Software Defined Networking

Network Infrastructures A.A 2020/21

Outline

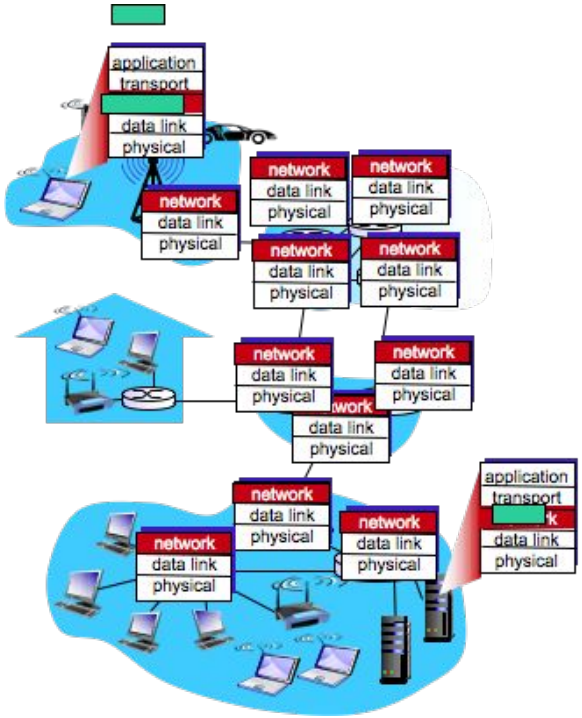
SAPIENZA
UNIVERSITÀ DI ROMA

- Control Plane and Data Plane
- Generalized Packet Forwarding
- Flow Table Entry
- Software Defined Networking
- OpenFlow Protocol
- SDN control/data plane interaction

Network Layer

SAPIENZA
UNIVERSITÀ DI ROMA

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in every host, router
- router examines header fields in all IP datagrams passing through it



Two key network-layer functions

SAPIENZA UNIVERSITÀ DI ROMA

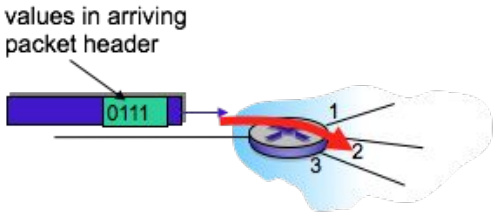
- network-layer functions:
  - **forwarding:** move packets from router's input to appropriate router output
  - **routing:** determine route taken by packets from source to destination
    - routing algorithms

- analogy: taking a trip
  - **forwarding:** process of getting through single interchange
  - **routing:** process of planning trip from source to destination

Network layer: data plane, control plane

SAPIENZA UNIVERSITÀ DI ROMA

- **Data plane**
  - local, per-router function
  - determines how datagram arriving on router input port is forwarded to router output port
  - forwarding function

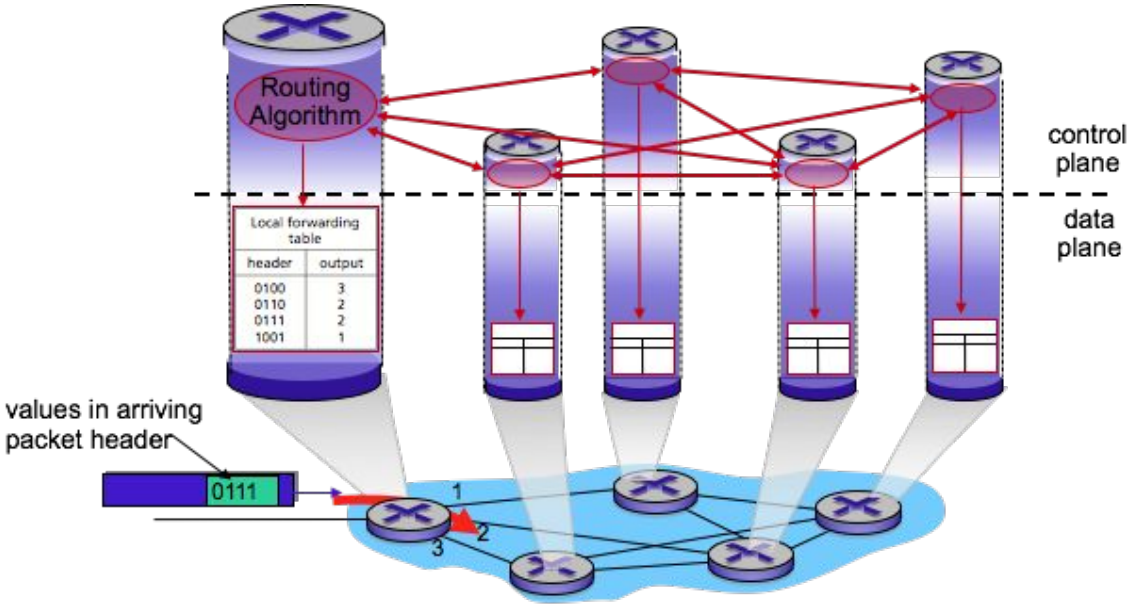


- **Control plane**
  - network-wide logic
  - determines how datagram is routed among routers along end-end path from source host to destination host
  - two control-plane approaches:
    - traditional routing algorithms: implemented in routers
    - software-defined networking (SDN): implemented in (remote) servers

