SNMP, ICMP and SDN

CE 352, Computer Networks

Salem Al-Agtash

Lecture 18

Slides are adapted from Computer Networking: A Top Down Approach, 7th Edition © J.F Kurose and K.W. Ross

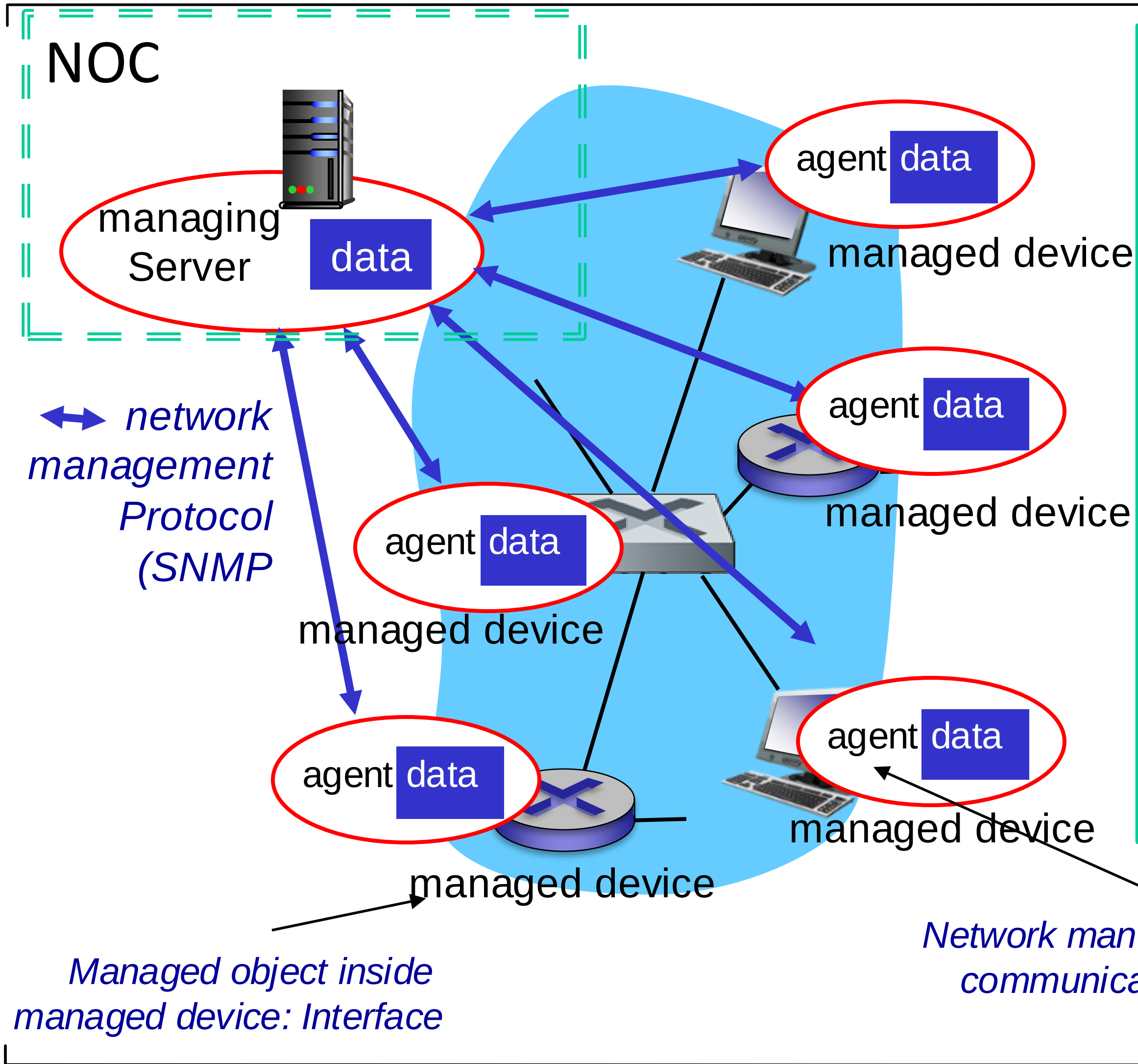
Network management

- autonomous systems (aka "network"): 1000s of interacting hardware/software components
- As other complex systems, networks require management and control

> "Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

- NOC: Network Operation Center that connects with devices, monitors, and controls the network (with human intervention)
- Simple Network Management Protocol (SNMP) – UDP application layer protocol - is used to collect information and configure devices (routers, switches, servers, printers). SNMPv1, SNMPv2, SNMPv3 (include security)
- e.g of SNMP network management tools: OpManager, Solarwinds, Zabbix, Cacti,…

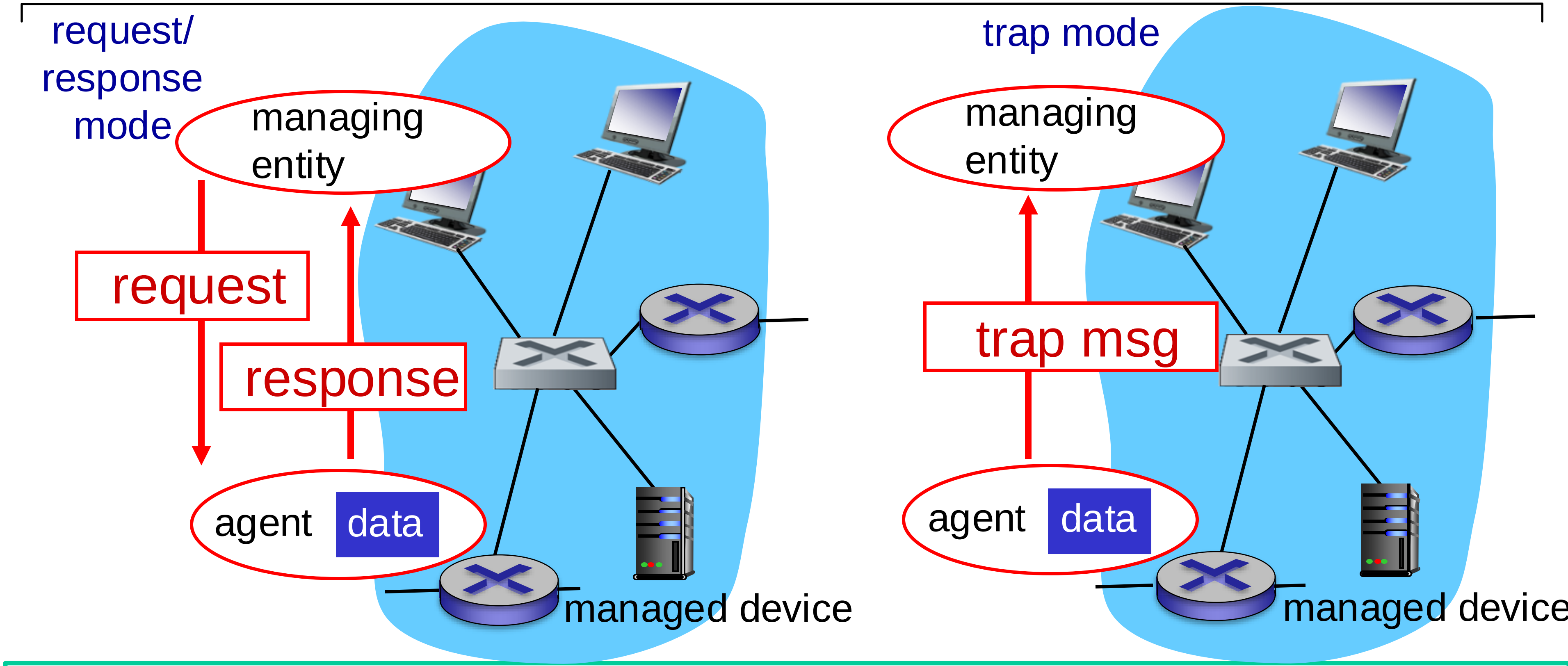
Infrastructure for network management



NOC
managing Server
data
agent
data
managed device
agent
data
managed device
agent
data
managed device
agent
data
managed device
agent
data
managed device
network management Protocol (SNMP
Managed object inside managed device: Interface
Network management agent (process) that communicates with the managing server (similar to routing agent)

- *managed devices* contain *managed objects* whose data is gathered into a *Management Information Base (MIB)*
- Agents use SNMP to inform the managing server of exceptional events (e.g. failures).
- Network administrator use SNMP to manage ("monitor, test, poll, configure, analyze, evaluate, and control") the network.

