Software Defined Networking & Cloud Computing

EE450: Computer Networks

Professor A. Zahid

The Internet (from outside)

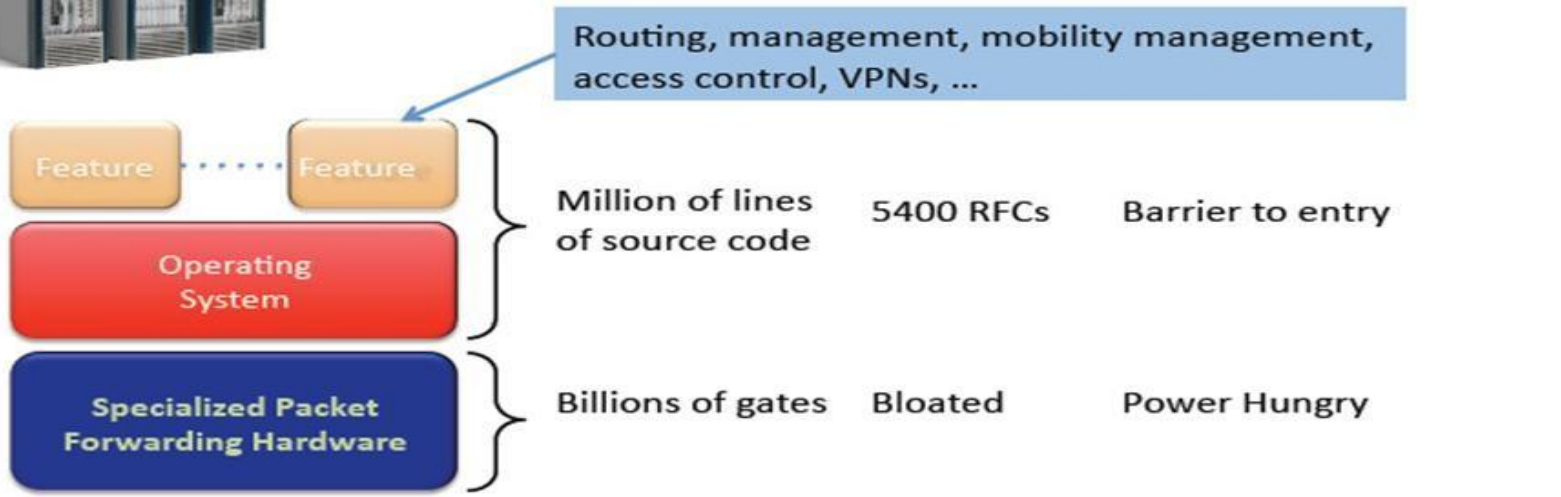
- Tremendous success
  - From research experiment to global infrastructure
- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Programmable hosts: arbitrary applications
- Enables innovation
  - Apps: Web, P2P, VoIP, social networks, ...
  - Links: Ethernet, fiber optics, Wi-Fi, cellular, ...
- Changes are easy at the edge!

The Internet (from inside)

- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features
- Lots of domain details
  - Whole bunch of protocols and header formats
  - Lots of boxes (Routers, Switches, Firewalls) and tools

Impacts performance, security, reliability, cost...

The Ossified Network



Many complex functions baked into the infrastructure

OSPF, BGP, multicast, differentiated services, Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …

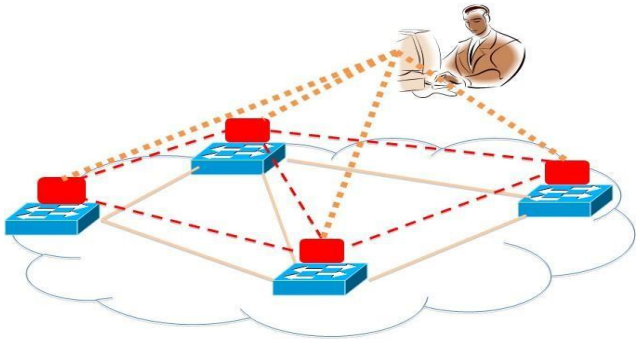
An industry with a "mainframe-mentality", reluctant to change

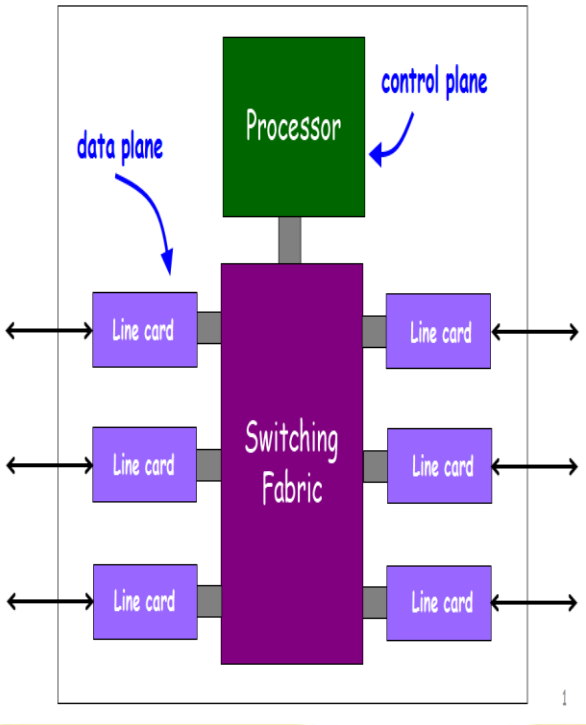
Basic Concepts of SDN (1)

- Separation of Data and Control Planes
  - The two planes (in addition to the management plane) are "tightly" integrated (Software and Hardware). Network operators have to configure each protocol separately on each individual device
  - In some cases (Such as Routing Protocols), each router is required to propagate control packets, receive responses, update tables accordingly, again resulting in increased delay and inefficient use of resources
  - SDN solution: Separate the control plane from the data plane. Use a Logical "centralized" controller that manages the routers/switching processing functionalities
    - Results? Better Scalability, Better management, better visibility of network resources, reduction of manual intervention, more control over the network, improved security, etc...

Network Planes

- Control Plane
  - Carries “Signaling Traffic”, Sets up Routing Tables, Admission Control, QoS , System configuration, Tracking topology changes
- Data Plane
  - Responsible for Switching/Forwarding packets, Mapping Header fields
  - Traffic Shaping, Filtering, Buffering, Scheduling, Policing, Measuring, etc...
- Management Plane
  - Responsible for administration, operation monitoring, device performance monitoring, collecting measurements, ex: SNMP

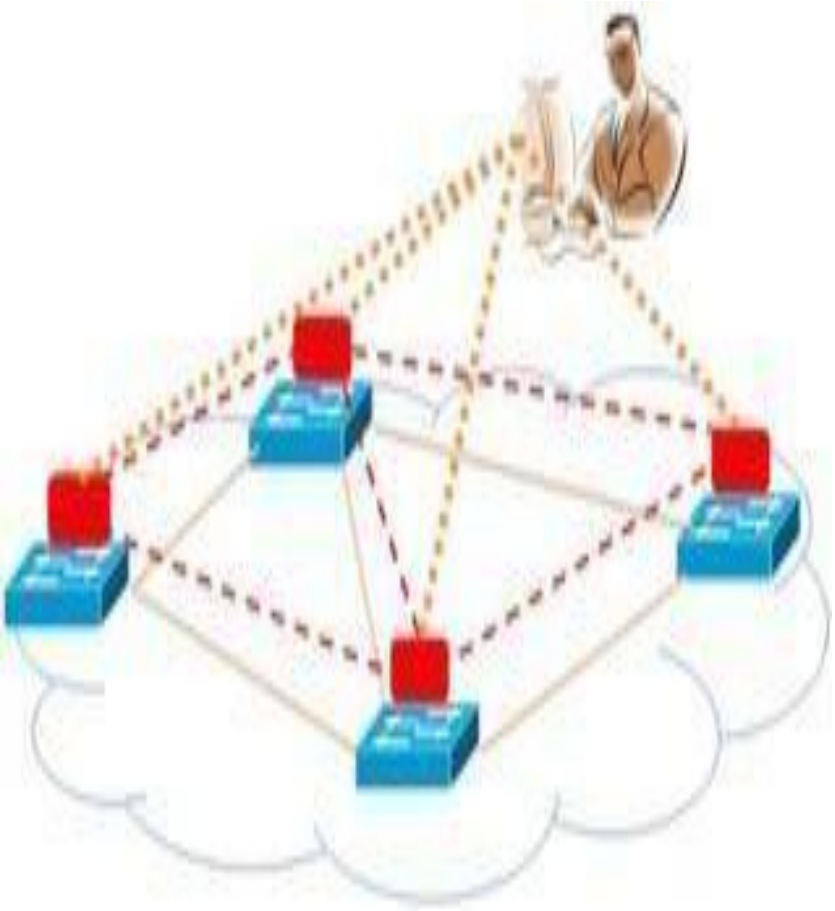


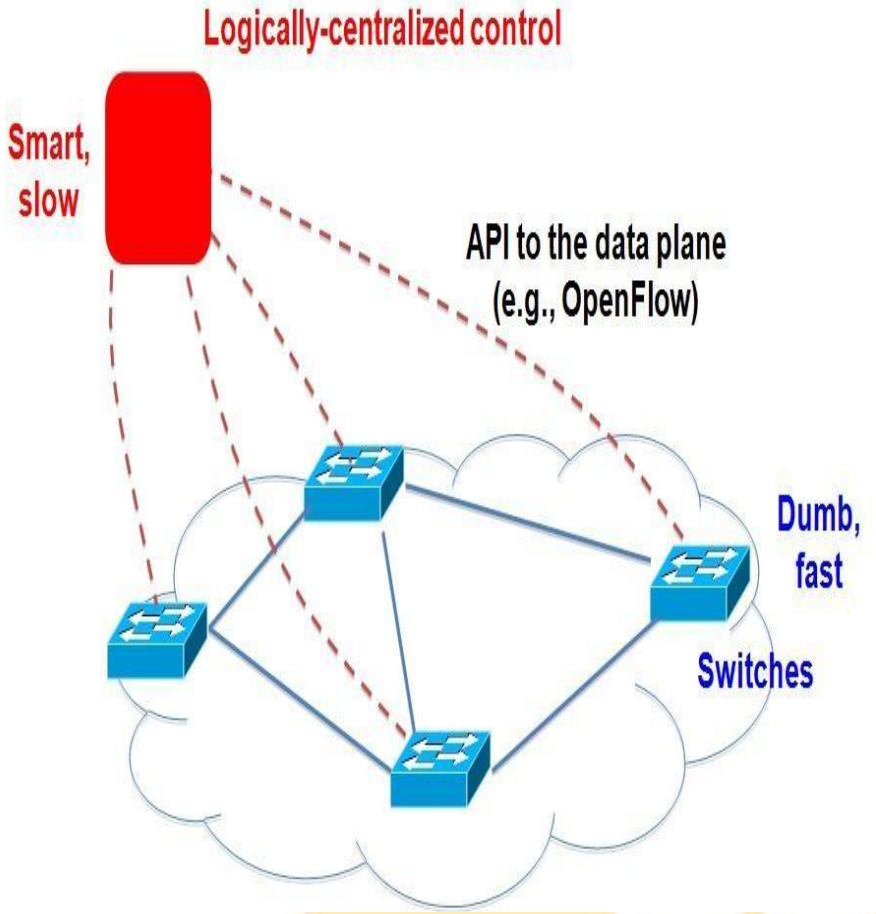


Timing Scale

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Tasks	Forwarding, buffering, filtering, scheduling	Routing, circuit set-up	Analysis, configuration
Location	Line-card hardware	Router software	Humans or scripts

