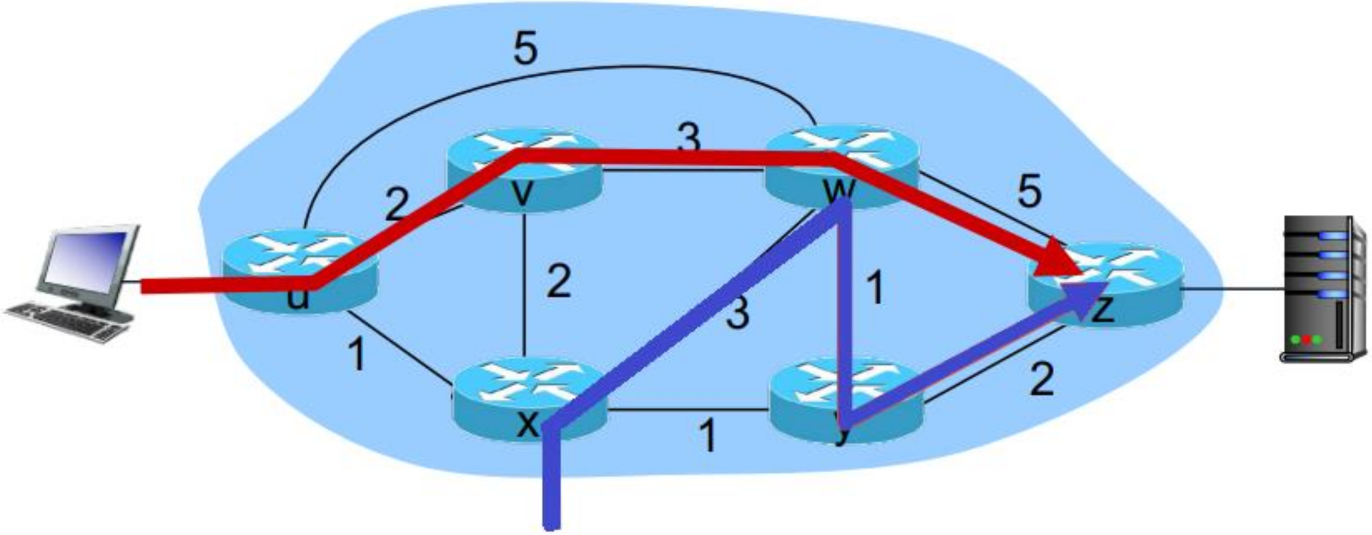
# Traffic engineering: difficult



Q: what if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

A: can't do it (or need a new routing algorithm)

# Traffic engineering: difficult



Q: what if w wants to route blue and red traffic differently?

A: can't do it (with destination based forwarding, and LS, DV routing)

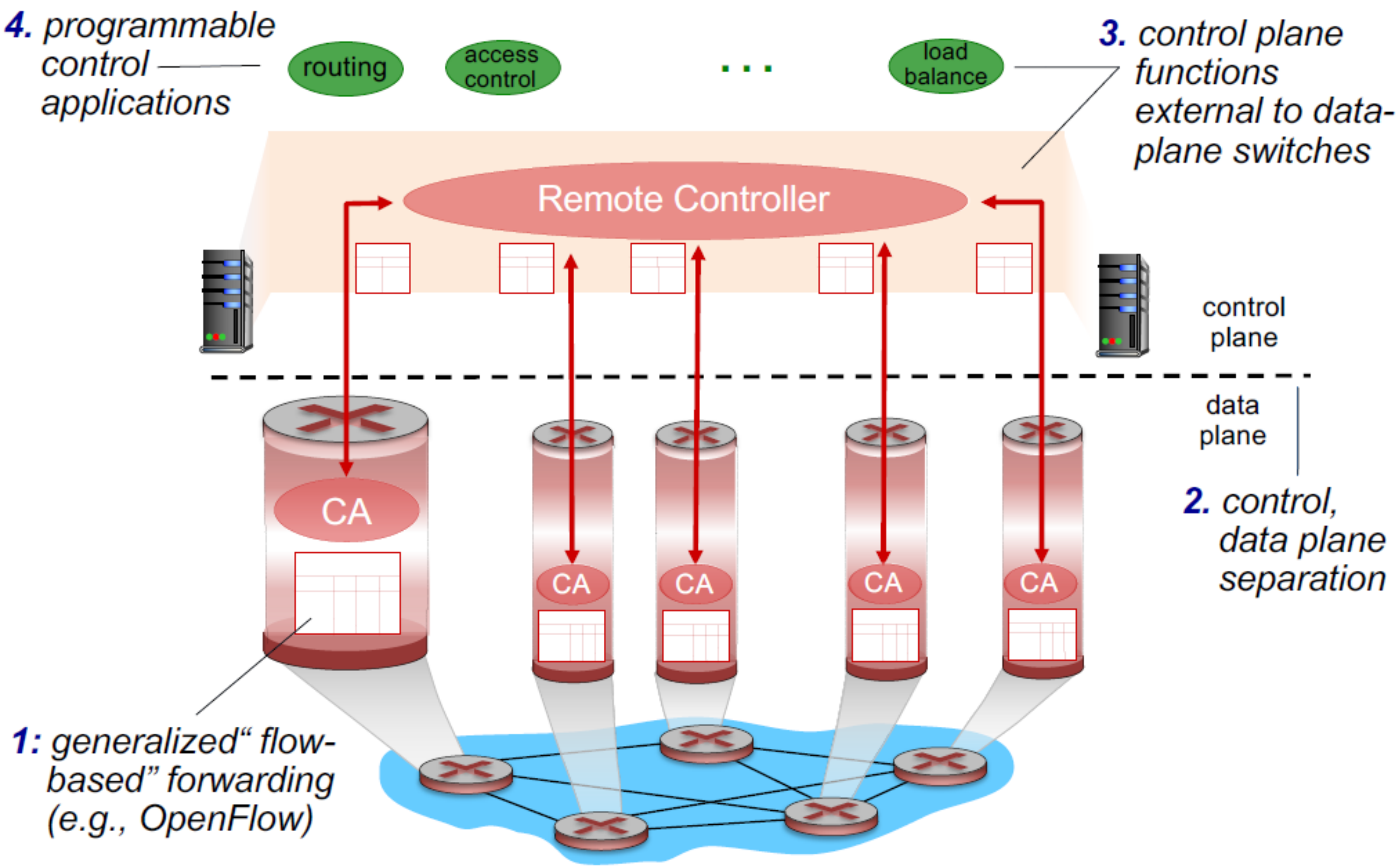
# Traffic engineering: difficult traditional routing

