بسمه تعالی

School of Computer Engineering

Advanced Computer Networks

Network Softwarization

Seyed Hamed Rastegar

Fall 1401

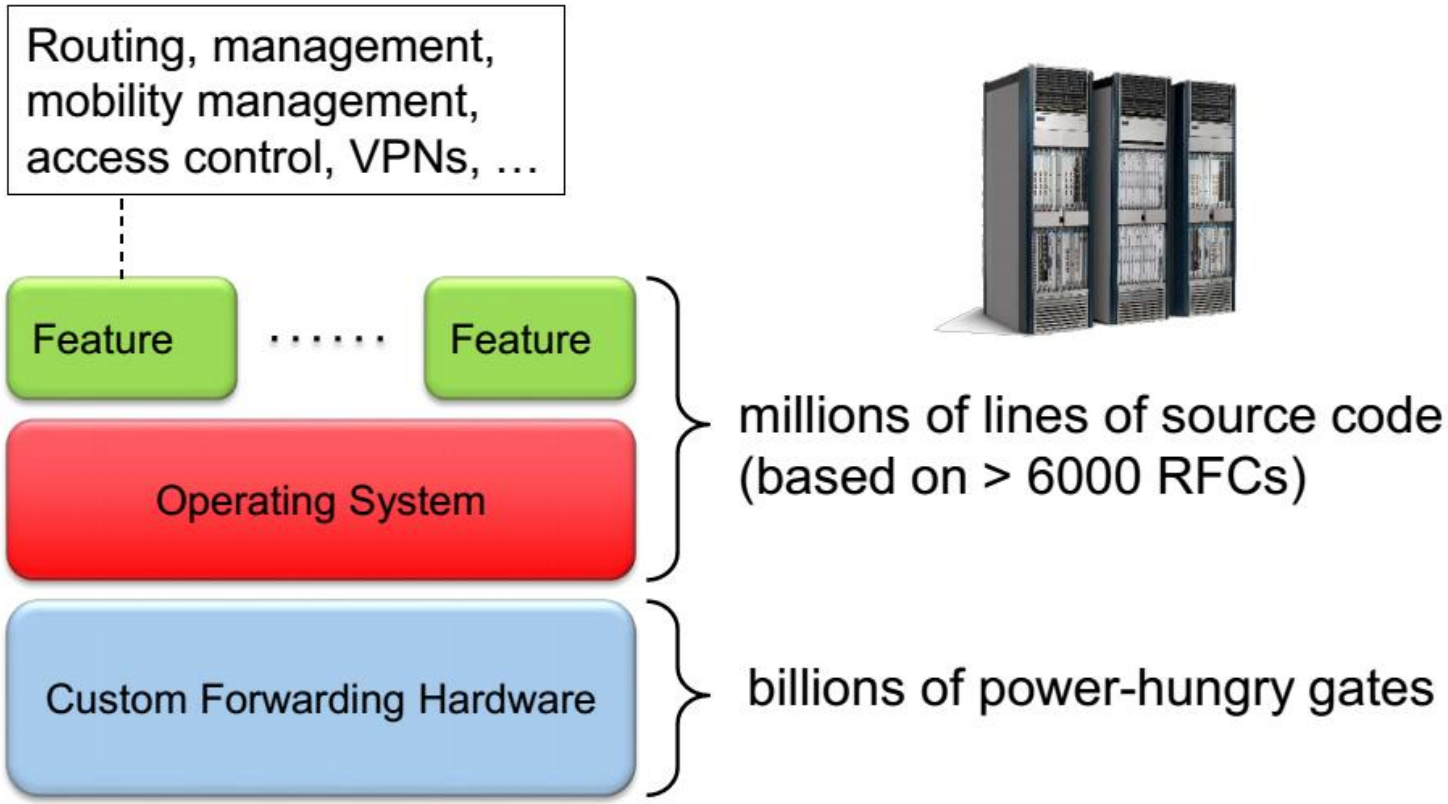
Contents

- Software Defined Networking
  - Review of Traditional Networks
  - Decoupling Software from Hardware Concept
  - Software Defined Networking Approach
  - SDN Architecture and Components
  - SDN Applications
- Network Function Virtualization

Traditional Network Equipment

- Equipment used in today's networks are vertically integrated.
- Vendor produce closed "Box"s with specific features.
- Hardware, OS and the applications all are developed by a unique entity.
- Improvement is restricted to that vendor and very complex.

Traditional Network Equipment



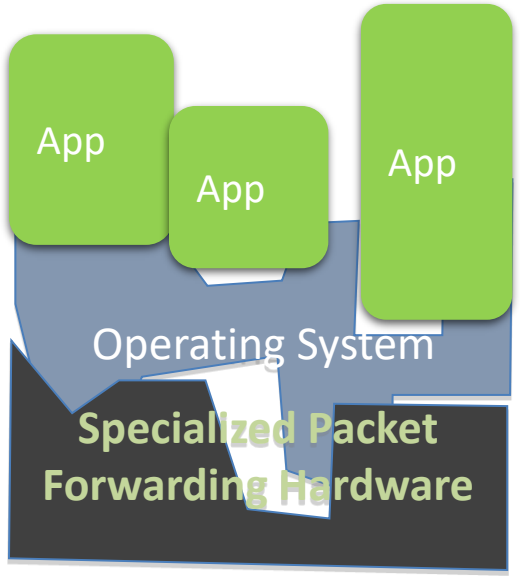
- extremely complex
- mainframe mentality
- very expensive

Disadvantages

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.



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

Buggy software in the equipment
- Routers with 20+ million lines of code
- Cascading failures, vulnerabilities, etc.

Solution?

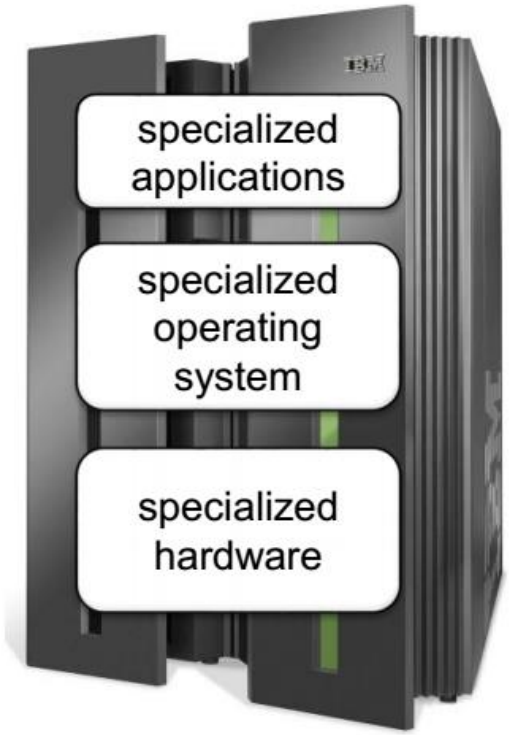
- These disadvantages caused researchers trying find better solution.
- In this path, several developments helped them:

✓ Shifting from mainframe mentality in computers.

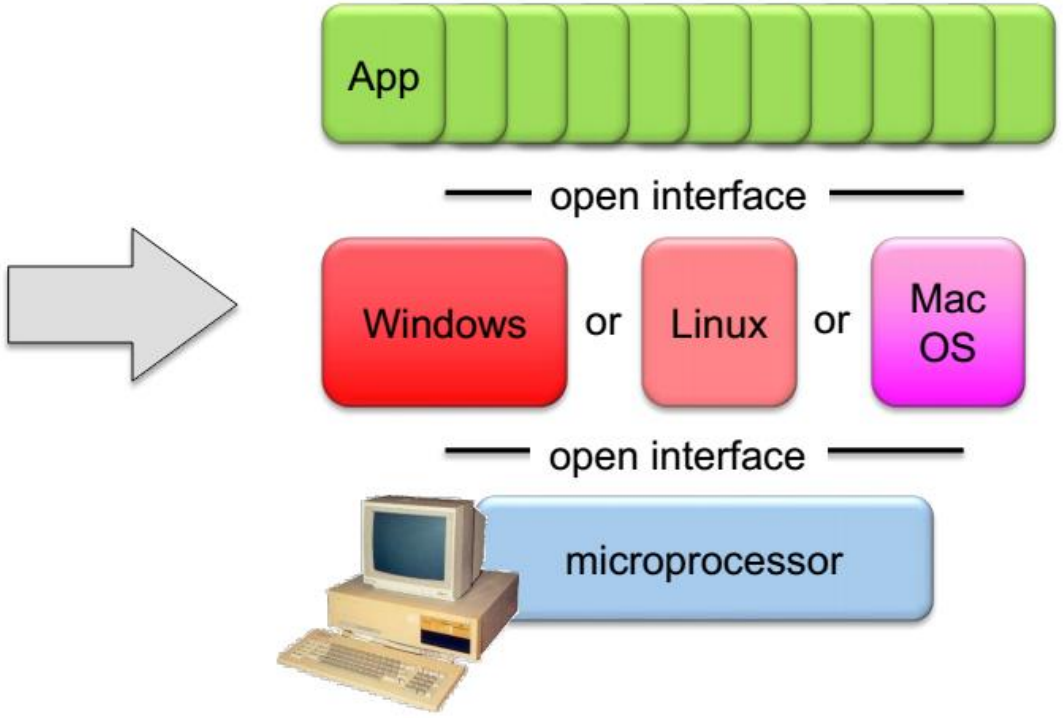
✓ Success of Large software companies e.g. Google, Facebook.

✓ Introduction of Software Defined Radio in physical communication.

History of Computers: Mainframe Mentality



- vertically integrated
- closed, proprietary
- slow innovation
- small industry



- horizontal
- open interfaces
- rapid innovation
- huge industry

Software Eng. Success:

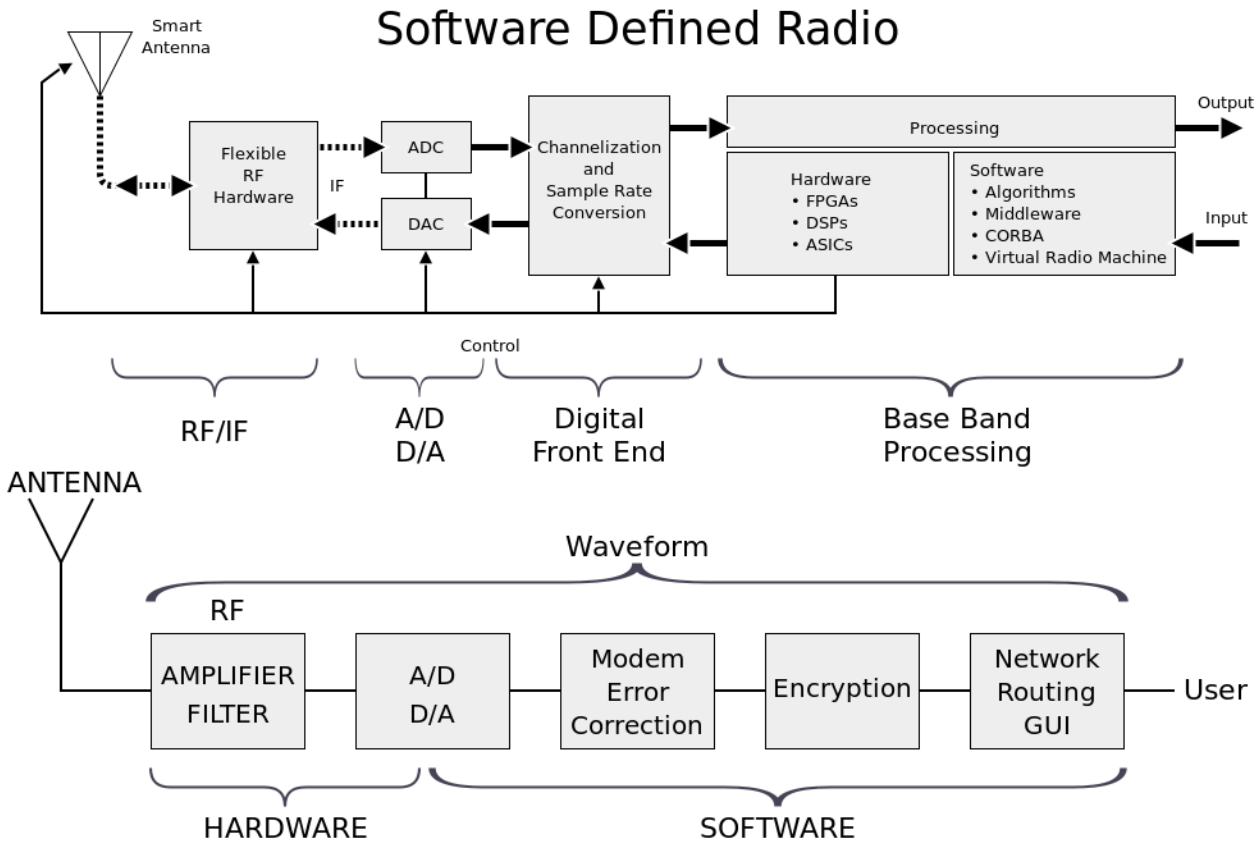
- Evolution of software companies and their success.
- Successful lessons and experience form open source operating systems, e.g. Linux, Android.
- Lower cost of software development than hardware implementation.
- Companies prefer to extend their software departments.