Per-router control plane

SAPIENZA
UNIVERSITÀ DI ROMA

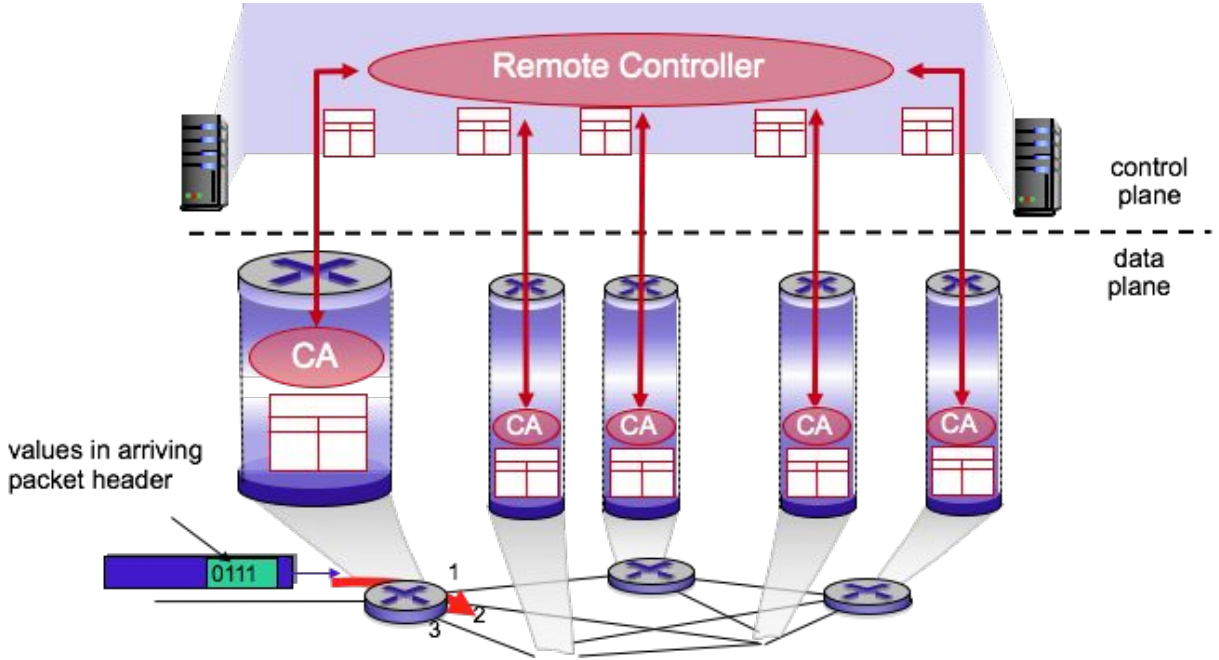
- Individual routing algorithm components in each and every router interact in the control plane



Logically centralized control plane

SAPIENZA
UNIVERSITÀ DI ROMA

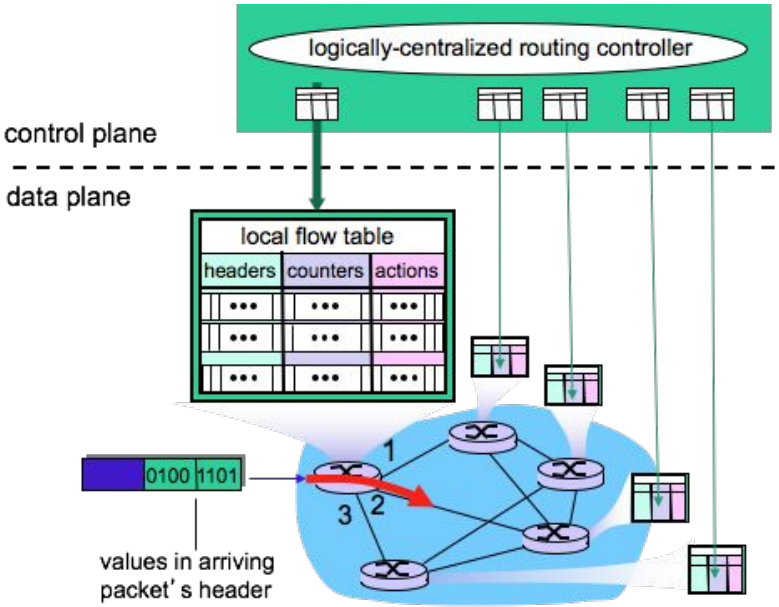
- A distinct (typically remote) controller interacts with local control agents (CAs)