- The desired traffic management and optimization objectives cannot be achieved using traditional routing techniques that rely on destination-based forwarding, link-state (LS), and distance-vector (DV) routing protocols.

- These traditional methods route traffic based solely on the destination address and pre-determined paths without considering the current network conditions, such as congestion, load balancing, or optimal path utilization.

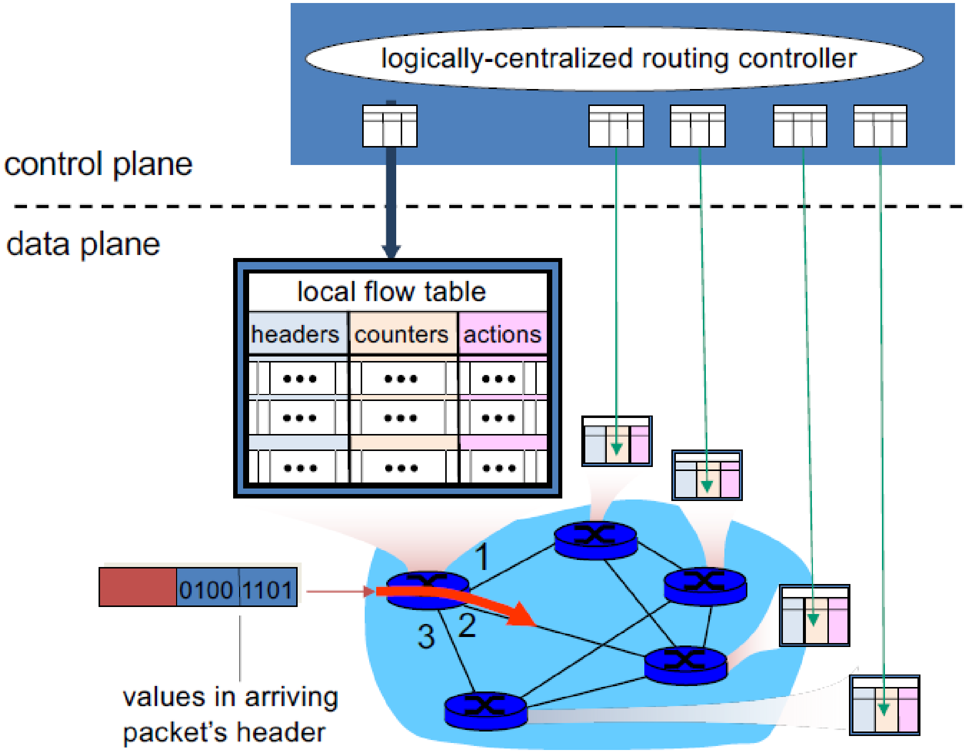
- **Traffic engineering** requires more advanced mechanisms, such as Multi-Protocol Label Switching (MPLS) or Software-Defined Networking (SDN), ***which can dynamically adjust paths and optimize the flow of traffic across the network***.

# Software defined networking (SDN)


4. programmable control applications
routing
access control
. . .
load balance
3. control plane functions external to data-plane switches
Remote Controller
control plane
data plane
2. control, data plane separation
CA
CA
CA
CA
CA
1: generalized" flow-based" forwarding (e.g., OpenFlow)

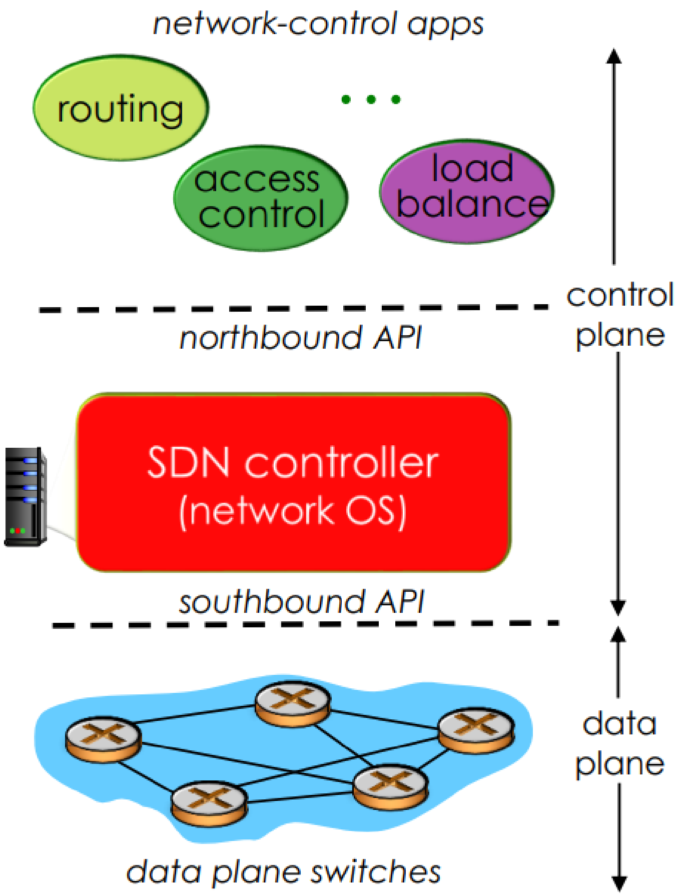
# Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