Traditional Network vs. SDN





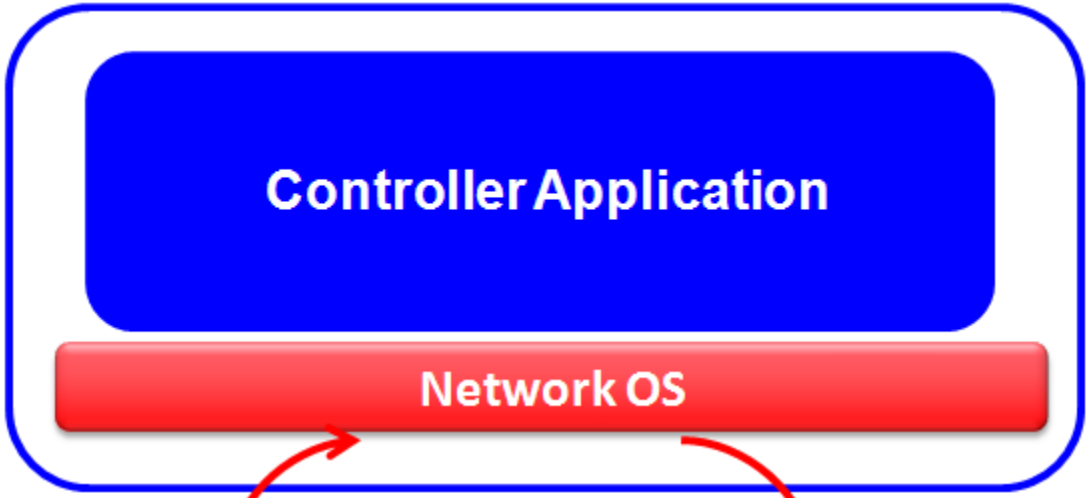
Basic Concepts of SDN (2)

- Programmability of the Control Plane
  - Moving the control plane to software (instead of firmware) allows for more flexible and dynamic access to network resources and administration
  - Network administer can control traffic, manage resources, reconfigure network devices, without having to touch "individual" devices!
  - The SDN controller will update tables, computes least cost paths, react to events (link/node failures, nodes joining in, perform load balancing (especially in locations close to data centers), implement network policies, etc...
  - SDN controller needs to establish connectivity with every device in network
    - Problem of Single point of failure in centralized control??? SDN suggests multiple "standby controllers"

Basic Concepts of SDN (3)

- Standardization of APIs
  - With a programmable centralized control plane, the network can be partitioned into several virtual networks (all sharing the same hardware infrastructure), each with different polices
  - Above is possible through the creation of standard APIs
  - SDN allows applications to interact with the network through the control plane. They can direct the configuration of networks, optimize resource allocations, etc...
- Southbound API: Communications between the control plane and the data plane. Example: Open Flow
- Northbound API: Communications between applications and the control plane

Controller Programmability

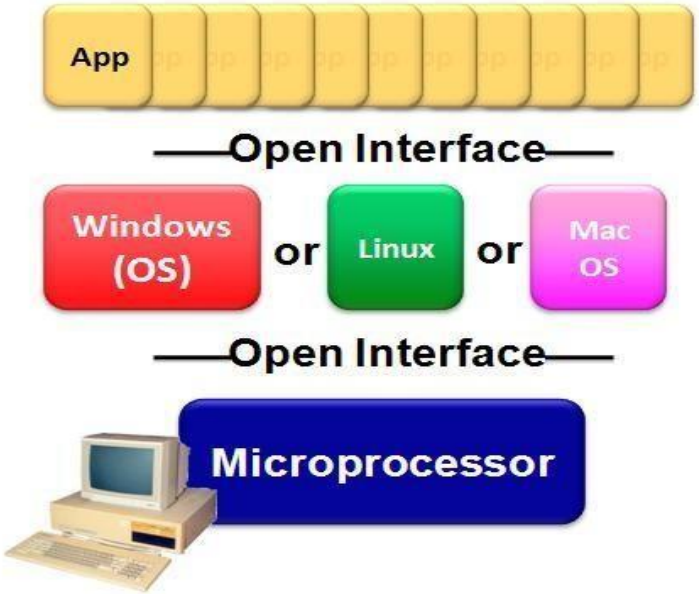


Events from switches
Topology changes,
Traffic statistics,
Arriving packets

Commands to switches
(Un)install rules,
Query statistics,
Send packets

Analogy: Mainframes vs. Microprocessors

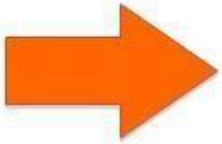
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

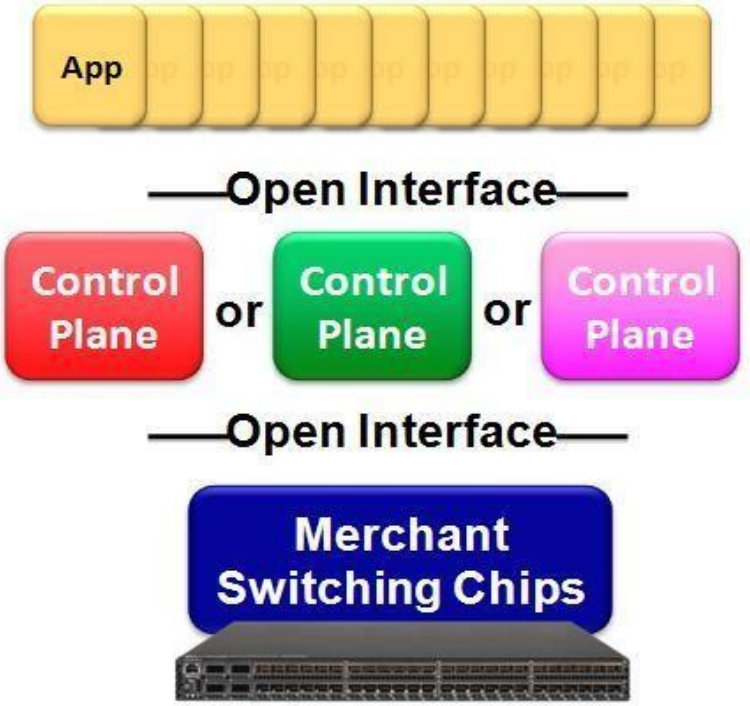


Horizontal
Open interfaces
Rapid innovation
Huge industry

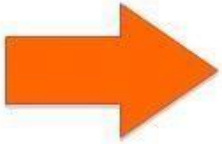
USC

Analogy: Routers/Switches





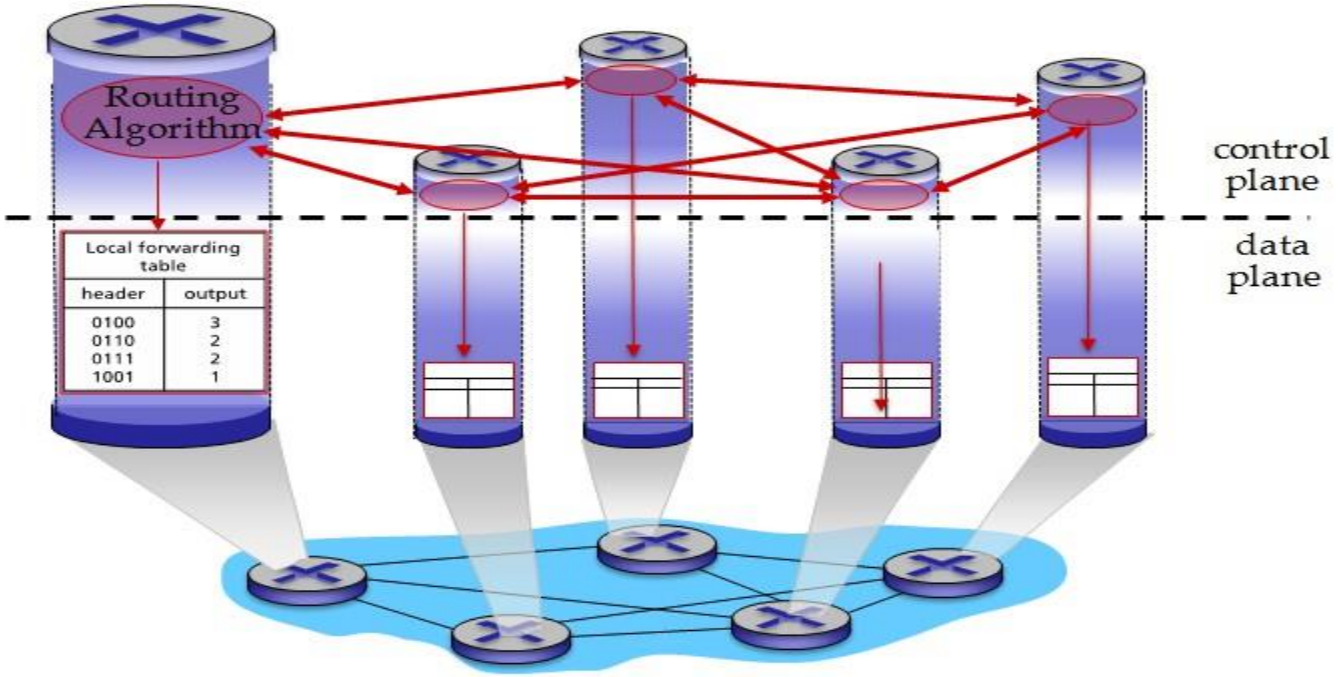
Vertically integrated
Closed, proprietary
Slow innovation



Horizontal
Open interfaces
Rapid innovation

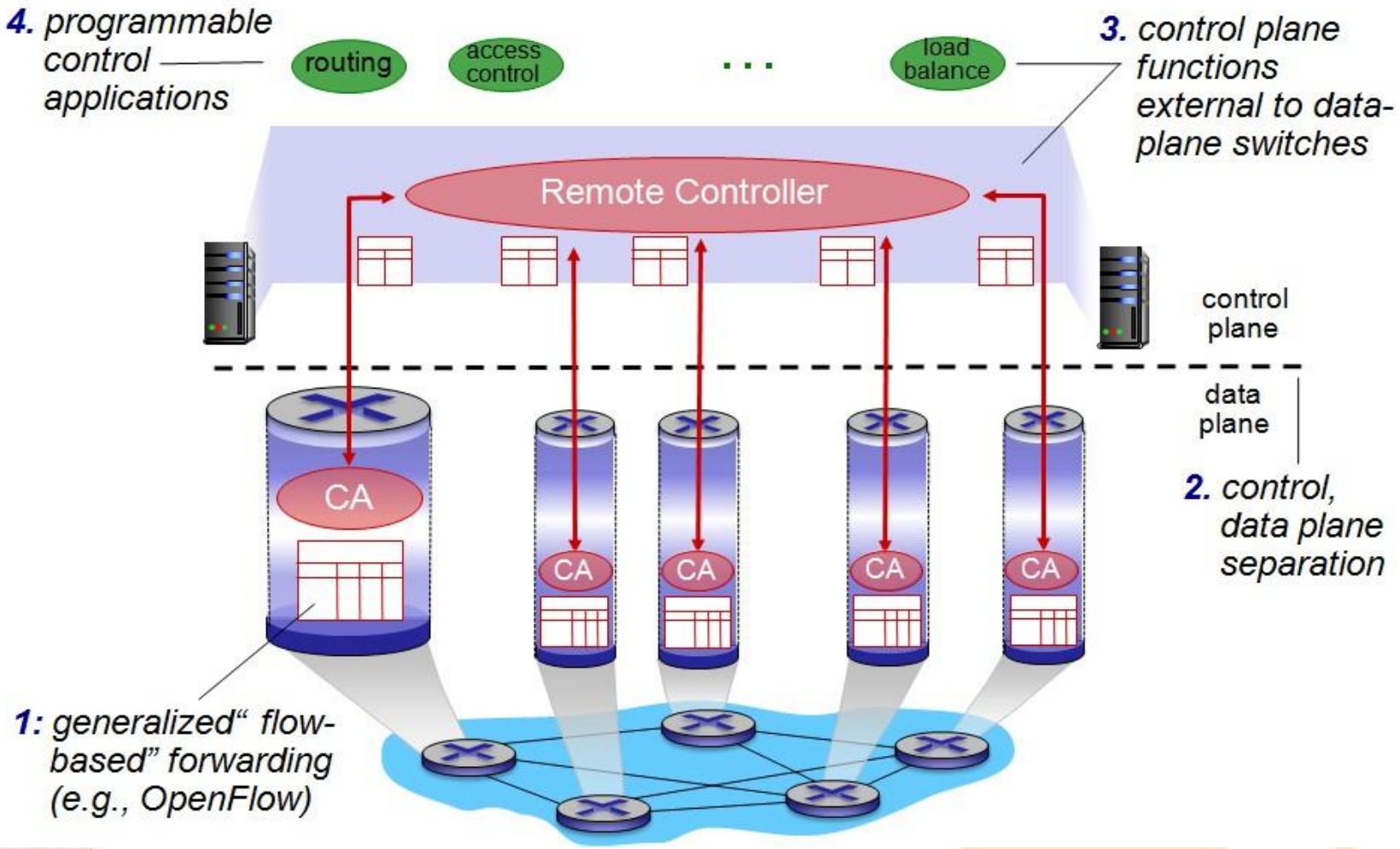
Per Router Control Plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



*R*outer contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS) with different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ..