Generalized Forwarding and SDN

SAPIENZA
UNIVERSITÀ DI ROMA

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



OpenFlow data plane abstraction

SAPIENZA
UNIVERSITÀ DI ROMA

- **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

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

OpenFlow data plane abstraction

SAPIENZA
UNIVERSITÀ DI ROMA

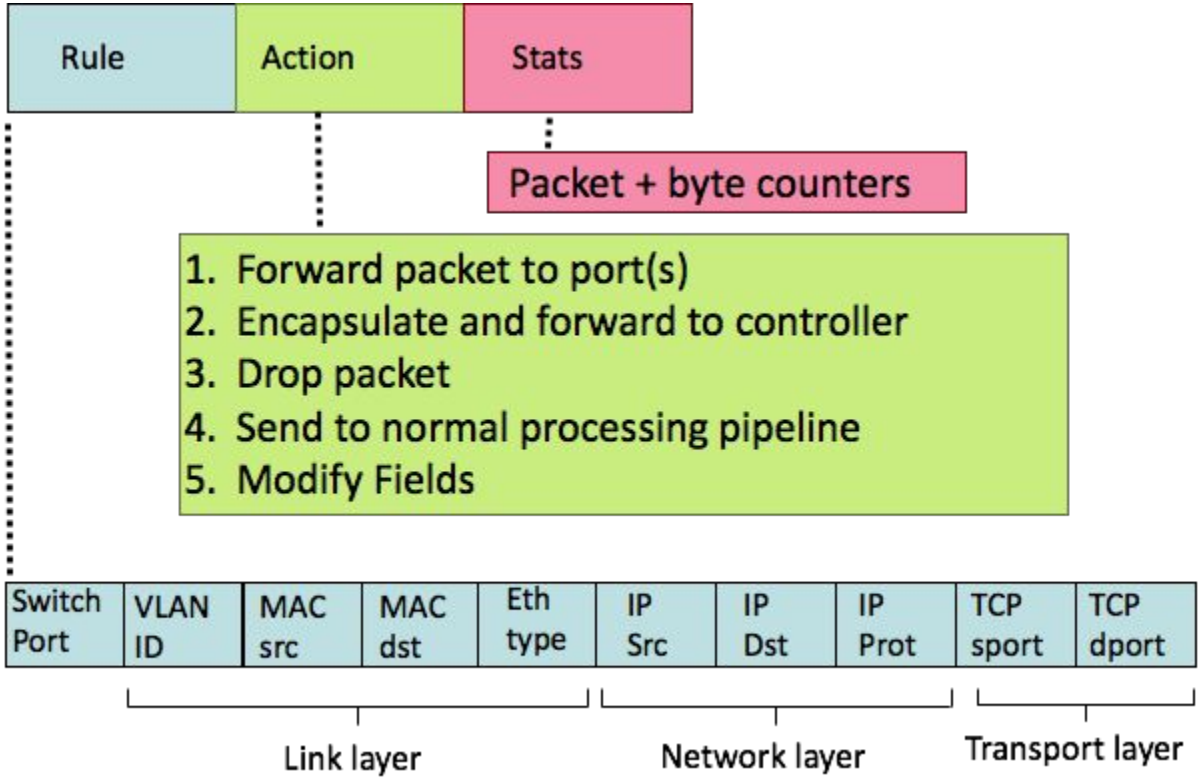
- **flow:** defined by header fields
- **generalized forwarding:** simple packet-handling rules
  - Pattern: match values in packet header fields
  - Actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - Priority: disambiguate overlapping patterns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

OpenFlow: Flow Table Entries

SAPIENZA
UNIVERSITÀ DI ROMA



Examples

SAPIENZA
UNIVERSITÀ DI ROMA

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

Examples

SAPIENZA
UNIVERSITÀ DI ROMA

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	*	*	*	*	*	*	*	*	port3

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

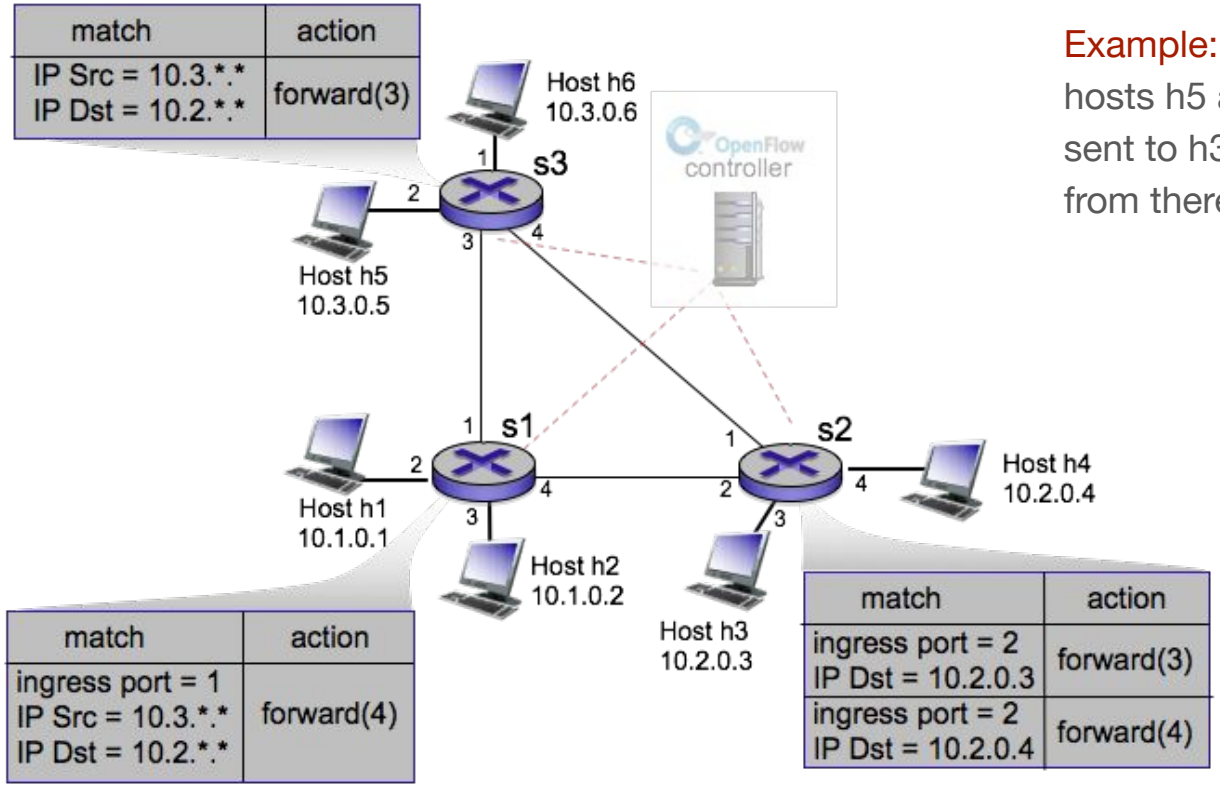
OpenFlow abstraction

SAPIENZA UNIVERSITÀ DI ROMA

- 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

OpenFlow example



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

Software defined networking (SDN)