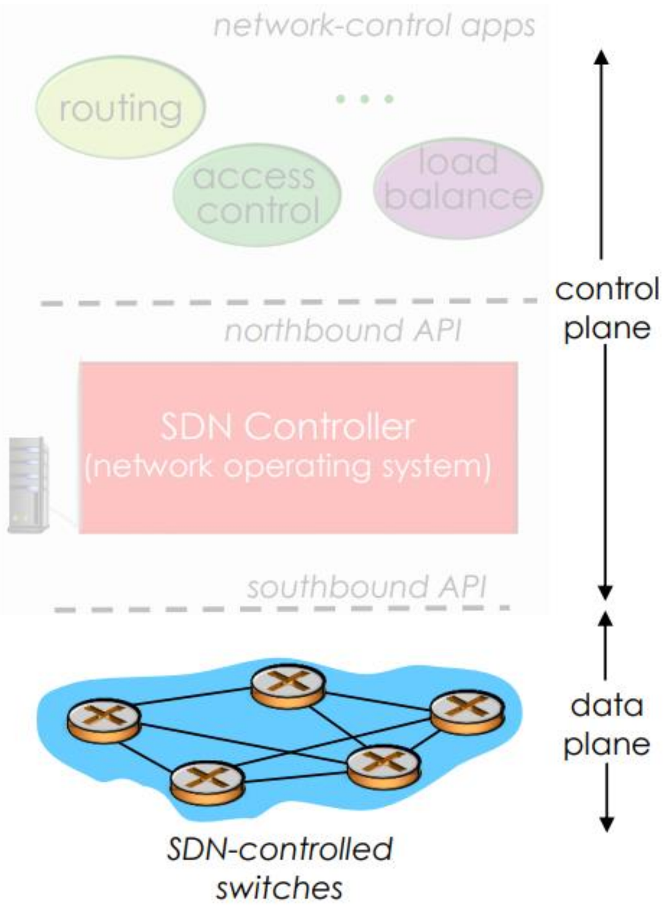
# Components of SDN

- Data plane switches
  - SDN controlled switches
- South Bound API
  - Interface between control and data plane (e.g. OpenFlow)
- SDN Controller
- North Bound API
  - Interface between controller and control apps
- Application layer:
  - It contains the typical network applications like intrusion detection, firewall, and load balancing



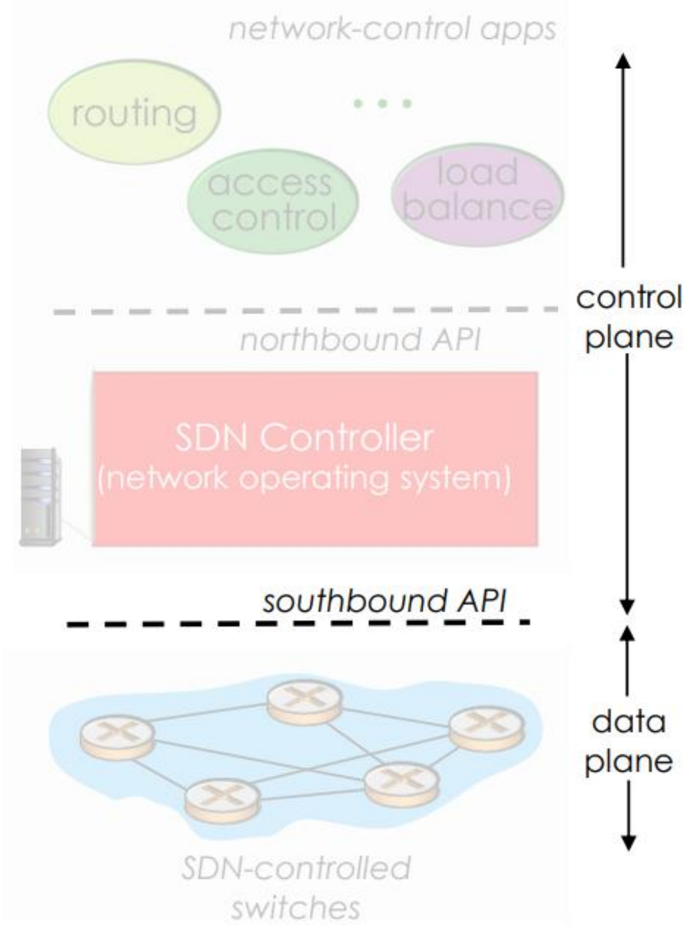
# Data plane switches

- Programmable Network Devices
  - fast, simple, commodity switches *implementing data-plane forwarding in hardware*
  - can be software (***virtual switch***)
  - typically has *no control functionality built-in*
  - programmed by a remote controller
    - switch flow table computed, installed by controller
  - Supports different SB-APIs (e.g., OpenFlow)
  - Available from different vendors
    - Pica8, Pronto, HP, NEC, etc.