SNMP protocol to convey MIB info



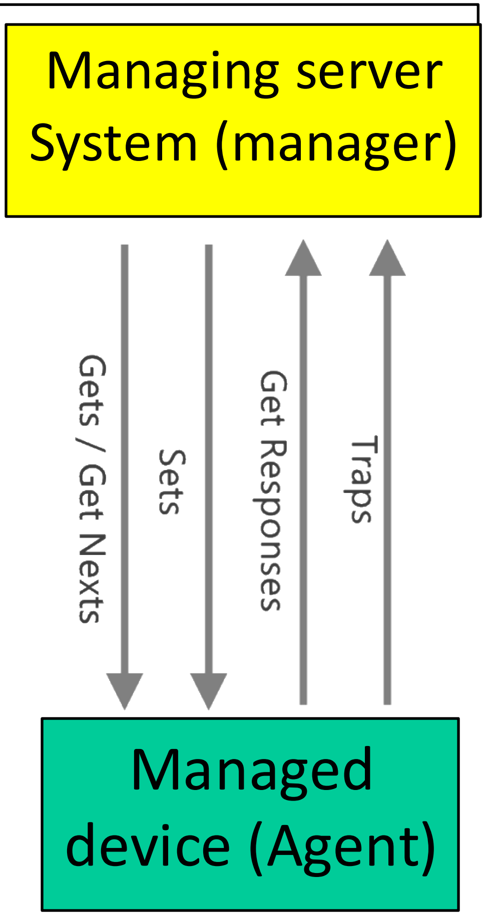
SNMP request-response mode:

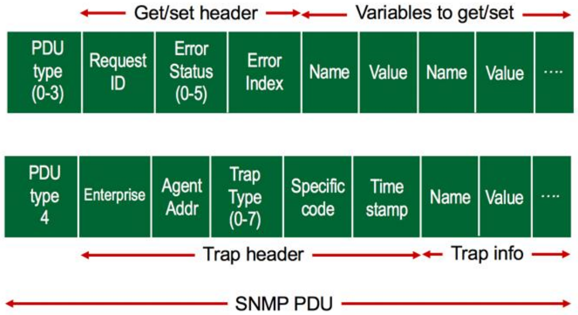
- SNMP managing server sends a request (query: retrieve, modify MIB object)to an SNMP agent
- SNMP agent receives the request, performs some action, and sends a reply

SNMP trap message: to notify managing server of an exceptional situation (e.g., up/down link interface).

SNMP protocol: message types (v2)

PDU type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: “get me data” (data instance, next data in list, block of data)
InformRequest	manager-to-manager: here’s MIB value
SetRequest	manager-to-agent: set MIB value
Response	Agent-to-manager: value, response to Request
Trap	Agent-to-manager: inform manager of exceptional event





PDU type (0-3)	Request ID	Error Status (0-5)	Error Index	Name	Value	Name	Value	....

PDU type 4	Enterprise	Agent Addr	Trap Type (0-7)	Specific code	Time stamp	Name	Value	....

SNMP protocol data unit (PDU), embedded in a UDP datagram (unreliable)

ICMP: internet control message protocol

used by hosts & routers to communicate network-level information

- error reporting: router reports unreachable host in http request, network, port, protocol
- echo request/reply (used by ping)

network-layer "above" IP:

- ICMP msgs carried in IP datagrams [protocol #1] similar to TCP (6), UDP(17)]

ICMP message has type and code plus first 8 bytes of IP datagram

Traceroute:

- sends a series of ordinary IP datagrams with unlikely UDP port no, each with TTL 1, 2, 3, ..etc.
- Router n discards TTL expired datagram and sends to source ICMP warning message to the source (type 11 code 0) –> source gets RTT and IP and name or router
- One datagram arrives, destination reports ICMP port unreachable, so no need to send additional probe messages

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

```
socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)
```

Traceroute and ICMP (recall Lab2)

source sends series of UDP segments to destination

- first set has TTL =1
- second set has TTL=2, etc.
- unlikely port number

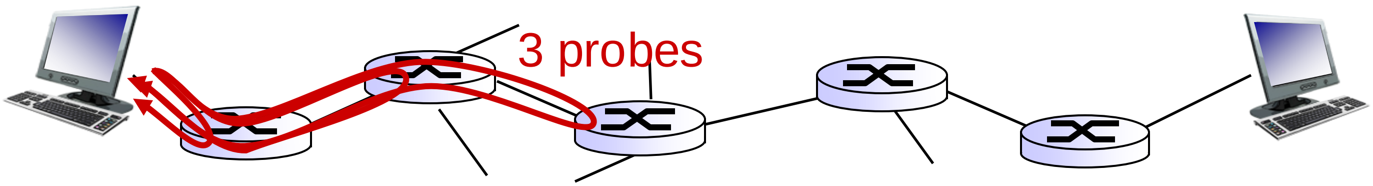
when datagram in *n*th set arrives to nth router:

- router discards datagram and sends source ICMP message (type 11, code 0)
- ICMP message include name of router & IP address

when ICMP message arrives, source records RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops



Software defined networking (SDN)

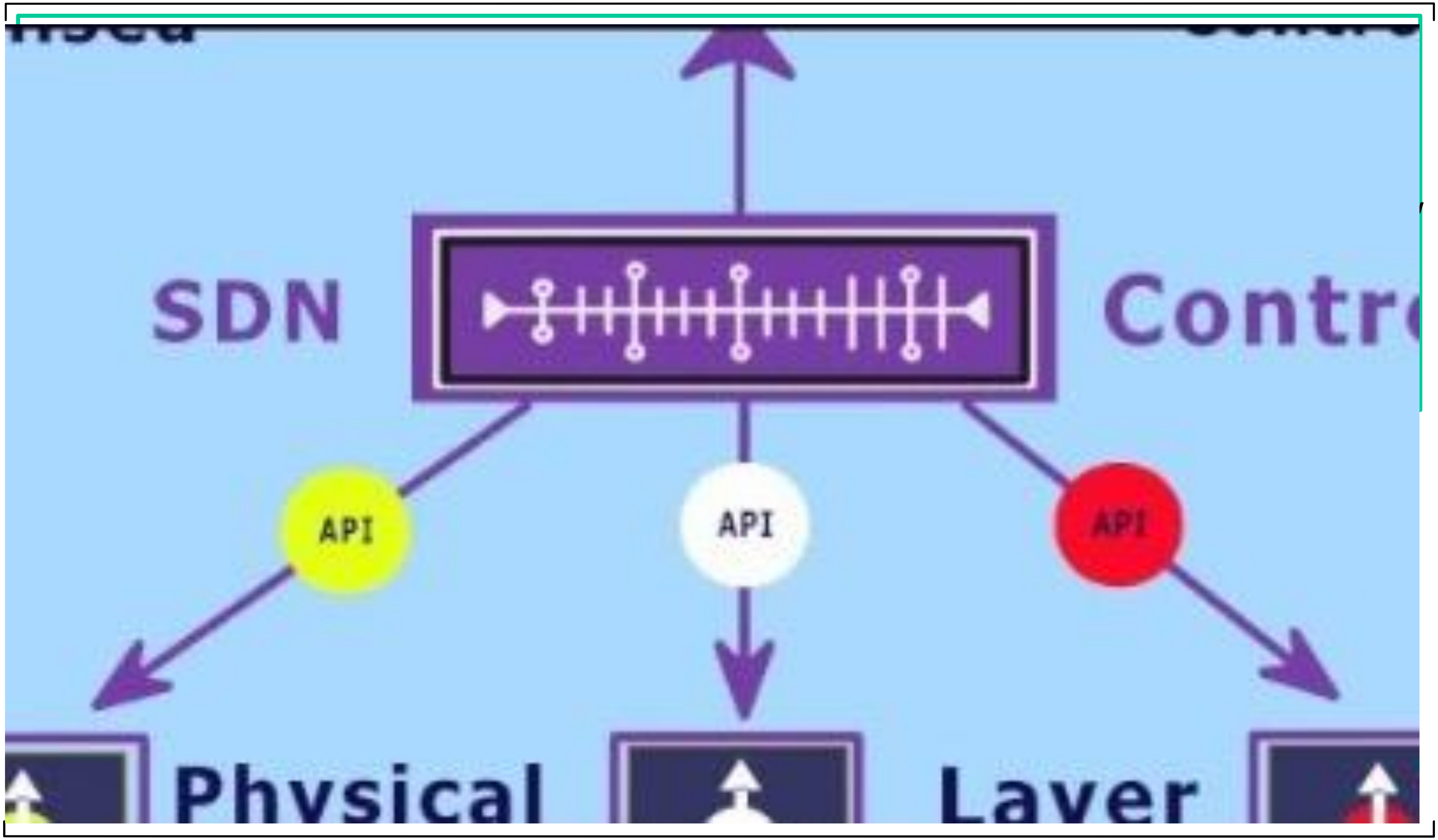
Internet network layer: historically has been implemented via distributed, per-router approach