SAPIENZA
UNIVERSITÀ DI ROMA

- 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

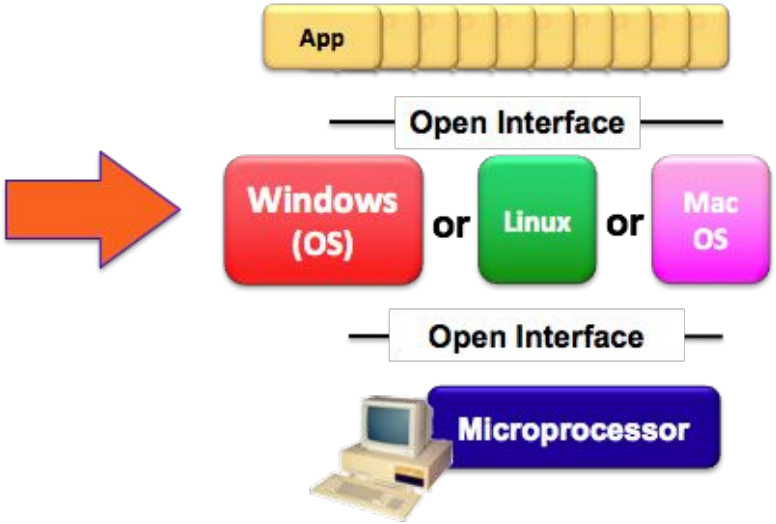
Software defined networking (SDN)

SAPIENZA
UNIVERSITÀ DI ROMA

- Why a logically centralized control plane?
  - easier network management: avoid router misconfigurations, greater flexibility of traffic flows
  - table-based forwarding (recall OpenFlow API) allows “programming” routers
    - 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

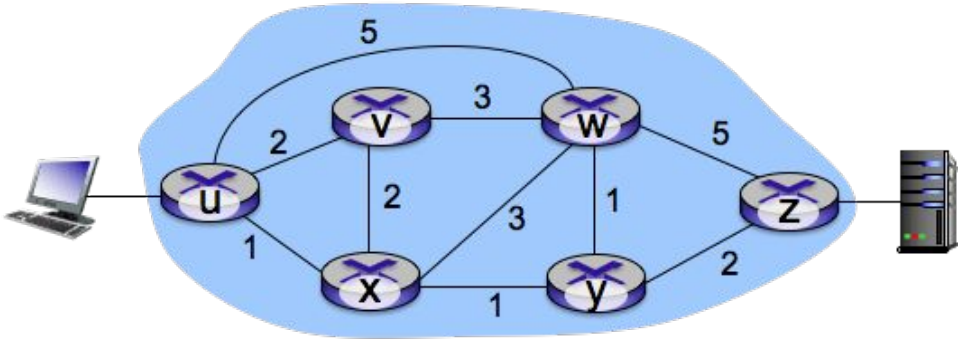
Vertically integrated
Closed, proprietary
Slow innovation
Small industry



Horizontal
Open interfaces
Rapid innovation
Huge industry

Traffic engineering: difficult traditional routing

SAPIENZA
UNIVERSITÀ DI ROMA



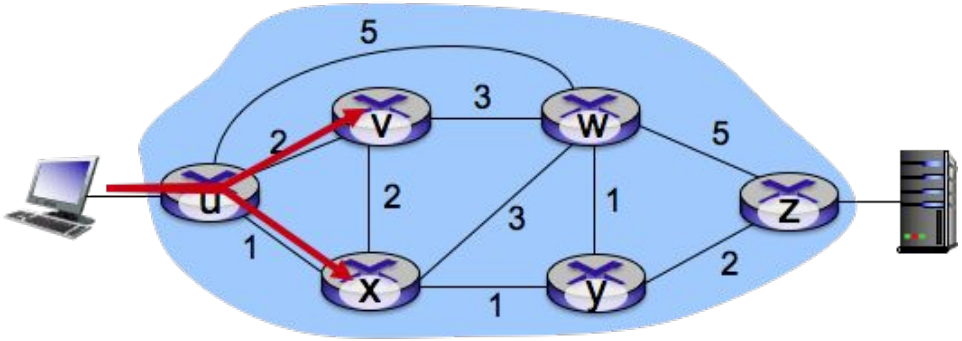
Q: what if network operator wants u-to-z traffic to flow along uvwz, x-to-z traffic to flow xwyz?

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

Link weights are only control “knobs”: wrong!

