AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

# 22AIE204
# COMPUTER NETWORKS

# NETWORK LAYER – Control Plane

- **Software Defined Networking (SDN)**

# Network layer: “control plane” roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

# Software defined networking (SDN)

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

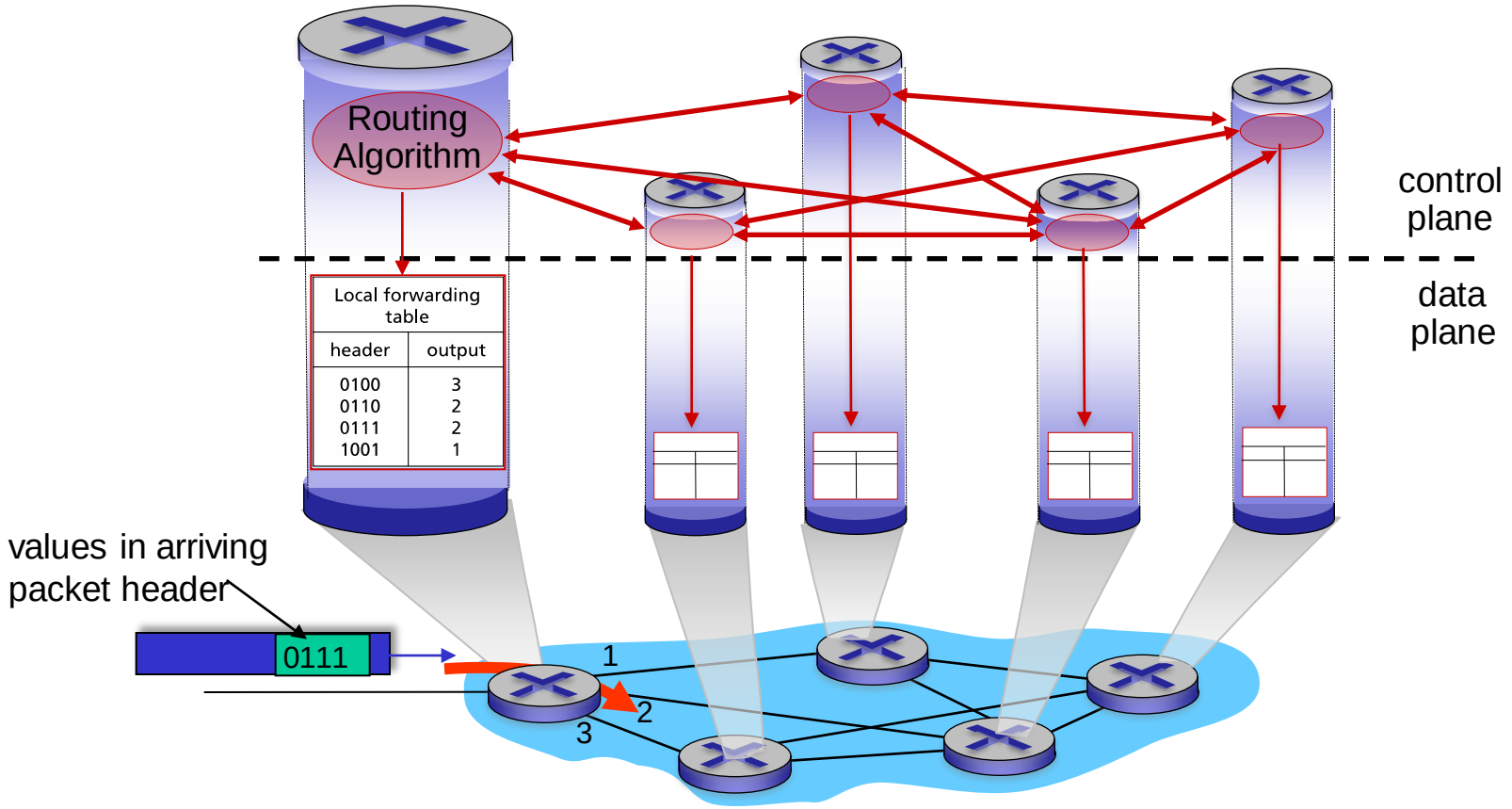
*monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ..