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

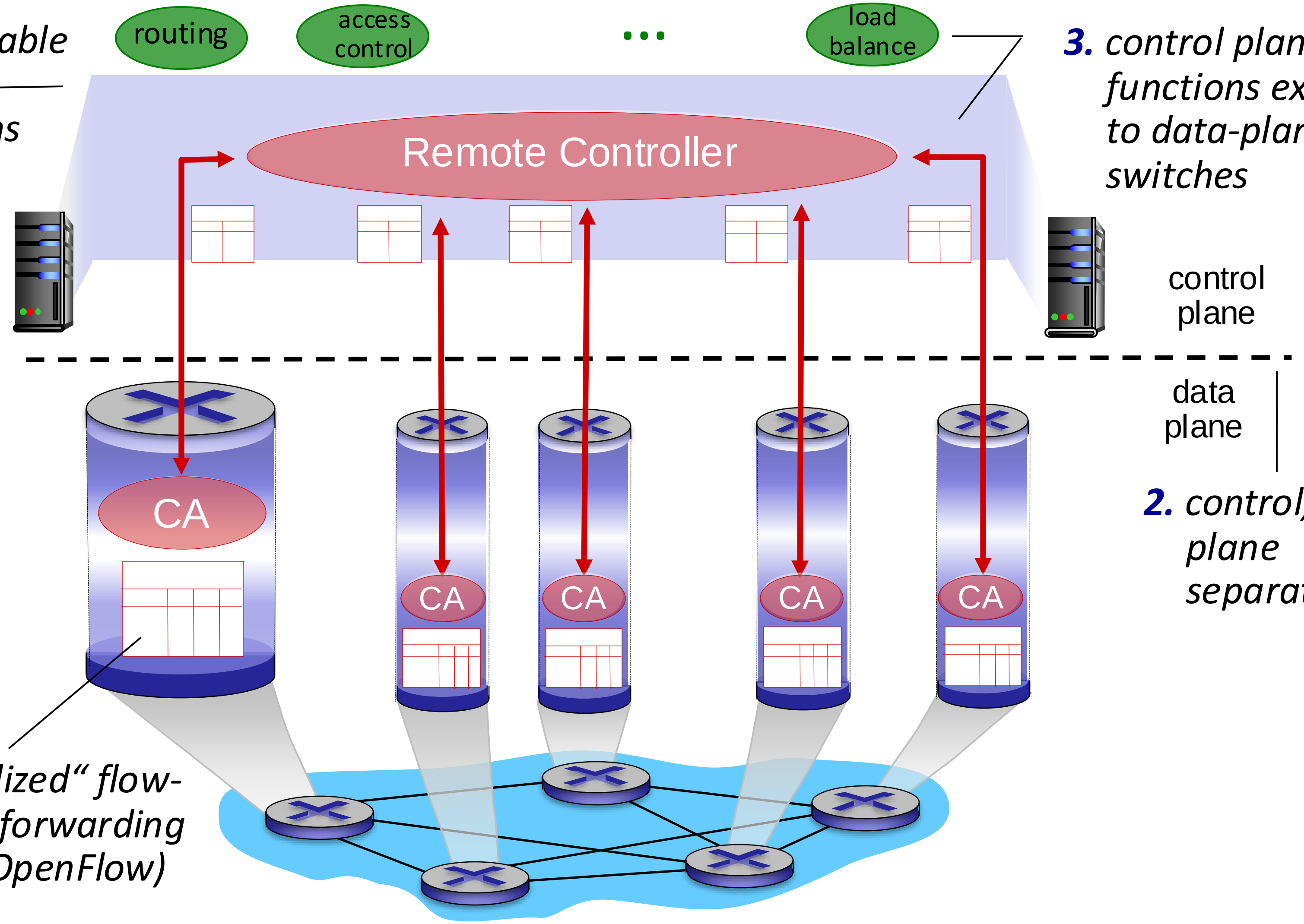
~2005: renewed interest in rethinking network control plane

Software defined networking (SDN)

https://youtu.be/53djBGNDXlo



Recall (SDN: centralized control plane)



4. *programmable control applications*

3. *control plane functions external to data-plane switches*

2. *control, data plane separation*

1: *generalized" flow-based" forwarding (e.g., OpenFlow)*

Why logically centralize control plane?

SDN represents a significant unbundling of network functionality:

- Data plane switches
- SDN controllers
- Network – control applications

Separate entities may be provided by different vendors

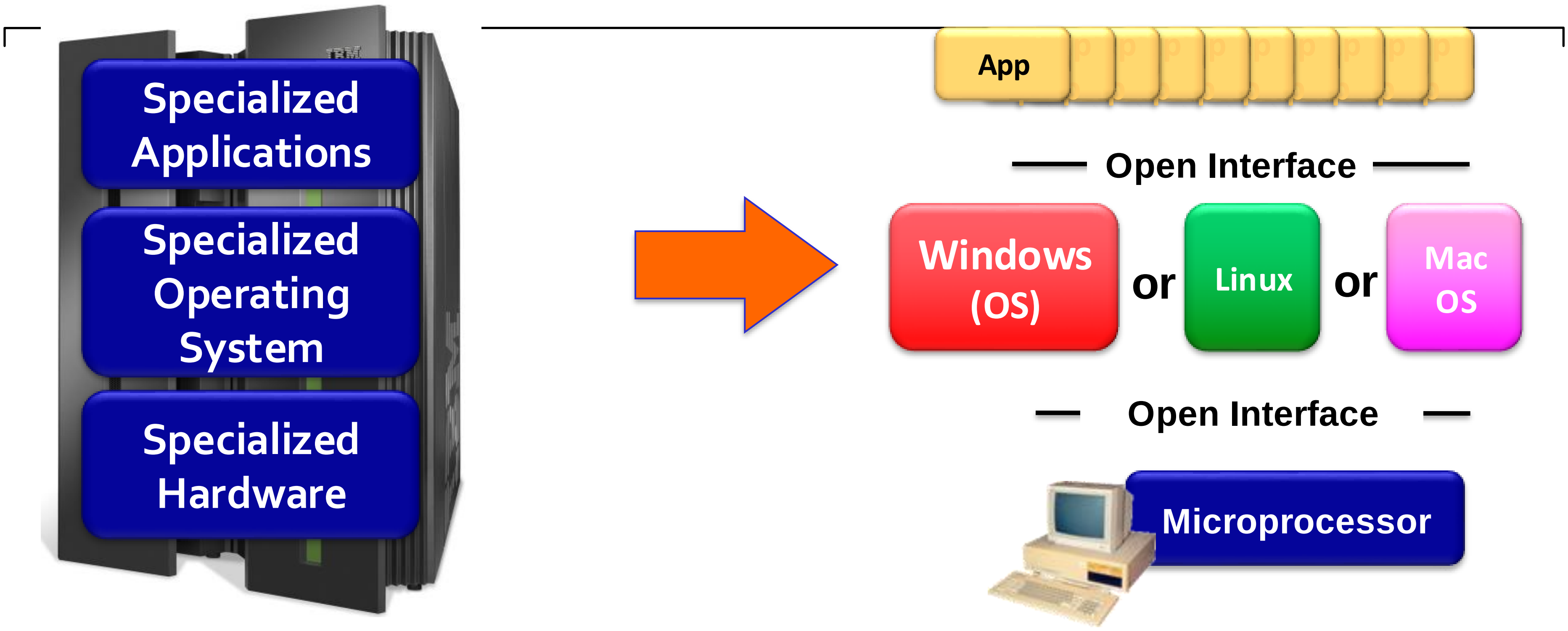
easier network management: avoid router misconfigurations, greater flexibility of traffic flows

table-based forwarding allows "programming"

- centralized "programming" easier: compute tables centrally and distribute
- distributed "programming: more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router

open (non-proprietary) implementation of control plane

Analogy: mainframe to PC evolution*



Specialized Applications

Specialized Operating System

Specialized Hardware

App

Open Interface

Windows (OS) or Linux or Mac OS

Open Interface

Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

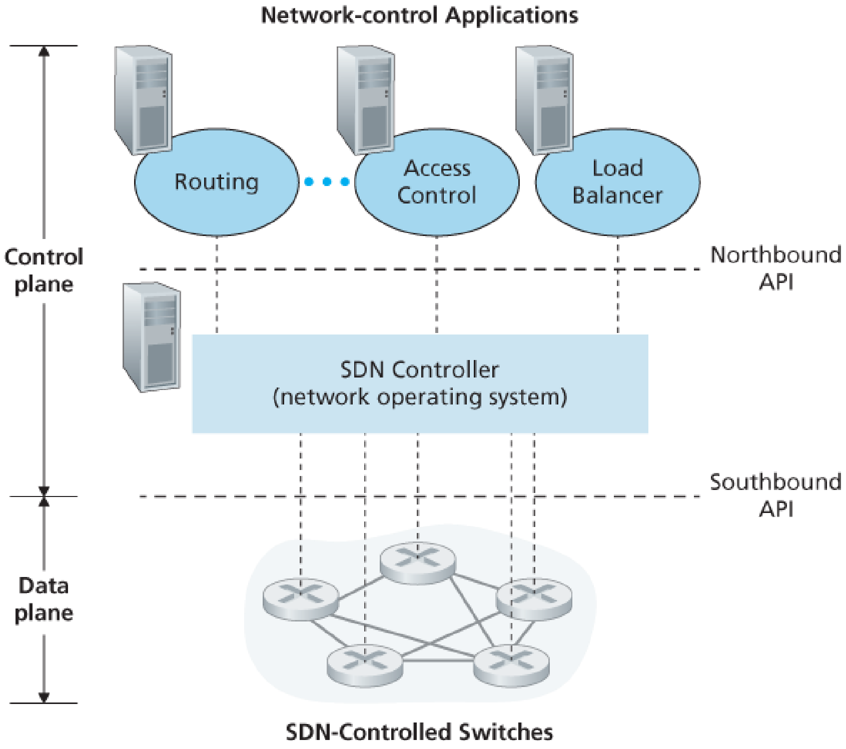
Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

SDN 4 architectural characteristics

1. Flow-based forwarding:
   - packet forwarding rules are specified in a switch's flow table (not only IP)
   - SDN control plane to compute, manage and install flow table entries in all of the network's switches.