Traffic engineering: difficult

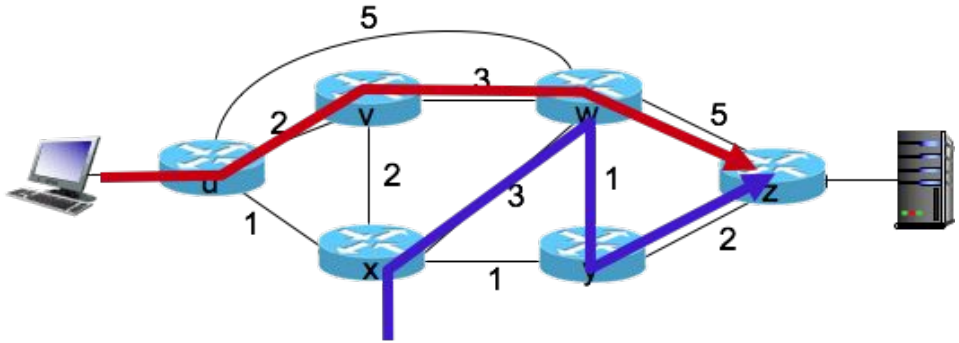
SAPIENZA
UNIVERSITÀ DI ROMA



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

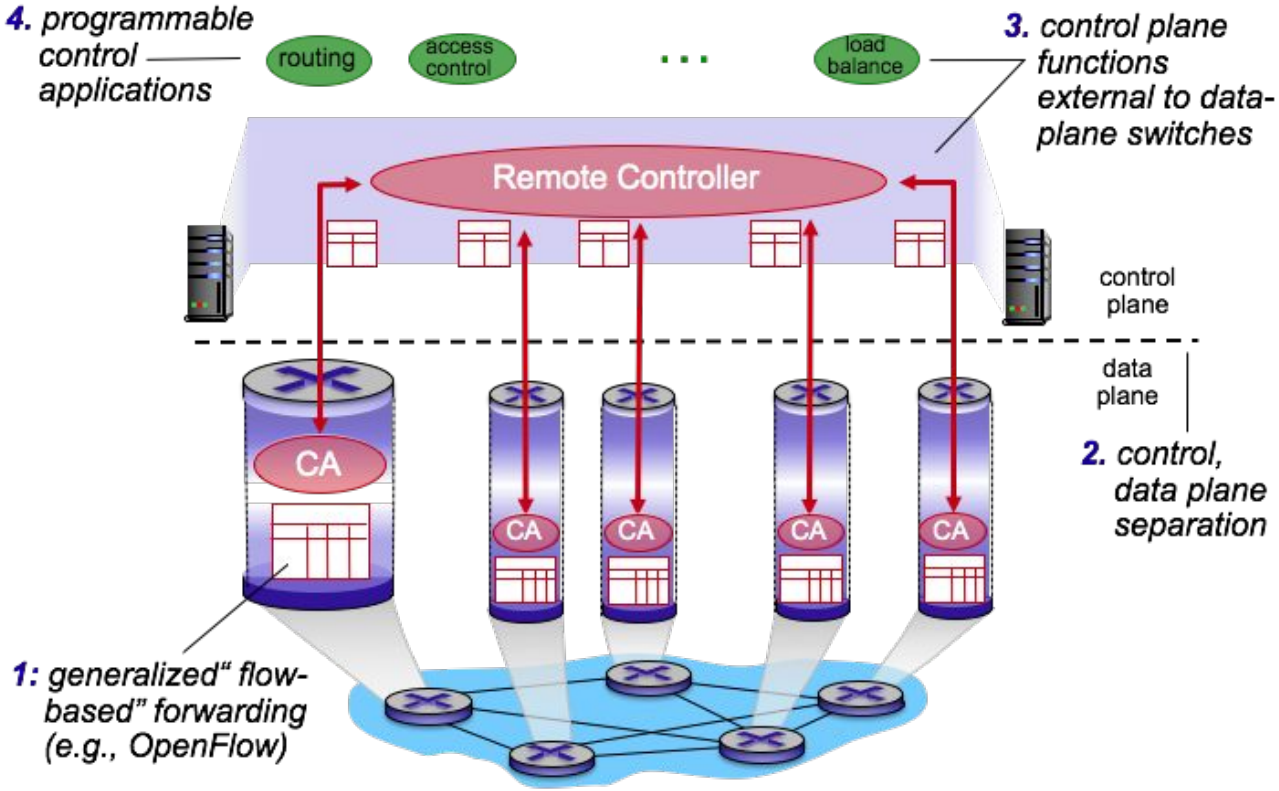


Q: what if w wants to route blue and red traffic differently?

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

Software defined networking (SDN)