SDN Principals



Control & Data Planes

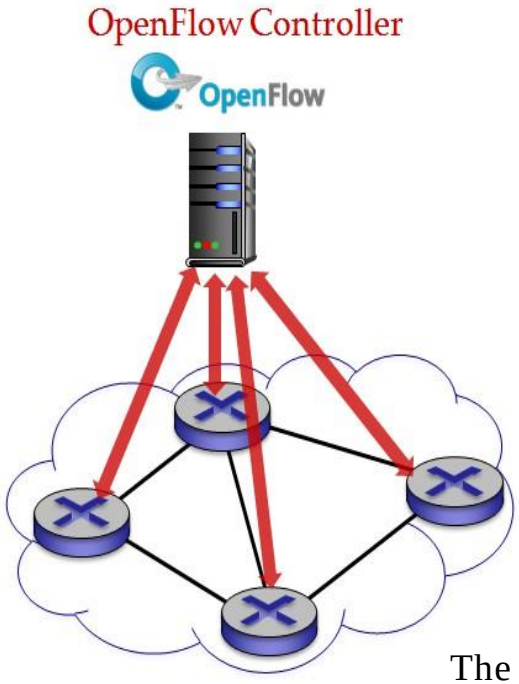
Data Plane Switches

- Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- Switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
- Protocol for communicating with controller (e.g., OpenFlow)

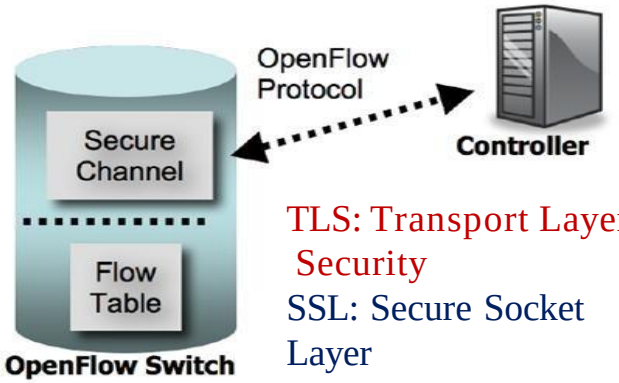
Control Plane: SDN controller

- Maintain network state information
- Interacts with network control applications "above" via northbound API
- Interacts with network switches "below" via southbound API
- Implemented as distributed system for performance, scalability, fault-tolerance, robustness

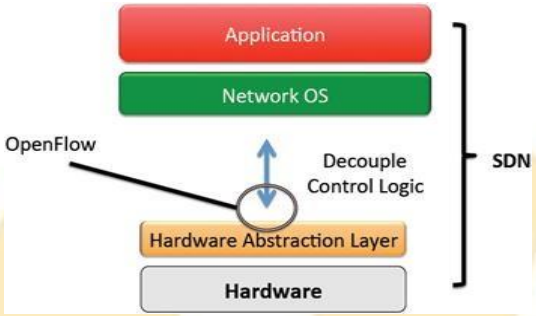
Open Flow Protocol



- 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)



TLS: Transport Layer Security
SSL: Secure Socket Layer



The switch is represented by a **logical abstraction** of a single flow table that performs packet lookup (header matching) and forwarding. OpenFlow does NOT dictate any hardware implementation or architecture of Switches

OpenFlow exploits the fact that Ethernet switches and routers contain flow-tables that run at line-rate to implement firewalls, NAT, QoS, and to collect statistics. While each vendor's flow-table is different , there is common set of functions that run in these switches and routers.

OpenFlow: Controller-to-Switch Messages

Key controller-to-switch messages:

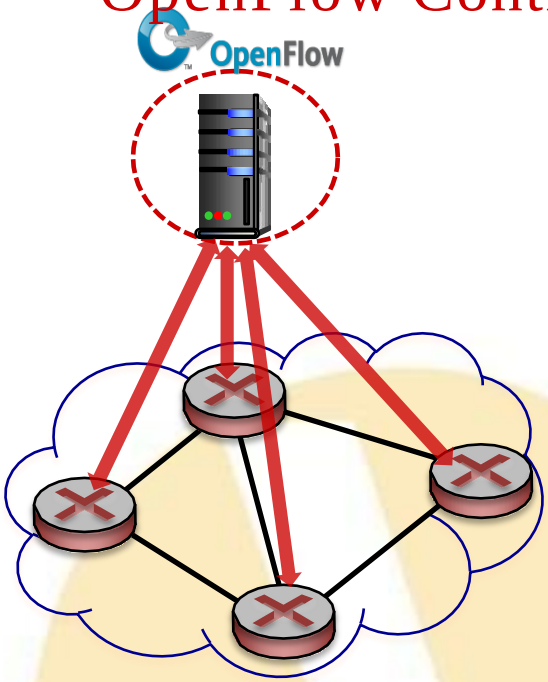
Features: controller queries switch features, switch replies

Configure: controller queries/sets switch configuration parameters

Modify-state: add, delete, modify flow entries in OpenFlow tables

Packet-out: controller can send this packet out of specific switch port

Read-State: controller can collect statistics (Counters)



OpenFlow: Switch to Controller Messages

Key switch-to-controller Messages:

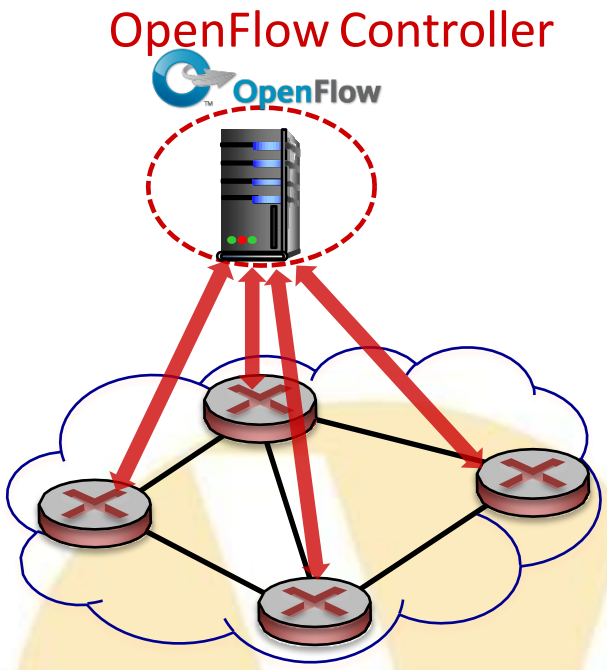
Packet-in: forward packet (and its control) to controller.

Flow-removed: flow table entry deleted at switch (based on a time-out)

Port status: inform controller of a change on a port (Port down for example)

Error Messages: from switch to controller.

Above messages are Asynchronous, i.e. they are sent without request from Controller



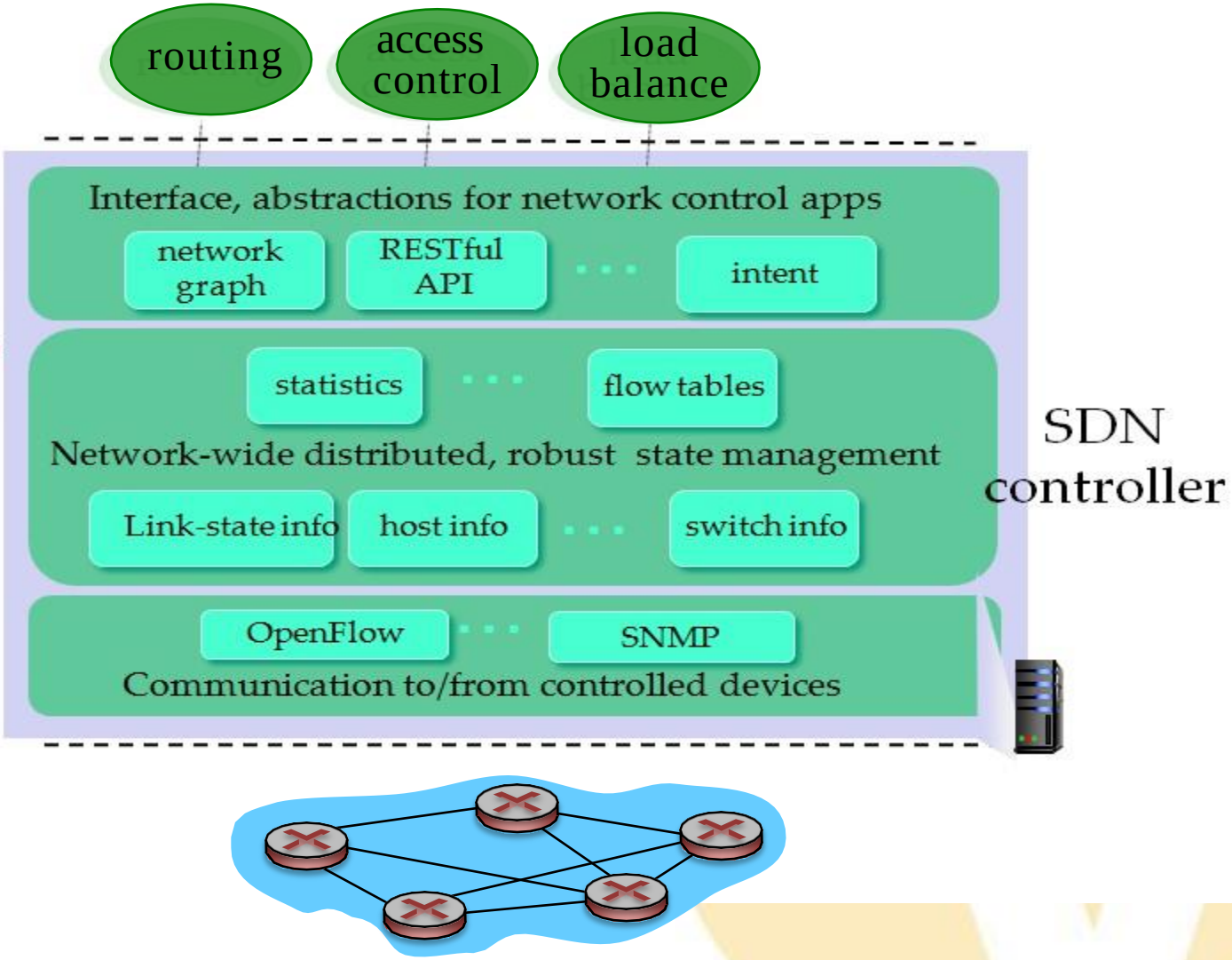
Network operators don't "program" switches by creating/sending OpenFlow messages directly. They use higher-level abstraction at Controller