2. Separation of data plane and control plane
   - Data plane consists of fast network's switches "match plus action" rules
   - Control plane consists of servers and software → switches' flow tables.



3. Network control functions: external to data-plane switches
   - SDN control plane is implemented in software that runs on distinct and remote servers
     - SDN controller that maintains accurate network state information (e.g., the state of remote links, switches, and hosts)
     - Network-control applications that monitor, program, and control the underlying network devices.

4. A programmable network
   - Programmable applications representing the "brains" of the SDN control plane
   - APIs to specify and control the data plane in the network devices.
     - Dijkstra's algorithm to determine end-end paths between sources and destinations
     - Access control, i.e., determine which packets are to be blocked at a switch
     - Server load balancing

Data plane switches

Data plane switches

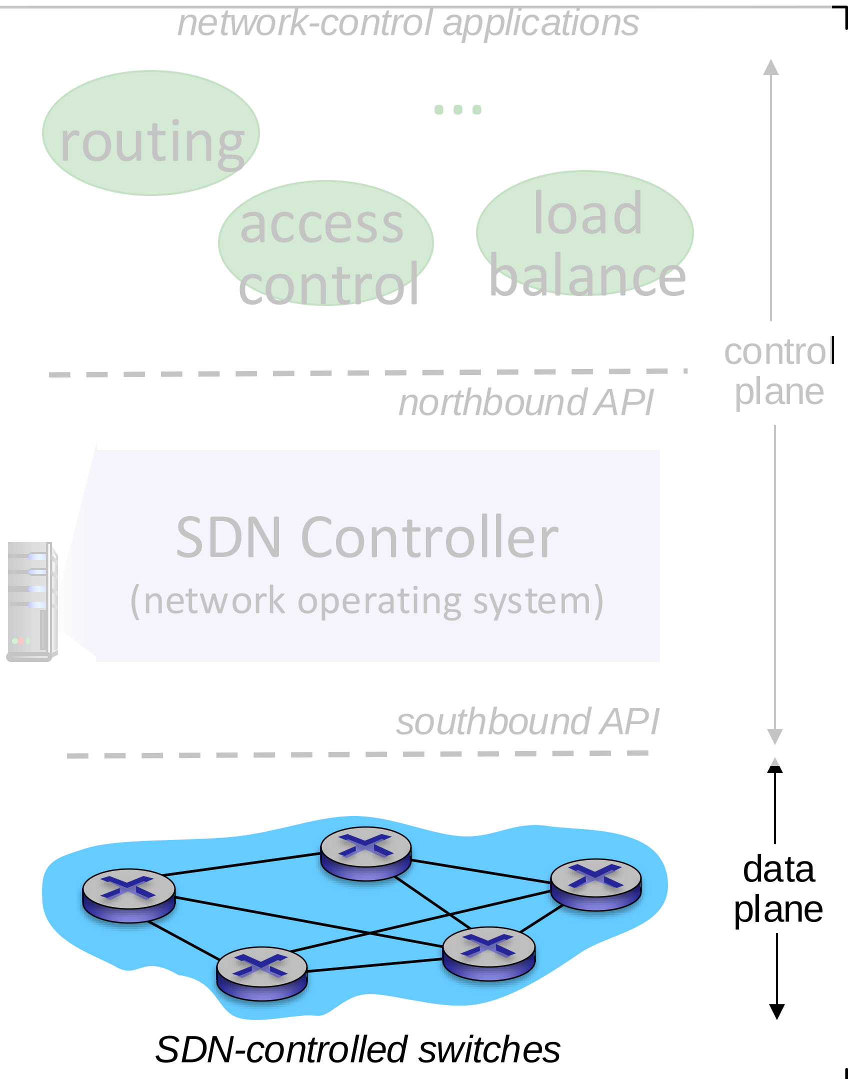
fast, simple, commodity switches implementing generalized data-plane forwarding in hardware

switch flow table computed, installed by controller

API for table-based switch control (e.g., OpenFlow)

- defines what is controllable and what is not

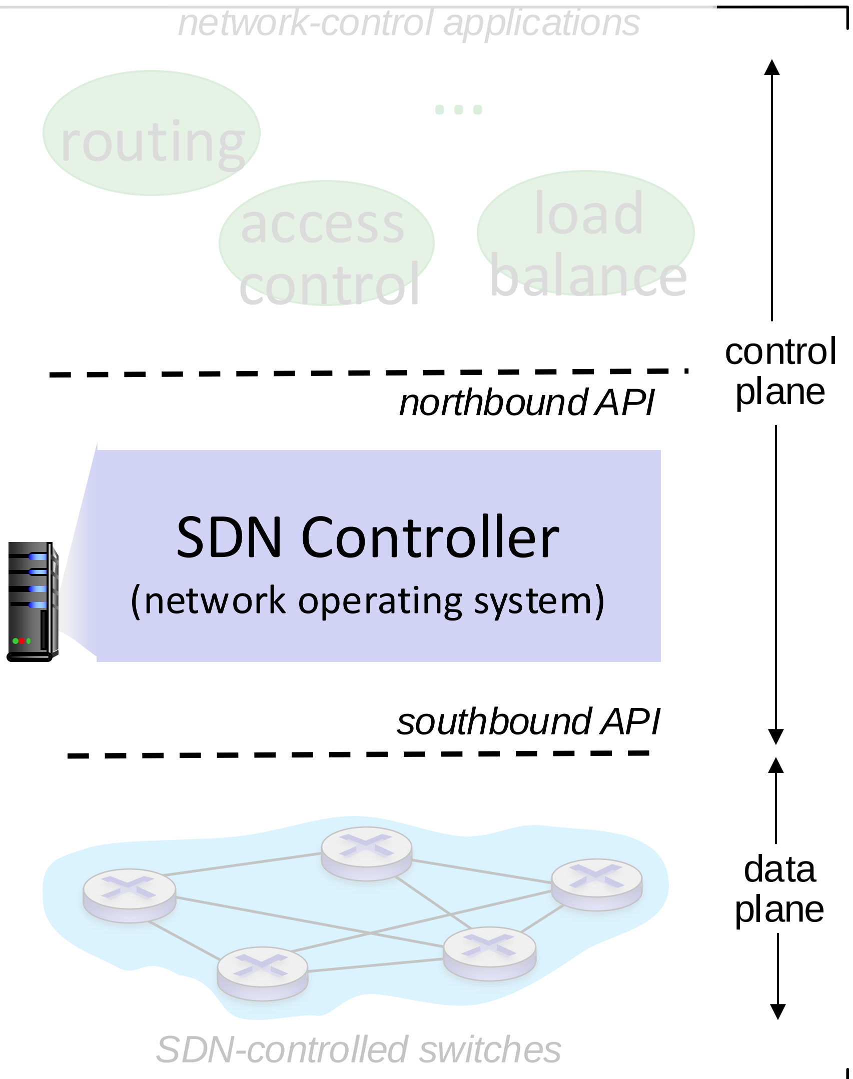
protocol for communicating with controller (e.g., OpenFlow)



SDN controller

SDN controller (network OS):