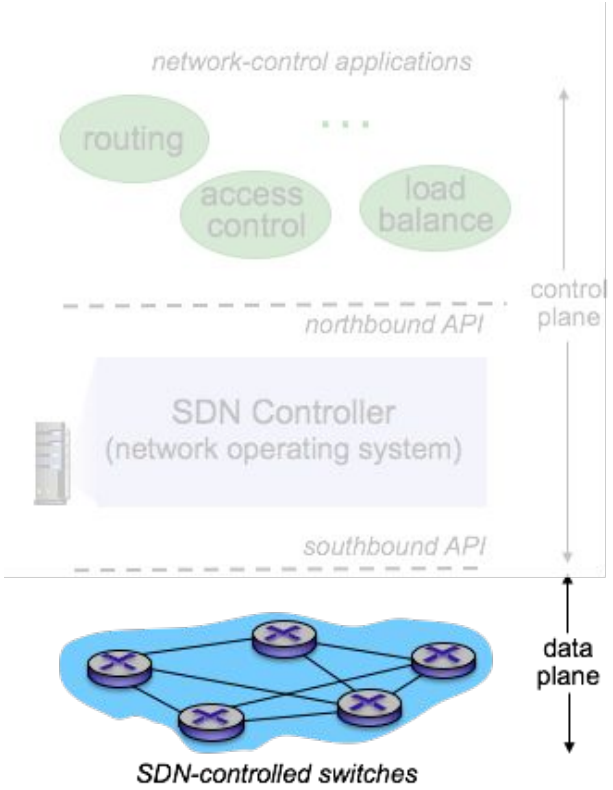
SAPIENZA
UNIVERSITÀ DI ROMA



SDN perspective: data plane switches

SAPIENZA UNIVERSITÀ DI ROMA

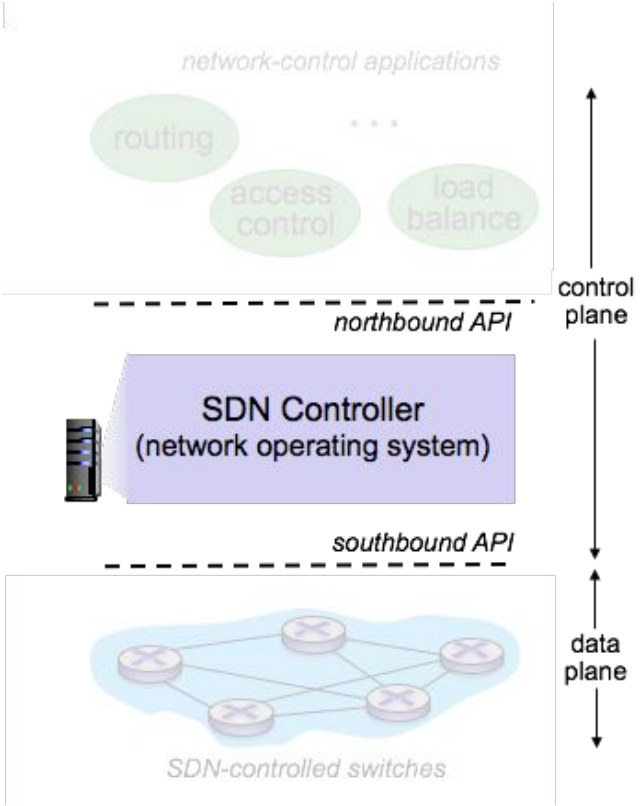
- Data plane switches
  - fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
  - switch flow table computed, installed by controller
  - API for table-based switch control (e.g., OpenFlow)
    - defines what is controllable and what is not
  - protocol for communicating with controller (e.g., OpenFlow)