Components of SDN Controller

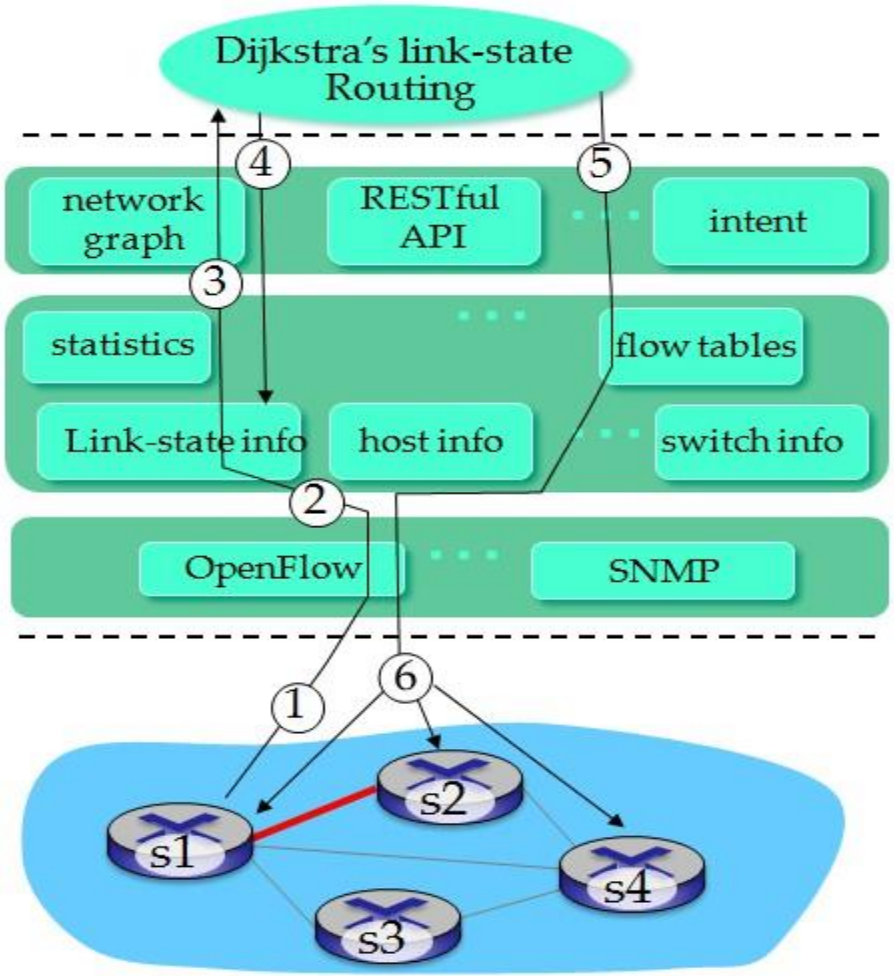
Interface layer to network control apps: abstractions API

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

Communication layer: communicate between SDN controller and controlled switches



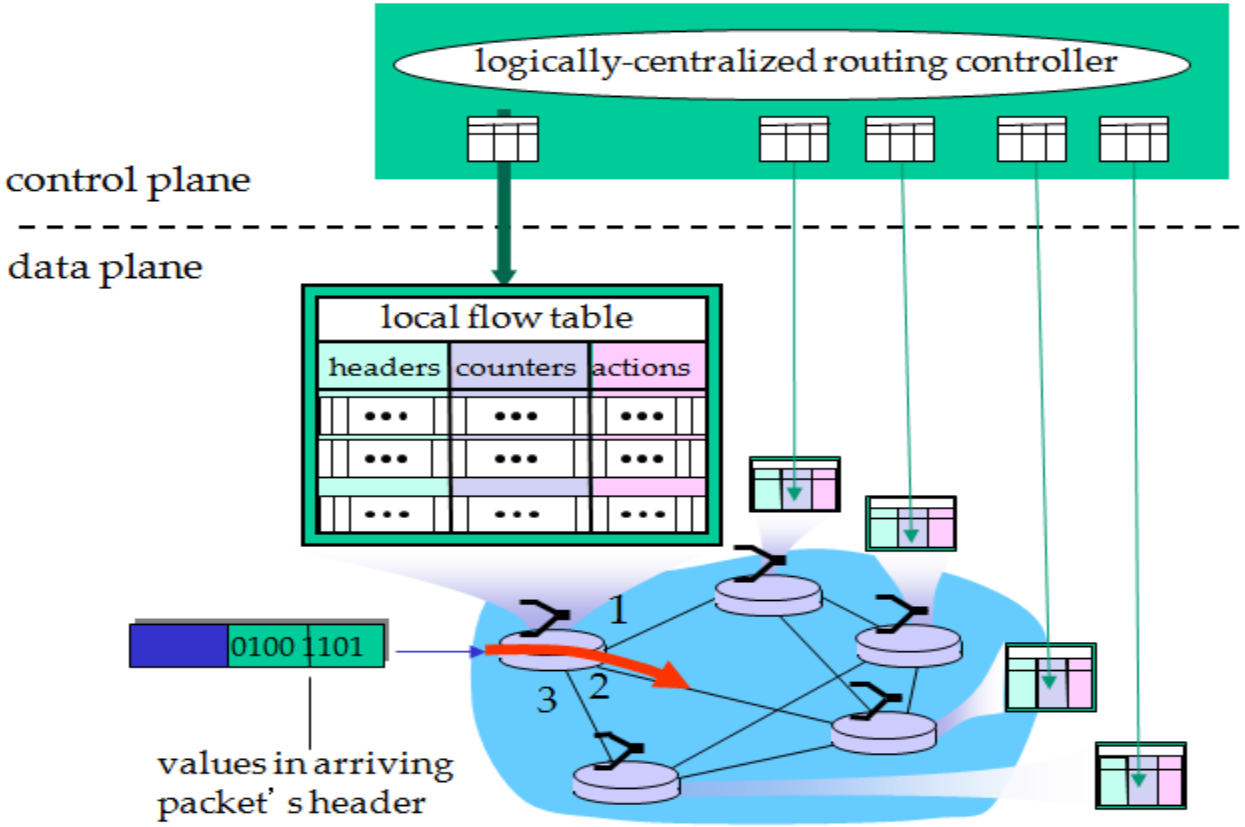
Example: SDN Controller-Data Plane Interaction



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
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

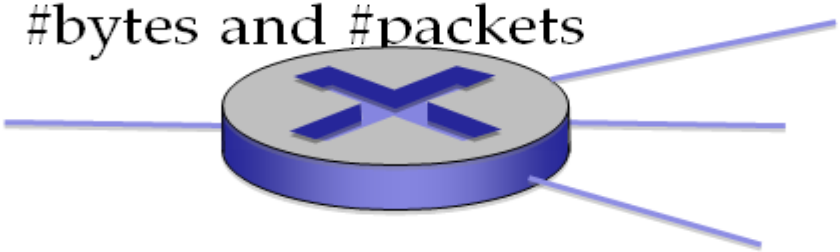
Generalized Forwarding & SDN

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



Open Flow Data Plane Abstraction

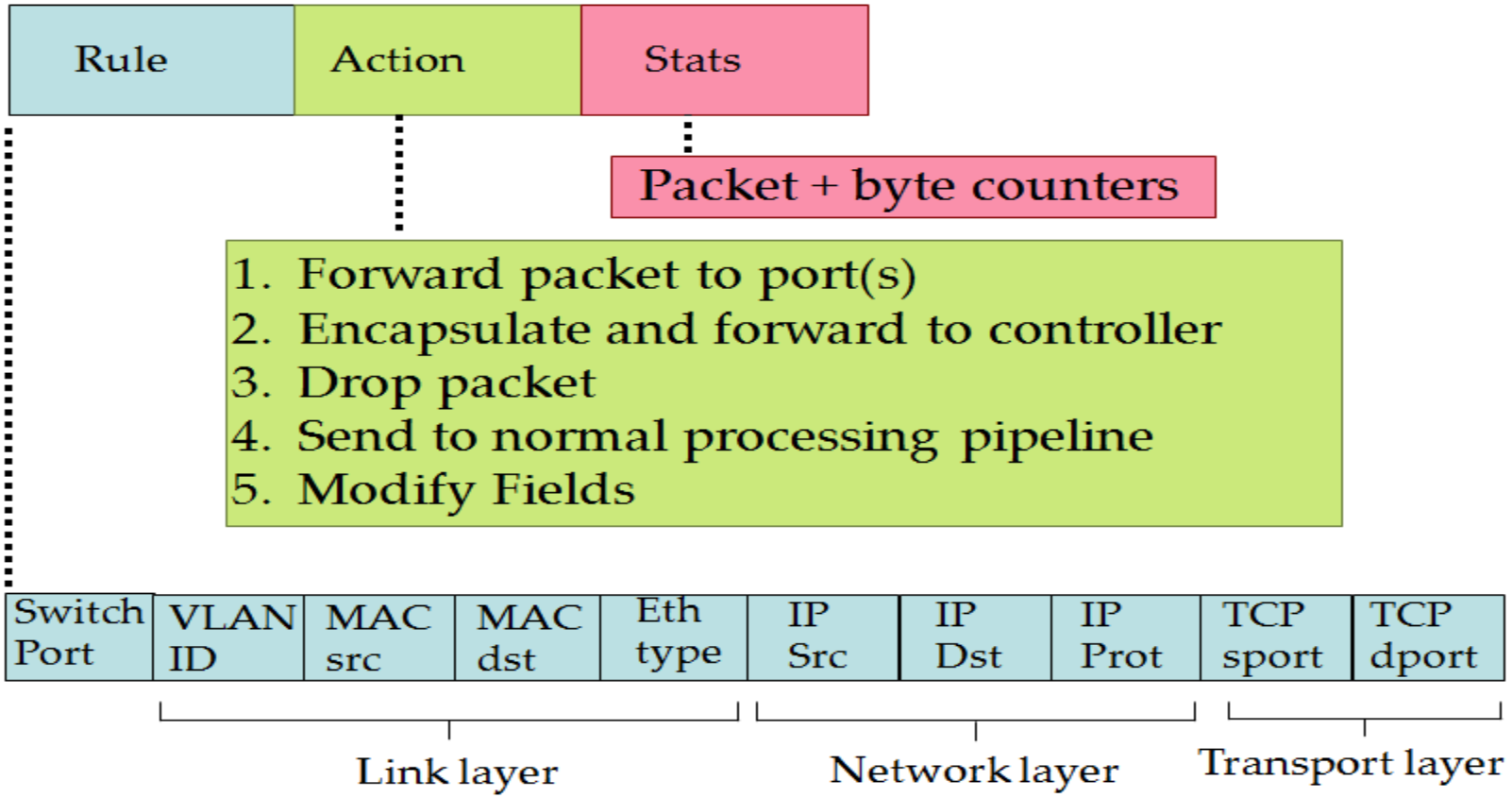
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



* : wildcard

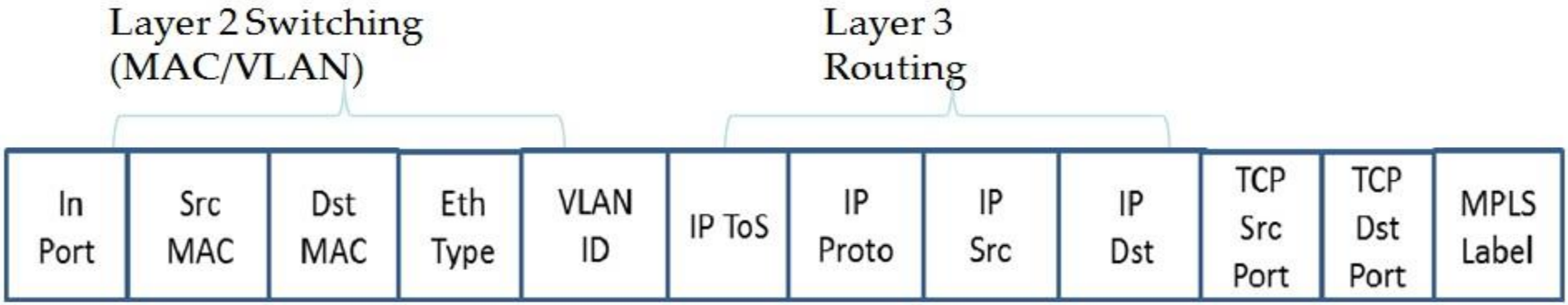
1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

Open Flow: Flow Tables Entries (example)



Packets are matched against flow entries based on prioritization.
An entry that specifies an exact match (i.e., it has no wildcards) is always the highest priority.
All wildcard entries have a priority associated with them.
Higher priority entries must match before lower priority ones.
If multiple entries have the same priority, the switch is free to choose any ordering.