~2005: renewed interest in rethinking network control plane

- Primary goal of SDN – Open and Programmable

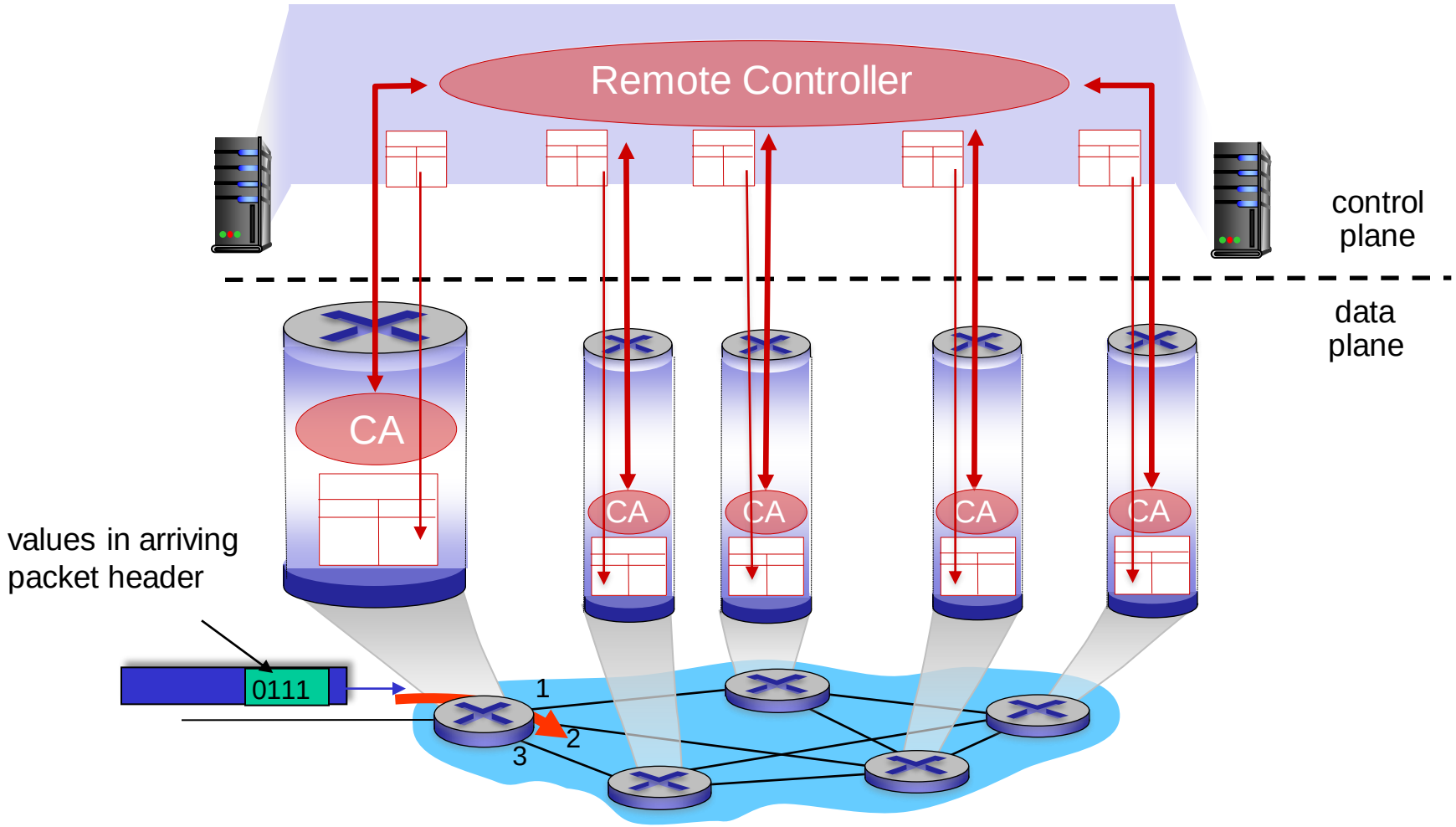
# Per-router control plane

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



# Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



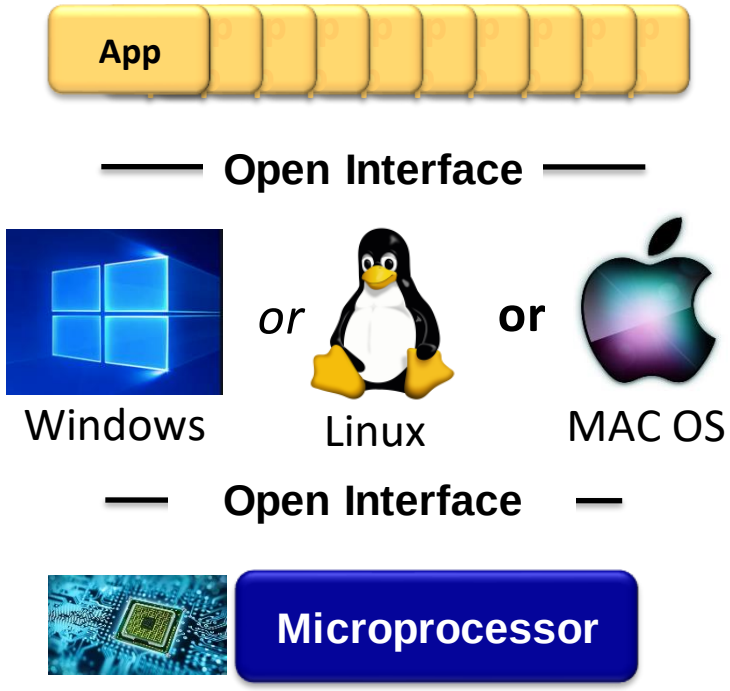
# Software defined networking (SDN)

*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
  - foster innovation: let 1000 flowers bloom