History of PHY Communication: Software Defined Radio

- Software-defined radio (SDR) is a radio communication system where components that have been typically implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded svstem.

- USRP® Networked, Bus, and X Series SDRs

History of PHY Communication: Software Defined Radio

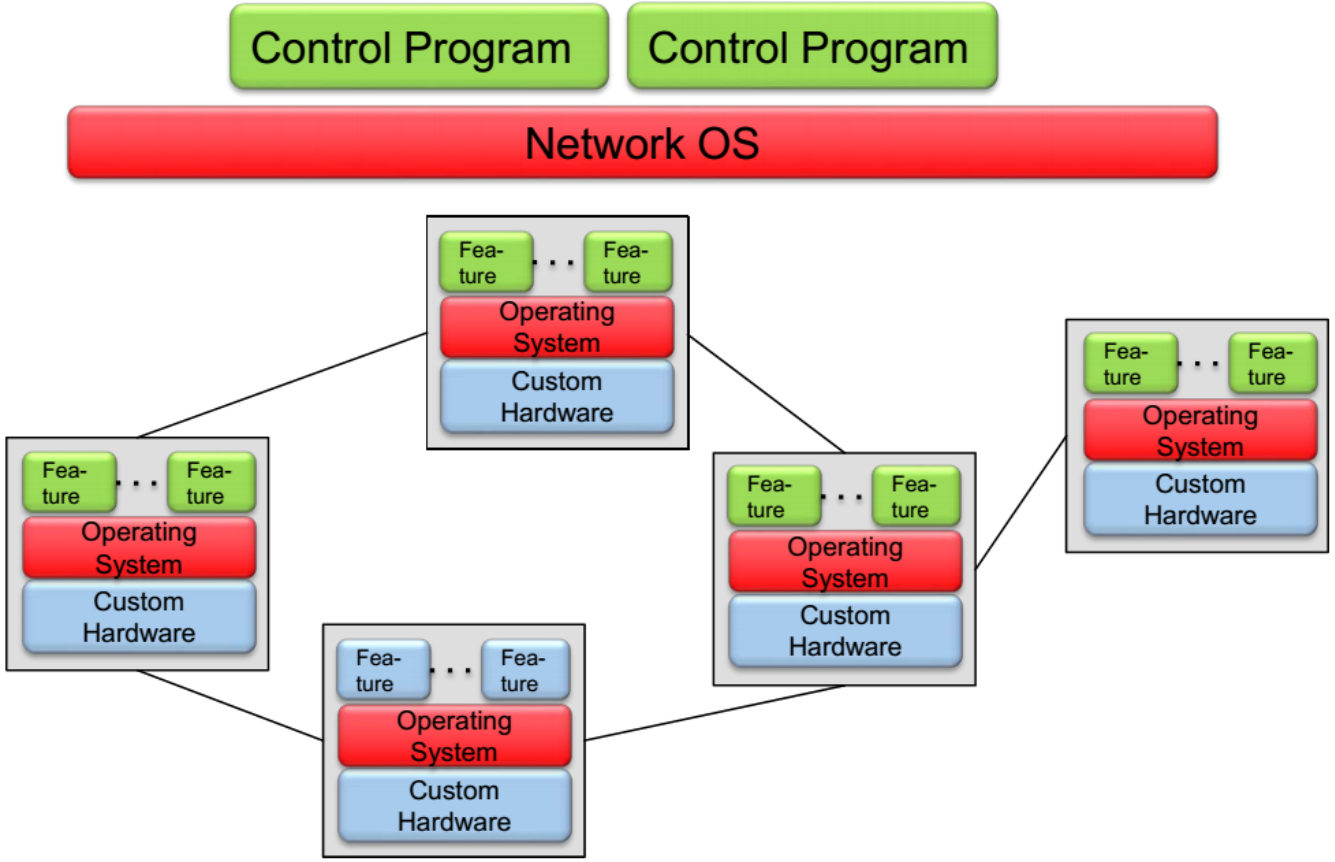


The Solution:

Concentrate more on Software, using low level Hardware

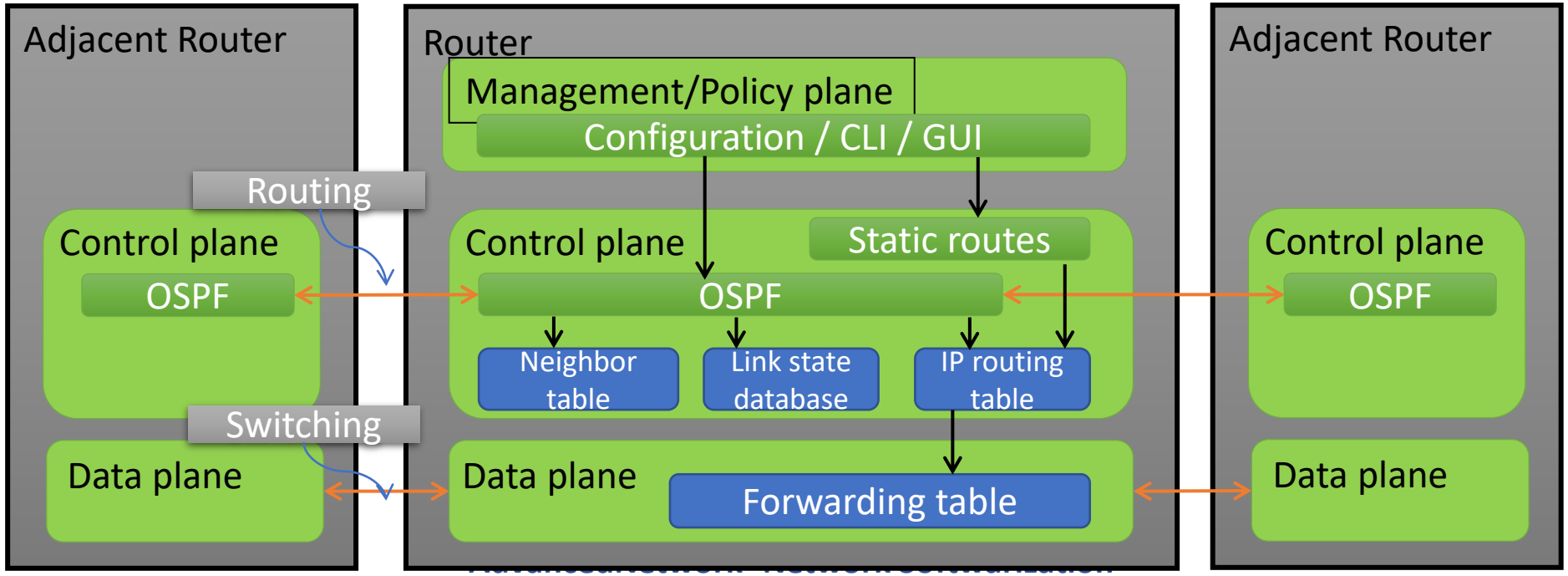
Restructuring Networks

- Current structure of a network is as follows



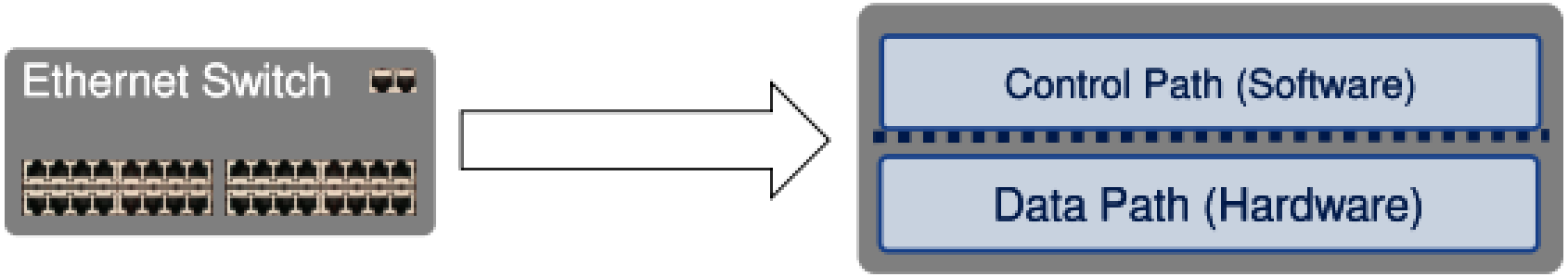
Detailed view of a Traditional Router

- Router can be partitioned into control and data plane
  - Management plane/ configuration
  - Control plane / Decision: OSPF (Open Shortest Path First)
  - Data plane / Forwarding



Traditional Router

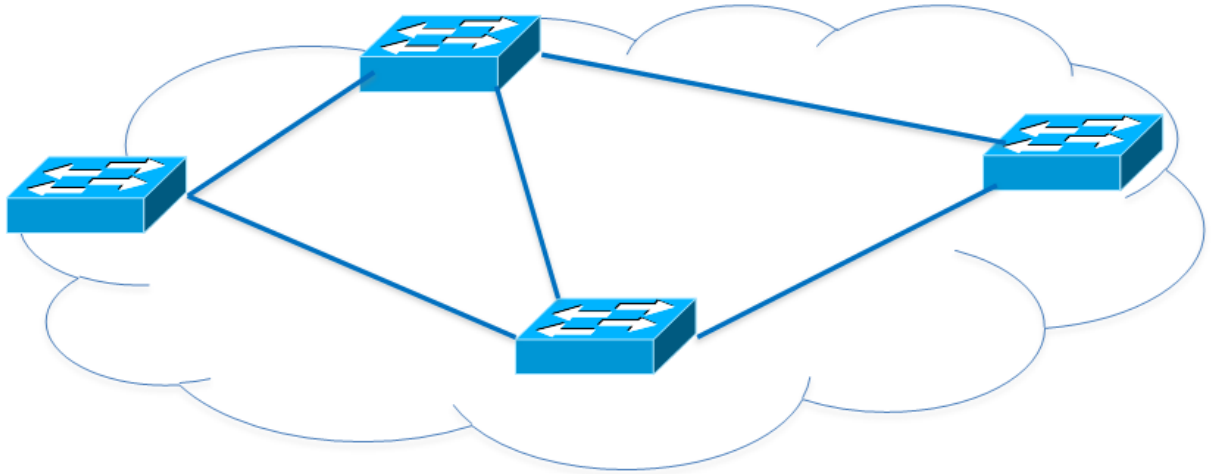
- Typical Networking Software
  - Management plane
  - Control Plane – The brain/decision maker – Implemented in Software - works in the large time-scale (second or even minute)
  - Data Plane – Packet forwarder -> Need High Speed -> Implemented in Hardware



Traditional Networks

- **Data Plane**

data plane:
packet handling

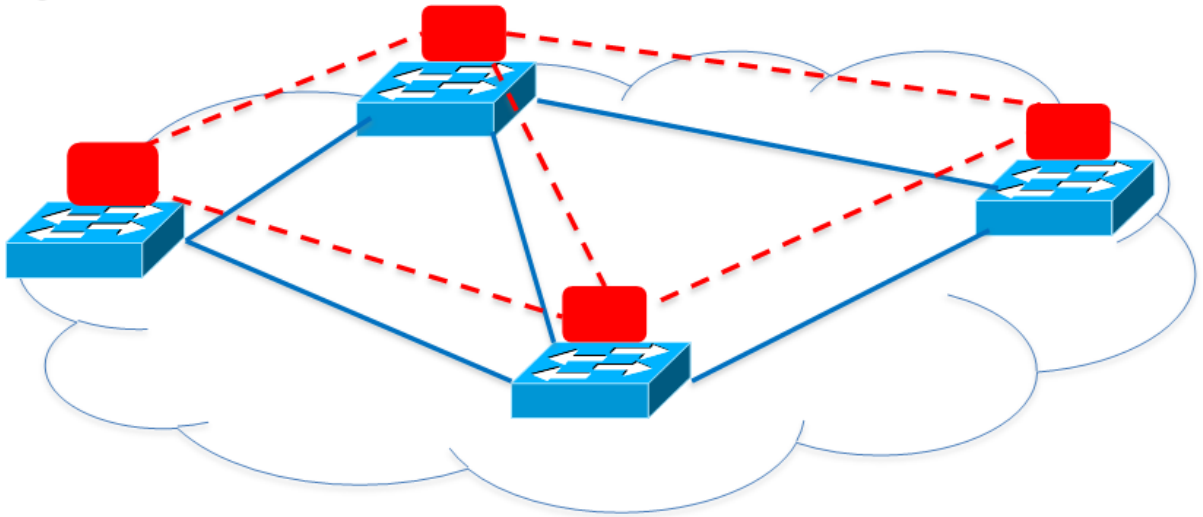


forward, filter, buffer, mark, rate-limit, measure packets

Traditional Networks

- **Control Plane**: Works based on algorithms that have been design to work individually on routers but arrive at a desired global result.