Another Example of an Open Flow Table



Fields to match against flows

Wild Card Filters

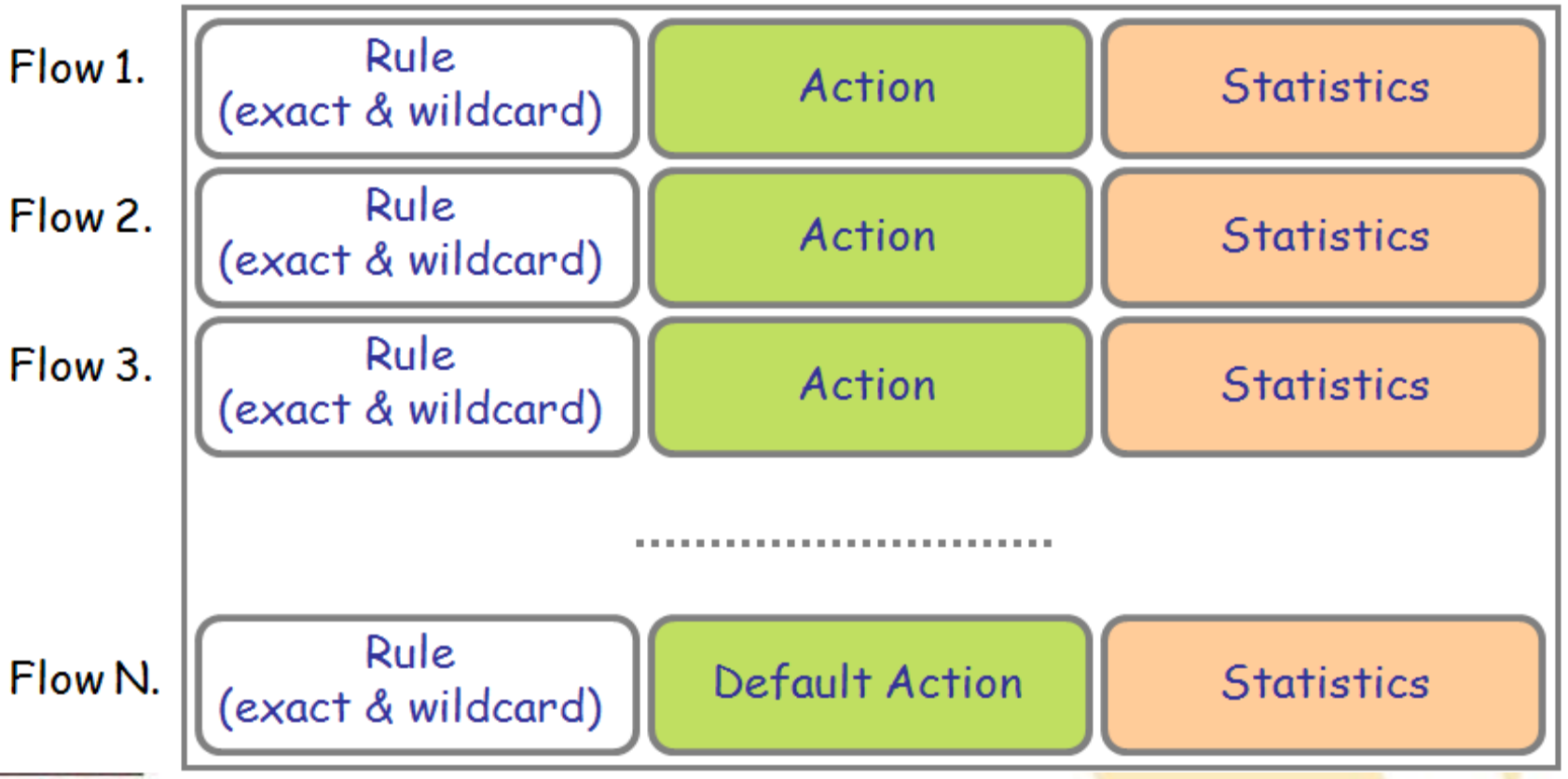
- IN Port
- VLAN ID
- VLAN Priority
- Ether Frame Type
- IP Type of Service
- IP Protocol
- TCP/UDP Src Port
- TCP/UDP Dst Port
- VLAN Priority
- MPLS Label
- IP Type of Service
- IP Src Address

Wild Card Matching:

- Aggregated MAC-subnet: MAC-src: A.*, MAC-dst: B.*
- Aggregated IP-subnet: IP-src: 192.168.*/24, IP-dst: 200.12.*/24

Open Flow Basics

Exploit the flow table in switches, routers, and chipsets



Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	Port 6

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6

Open Flow Abstraction (examples)

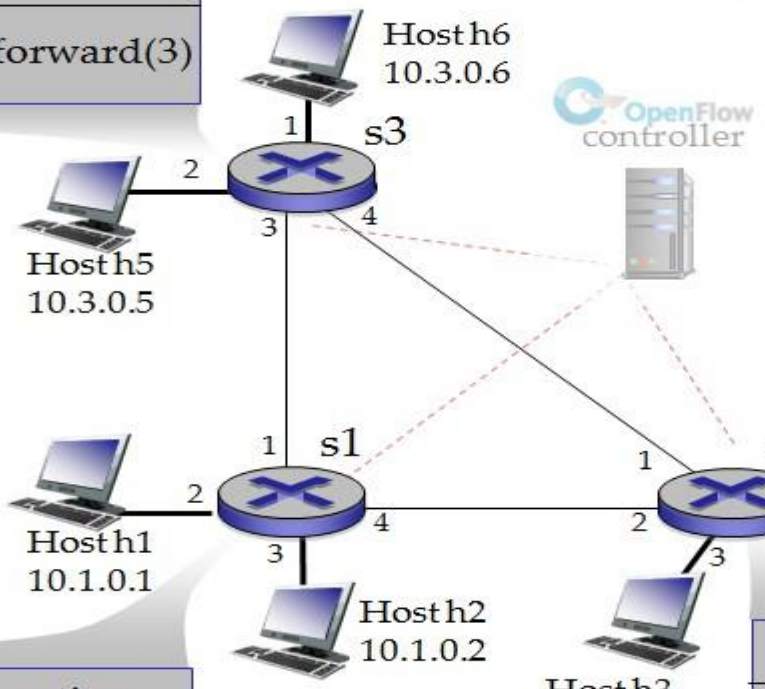
- match+action: unifies different kinds of devices

- Router
  - match: longest destination IP prefix
  - action: forward out a link
- Switch
  - match: destination MAC address
  - action: forward or flood
- Firewall
  - match: IP addresses and TCP/UDP port numbers
  - action: permit or deny
- NAT
  - match: IP address and port
  - action: rewrite address and port

Open Flow Example

Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Cloud Computing Definition

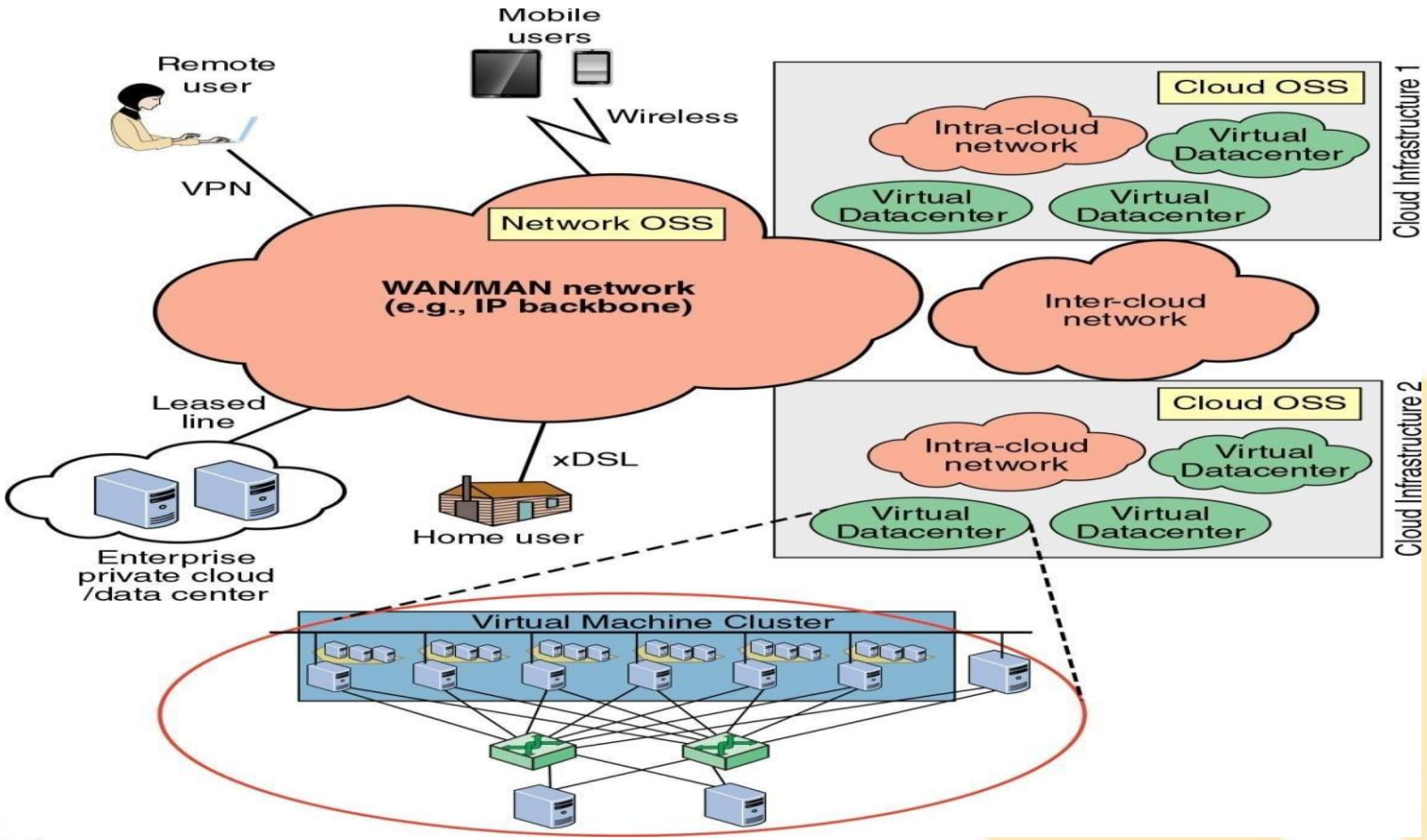
- "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (for example, Networks, Servers, Storage, Applications, and Services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.“
  - These services are provided from large Data Centers (DCs)