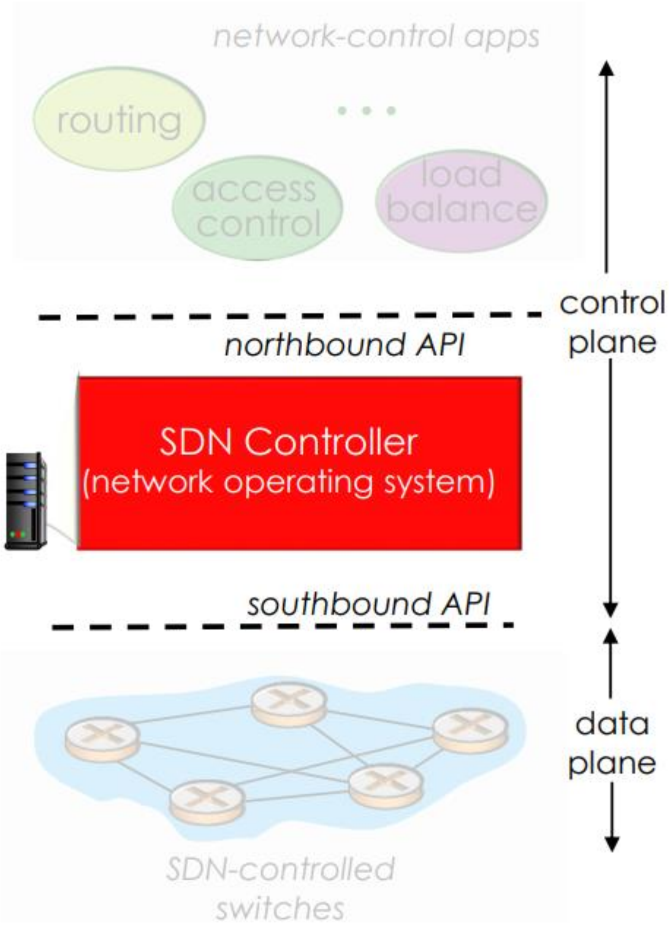
# South-Bound API

- The SB-API is used by the controller to communicate with the data plane switches
  - defines what is controllable and what is not
- SB-API can be for
  - Control (e.g., OpenFlow)
  - Configuration (e.g., OF-CONFIG)
- OpenFlow is the most popular SB-API for SDN
  - Implemented over TCP
  - Supports both in-band and out-of-band control



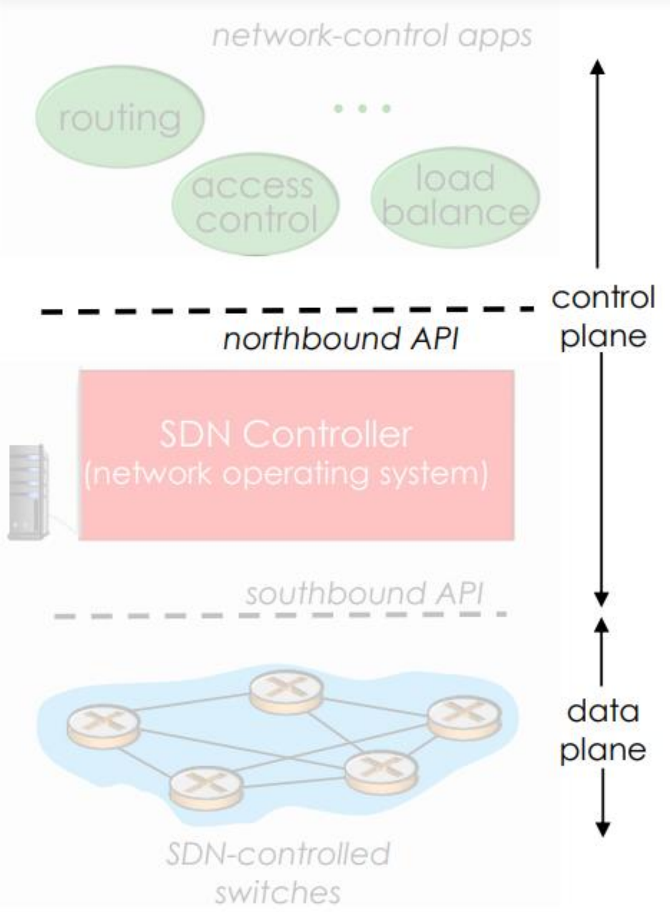
# SDN Controller

- Network OS
  - logically centralized software to control data plane switches
  - Maintain network state information
  - interacts with network control applications “above” via northbound API
  - interacts with network switches “below” via southbound APIs
    - supports one or more SB-APIs
- Controllers
  - **OpenFlow**: NOX/POX, Floodlight, OpenDaylight, ONOS, etc.
  - Other: Path Computation Element (PCE) in MPLS/GMPLS, ForCES, etc.



# North Bound API

- Used by control apps to communicate with the controller
  - obtain network state
    - flow table content, counters, device health, etc.
  - modify network state
    - add/remove flow table entry, turn on/off a switch port, etc.
- Used by controller to provide network-wide view to control apps

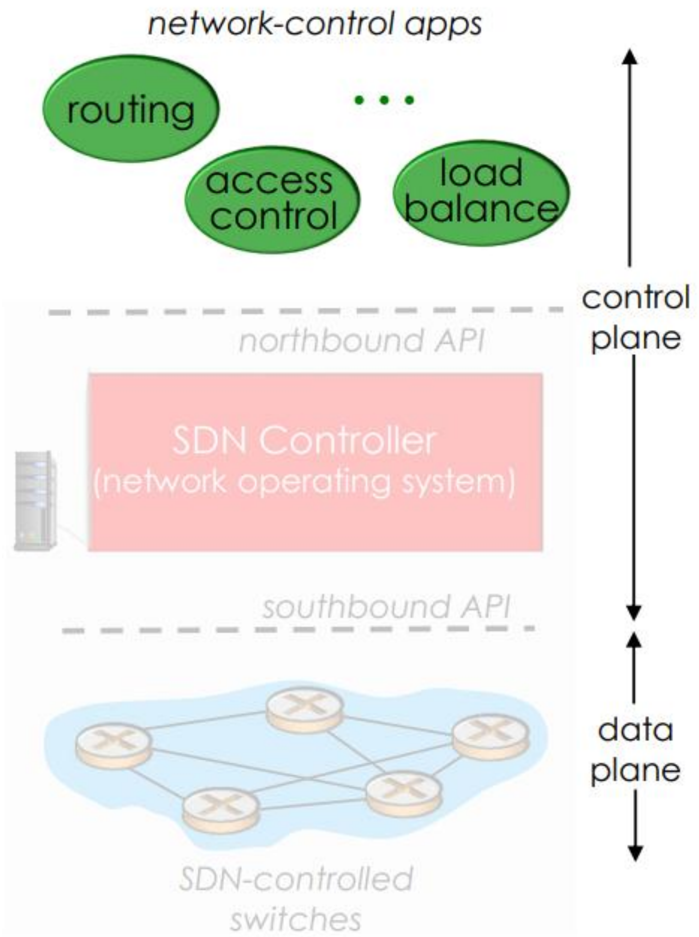


# Control apps

- “brains” of control: implement control functions using lower-level services, API provided by SDN controller

- unbundled: can be provided by 3rd party: distinct from routing vendor, or SDN controller

- Example apps: *load balancing, access control, QoS management,* etc.