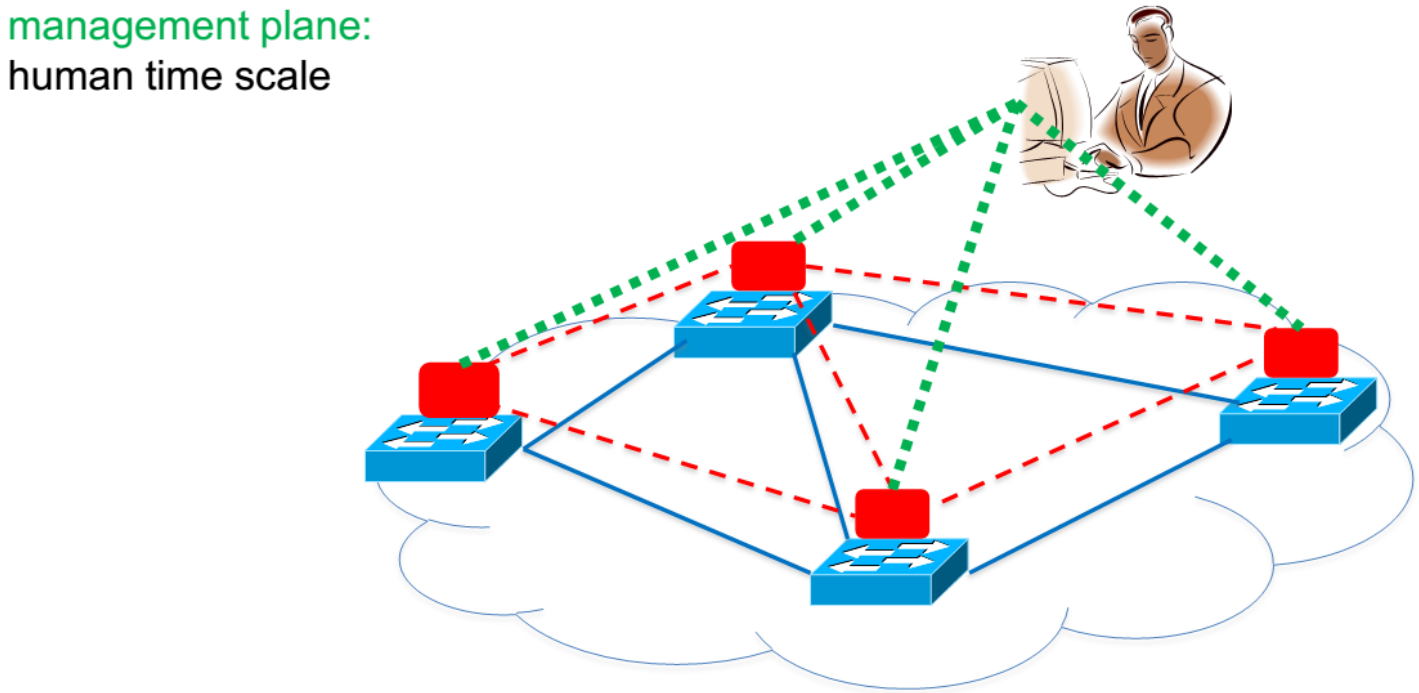
control plane:
distributed algorithms



track topology changes, compute routes, install forwarding rules

Traditional Networks

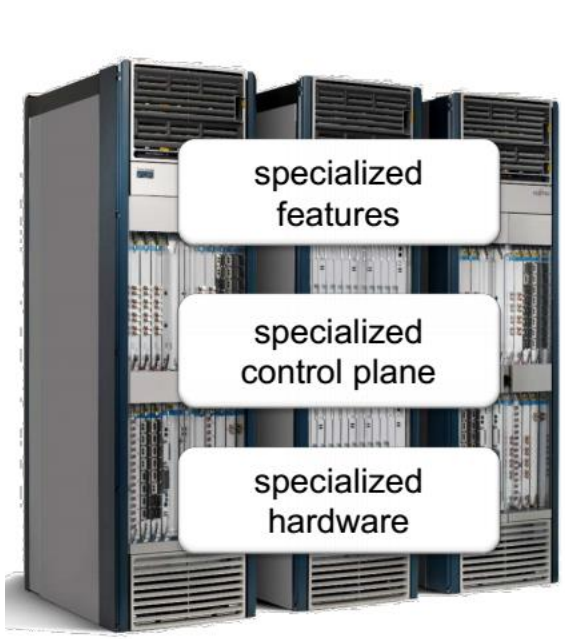
- **Management Plane**: Vendor-based operational service and software updating. Not open to innovate and define arbitrary features.

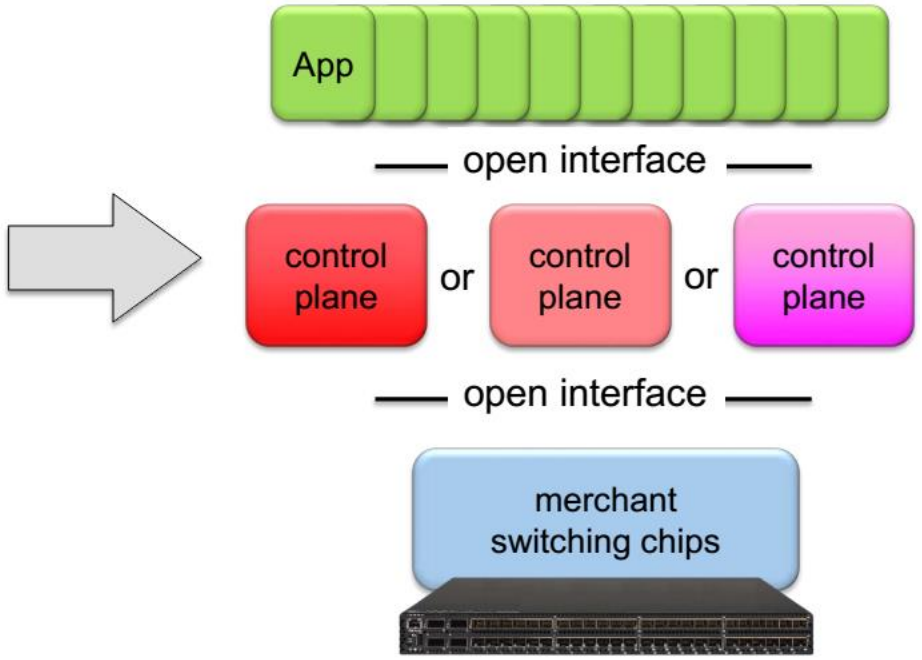


Using the Concept in Networking

Decoupled Planes

- Convert closed “Box” to a simple hardware/ transfer the Brain to Network Controller.

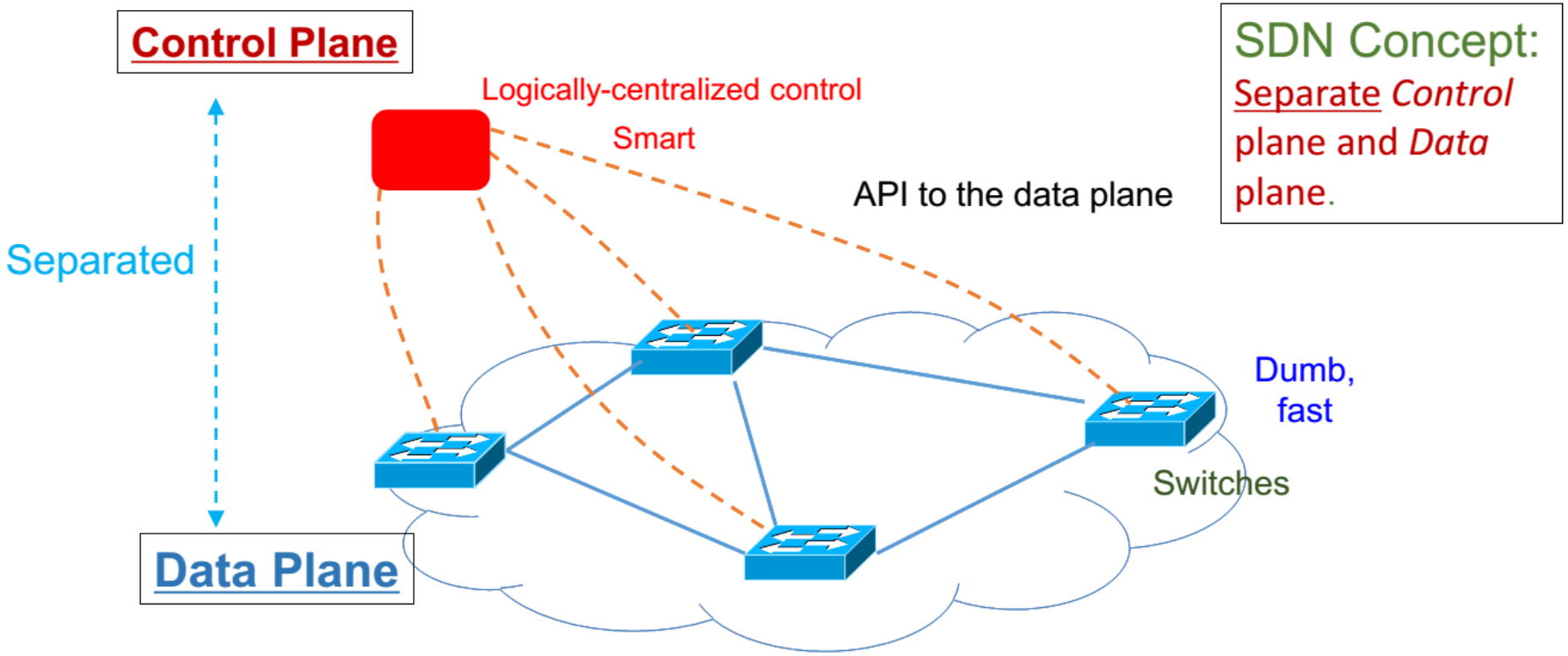




- vertically integrated
- closed, proprietary
- slow innovation

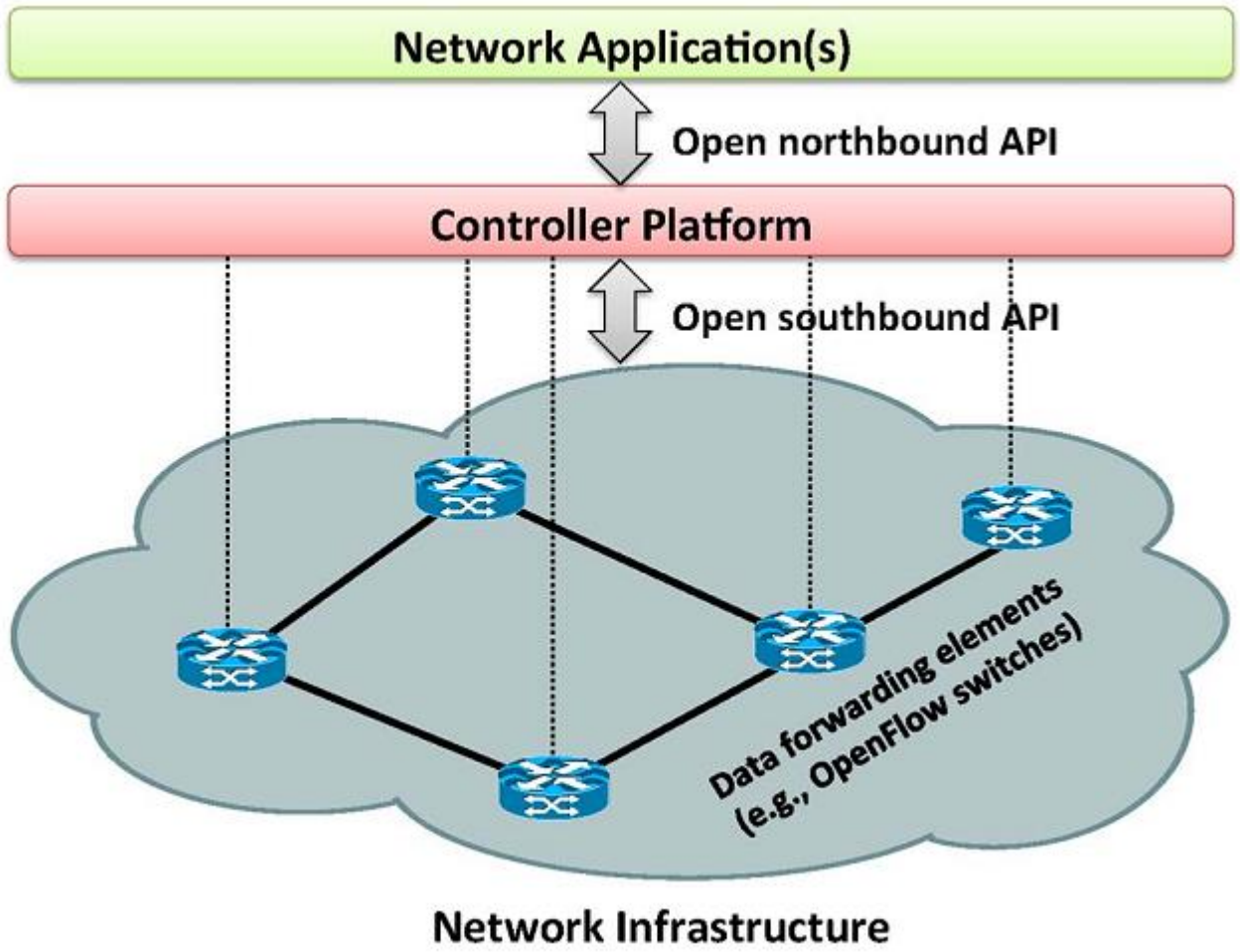
- horizontal
- open interfaces
- rapid innovation