Examples of Data Centers (> 100,000Servers)

Motivation for Cloud Computing

- Overcapacity at large corporate Data Centers
- Falling Cost of Storage
- Ubiquitous Broadband and Wireless Access Technologies
- Significant improvements in Networking Technologies

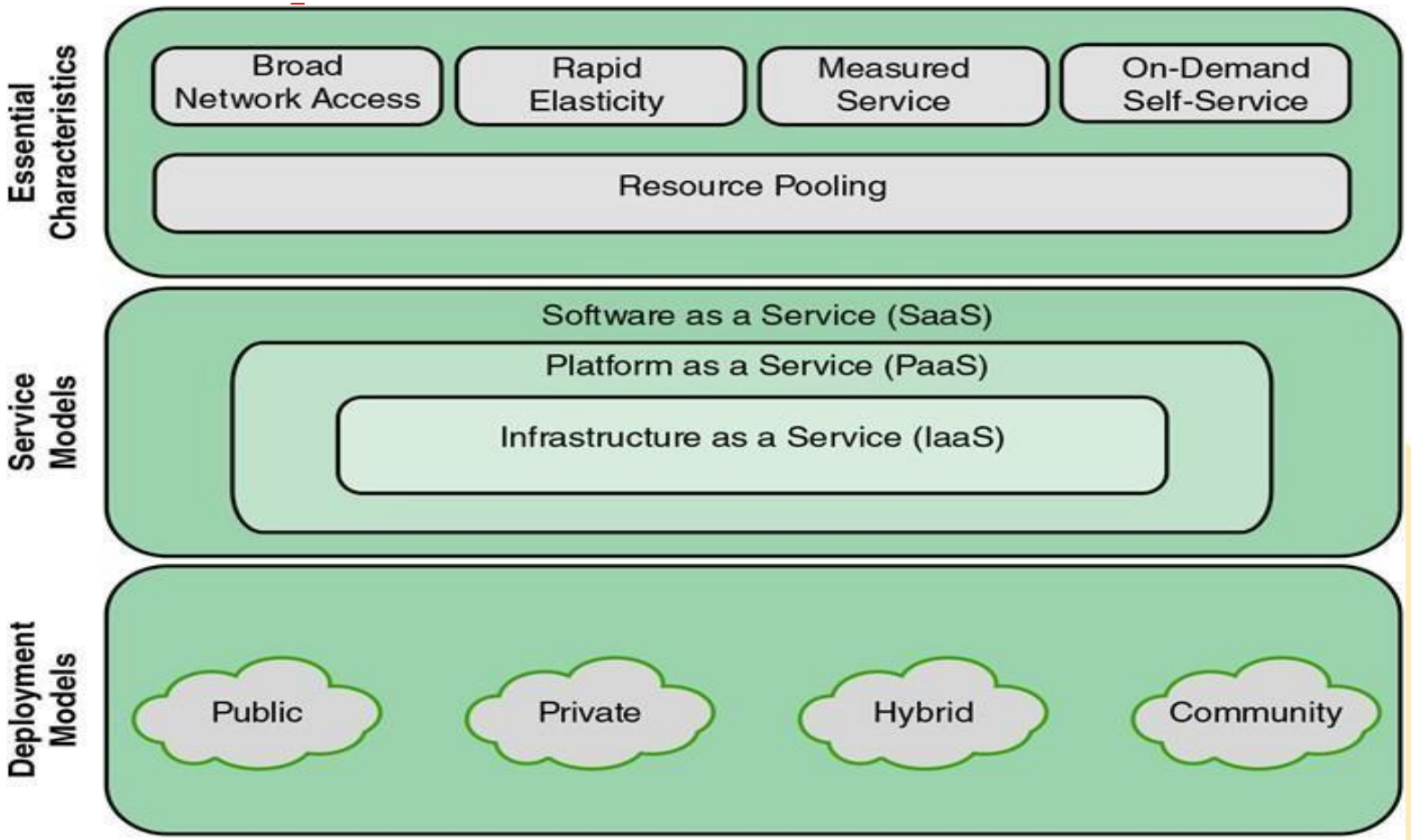
Cloud Networking Model



Cloud Computing Effects on Network

- The Network is a big part of cloud computing. A cloud user connects to the network to access the cloud resources.
- The cloud is accessible through a public network (Internet) or through a private network
  - Response-time guarantees depend upon this connectivity. Some cloud vendors offer dedicated links to their data centers and provide (and charge) appropriate SLAs for response time .
  - Others might implement a best-effort scheme but provide tools for monitoring and characterizing application performance and response time, so that users can plan their bandwidth needs.

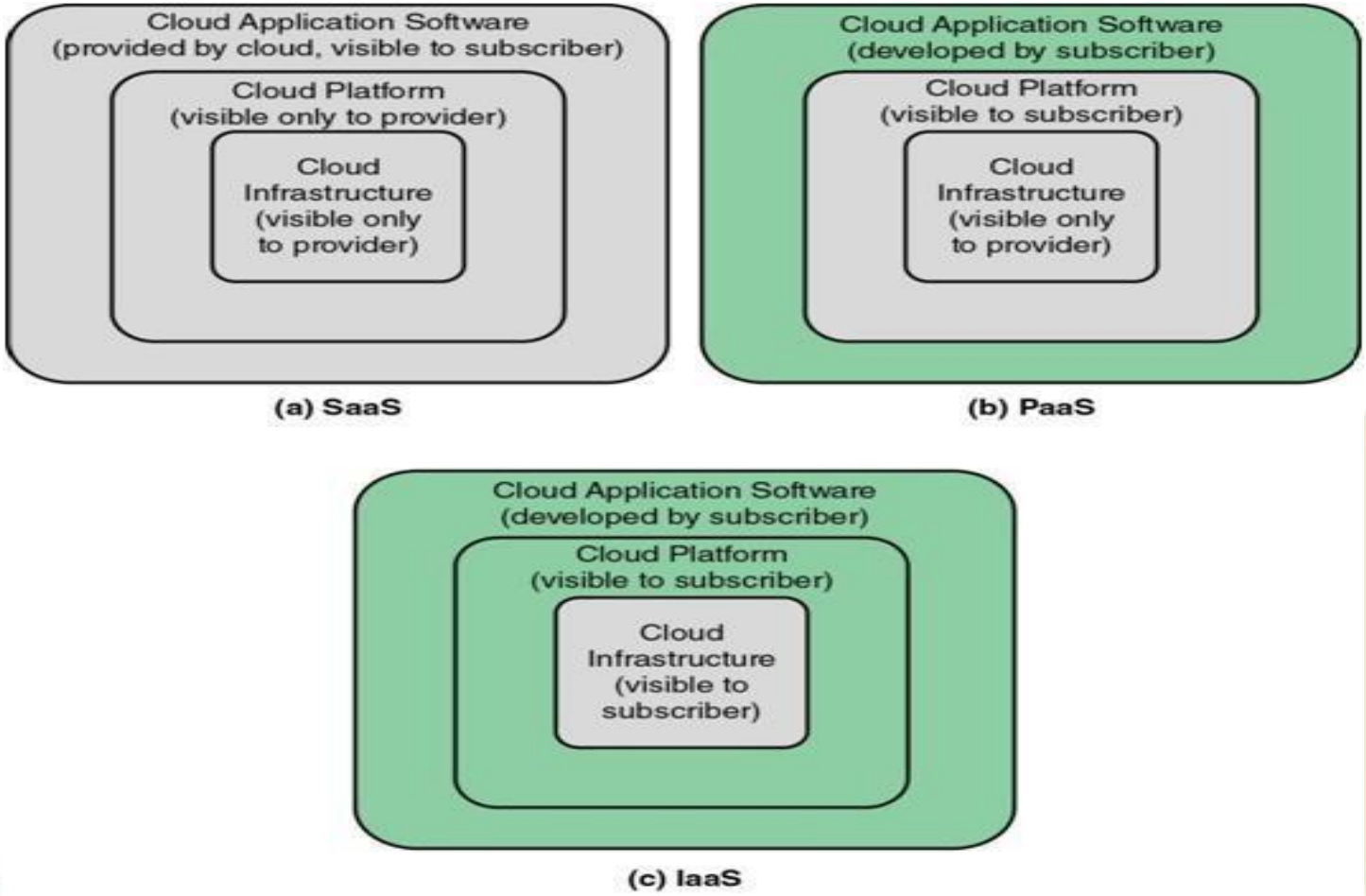
Cloud Computing



Cloud Computing Characteristics

- Scalability: Cloud computing gives you the ability to expand or reduce resources according to your specific service requirement.
- Pay-per-use: You pay for cloud services only when you use them. This is referred to as "Measured Service"
- On demand: With cloud services there is no need to have dedicated resources waiting to be used, as is the case with internal services.
- Resiliency: The resiliency of a cloud service offering can completely isolate the failure of server and storage resources from cloud users. Work maybe migrated to a different physical resource in the cloud with or without user awareness and intervention.
- Multitenancy: Public cloud services providers often can host the cloud services for multiple users (Resource Pooling) within the same infrastructure. Server and storage isolation may be physical or virtual—depending upon the specific user requirements.

Cloud Service Model



Cloud Computing Service Models

- **Software as a Service (SaaS)**
  - Instead of obtaining desktop and server licenses for software products it uses, an enterprise can obtain the same functions through a hosted service from a provider through a network connection. The interface to the software is usually through a web browser. An example is a web-based e-mail like Google mail.
  - SaaS saves the complexity of software installation, maintenance, upgrades, etc.. for the IT team within the enterprise. The software is now managed centrally at the SaaS provider's facilities.
  - SaaS provider can provide the service to multiple customers and enterprises, resulting in a multitenant model. Pricing is on a per-use basis, for fixed Bandwidth and Storage.
  - Example: Salesforce.com

Cloud Computing Service Models

- **Platform as a Service (PaaS)**
  - Provides a software platform on which users can build their own applications and host them on the PaaS provider's infrastructure.
  - Platform is used as a development framework to build, debug, and deploy applications.
  - Applications do not need to worry about the scalability of the underlying platform (hardware and software), the scalability is guaranteed transparently by the PaaS platform.
  - Customer does NOT manage the underlying Cloud Infrastructure, including Network, Servers, Storage or OS.
  - Examples: Google (App-Engine) or Force.com (the PaaS offering from Salesforce.com), Microsoft Azure, etc…
    - Require the applications to follow their own API and be written in a specific language.
    - Pricing for PaaS can be on a per-application developer license

Cloud Computing Service Models

- **Infrastructure as a Service (IaaS)**
  - IaaS provider offers you "raw" computing, storage, and network infrastructure. You can load your own software (OS & Applications) onto this infrastructure.
  - IaaS offers the greatest degree of control of the three models. You need to know the resource requirements for your specific application to exploit IaaS well. Scaling is your (not the provider‘s) responsibility.
  - Example: Amazon (EC2 Service: AWS and S3). Amazon lets you rent servers with a certain CPU speed, memory, and disk capacity along with the OS and applications that you need to have installed on them
    - Amazon uses virtualization service. You actually can get a VM when you ask for a specific machine configuration, though VMs are not a prerequisite for IaaS.
  - Pricing for the IaaS can be on a usage or subscription basis.

Public vs. Private Clouds

- Public Cloud Service providers are those whose data centers are external to the users (Businesses or Individuals) of the service
  - The infrastructure and control of these clouds is with the service provider.
  - Resources are available (dynamically) on demand and accessed through web applications and API
- Private Cloud Providers are responsible only for the infrastructure and not for the control.
  - Similar to a section of a shared data center being partitioned for use by a specific customer.
  - A Private Cloud Provider can offer SaaS, PaaS, or IaaS services, though IaaS might appear to be a more natural fit.