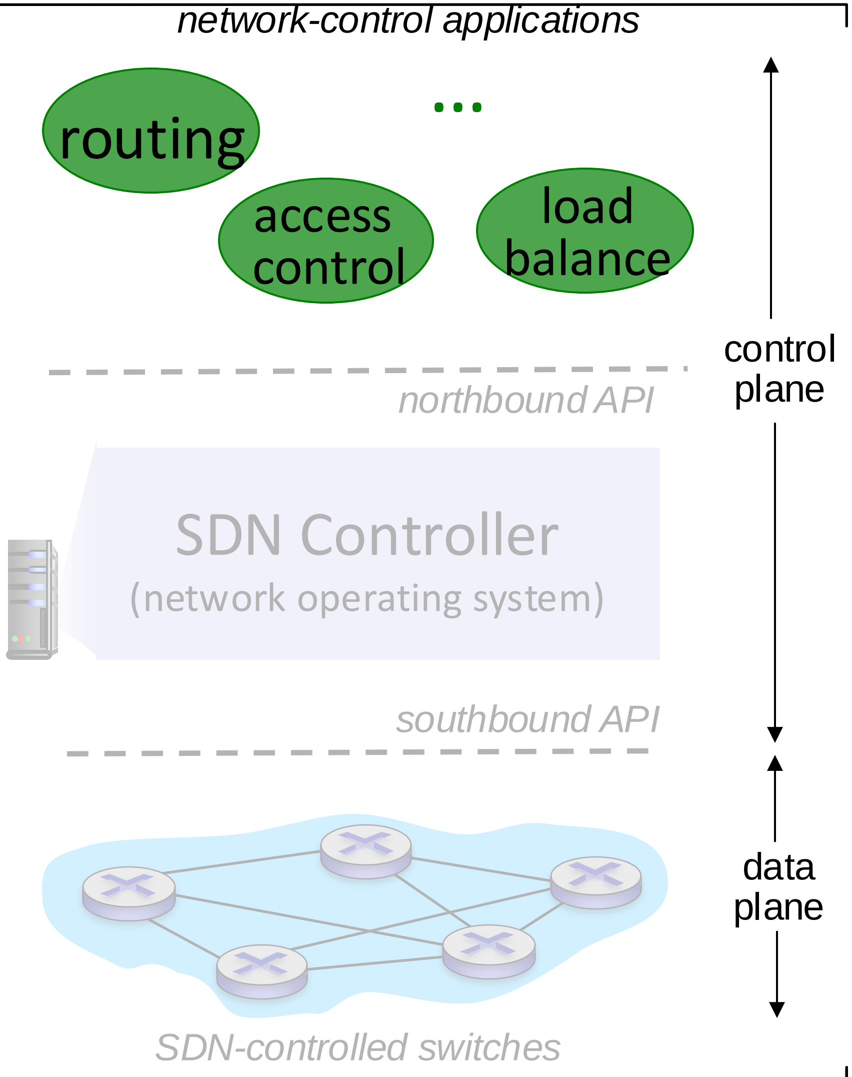
- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness
- Proprietary SDN: ONIX, Contrail, Google's controller
- Open Source SDN: ODL, ONOS



SDN control applications

network-control apps:

- "brains" of control: implement control functions using lower-level services, API provided by SND controller
- *unbundled:* can be provided by 3rd party: distinct from routing vendor, or SDN controller

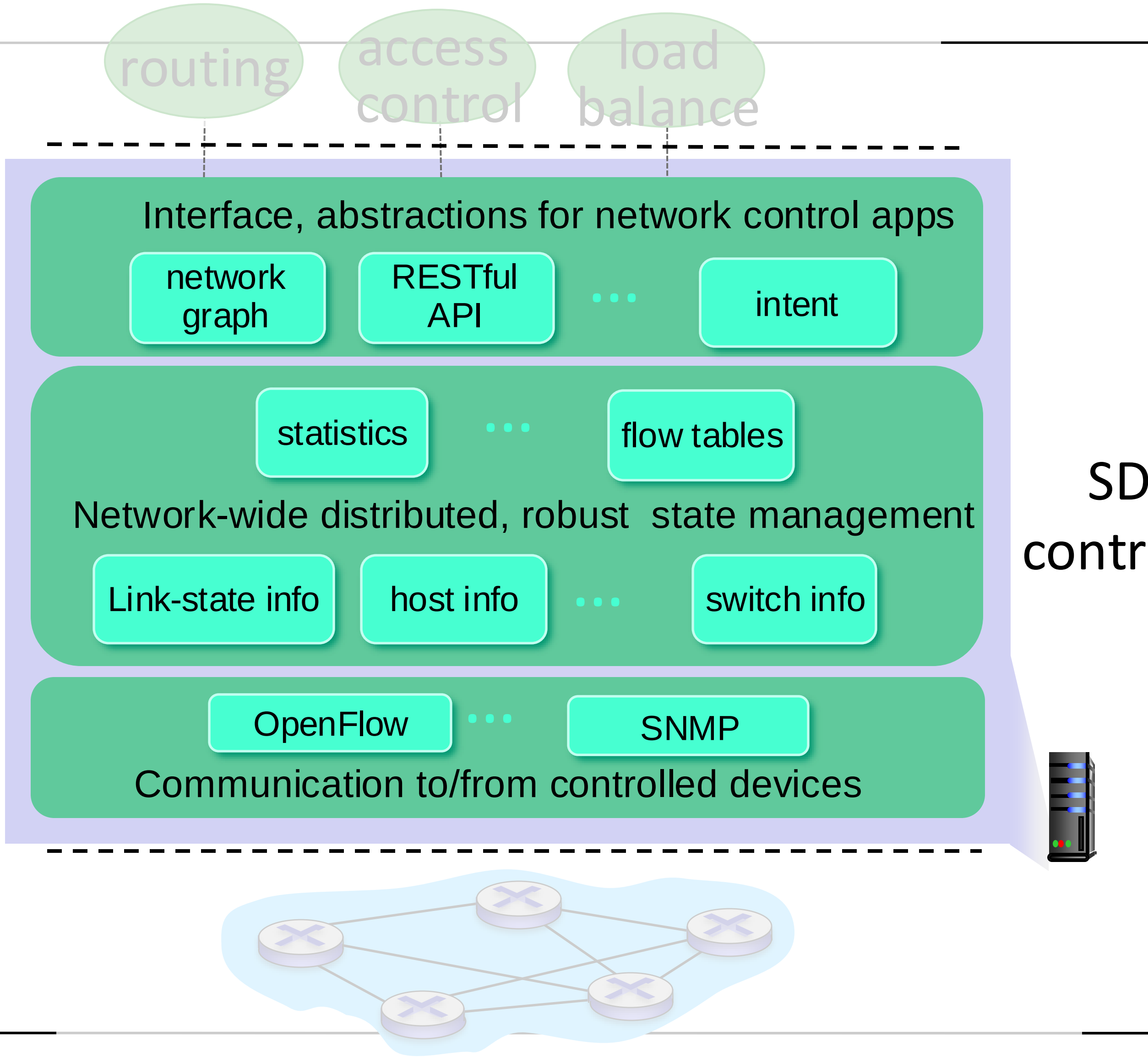


Components of SDN controller

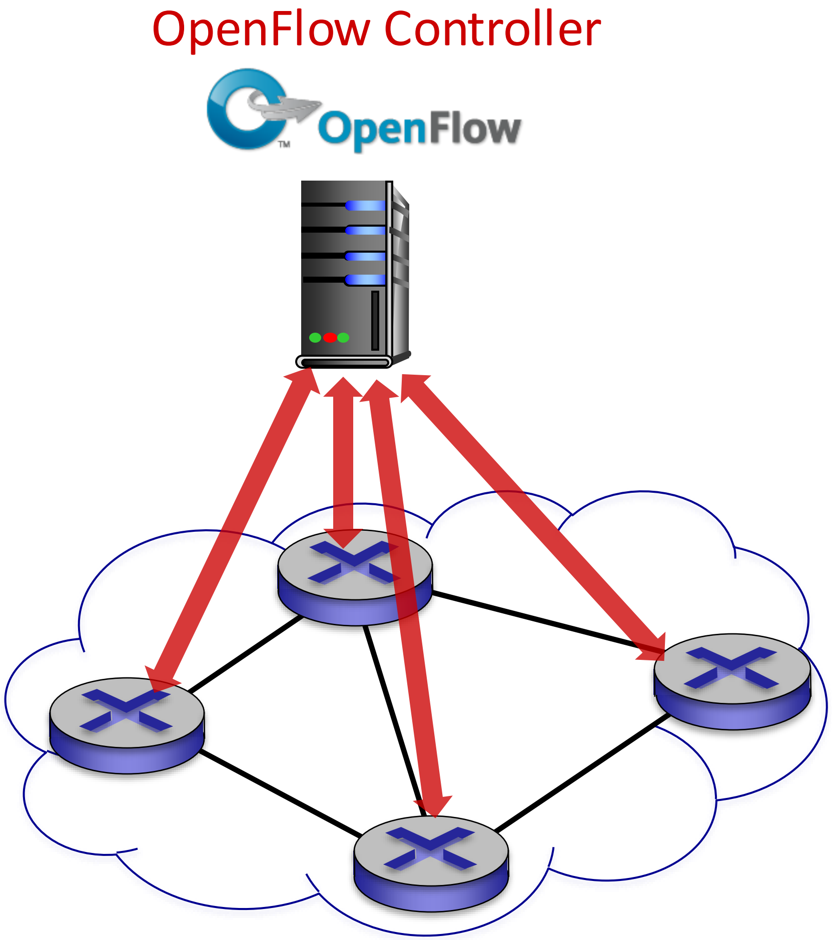
Interface layer to network control apps: abstractions API

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

communication layer: communicate between SDN controller and controlled switches



OpenFlow protocol



Operates between controller, switch

TCP used to exchange messages

- optional encryption

OpenFlow messages:

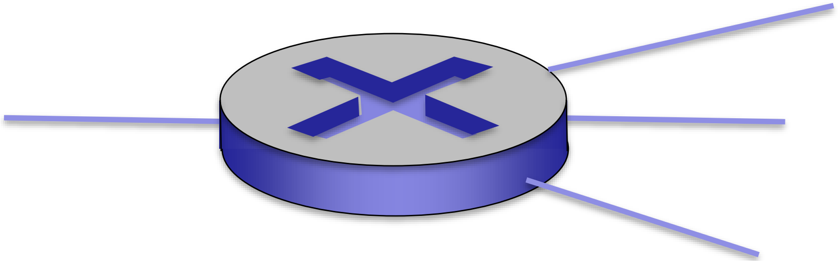
- controller-to-switch
- Switch-to-controller
- symmetric (in either direction)

Recall (Lec14. OpenFlow data plane abstraction)

OpenFlow (OF): SDN standards and the flow is defined by header fields

generalized forwarding: simple packet-handling rules which allows router to perform IP forwarding as well as a rich set of other functions (firewalling, NAT,..), traditionally implemented in separate devices.