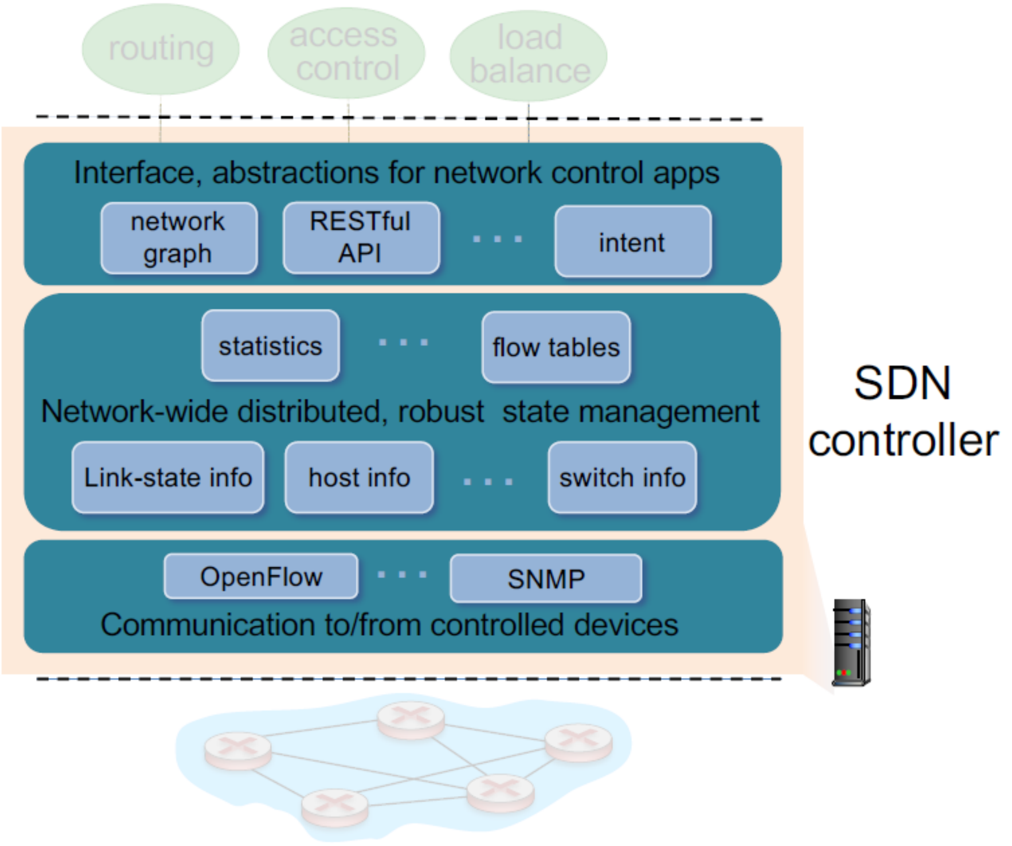


# Components of SDN controller

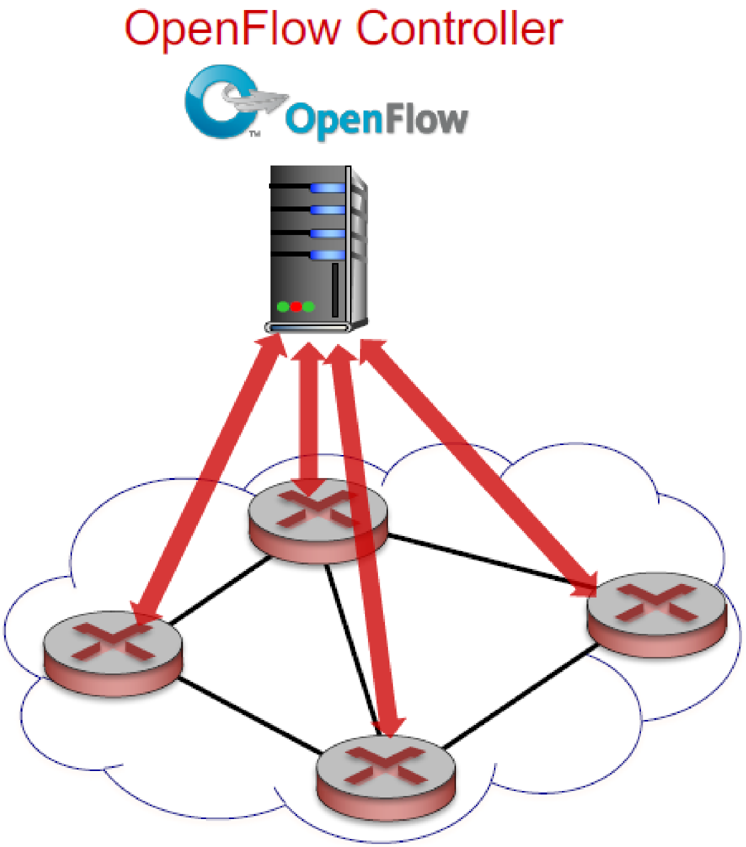
Interface layer to network control apps: abstractions API