Public vs. Private Clouds

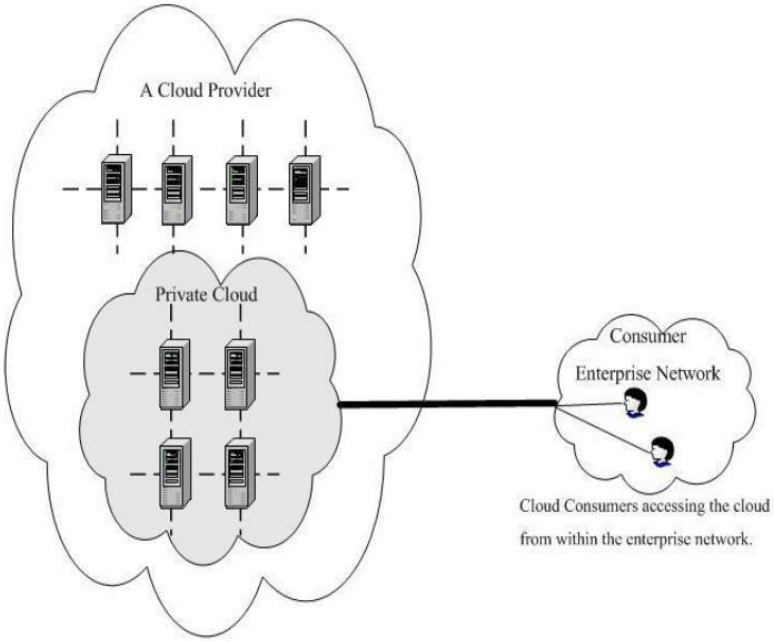


Figure 11: Out-sourced Private Cloud

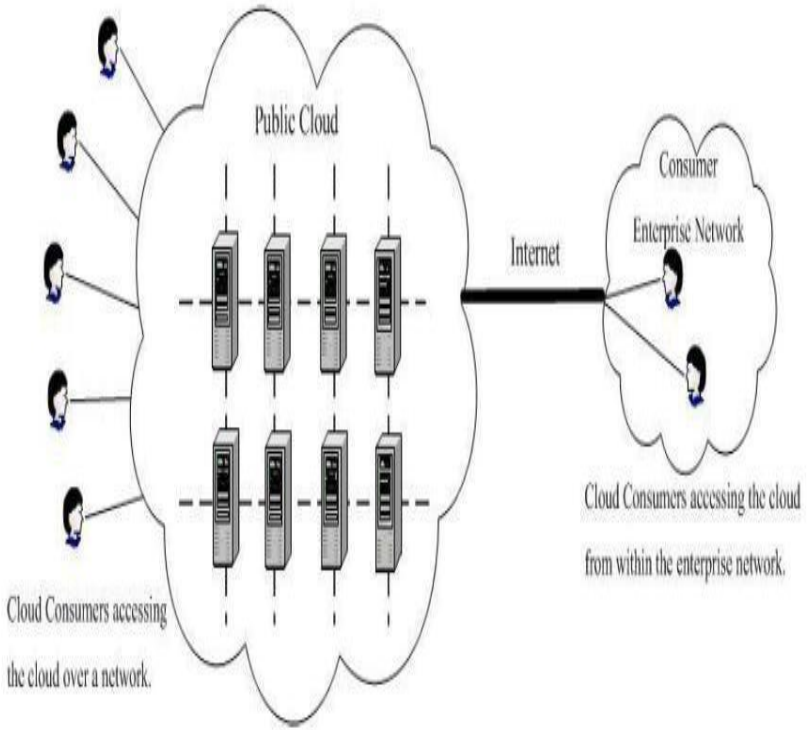


Figure 9: Public Cloud

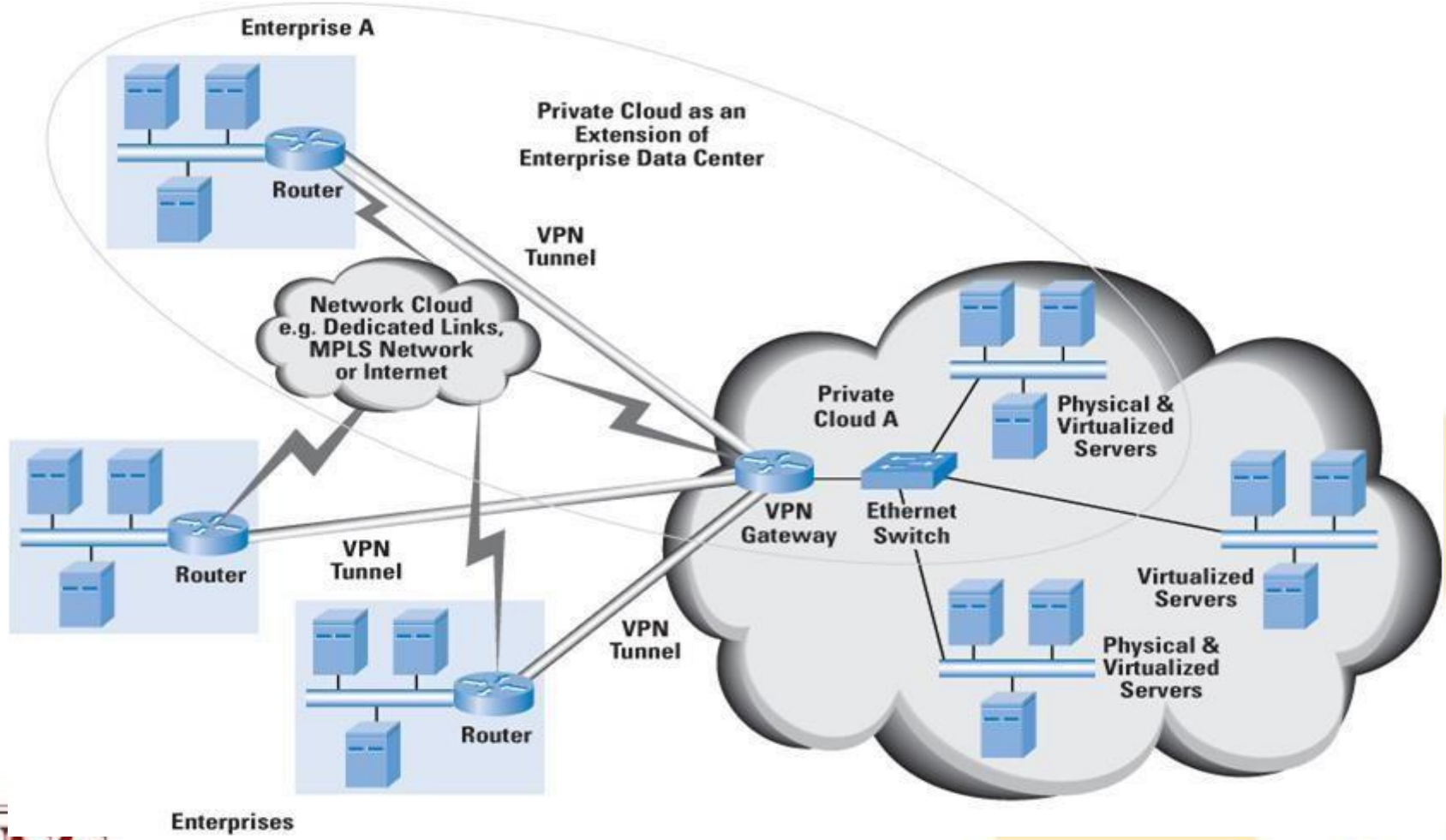
IaaS Private Cloud Example

- Consider an IaaS cloud to which an enterprise connects to augment its server capacity for a limited period of time.
- Assume that the enterprise uses a 10.x.x.x private addressing scheme for all its servers because they are internal to the enterprise. It would be ideal if the additional servers provided by the IaaS cloud were part of the same addressing scheme (the 10.x.x.x scheme).
- IaaS cloud service provider can partitioned a portion of its public cloud to realize a private cloud for that enterprise . The private cloud is reachable as a LAN extension to the servers in enterprise A's data center.
- How is that done?

IaaS Private Cloud Example

- A secure Virtual Private Network (VPN) tunnel is first established between the enterprise data center and the public cloud.
- This tunnel uses public IP addresses to establish the site-to-site VPN connection. The VPN gateway on the cloud service provider side uses multiple contexts—each context corresponding to a specific private cloud.
- Traffic from enterprise A is decrypted and forwarded over to an Ethernet switch to the private cloud for enterprise A.
- A server on enterprise A's internal data center sees a server on private cloud A to be on the same network.

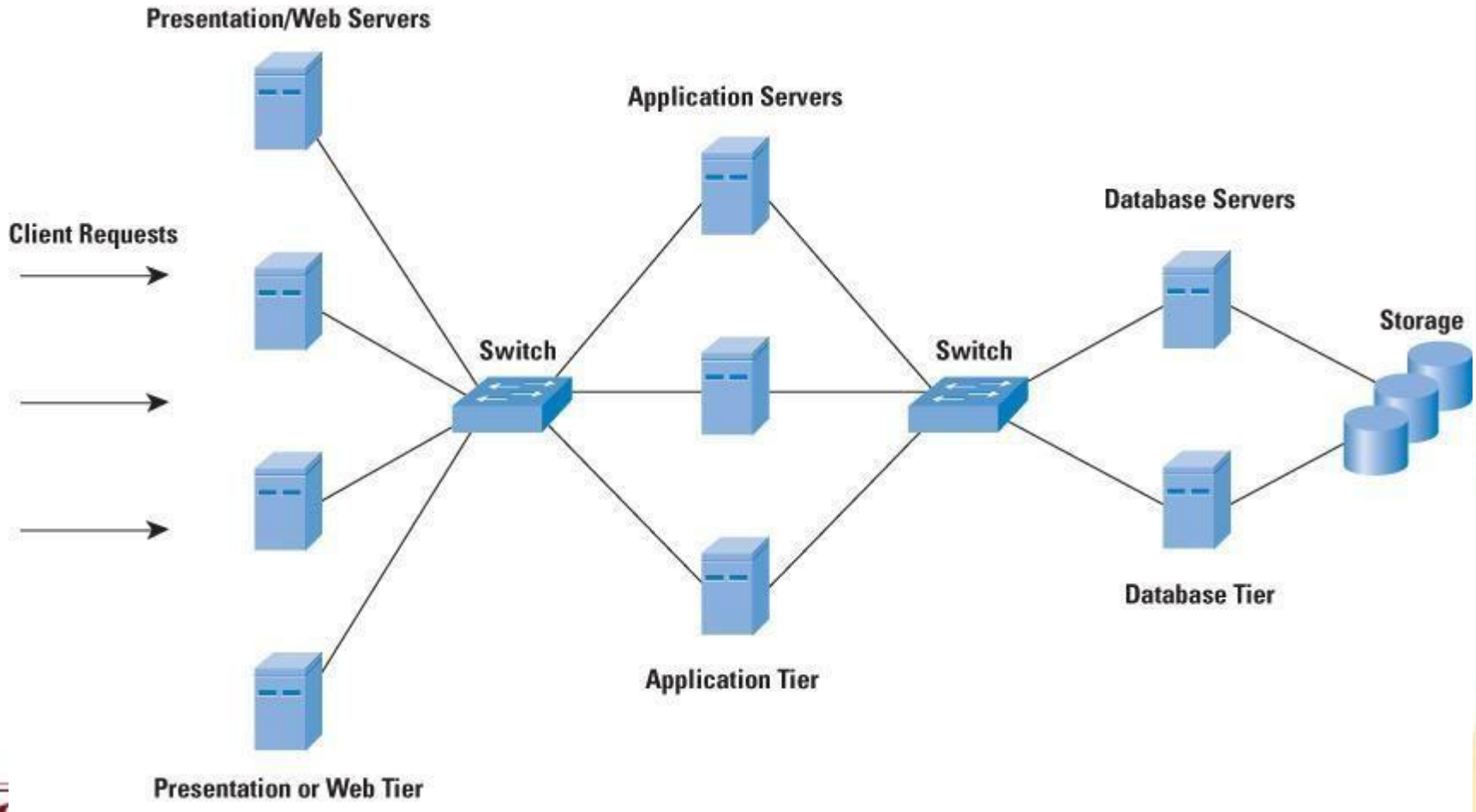
IaaS Private Cloud



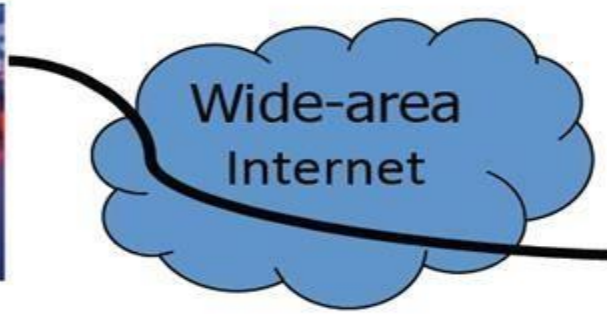
Cloud Network Functional Perspective

- From a functional perspective, data-center server organization has often adopted a three-tier architecture. The three-tier functional architecture has a web or Presentation Tier on the front end, an Application Tier to perform the application and business-processing logic, and finally a Database Tier (to run the database management system), which is accessed by the Application Tier for its tasks