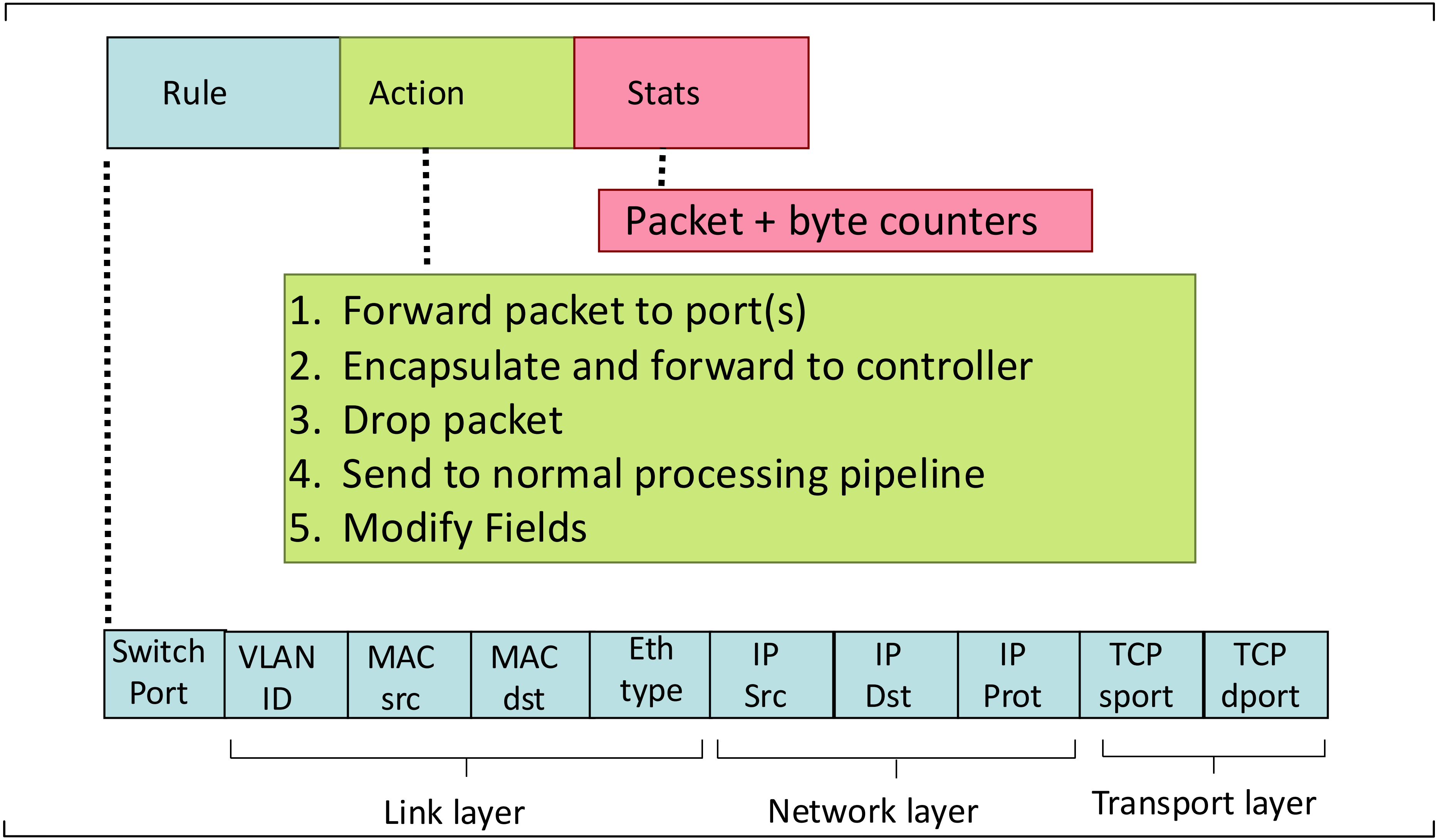
- *Pattern:* match values in packet header fields
- *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller



Flow table in a router (computed and distributed by controller) define router's match + action rules

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

Recall (Lec.14 OpenFlow: Flow Table Entries)



Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

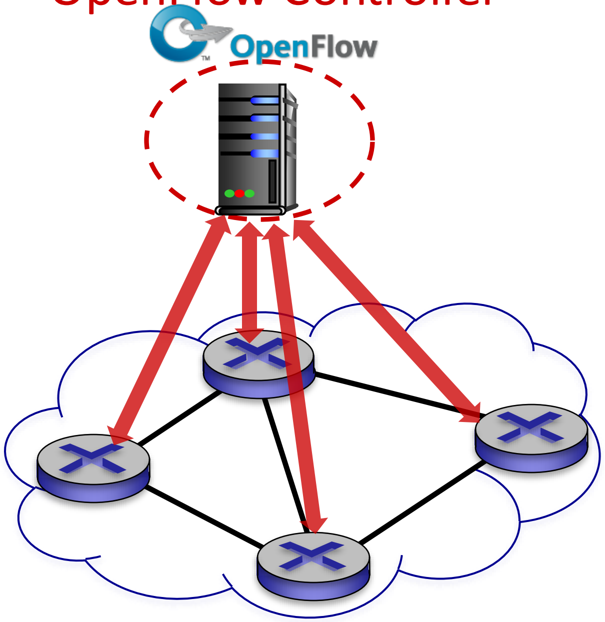
Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

OpenFlow: controller-to-switch messages

Key controller-to-switch messages

- *read-state:* controller queries switch to collect statistics from switch flow table, ports
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *send-packet:* controller send this packet out of specific switch port

OpenFlow Controller



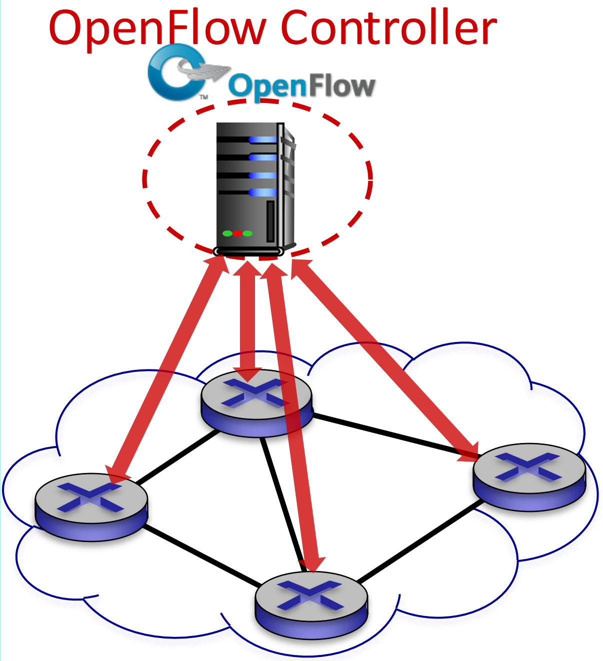
OpenFlow: switch-to-controller messages

Key switch-to-controller messages

packet-in: not-matched packet at switch port transferred to controller for further processing

flow-removed: flow table entry deleted at switch

port status: inform controller of a change in a port status



OpenFlow: symmetric messages

Either direction messages (no solicitation)

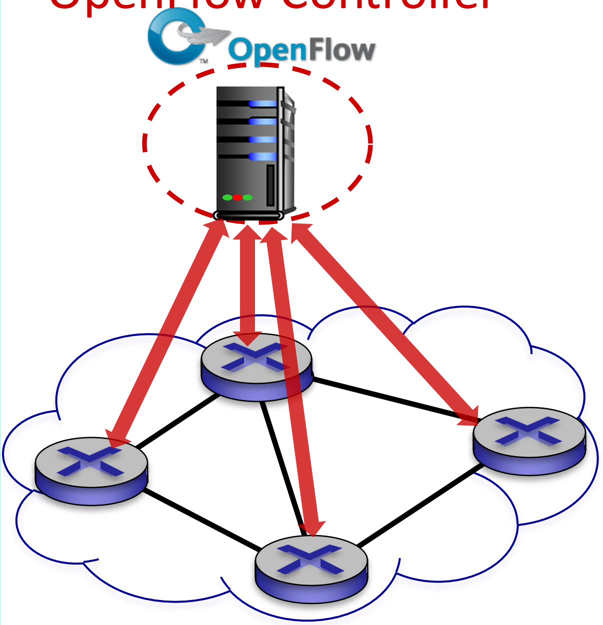
hello: exchanged between the switch and controller upon connection startup.