# SDN analogy: mainframe to PC revolution

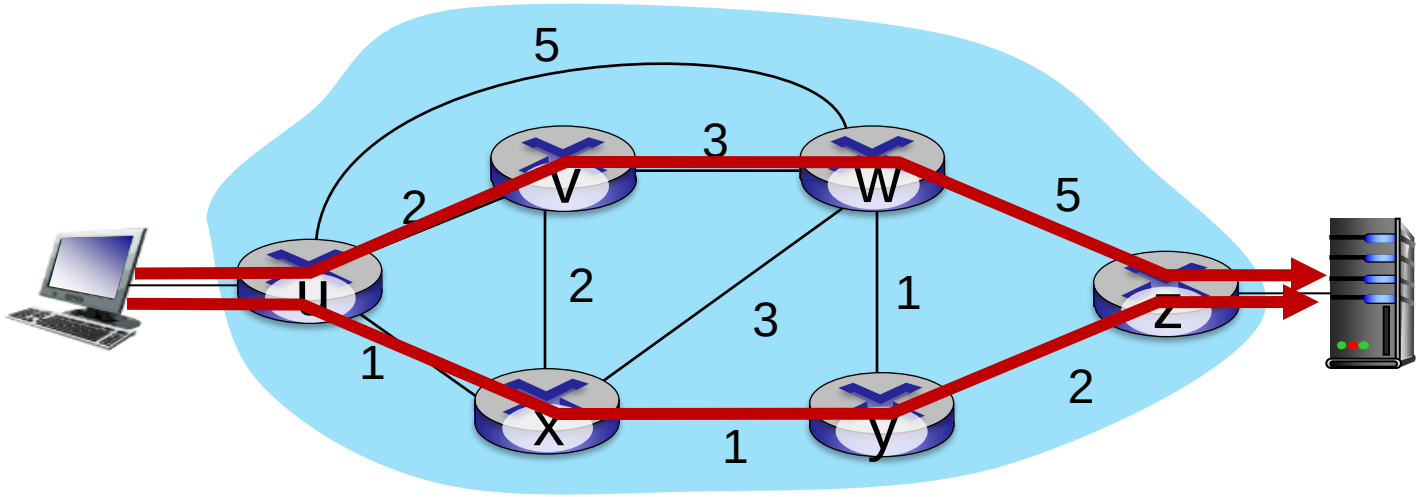

Vertically integrated
Closed, proprietary
Slow innovation
Small industry




Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

# Traffic engineering: difficult with traditional routing

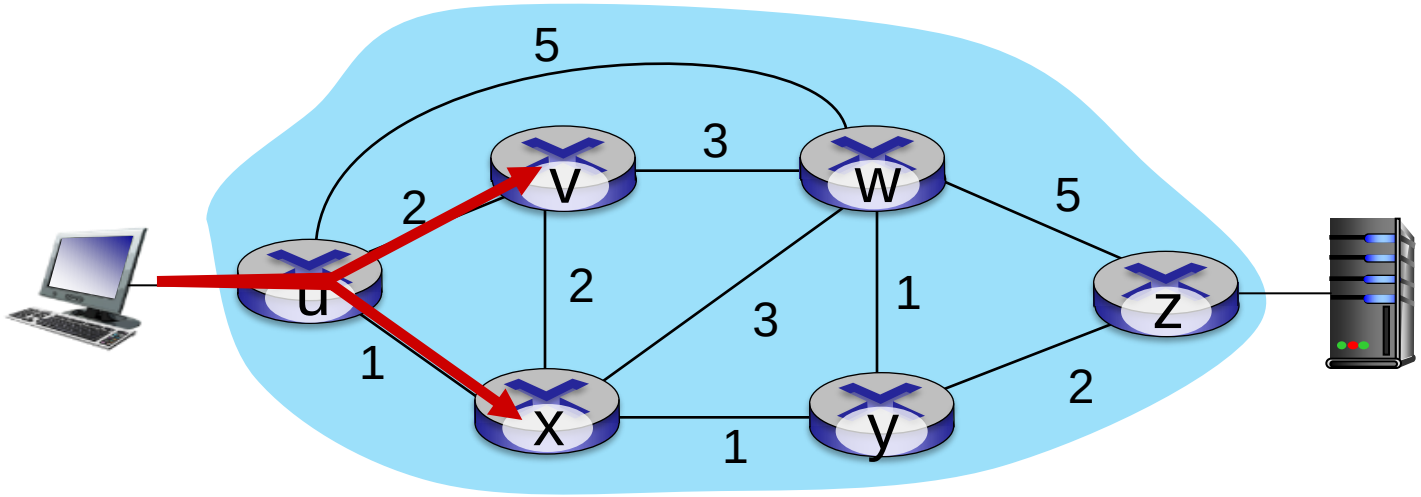



*Q:* what if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*?

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

*link weights are only control "knobs": not much control!*

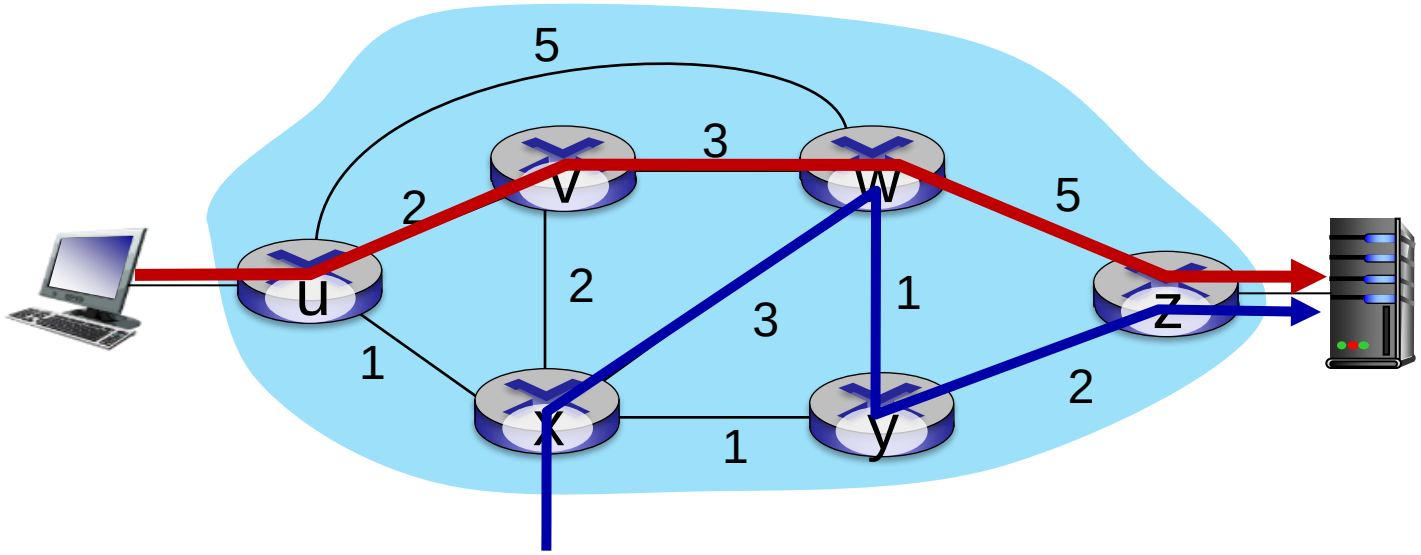
# Traffic engineering: difficult with traditional routing