Approach in network scale



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

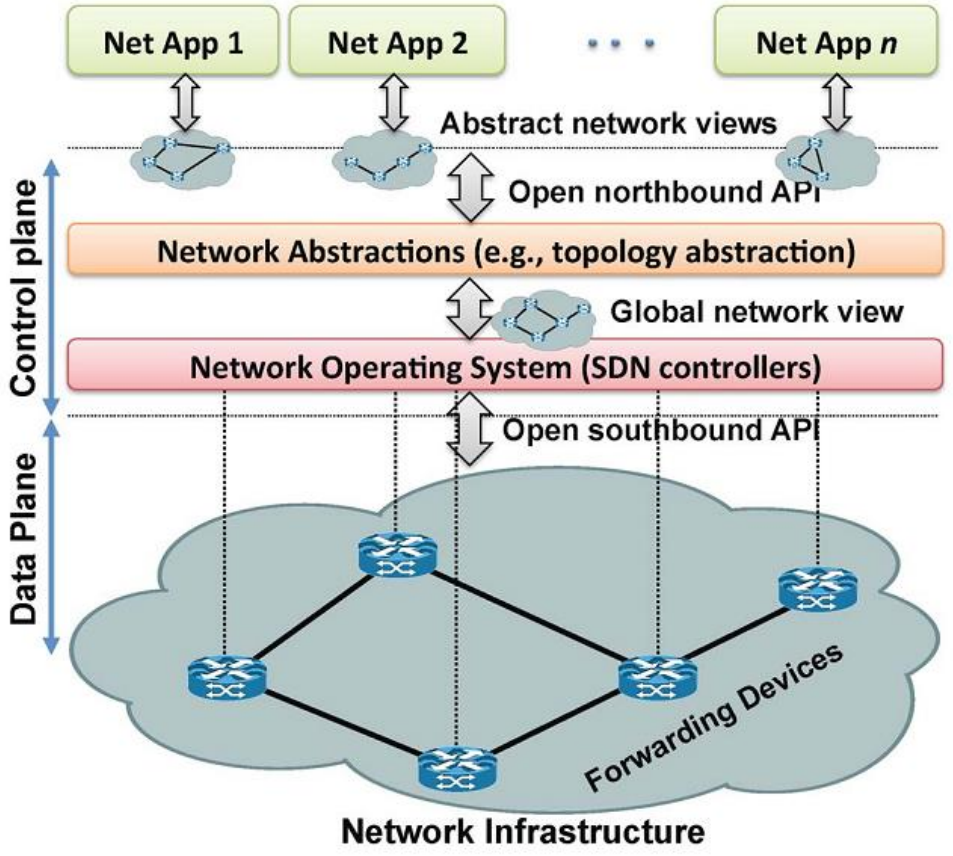
Simple View of SDN Architecture



SDN and Network Abstraction

- From a network administrator point of view SDN makes the network programmable through abstraction.

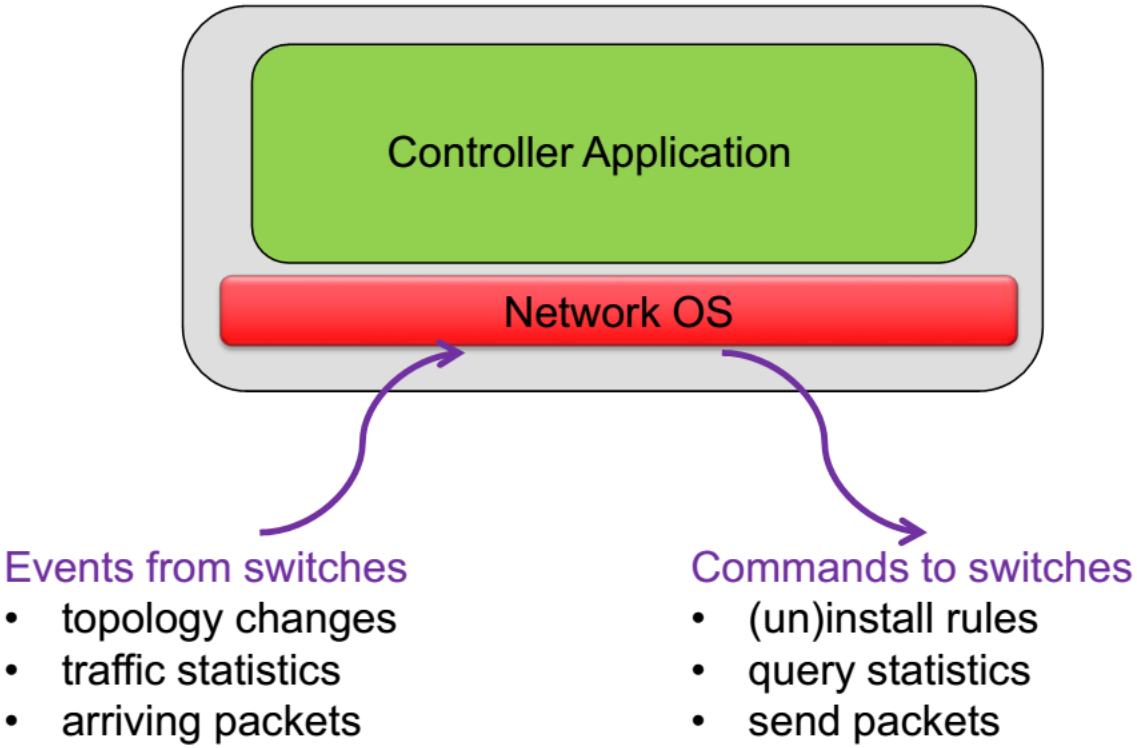
- It is a more general concept than data/control plane decoupling.



Important Issues in SDN

Controller Functionality

- Input Data/Output Command



Different Function Can be Applied

- Network Devices
  - Router
    - Match: longest destination IP prefix
    - Action: forward out via a link
  - Switch
    - Match: destination MAC address
    - Action: forward or flood
  - Firewall
    - Match: IP addresses and TCP/UDP port numbers
    - Action: permit or deny
  - Network Address Translator
    - Match: IP address and port
    - Action: rewrite address and port

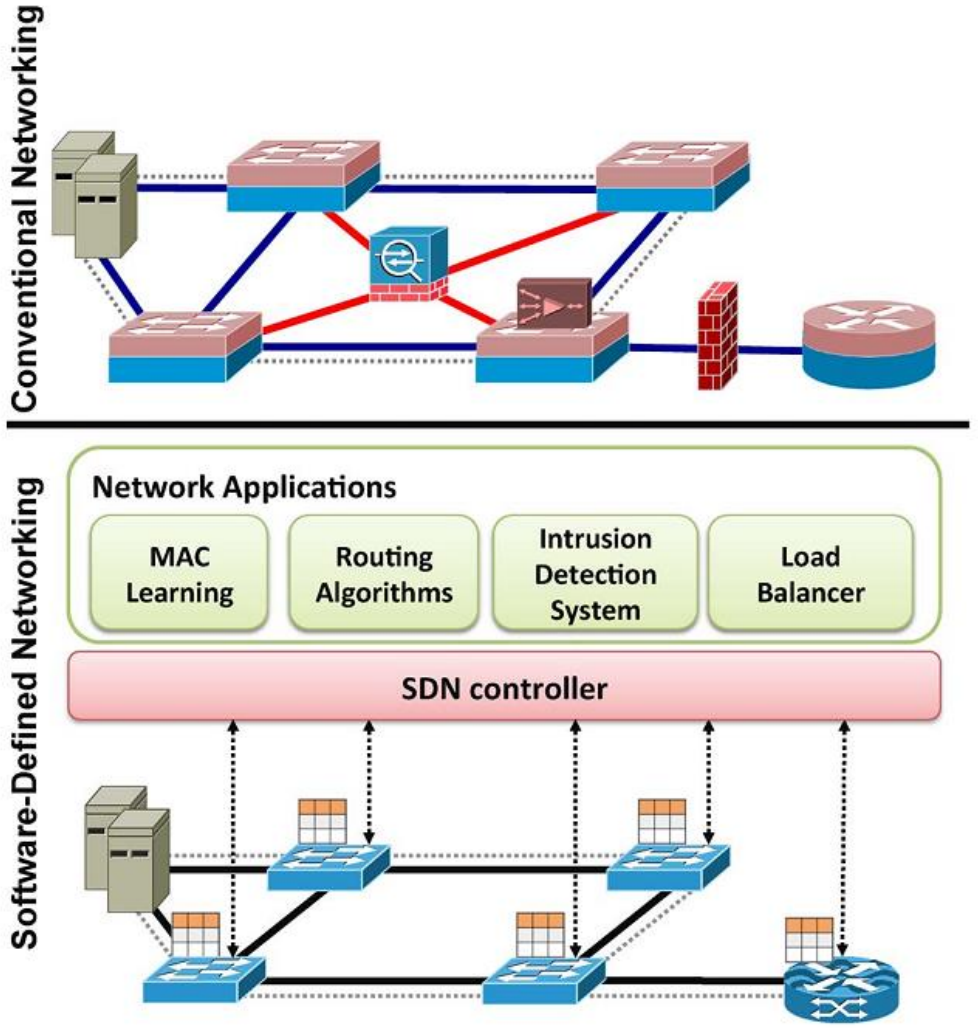
Forwarding Devices

Simple packet-handling rules

- Pattern: match packet header bits
- Actions: drop, forward, modify, send to controller
- Priority: disambiguate overlapping patterns
- Counters: #bytes and #packets
  (received, transmitted, dropped, erroneous, …)

MAC src	MAC dst	IP src	IP dst	TCP dst port	…	Action	Count
*	10:20:.	*	*	*	*	Port1	250
*	*	*	5.6.7.8	*	*	Port2	300
*	*	*	*	25	*	Drop	892
*	*	*	192.*	*	*	Local	120
*	*	*	*	*	*	Controller	11

Traditional vs SDN:
The same forwarding, flexible features



Controller Design

- Different questions comes to mind regarding SDN Controller:

➢ Is controller physically centralized?

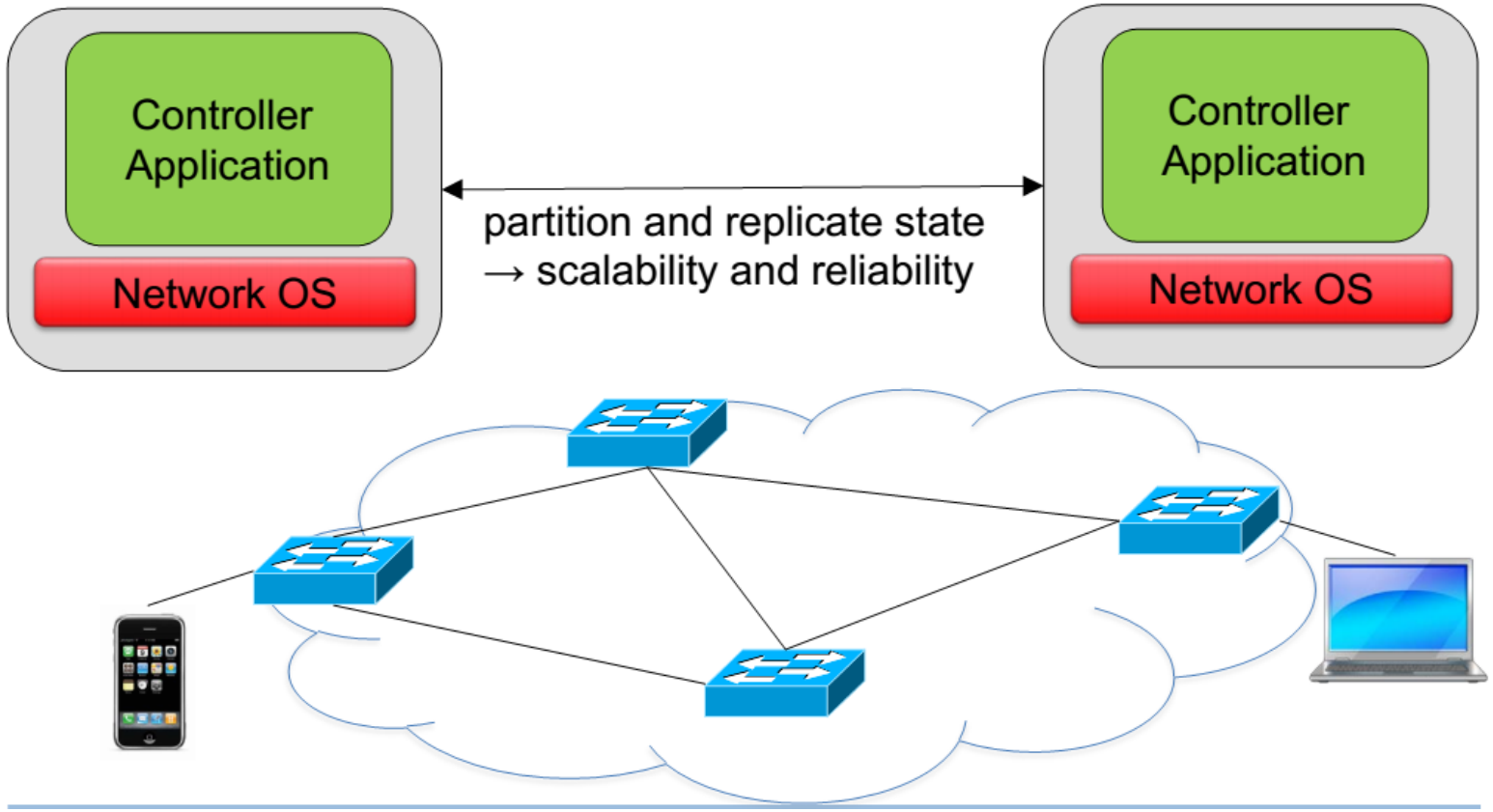
➢ Only one controller is used in a network?