Network-wide state management layer: state of networks links, switches, services: a *distributed database*

*communication layer*: communicate between SDN controller and controlled switches



# OpenFlow protocol



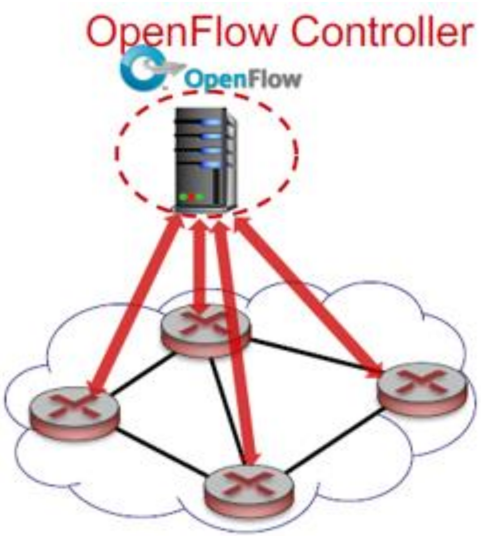
- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

# OpenFlow: controller-to-switch messages

❑ The primary types of controller-to-switch messages include:

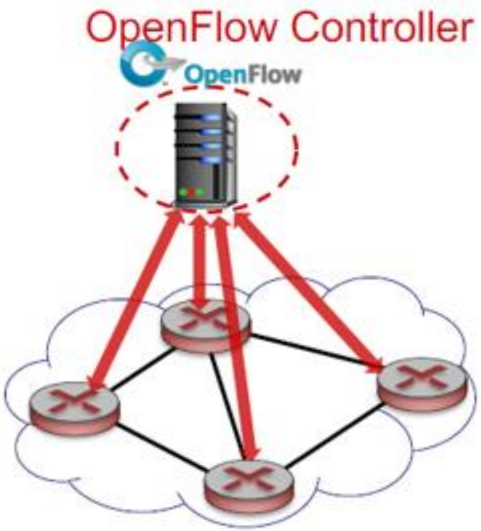
1. **Features**: The controller requests the switch's capabilities, and the switch responds with its features, such as supported ports and actions.

2. **Configuration**: The controller sets or queries configuration parameters of the switch, such as the length of time the switch should wait before sending flow statistics.



# OpenFlow: controller-to-switch messages

3. **Modify-State:** These messages alter the state of the switch, such as adding, modifying, or deleting flow entries in the switch's flow table. This includes:
   - Flow Mod (Flow Modification): Directs the switch to add, update, or delete flow entries based on specified match conditions and actions.
   - Port Mod (Port Modification): Changes the configuration of the switch ports, such as enabling or disabling a port.

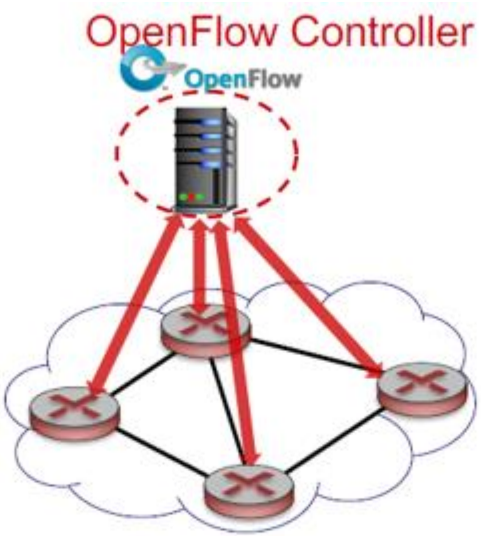


4. **Packet-Out:** The controller sends a specific packet out through one or more ports of the switch, used for forwarding packets that do not match any flow entry or for implementing custom forwarding behavior.

# OpenFlow: switch-to-controller messages

❑ The primary types of switch-to-controller messages include:

1. **Packet-In:** The switch sends this message to the controller when it receives a packet that does not match any existing flow entries. The controller then decides how to handle the packet, such as creating a new flow entry or sending the packet out through a specific port.

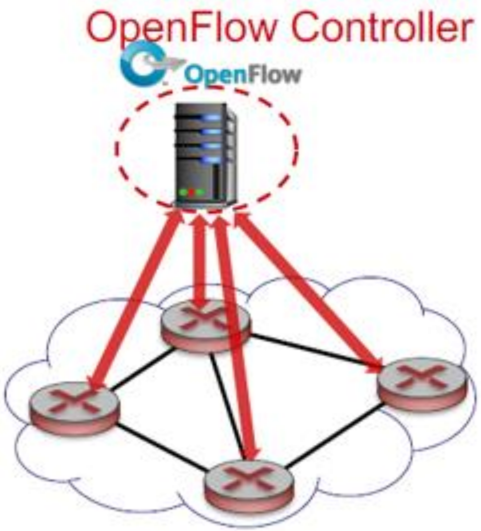


2. **Flow-Removed:** This message is sent by the switch to inform the controller that a flow entry has been removed from the flow table, either because it expired or was explicitly removed.