echo: verify the liveness of a controller-switch connection, and may as well be used to measure its latency or bandwidth

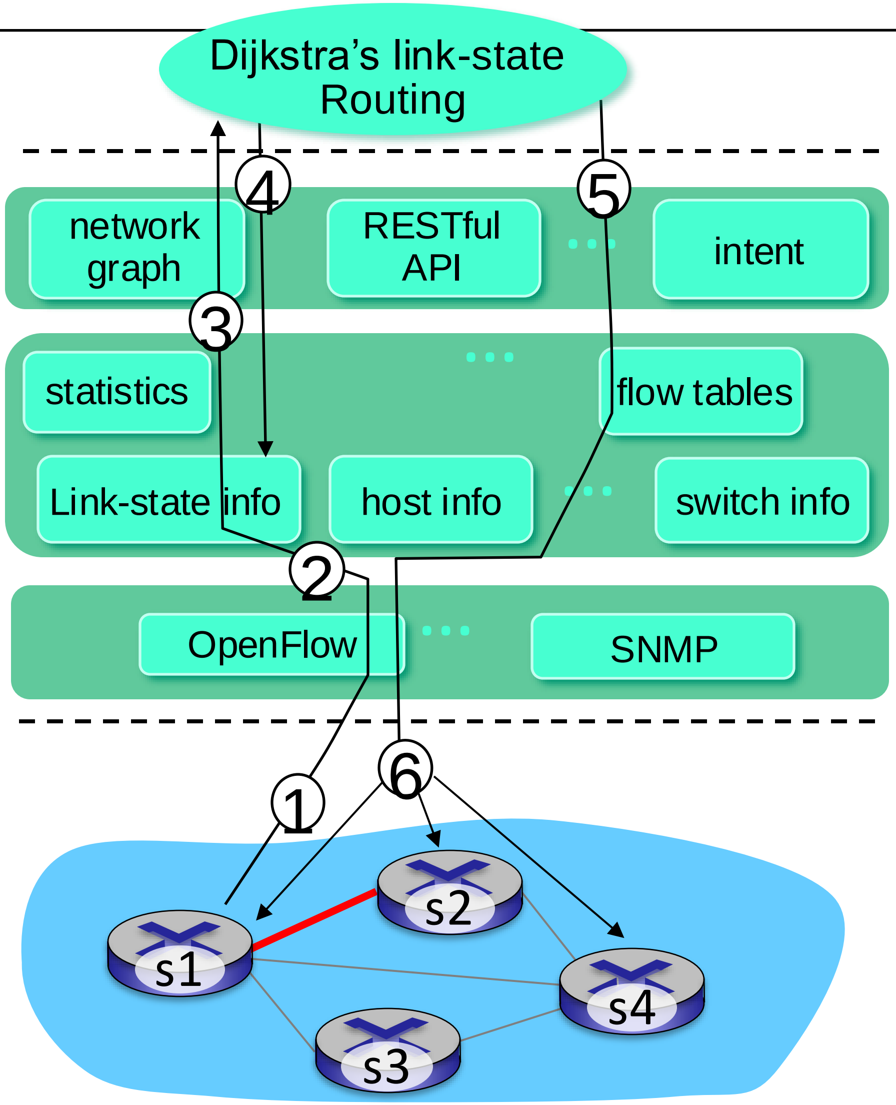
error: notify the other side of the connection of problems

experimenter: offer additional functionality within the OpenFlow message type space.

OpenFlow Controller



SDN: control/data plane interaction example



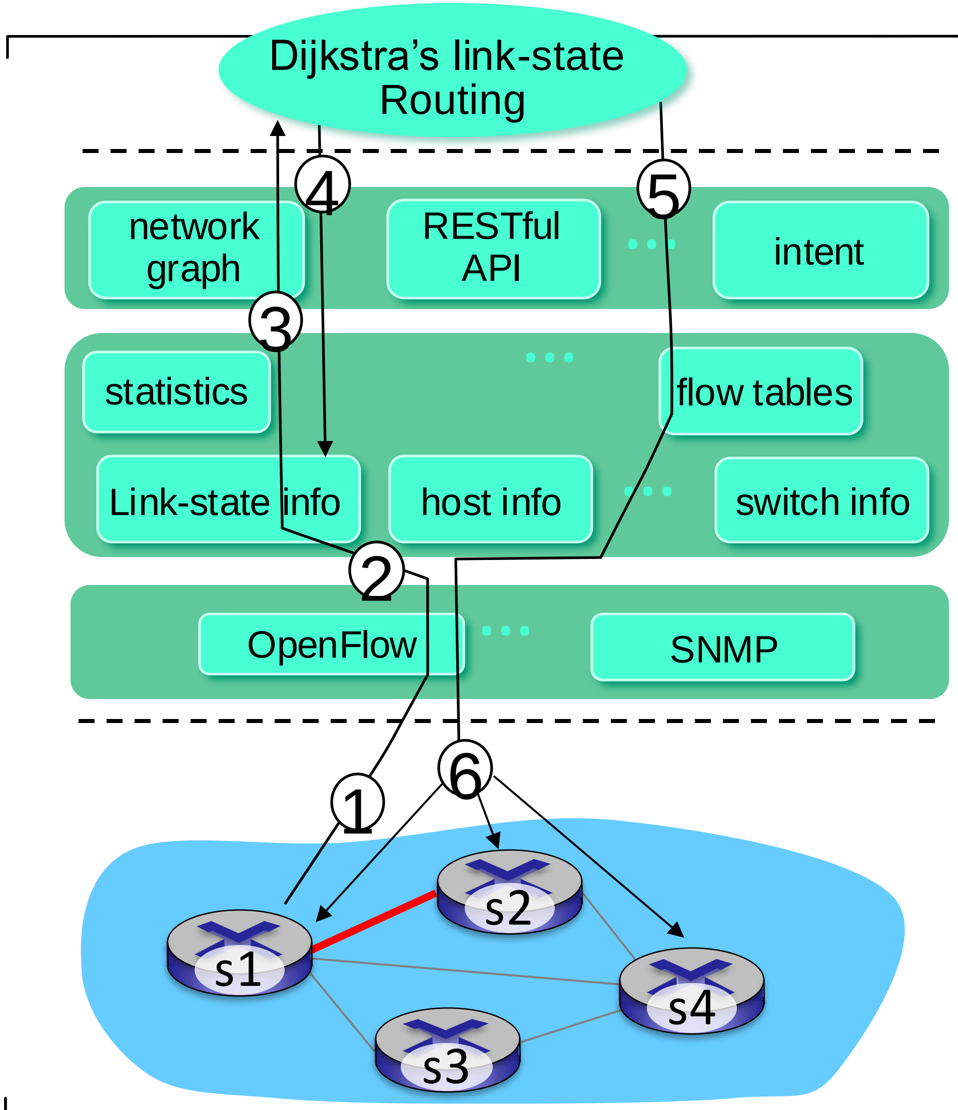
① S1, experiencing link failure using OpenFlow port status message to notify controller

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

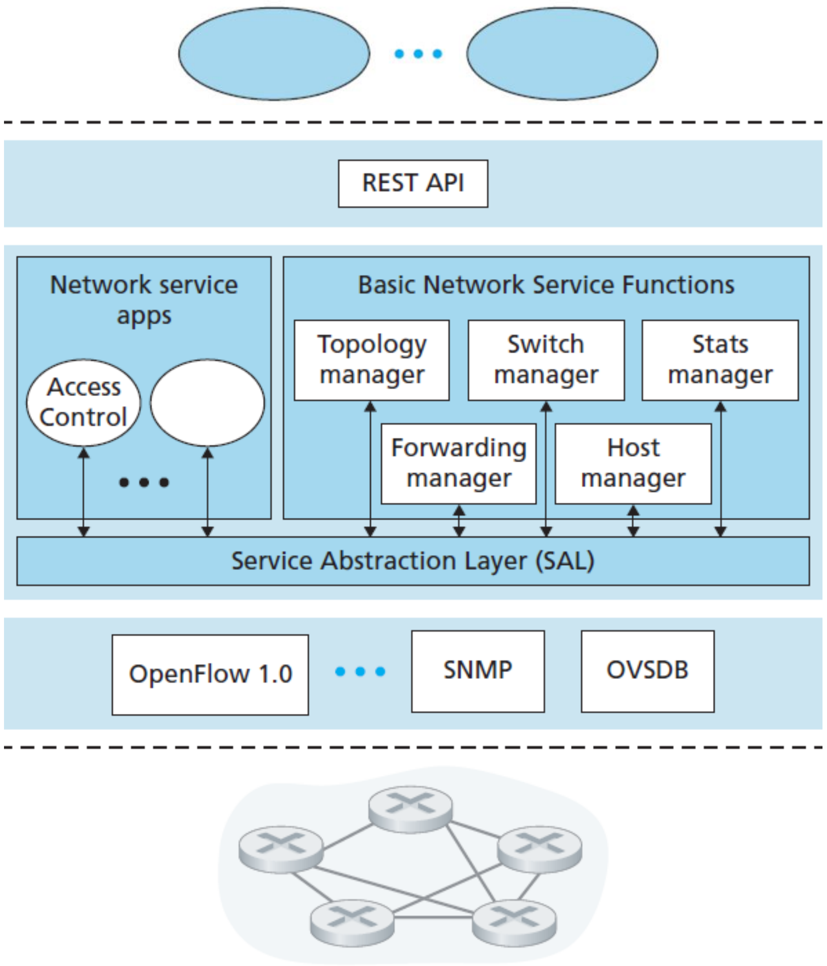
SDN: control/data plane interaction example