*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 with traditional routing

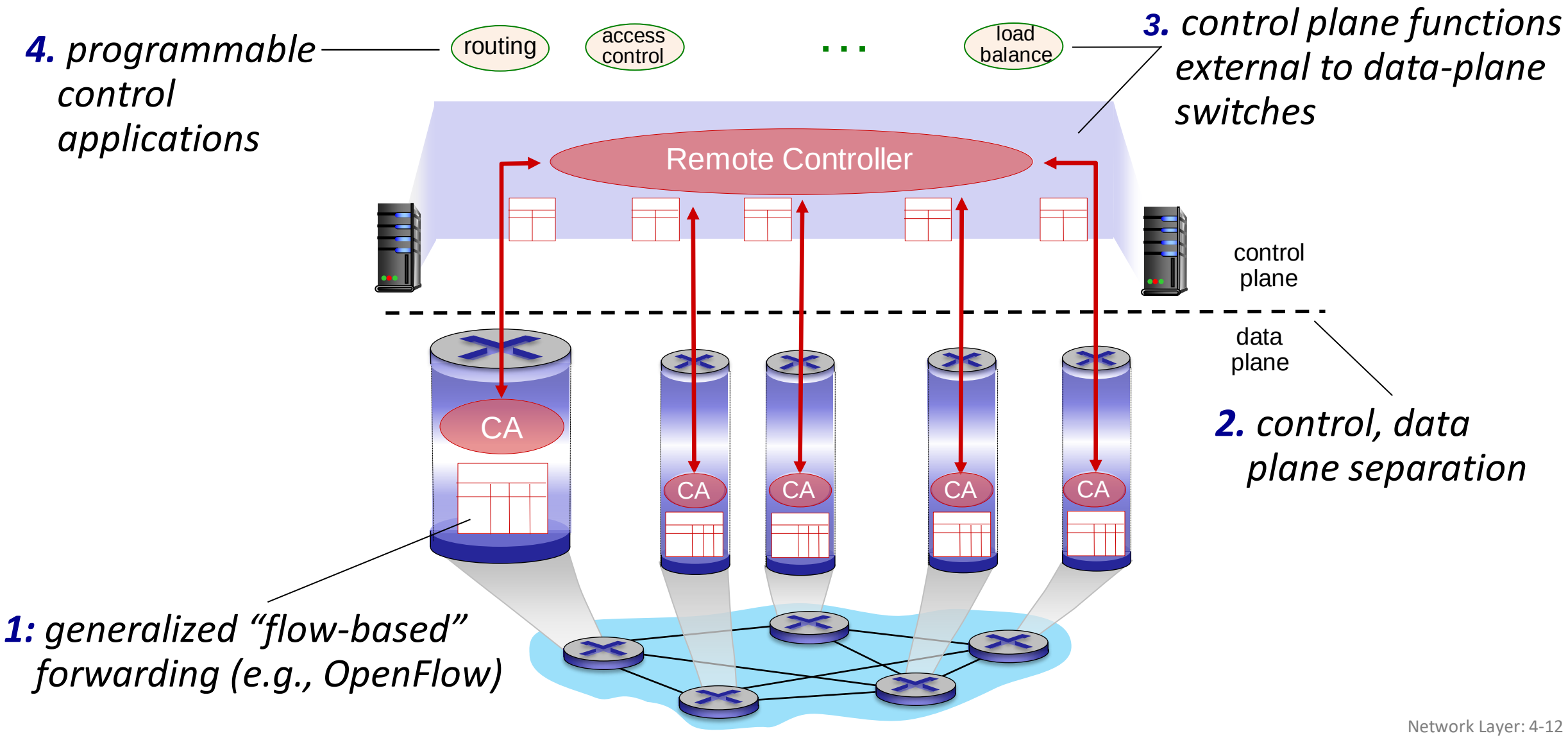



*Q:* what if w wants to route blue and red traffic differently from w to z?

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

We learned in Chapter 4 that generalized forwarding and SDN can be used to achieve *any* routing desired

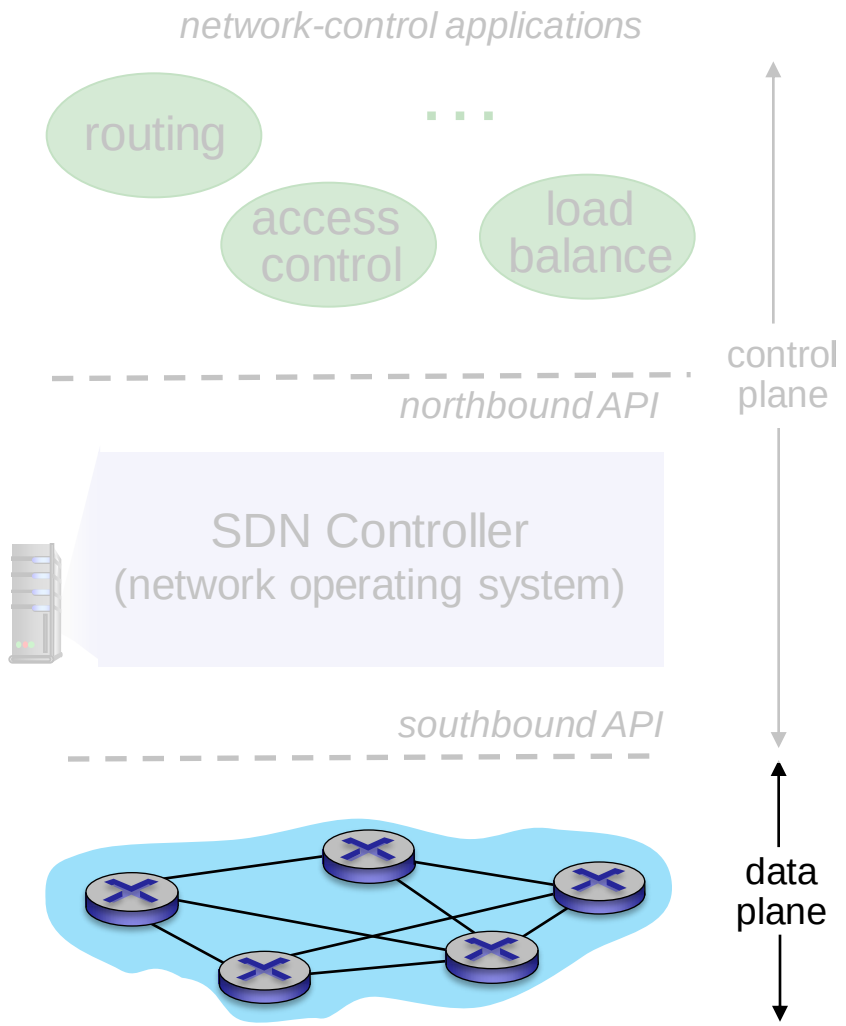
# Software defined networking (SDN)