➢ What should we do with high processing load of controller?

➢ How to place the controller regarding latency requirements?

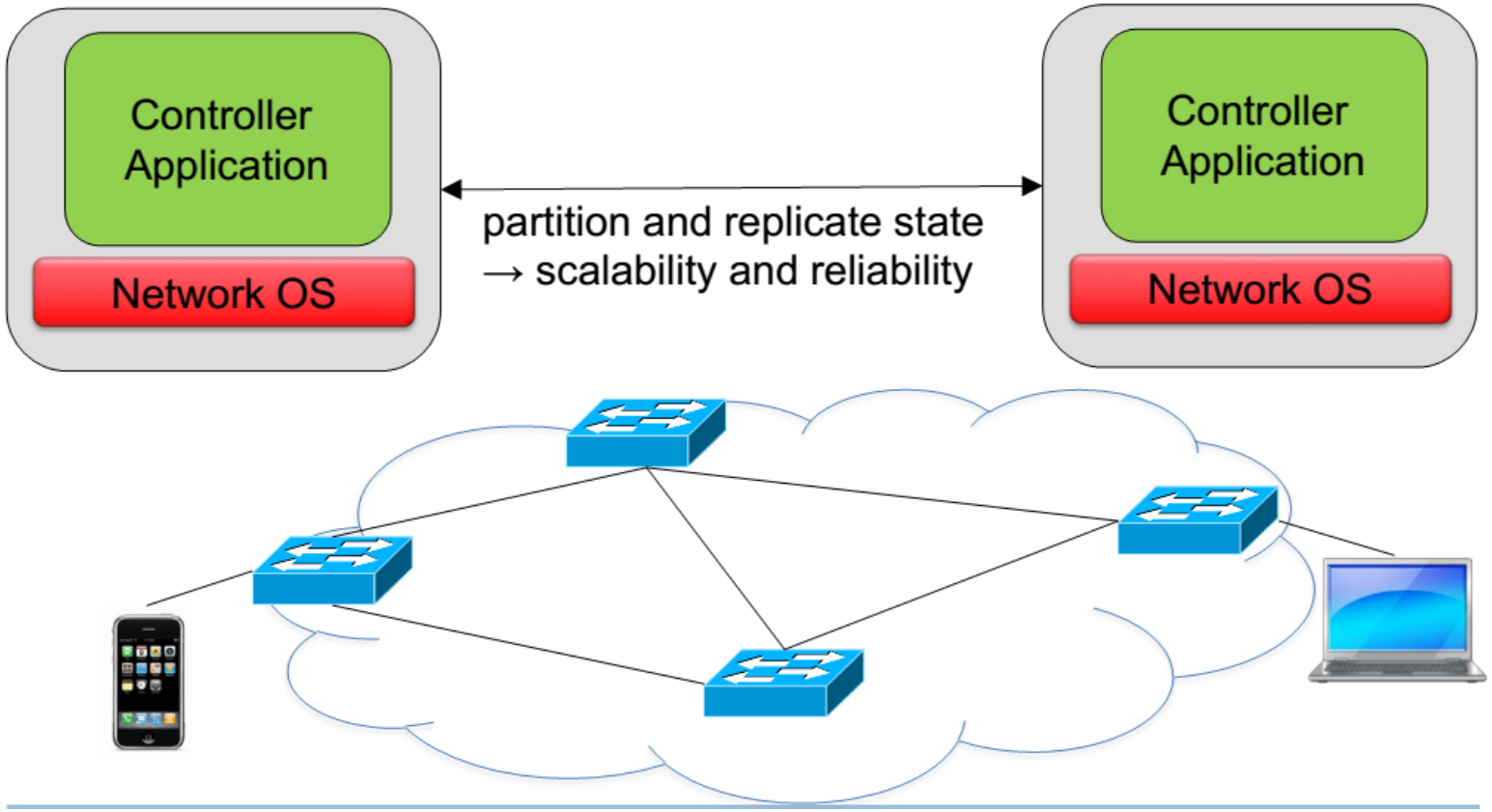
Controller Design

- **Centralized Controller**: The controller is a logically centralized entity but may be physically distributed.



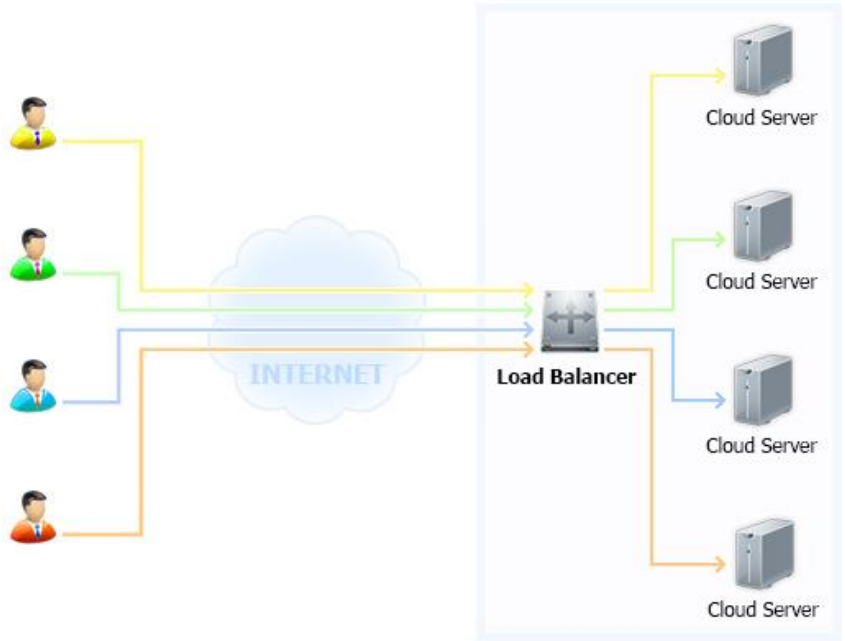
Controller Design

- **How many Controller**: On the other hand due to large network more controller may be required.



Controller Design

- **Computational Load**: Controller is usually a cluster of cloud server with high computational power.
- Load balancer is used to prevent server crashes.



Controller Design

- **Latency**: Controller placement problem is important due to delay requirement of the network [Heller2012]

- Two metric has been used

1. Average Latency

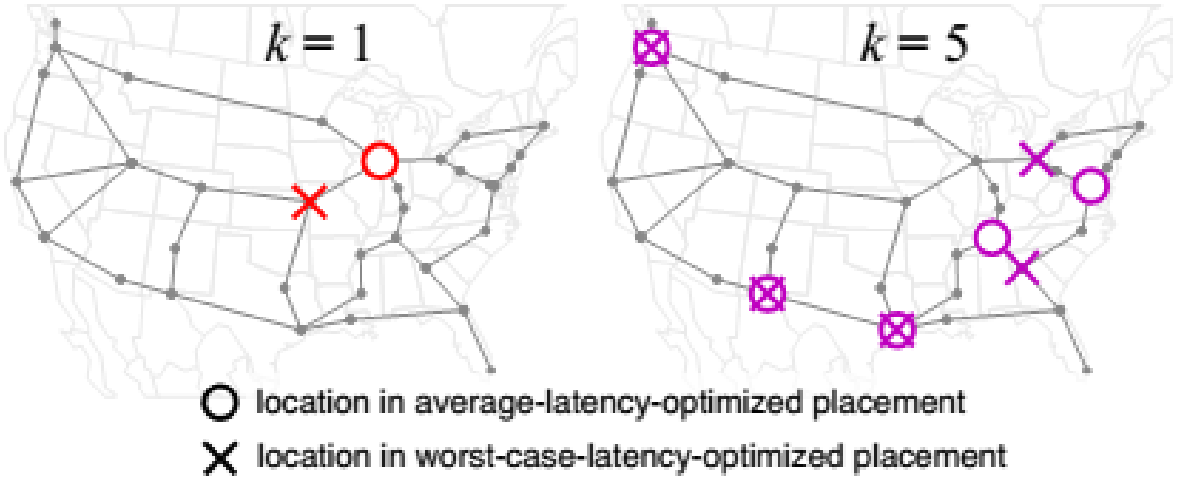
$$L_{avg}(S') = \frac{1}{n} \sum_{v \in V} \min_{(s \in S')} d(v, s)$$

2. Worst-case Latency

$$L_{wc}(S') = \max_{(v \in V)} \min_{(s \in S')} d(v, s)$$

Controller Design: Latency

- The results for Internet2 OS3E is as follows, k = 1,5 are the number of controllers

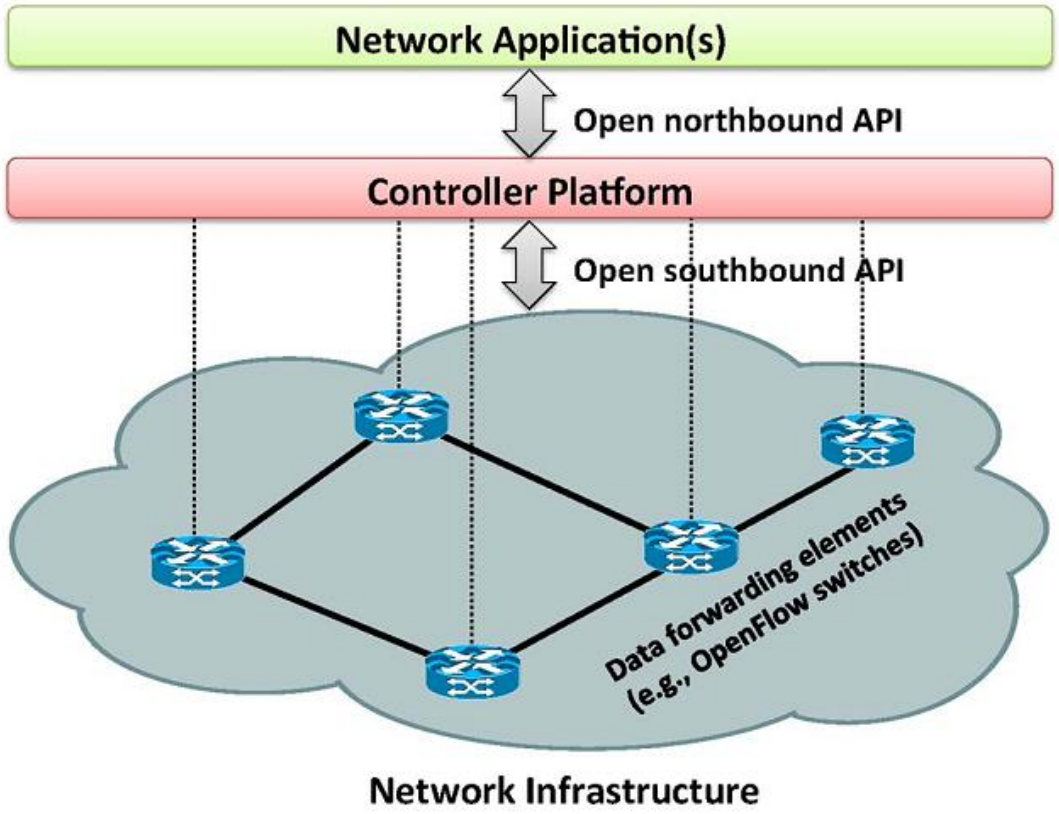


- The solution is standard problem in computer science:

  Average → k-median Worst-case→ k-center

SDN architecture revisit

- Southbound interface is the major enabler of the SDN concept.