The 3-Tier Functional Architecture

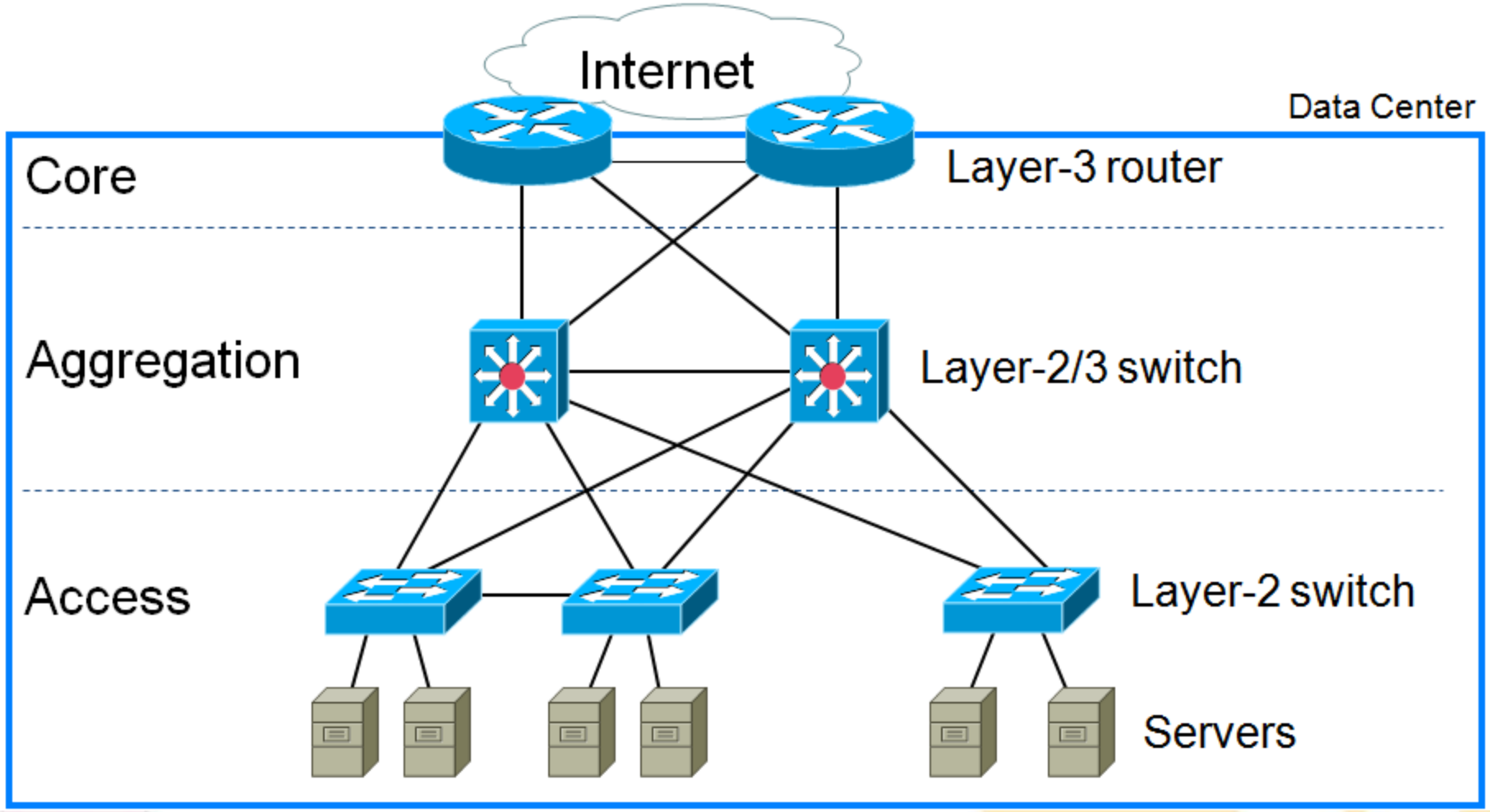


Front-End Type of Applications



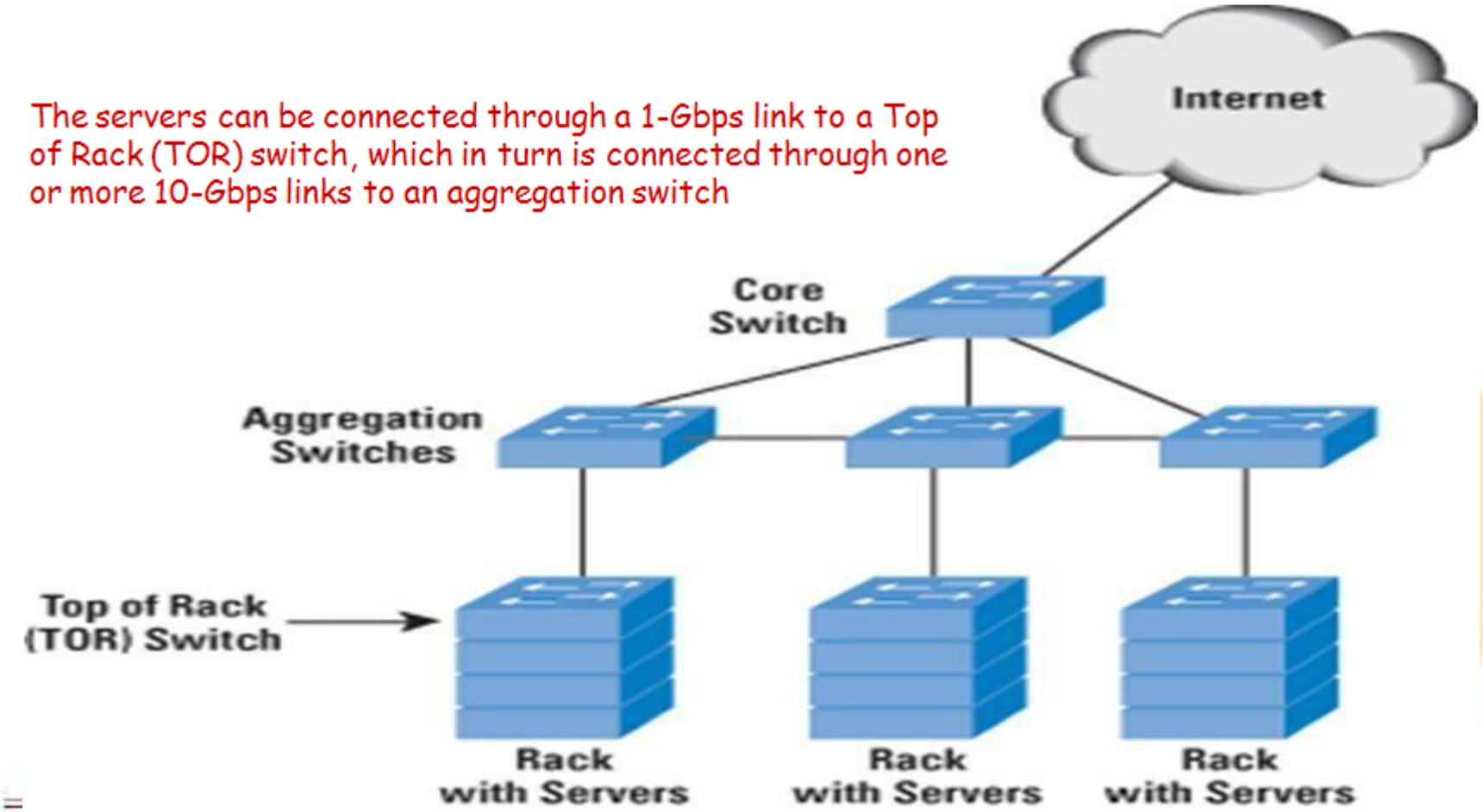
- Data sizes driven by the content that users actually consume
  - Growth largely due to higher bitrate content (IP TV/movies, iPhone Facetime)
- Mobile Internet source of new users
- Often constrained by the "last mile"

Data Center Network Topology



Data Center Switching Network Architecture

The servers can be connected through a 1-Gbps link to a Top of Rack (TOR) switch, which in turn is connected through one or more 10-Gbps links to an aggregation switch

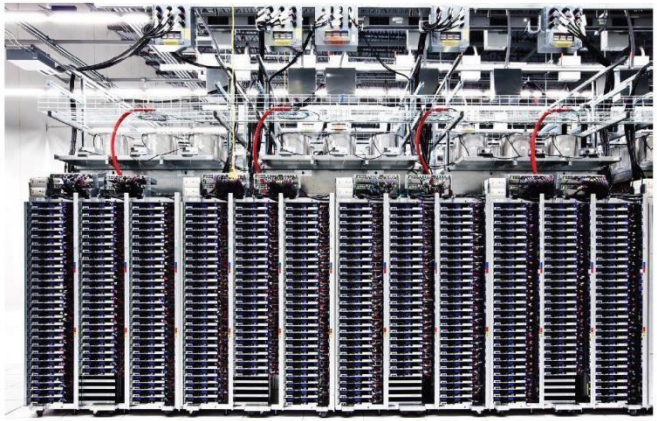


Top of Rack Architecture

- Rack of servers
  - Commodity servers
  - And top-of-rack switch
- Modular design
  - Preconfigured racks
  - Power, network, and storage cabling
- Aggregate to the next level

A rack has 20-40 servers.

The servers in a rack connect to a 48-port "top of rack" (ToR) switch.



Google Rack of Servers

"Top of Rack" switch

Aggregation/Core Switches

- Aggregation switch connects to other aggregation switches and through these switches to other servers in the data center.
- A core switch connects to the various aggregation switches and provides connectivity to the outside world, typically through Layer 3(IP).
- Most of intra-data center traffic traverses only the TOR and the aggregation switches. Hence the links between these switches and the bandwidth of those links need to account for the traffic patterns.
  - Note: The presence of virtualized servers adds an extra dimension. Network connections to physical servers will need to involve "fatter pipes" because traffic for multiple VMs will be Multiplexed onto the same physical Ethernet connection.

Basic Building Blocks of a DC