# Software defined networking (SDN)

## Data-plane switches:

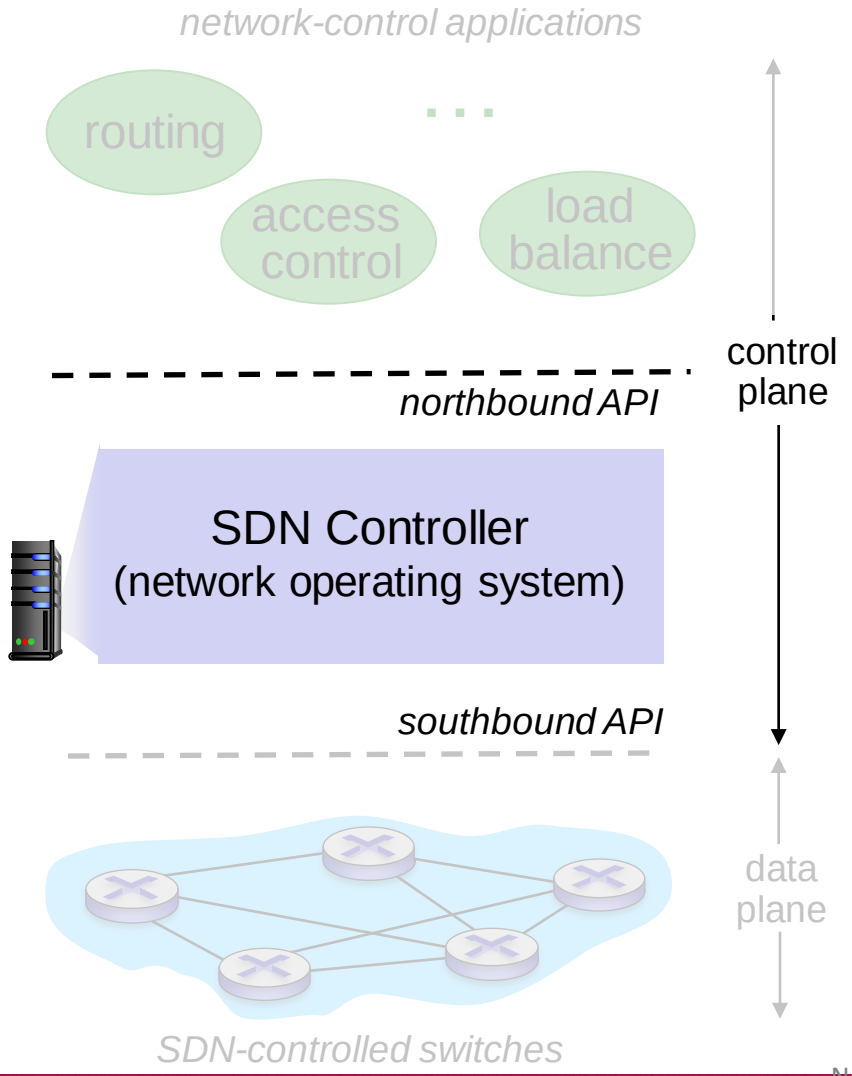
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- protocol for communicating with controller (e.g., OpenFlow)



# Software defined networking (SDN)

## SDN controller (network OS):

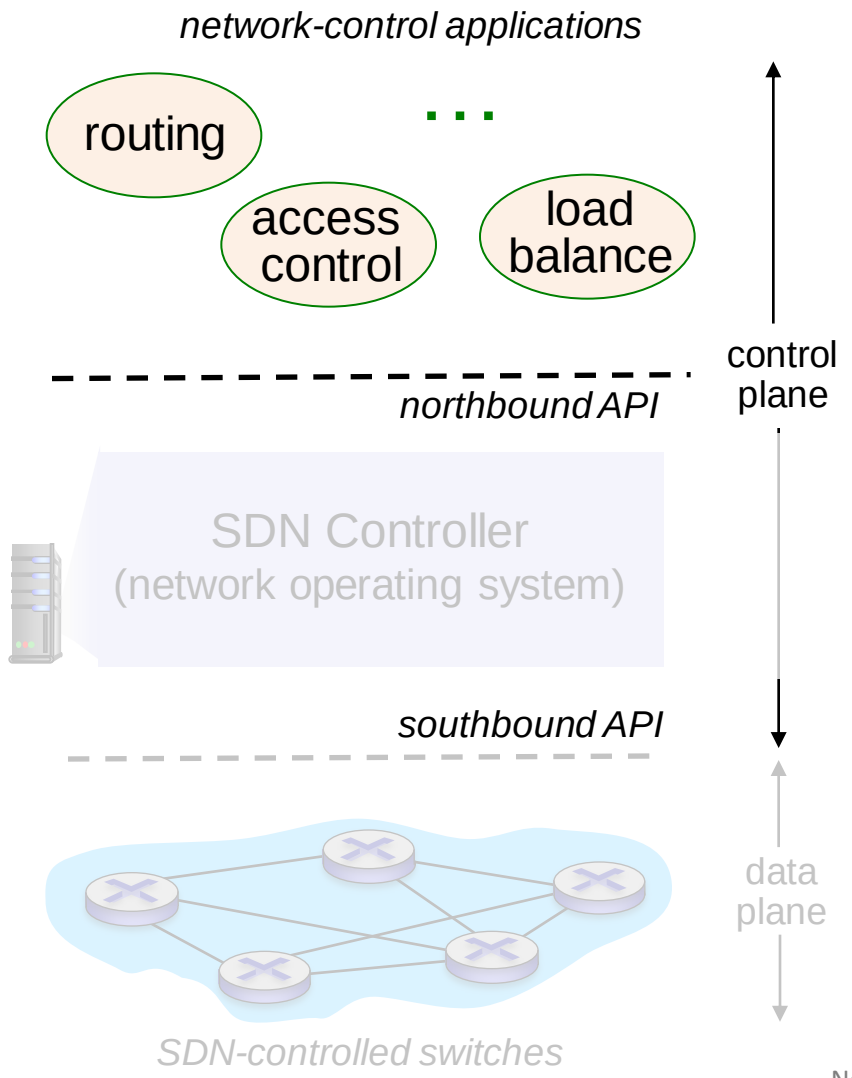
- maintain network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