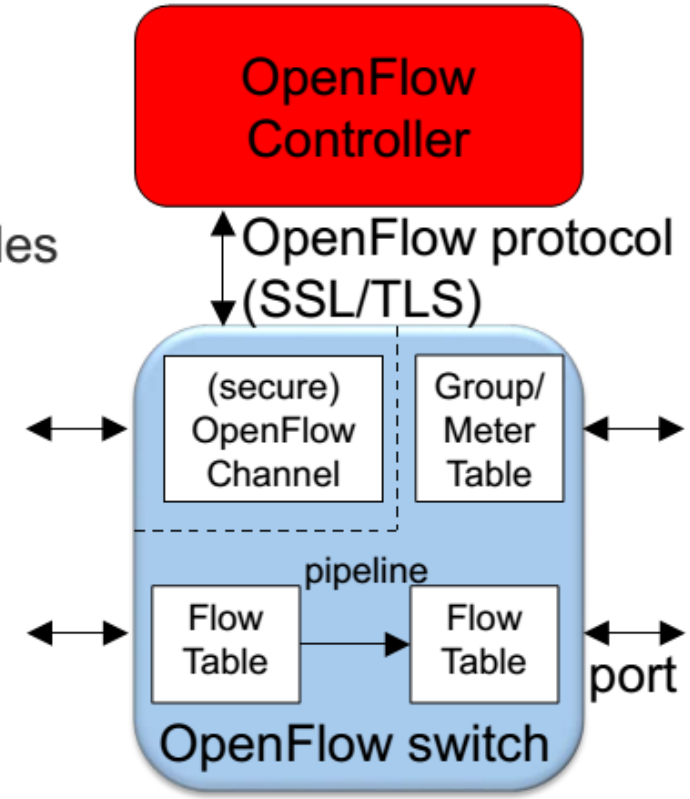


Southbound Interface

- Without efficient southbound interface SDN ideas will not be realized. SDN achieves its goals base on this framework.
- The most well-known southbound interface is the OpenFlow protocol which was established by joint collaboration of Stanford and Berkeley.
- There exist other interfaces such as Cisco OpFlex but less used.
- Many thought SDN as OpenFlow but it is wrong. SDN is a concept while OpenFlow is a Tool.

OpenFlow: Brief Intro.

- Most Ethernet switches and routers contain flow tables based on content addressable memory running at line-rate to
  - support QoS
  - implement firewalls and NATs
  - collect statistics
- Switches and routers provide similar functionality, but have proprietary flow tables (data path = flow table + actions)
- OpenFlow aims to provide a standard interface to program flow tables.
- OpenFlow switch
  1. Flow / group / meter tables
  2. Secure channel for configuration
  3. OpenFlow protocol
- OpenFlow Controller
  - adds/changes/removes table entries.

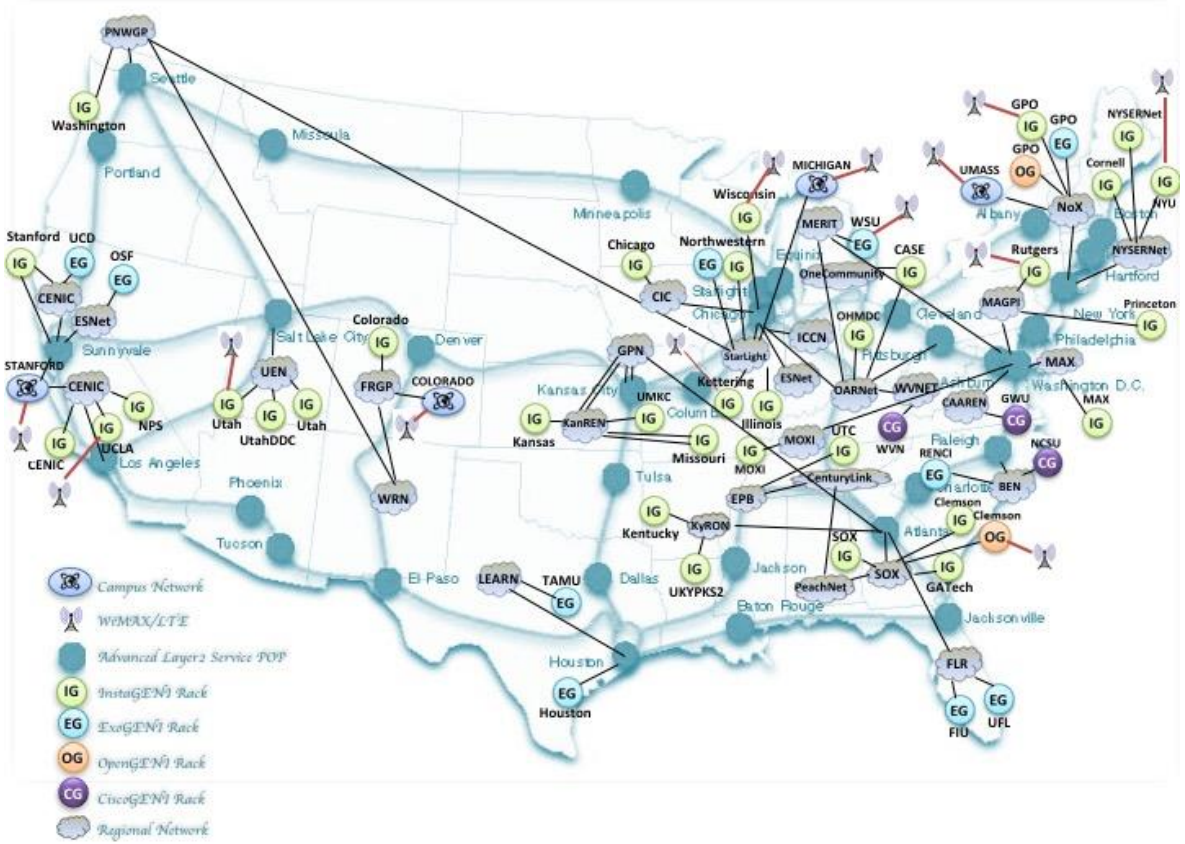


OpenFlow Origin: Network Experiments

- How do evaluate your proposed algorithm in computer networking?
- There are some tools such as NS2/3, and OPNET for simulation and Mininet for emulation. (We talked about simulation and emulation previously in class!)
- However, simulation and emulation can not provide a practical and ideal test environment.
- The ideal case is to be able to test on a real environment.

OpenFlow Origin: Network Experiments

- GENI (Global Environment for Network Innovations) provides a virtual laboratory for networking and distributed systems research and education.



OpenFlow Origin: Network Experiments

- GENI is well suited for exploring networks at scale, thereby promoting innovations in network science, security, services and applications.

- GENI allows experimenters to:
  - Obtain compute resources from locations around the United States;
  - Connect compute resources using Layer 2 networks in topologies best suited to their experiments;
  - Install custom software or even custom operating systems on these compute resources;
  - Control how network switches in their experiment handle traffic flows;
  - Run their own Layer 3 and above protocols by installing protocol software in their compute resources and by providing flow controllers for their switches.

OpenFlow : Memory Problem

- An issue regarding the scalability of an OpenFlow network is memory limitation in forwarding devices.

- OpenFlow rules are more complex than forwarding rules in traditional IP routers. They support more flexible matching and matching fields and also different actions to be taken upon packet arrival.

- A commodity switch normally supports between a few thousand up to tens of thousands forwarding rules.

OpenFlow : Memory Problem

- Ternary Content Addressable Memory (TCAM) has been used to support forwarding rules, which can be expensive and power-hungry.

- The rule space is a bottleneck to the scalability of OpenFlow.

- The optimal use of the rule space to serve a scaling number of flow entries while respecting network policies and constraints is a challenging and important topic

OpenFlow: Open Networking Foundation

- The Open Networking Foundation (ONF) was founded in 2011 by Deutsche Telekom, Facebook, Google, Microsoft, Verizon, and Yahoo to promote the implementation of SDN and OpenFlow-based networks.

- 2013: 123 members, 2014: 150 members with 24 start-up.
- As of each protocol OpenFlow needs development and releases.
- ONF is responsible for OpenFlow standard release.

OpenFlow development

- In the first generation “Type 0” switches (version 0), the flow header is a 10-tuple.

- A TCP flow could be specified by all ten fields, whereas an IP flow might not include the transport ports in its definition.

In Port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

OpenFlow development

- In the last slides, OpenFlow type “0” was introduced. It was the first version. It also represented as 0.X.Y with X,Y Integer for further modifications.
- The unique characteristic of this switch is its 10-tuple header format.
- The first version 0.2.0 released in March 2008, versions 0.8.1, 0.8.2, ... , 0.8.9 introduced afterward in 2008 until the release of 0.8.9 in July 2009.
- It was a beginner version not extensively deployed.

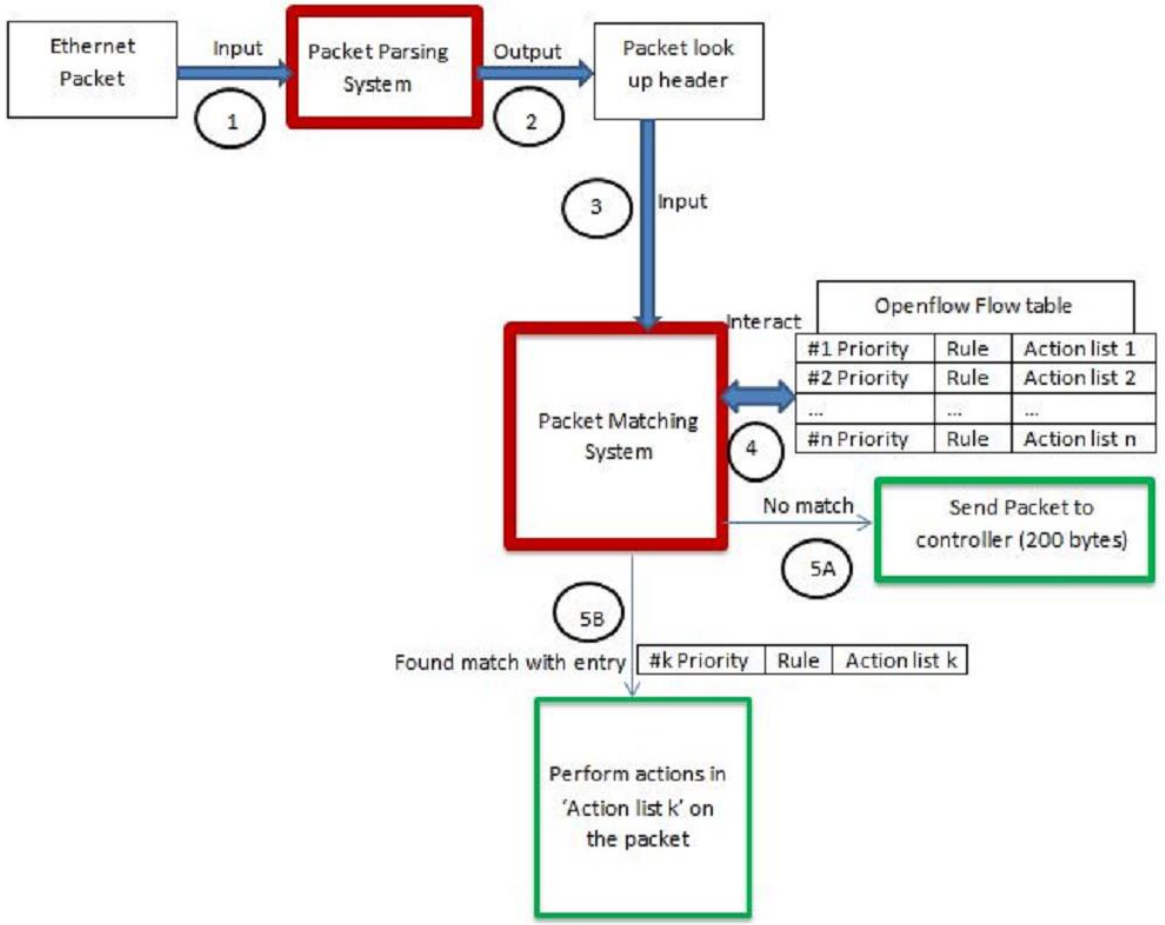
OpenFlow development (1.0.0)

- OpenFlow version 1.0 the most widely deployed version, was released in December, 2009.
- An important modification is the header field in each version.
- 1.0.0 Header fields:

Ingress Port
Ether src
Ether dst
Ether type
VLAN id
VLAN priority CoS
IP src
IP dst
IP Proto
IP ToS bits
TCP/UDP src port
TCP/UDP dst port

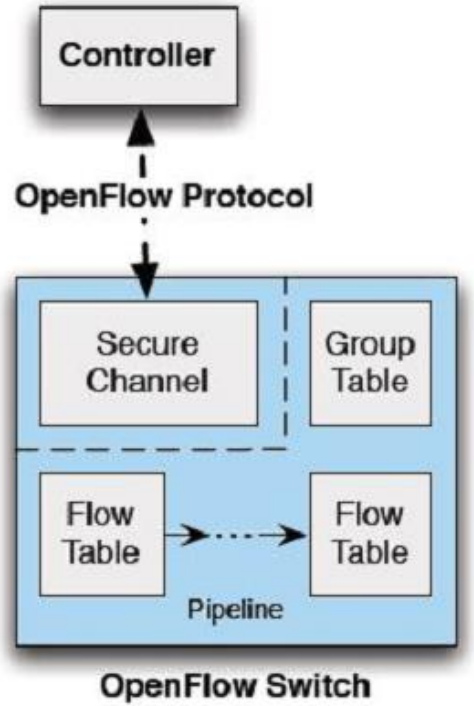
OpenFlow development (1.0.0)

How a packet is processed and forwarded in an OpenFlow 1.0.0 swi



OpenFlow development (1.1.0)

- In the **OpenFlow 1.1.0** specification, a switch contains
  several flow tables and a group table, instead of just one single flow table, as in OpenFlow 1.0.0.