SDN perspective: SDN controller

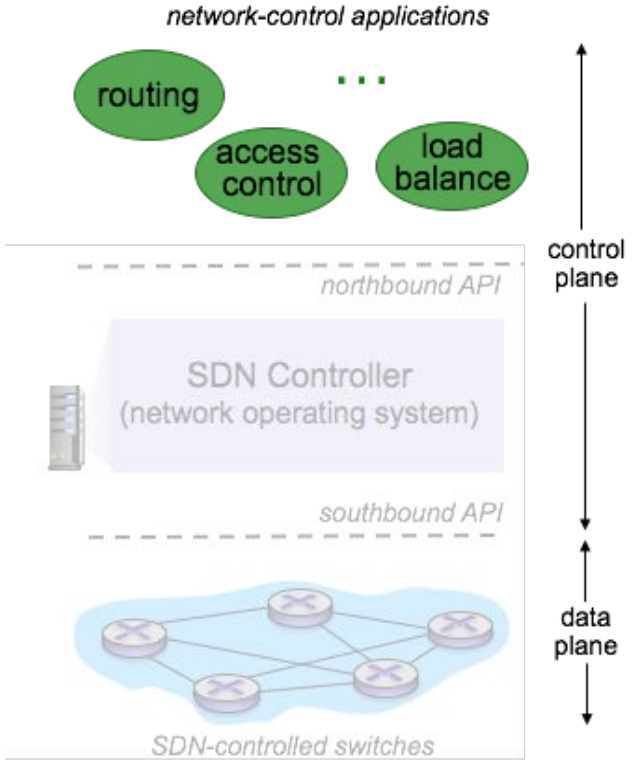
SAPIENZA
UNIVERSITÀ DI ROMA

- 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



SDN perspective: network apps

- 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



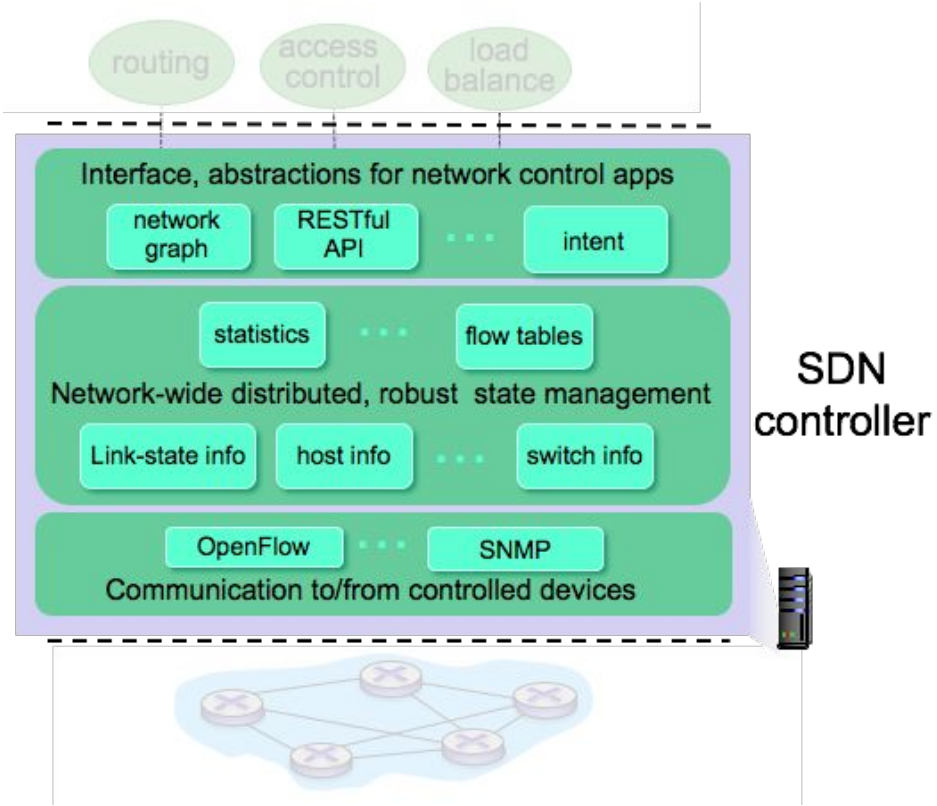
Components of SDN controller

SAPIENZA
UNIVERSITÀ DI ROMA

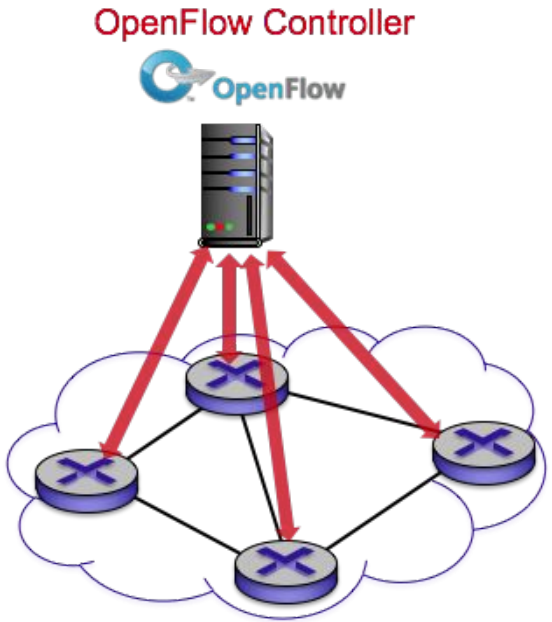
Interface layer to network control apps: abstractions API

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

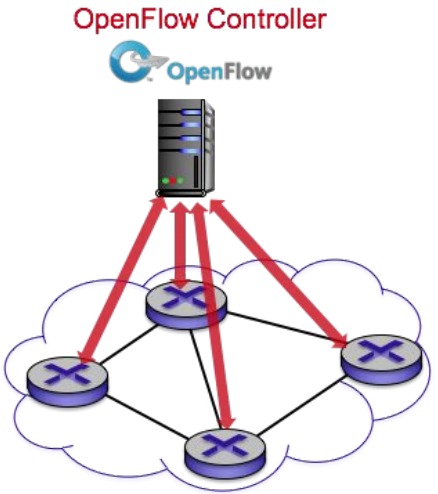


- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

OpenFlow: controller-to-switch messages

SAPIENZA UNIVERSITÀ DI ROMA

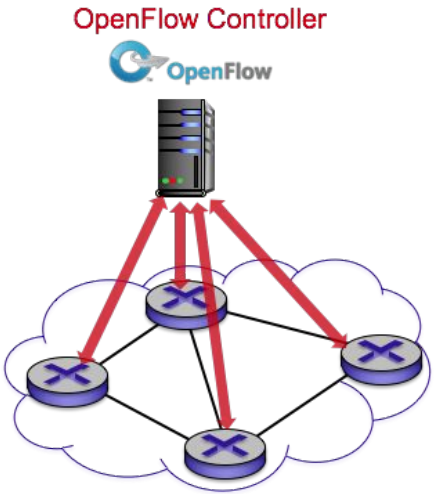
- Key **controller-to-switch** messages
  - features: controller queries switch features, switch replies
  - configure: controller queries/sets switch configuration parameters
  - modify-state: add, delete, modify flow entries in the OpenFlow tables
  - packet-out: controller can send this packet out of specific switch port



OpenFlow: switch-to-controller messages

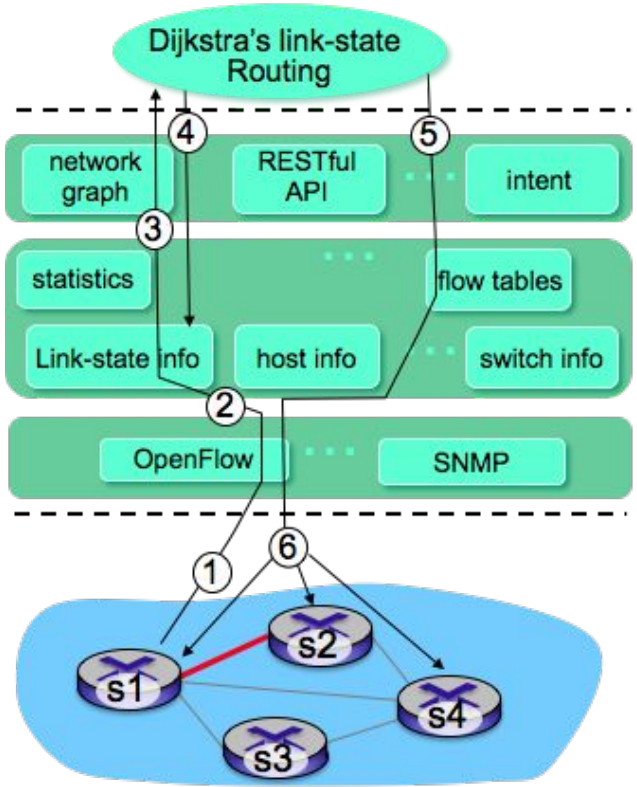
SAPIENZA UNIVERSITÀ DI ROMA

- Key **switch-to-controller** messages
  - packet-in: transfer packet (and its control) to controller. See packet-out message from controller
  - flow-removed: flow table entry deleted at switch
  - port status: inform controller of a change on a port.