# Software defined networking (SDN)

## network-control apps:

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

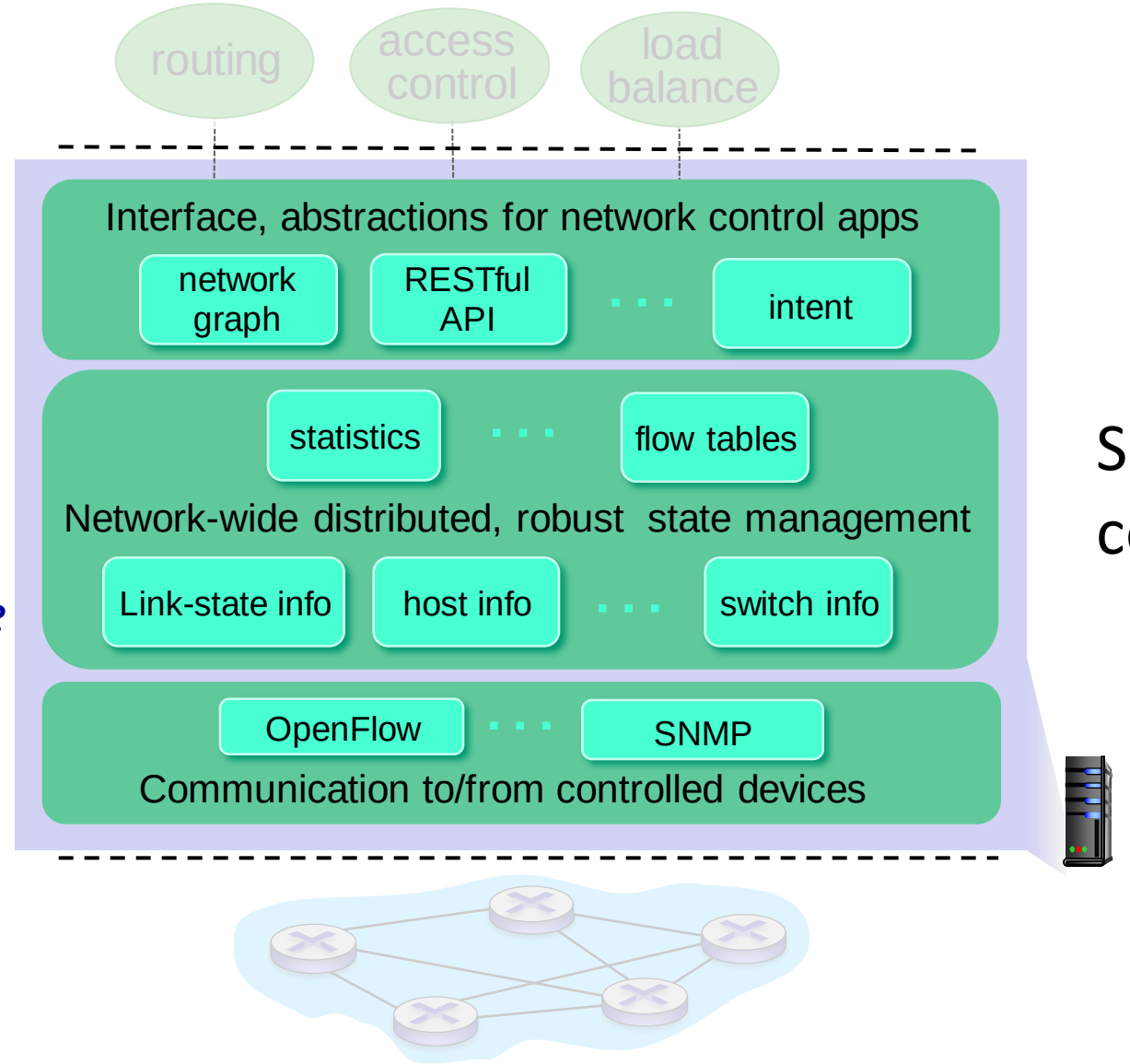


# Components of SDN controller

interface layer to network control apps: abstractions API

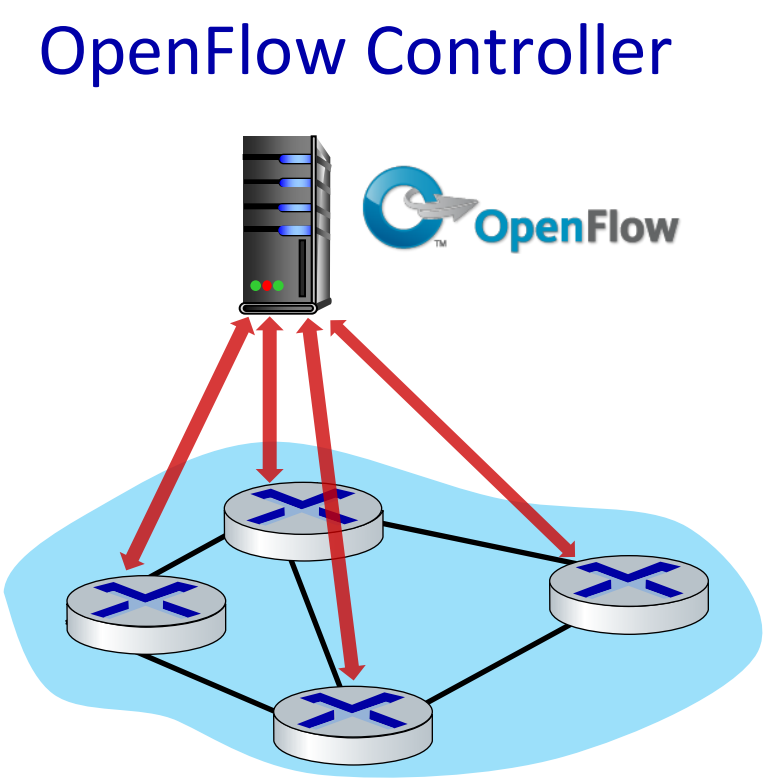
network-wide state management : state of networks links, switches, services: a *distributed database*

*communication*: 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.)
- distinct from OpenFlow API
  - API used to specify generalized forwarding actions

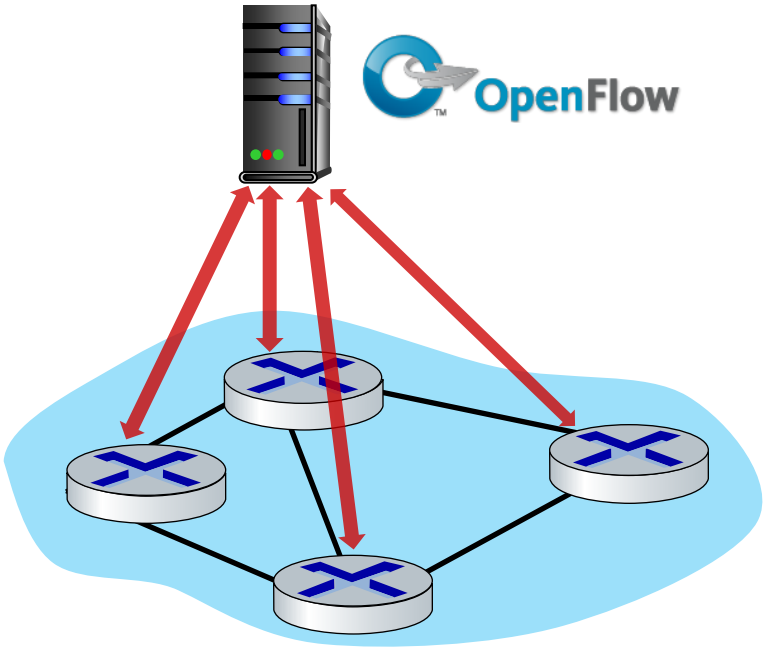


# OpenFlow: controller-to-switch messages

## 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 the OpenFlow tables
- *packet-out:* controller can send this packet out of specific switch port