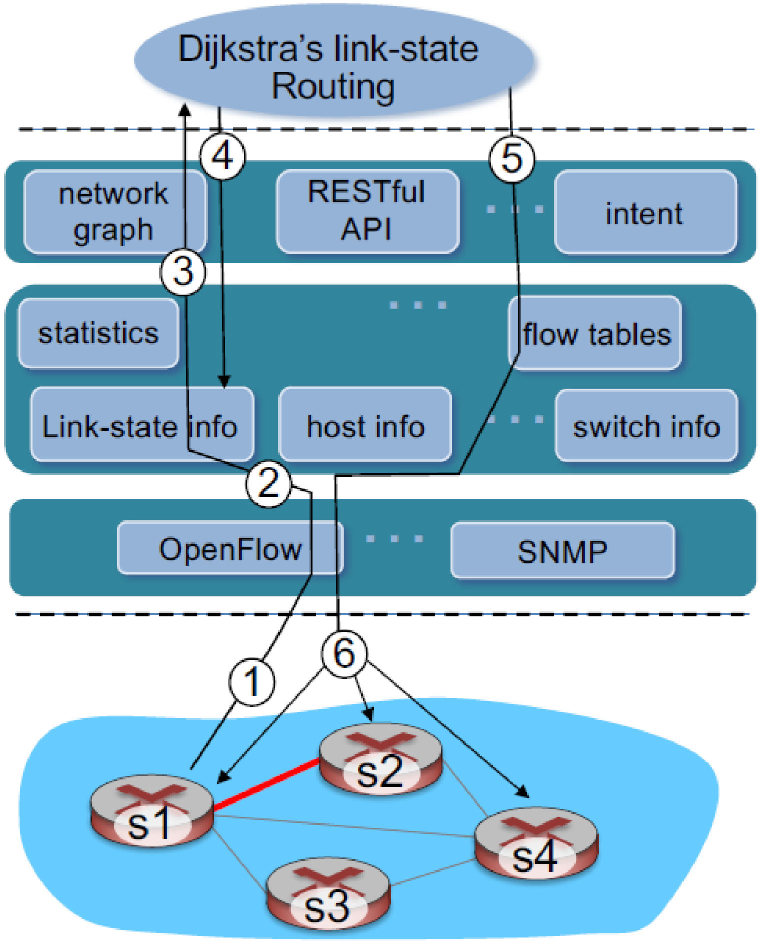
# OpenFlow: switch-to-controller messages

3. **Port Status:** The switch uses this message to notify the controller of changes in the status of a port, such as when a port goes up or down.

Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller



# SDN: control/data plane interaction example



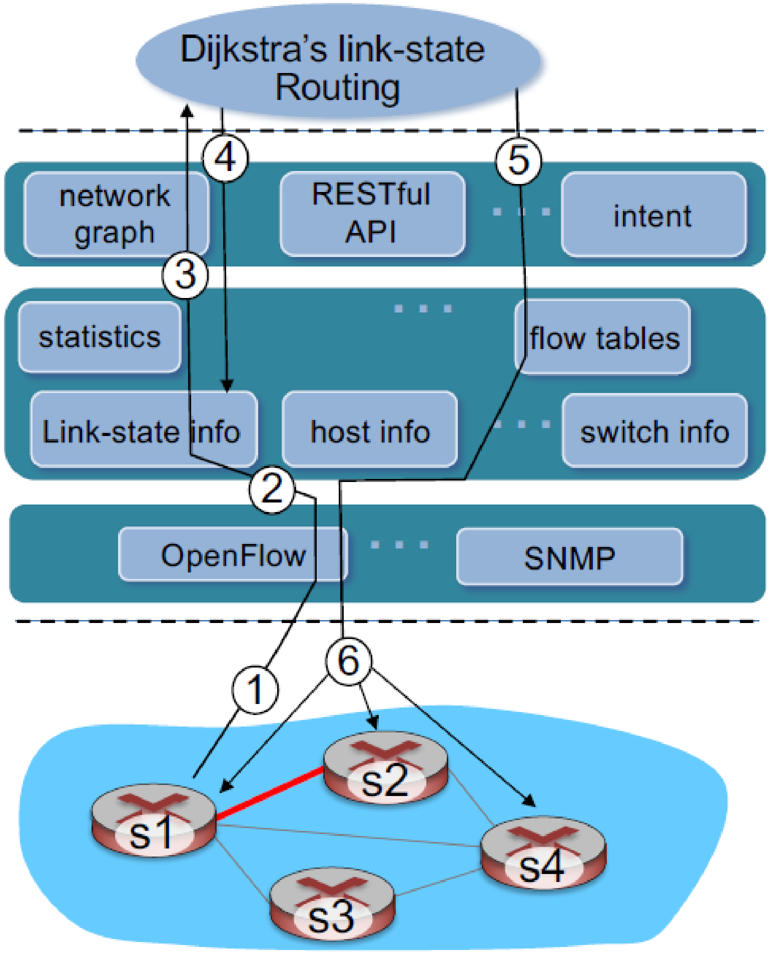
1. S1, experiencing link failure using OpenFlow port status message to notify controller

2. SDN controller receives OpenFlow message, updates link status info

3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.

4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

# SDN: control/data plane interaction example



⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

⑥ Controller uses OpenFlow to install new tables in switches that need updating

# SDN Benefits

- Greater span of control and network analytics and response.
- Better intelligence with a global view of the network rather than each network element looking at the network from its own viewpoint.
- Improved application experience and empower the network owner/operator
- Rapid deployment of applications using networking that supports the application's specific needs.
- Simplified and automated IT administration
- Opportunity to open up the network to a diverse set of vendors and disaggregation

# Opportunities Where Separation Helps

- Data Centers
  - VM migration, L2 Routing
- Routing
  - More control over decision logic
- Security
  - Traffic Filtering, Multi-Tenancy, Network Access Control

# Example Data Centers (Yahoo!)

- **Yahoo!'s data centers**, equipped with 20,000 servers capable of supporting 400,000 VMs, are designed for high performance and scalability.
- The networks "any-to-any" connectivity with 1024 distinct inter-host links provides robust, low-latency communication between server.
- The capability for sub-second VM migration with guaranteed consistency ensures minimal downtime and reliable service continuity, which is crucial for maintaining the high availability and reliability of Yahoo!'s diverse range of internet services.

# Example Data Centers (Yahoo!)

- Problem: Keeping all switches in sync. with 400,000 VMs
- Solution: Program switch from a central point

# Example: SDN in Real World – Google's Story

- The industries were skeptical whether SDN was possible.