Ingress port
Metadata
Ether src
Ether dst
Ether type
VLAN id
VLAN priority
MPLS label
MPLS EXP traffic class
IPv4 src
IPv4 dst
IPv4 proto / ARP opcode
IPv4 ToS bits
TCP/UDP/SCTP src port. ICMP Type
TCP/UDP/SCTP dst port. ICMP Code

OpenFlow development (1.1.0)

- The processing of a packet entering the switch has changed as there are multiple flow tables available in the switch.
- The flow tables in the switch are linked to each other through a process termed as **pipeline processing**.
- Pipeline processing involves a set of flow tables linked together to process the packet coming in.
- When the packet first enters the switch, it is sent to the first table to look for the flow entry to be matched.

OpenFlow development (1.1.0)

- The flow entries in the flow tables can also point to the group table.
- The group table is a special kind of table designed to perform operations that are common across multiple flows.
- Complex forwarding actions such as multipath and link aggregation are enabled through the group table.
- Finally, specification 1.1.0 introduces instructions instead of actions.

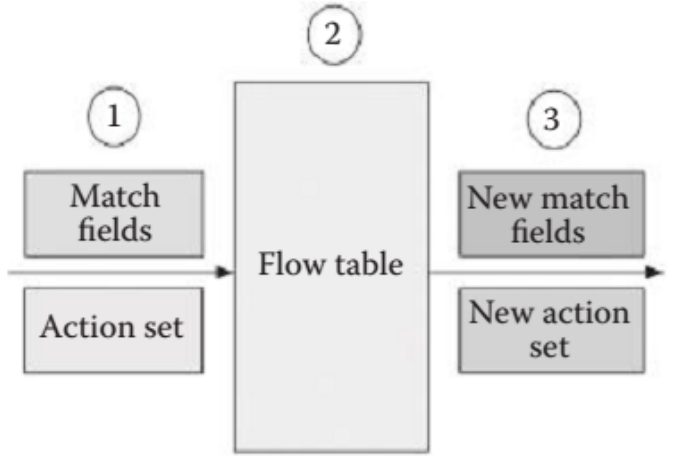
OpenFlow development (1.1.0)

- Multiple flow tables: table 0 to table n

1. The entry flow first checked by the table 0. Depending on the matching results further instruction will be done.
2. The instructions modify the action set by adding or deleting actions, such as packet forwarding, packet modification, group table processing, and pipeline processing.
3. Pipeline processing is performed by explicitly directing the packet to another flow table using the GoTo instruction.
4. In this case, the instructions to modify the packet header and update the match fields are performed before the new match is executed.
5. In case of match success in the next table, another set of instructions will be applied to the action set.

OpenFlow development (1.1.0)

6. Before sending the packet to the output port(s), every instruction in the action set is performed. If no match is found in a table, the switch response will depend on the table configuration.



1) – Find the highest priority matching flow entry, based on the ingress port, packet headers, and metadata.
2) – If there is a match, apply instructions, creating the new match fields and the new action set:
a) Apply action instruction, which modify the packet and update the match fields.
b) Update action set, which means to clear actions and/ or write action instructions.
c) Update metadata.
3) – Send the new match fields and the new action set to the next table.

OpenFlow development (1.2.0)

- This version was released in December 2011 and it includes a few major features.
- Support to **IPv6** addressing is added.
- Another important feature supported is the possibility of connecting a switch to **multiple controllers** concurrently.
- The switch maintains connections with all the controllers and these can communicate with each other to do **hand overs**.
- Having multiple controllers provides **faster recovery** during failure and it is also possible to achieve **load balancing**.

OpenFlow development (1.3.0)

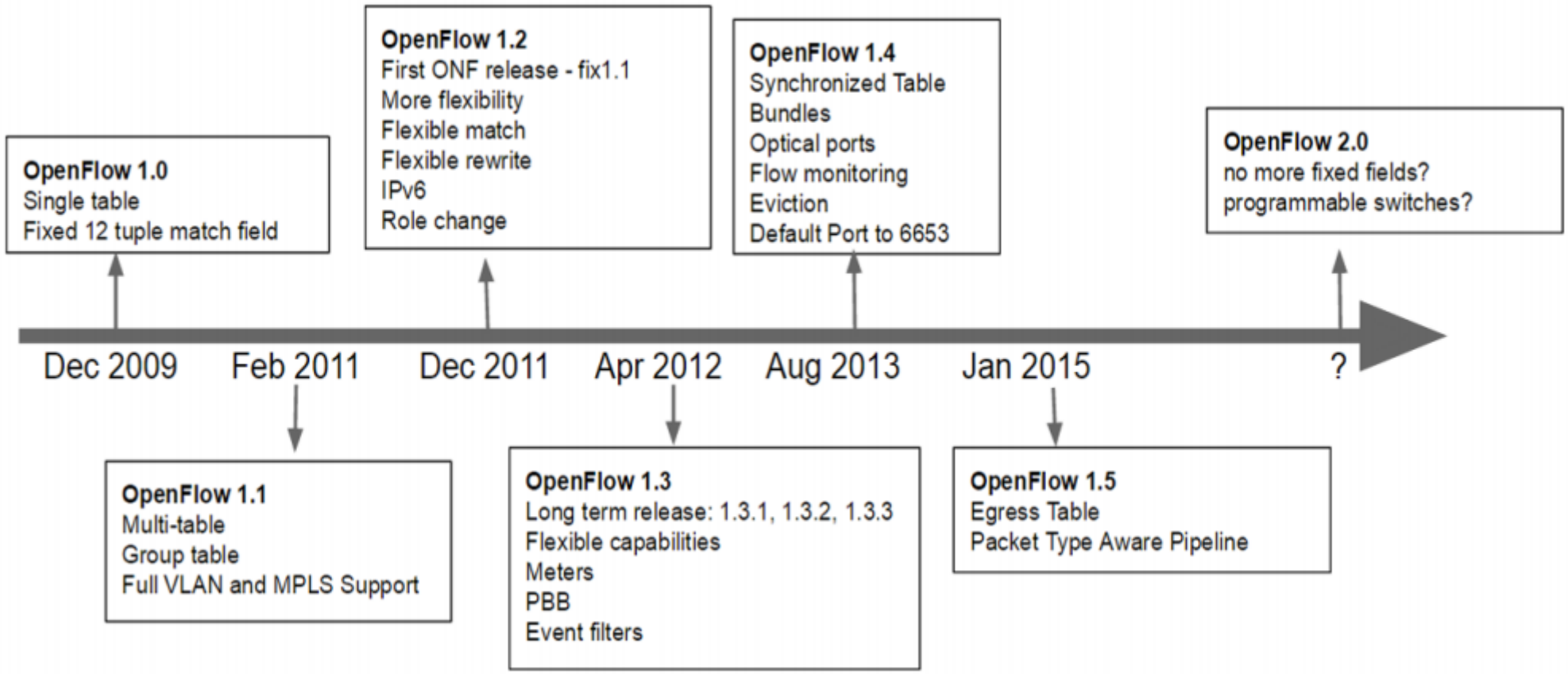
- This version was released in June 2012 and it includes a few major features.
- It is possible to **control** the rate of packets through per flow meters.
- **Auxiliary** connections between the switch and the controller have been enabled.
- Another improvement is that **cookies can be added** to the packets sent from the switch to the controller and **specific durations field** have been added to most statistics.

OpenFlow development

- Comparison (2014-2015)

Specification	1.0.0	1.1.0	1.2	1.3.0
Widely deployed	Yes	No	No	No
Flow table	Single flow table	Multiple flow tables	Multiple flow tables	Multiple flow tables
MPLS matching	No	Yes	Yes	Yes, bottom of stack bit added
Group table	No	Yes	Yes	Yes, more flexible table miss support
IPv6 support	No	No	Yes	Yes, new header field added
Simultaneous communication with multiple controllers	No	No	Yes	Yes, auxiliary connections enabled