OpenFlow Controller

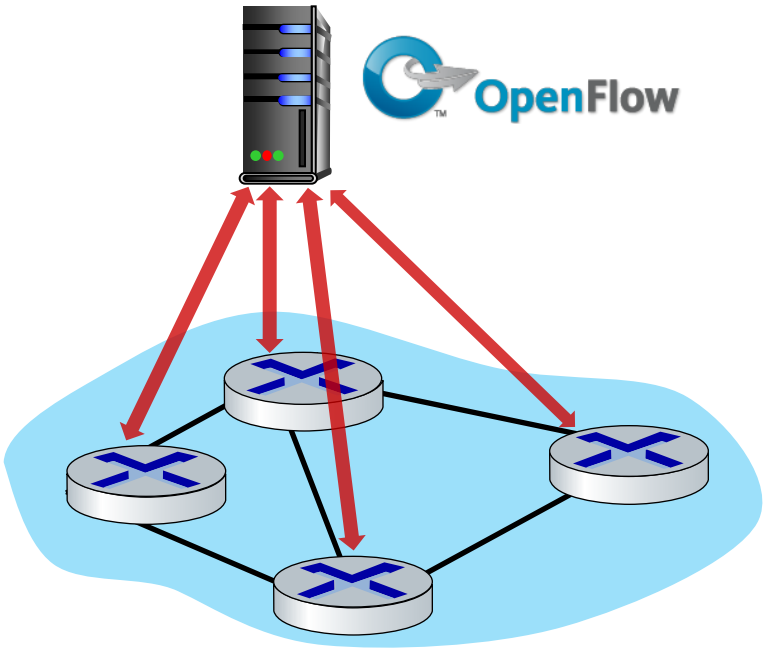


# OpenFlow: switch-to-controller messages

## Key switch-to-controller messages

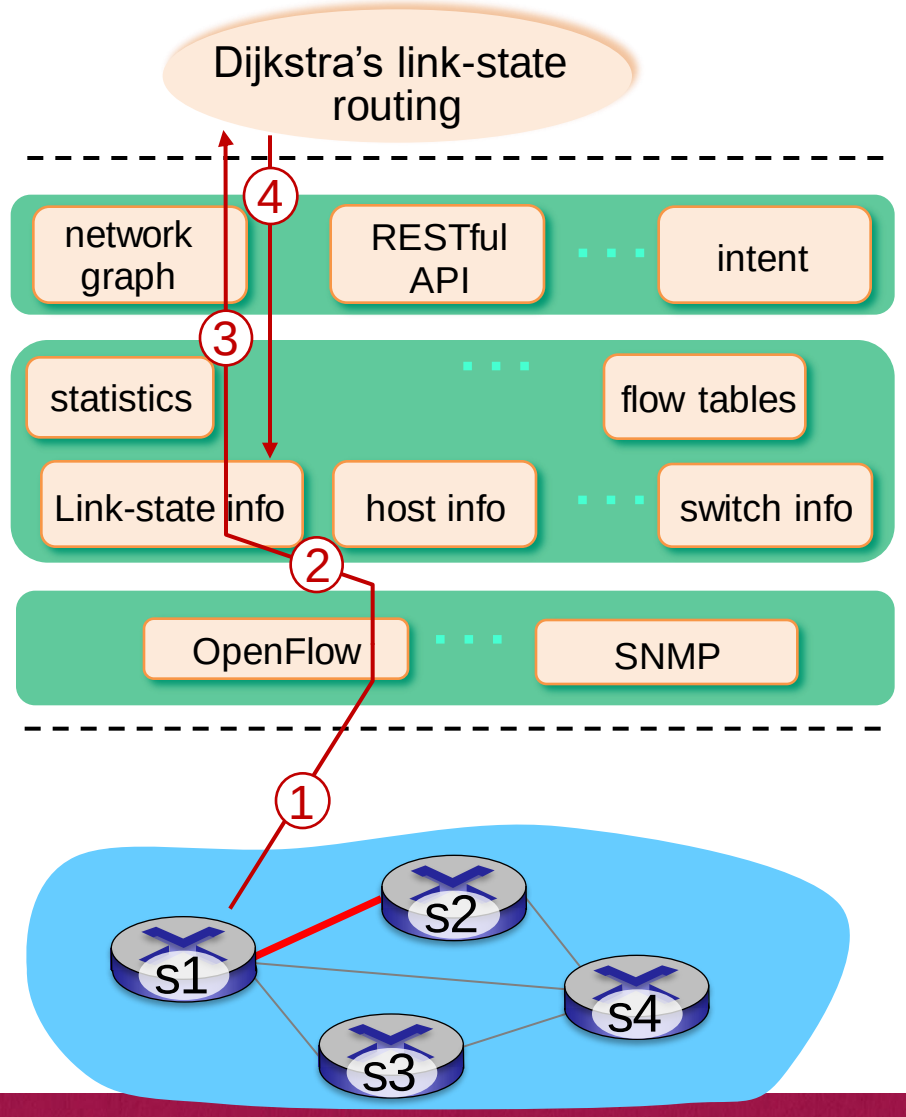
- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* flow table entry deleted at switch