- **Google had big problem**.
  - High financial cost management their datacenters: Hardware and software upgrade, over provisioning (fault tolerance), manage large backup traffic, time to manage individual switch, and a lot of men power to manage the infrastructure.
  - Delay caused by rebuilding connections after link failure:
    - Slow to rebuild the routing tables after link failure
    - Difficult to predict what the new network may perform

- **Google went a head and implemented SDN**
  - Built their hardware and wrote their own software for their internal datacenters.
  - Surprised the industries when Google announced SDN was possible in production

- How did they do it?
  - Read "B4: Experience with a Globally-Deployed Software Defined WAN"

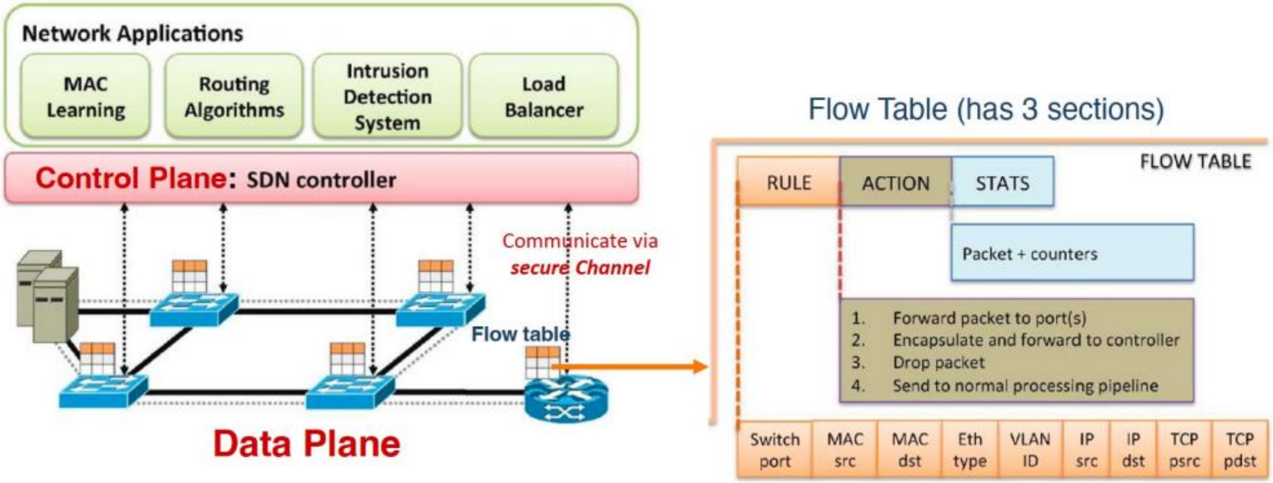
# What is OpenFlow?

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

- **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.

# What is OpenFlow?

- 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.


Network Applications
MAC Learning
Routing Algorithms
Intrusion Detection System
Load Balancer
Control Plane: SDN controller
Communicate via secure Channel
Flow table
Data Plane
Flow Table (has 3 sections)
FLOW TABLE
RULE
ACTION
STATS
Packet + counters
1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
Switch port
MAC src
MAC dst
Eth type
VLAN ID
IP src
IP dst
TCP psrc
TCP pdst

# What is OpenFlow?

- **Match Fields:** This section defines the criteria used to match incoming packets against existing flow entries in the flow table. These criteria can include fields such as source and destination MAC addresses, source and destination IP addresses, IP protocol, source and destination ports, VLAN tags, etc. When a packet arrives at a switch, it is compared against the entries in the flow table based on these match fields to determine how it should be processed.

- **Actions:** This section specifies the actions to be taken on packets that match the criteria defined in the match fields. Actions can include forwarding packets to a specific port, dropping packets, modifying packet headers, sending packets to the controller for further processing, etc. Each flow entry in the table contains one or more actions that are executed when a packet matches the corresponding match fields.

# What is OpenFlow?

- **Counters and Statistics:** This section keeps track of various statistics related to the flow entries, such as the number of packets and bytes processed by each flow, packet drop count, etc. These statistics can be useful for monitoring and troubleshooting network behavior and performance.

# SDN Controllers

- NOX/POX
- Ryu
- Floodlight
- OpenDaylight
- Pyrectic
- Frenetic
- Procera
- RoutFlow
- Trema

# SDN Controllers

## NOX Controller

- First-generation OpenFlow controller
  - Open source, stable, widely used
- Two “flavors” of NOX
  - NOX-Classic: C++/Python. No longer supported
  - NOX(the new NOX)
    - » C++ only
    - » Fast, clean codebase
    - » Well maintained and supported

# SDN Controllers

## POX Controller

- POX in Python
  - – Supports OpenFlowv.1.0 only
- Advantages
  - – Widely used, maintained, supported
  - – Relatively easy to read and write code
- Disadvantages:
  - performance

# SDN Controllers

## Ryu Controller

- Open source Python controller
  - Supports OpenFlowv.1.0, v.1.2, v.1.3, v.1.4, v.1.5
  - Work with OpenStack
- Advantages
  - OpenStack integration
- Disadvantages:
  - performance

# SDN Controllers

## Floodlight Controller

- Open source Java controller
  - Supports OpenFlowv.1.0
  - Fork from the Beacon Java OpenFlow controller
  - Maintained by Big Switch Networks
- Advantages
  - Good documentation
  - Integration with REST API
  - Production-level, OpenStack/Multi-Tenant Clouds
- Disadvantages:
  - Steep learning curve

Project
Floodlight

# SDN Controllers

## OpenDaylight Controller

- Open source Java controller
  - Is a modular open platform for customizing and automating networks of any size and scale
- Advantages
  - Industry acceptance
  - Integration with OpenStack
- Disadvantages:
  - Complex
  - Steep learning curve