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

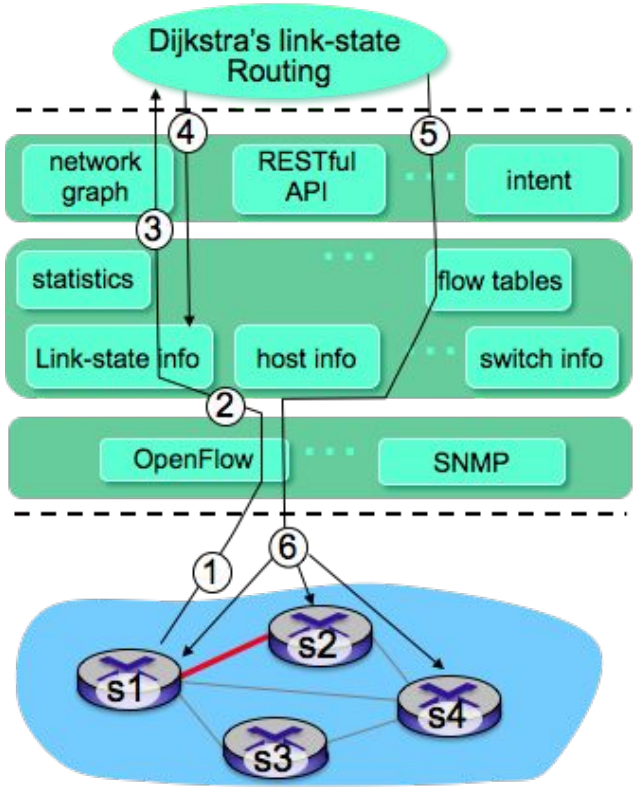
SDN control/data plane interaction example



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

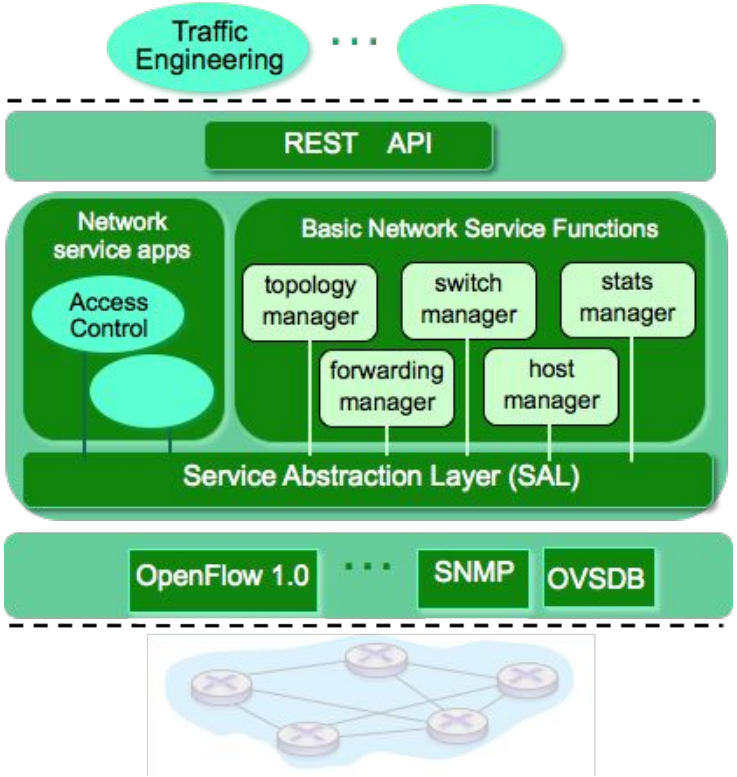
SDN control/data plane interaction example

SAPIENZA UNIVERSITÀ DI ROMA



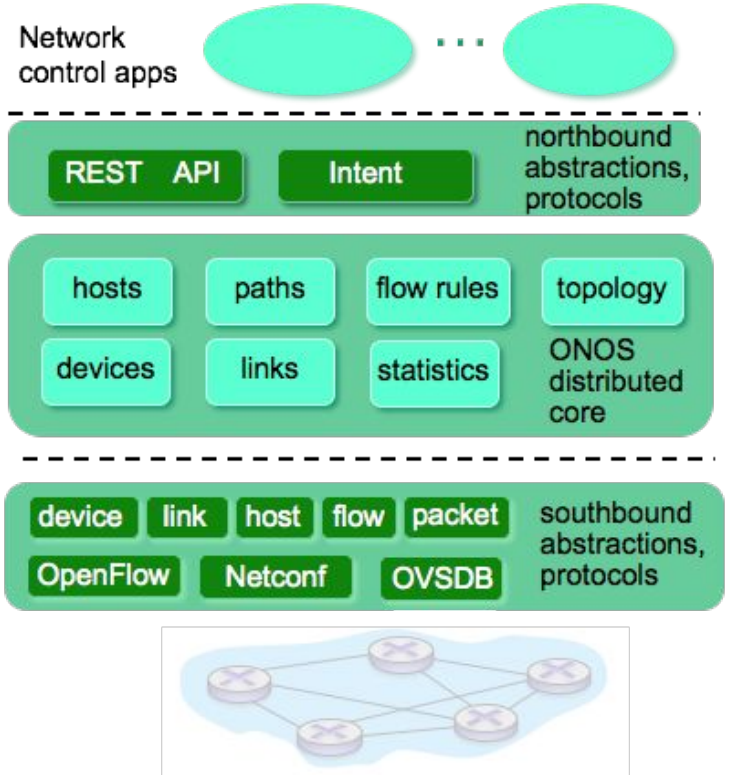
5. link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed
6. Controller uses OpenFlow to install new tables in switches that need updating

OpenDaylight Controller



- ODL Lithium 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
- intent framework: high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

SDN: selected challenges

SAPIENZA
UNIVERSITÀ DI ROMA

- 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: “baked in” from day one?
- networks, protocols meeting mission-specific requirements
  - e.g., real-time, ultra-reliable, ultra-secure
- Internet-scaling