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

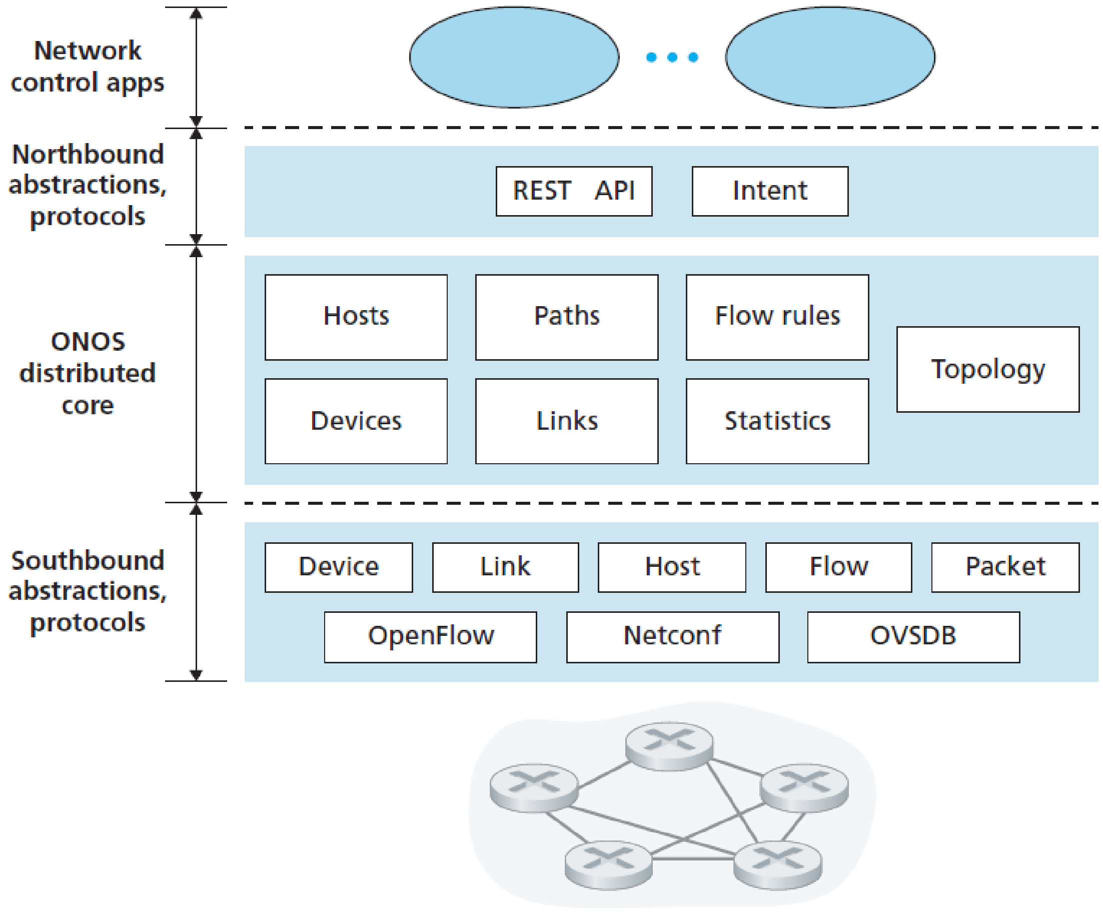
⑥ Controller uses OpenFlow to install new tables in switches that need updating

OpenDaylight (ODL) controller



- network apps may be contained within, or be external to SDN controller
- Service Abstraction Layer: interconnects internal, external applications and services

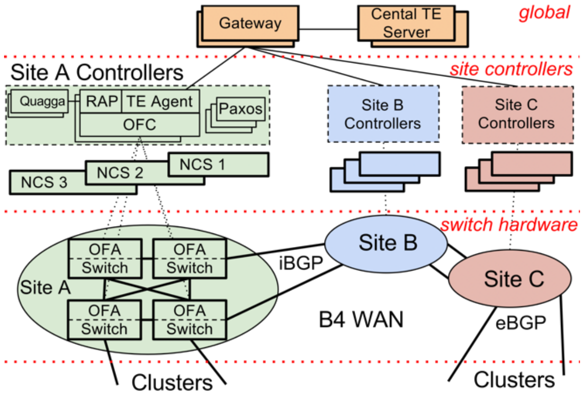
ONOS controller

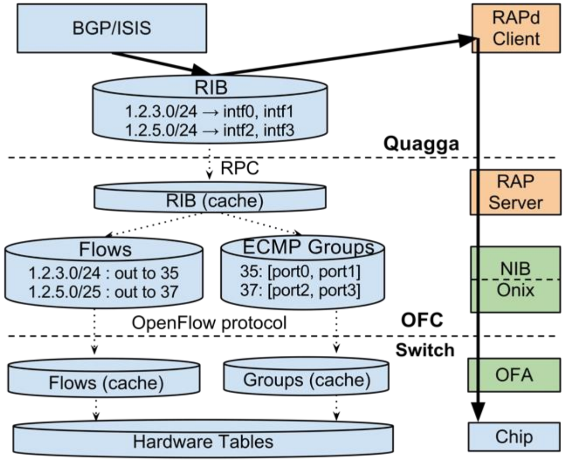


- control apps separate from controller
- high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

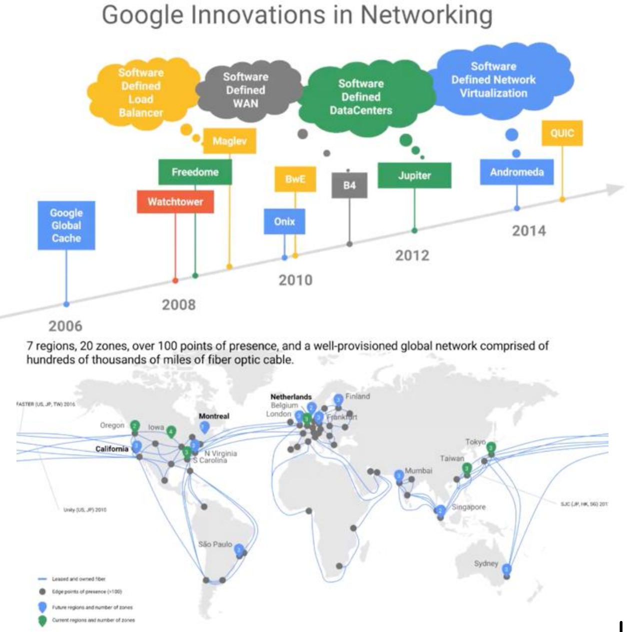
Google’s Software-Defined Global Network

- ONIX SDN controller
- Open Flow Controller (OFC) in the network control server (NCS)
- Custom switches:
  - B4 128-port 10GE switches in two-stage topology with a copper backplane switch built from 24 individual 16x10GE non-blocking switch chips
- Extended version of OpenFlow, with a local Open Flow Agent (OFA)
- Two routing protocols, BGP (for routing between the data centers) and IS-IS (a close relative of OSPF, for routing within a data center)





Source: Sushant Jain *et al*, “B4: Experience with a Globally-Deployed Software Defined WAN”, Online: https://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf


Google Innovations in Networking

7 regions, 20 zones, over 100 points of presence, and a well-provisioned global network comprised of hundreds of thousands of miles of fiber optic cable.

SDN: selected challenges

hardening the control plane: dependable, reliable, performance-scalable, secure distributed system

- robustness to failures: leverage strong theory of reliable distributed system for control plane
- dependability, security

networks, protocols meeting mission-specific requirements

- e.g., real-time, ultra-reliable, ultra-secure

Internet-scaling

Summary

Today:

- SNMP, ICMP
- SDN
- Data plane
- Control plane

Canvas discussion:

- Reflection
- Exit ticket

Next time:

- read 6.1 and 6.2 of KR (Data Link layer, Error detection/ correction)
- follow on Canvas! material and announcements

Any questions?