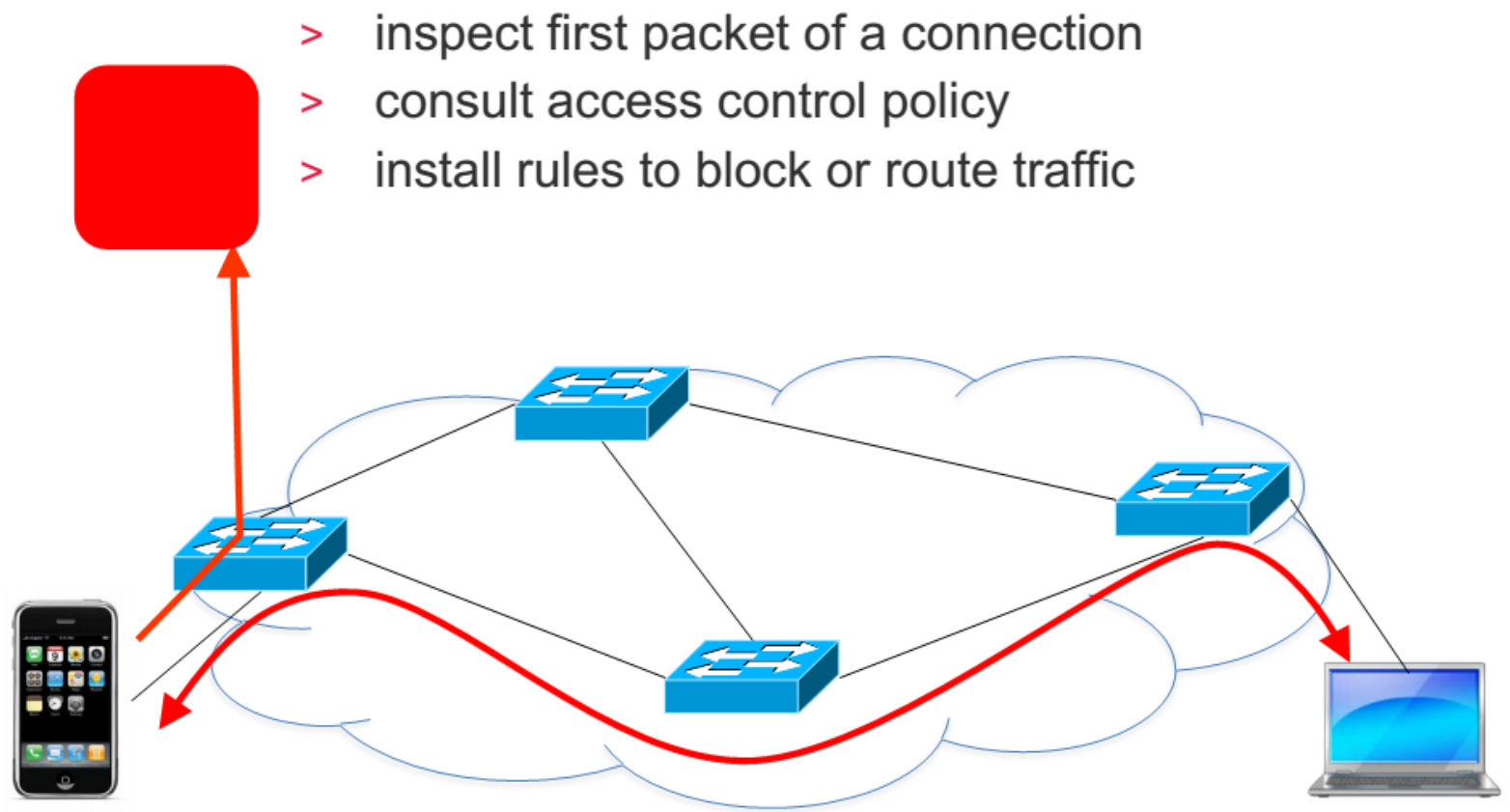
OpenFlow development



SDN Applications:
Several Examples

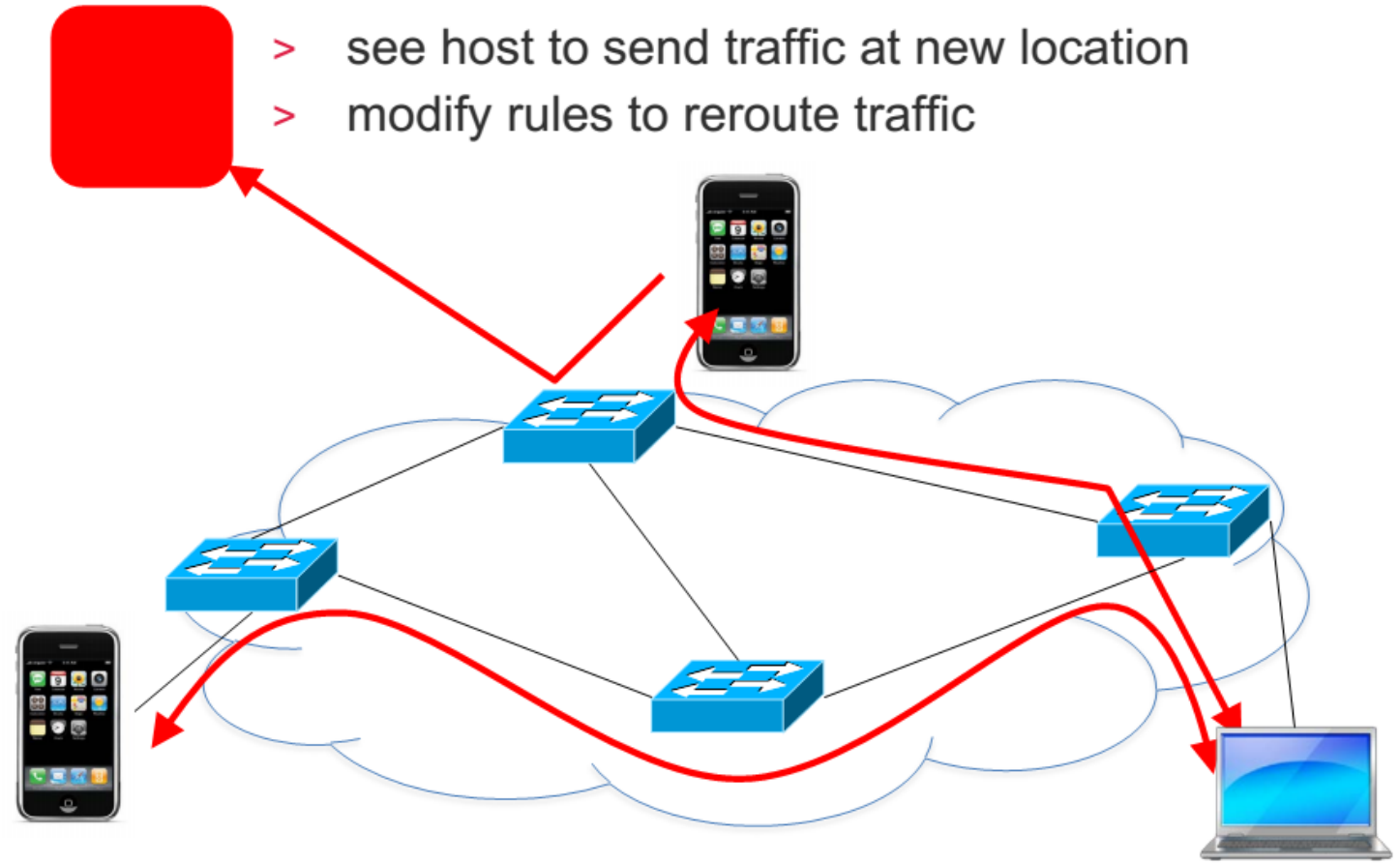
SDN Applications

- Dynamic Configuration, policy management, Access Control



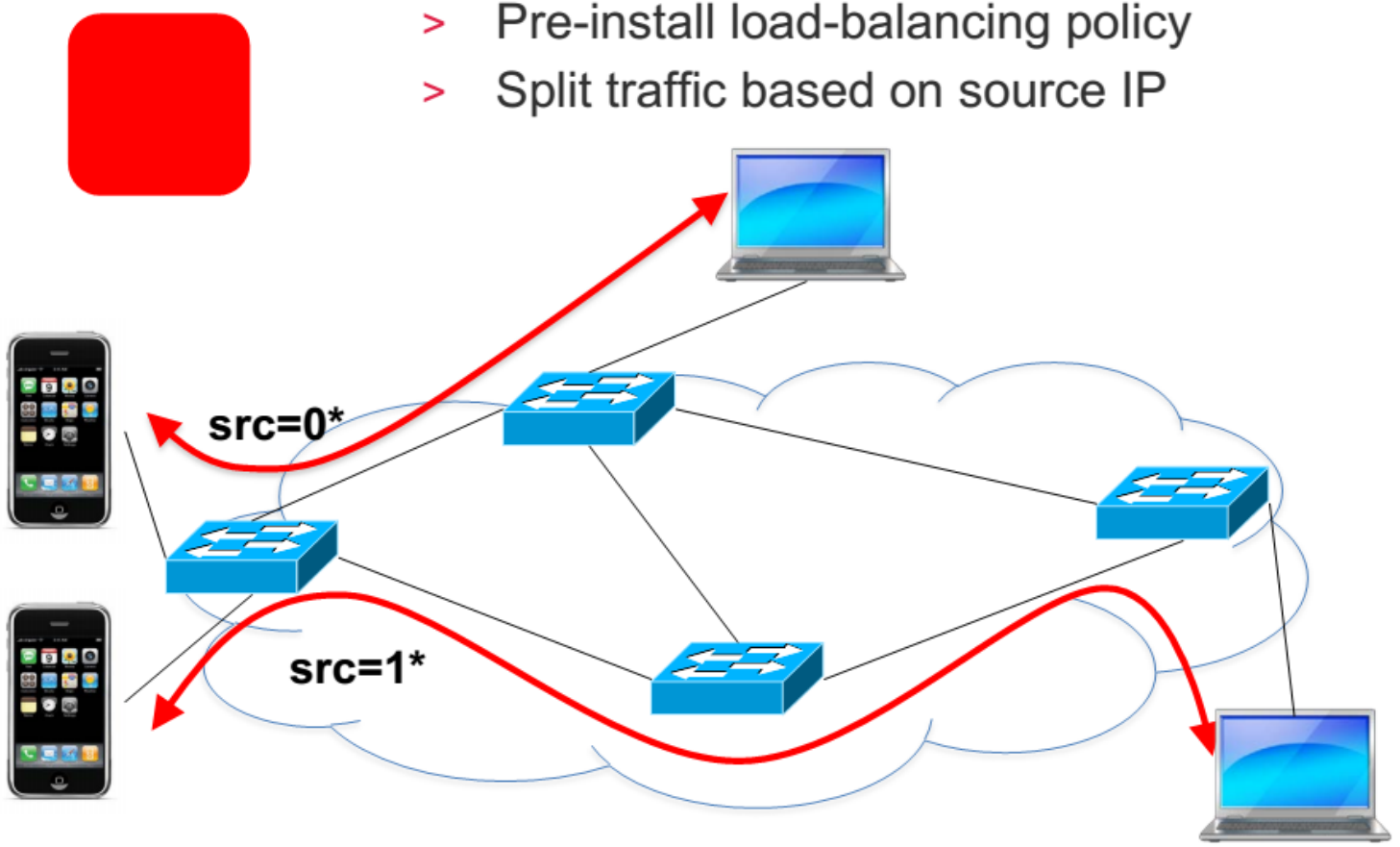
SDN Applications

- Seamless Mobility Migration



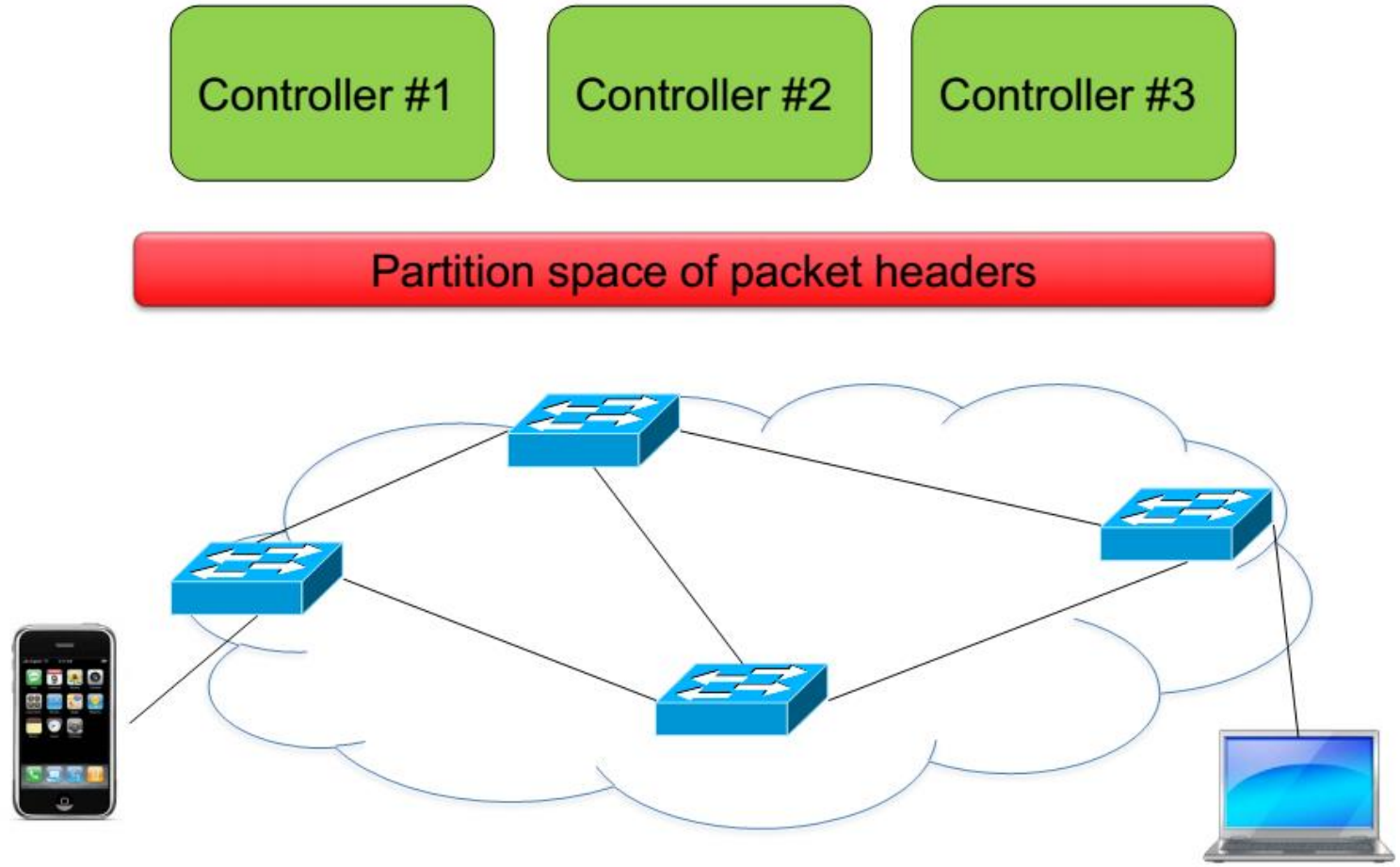
SDN Applications

- Server Load Balancing



SDN Applications

- Network Virtualization

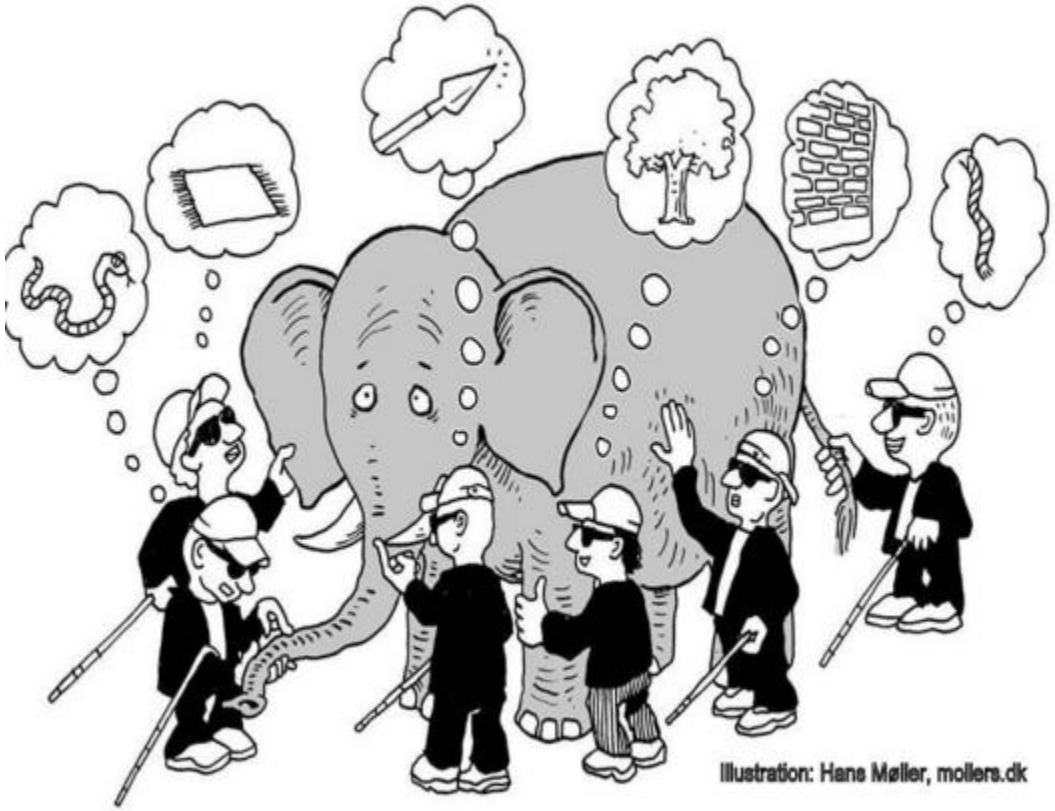


SDN Applications: More

- *Dynamic access control*
- *Seamless mobility/migration*
- *Server load balancing*
- *Network virtualization*
- Link failure recovery
- Data center networking
- Multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection
- Network management and control
- Virtual networks
- Non-IP networks, Future Internet protocols
- Deep-packet inspection, intrusion detection

Important Note

- Misunderstanding About SDN



Many Definitions

- Openflow
- Controller
- Openstack
- Overlays
- Network virtualization
- Automation
- APIs
- Application oriented
- Virtual Services
- Open vSwitch
- …