- *port status:* inform controller of a change on a port.

## OpenFlow Controller




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



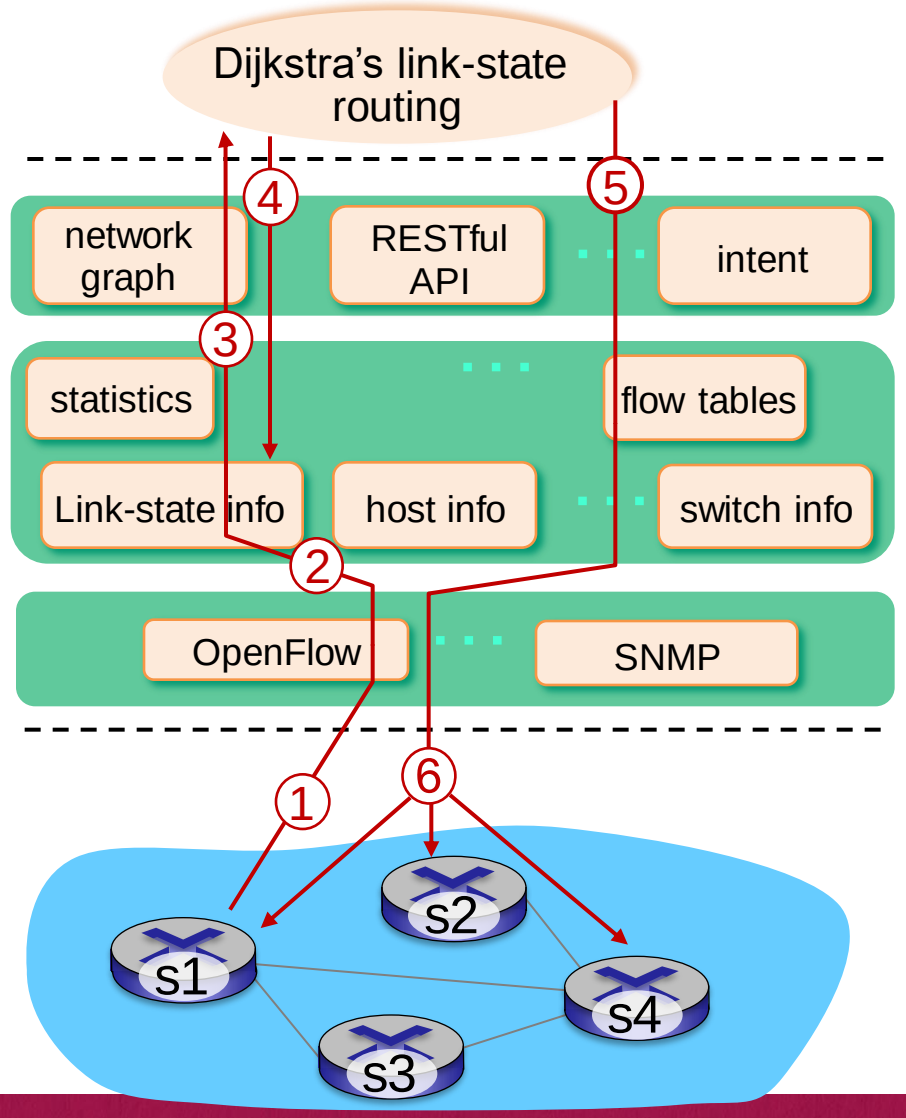

① S1, experiencing link failure uses OpenFlow port status message to notify controller

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

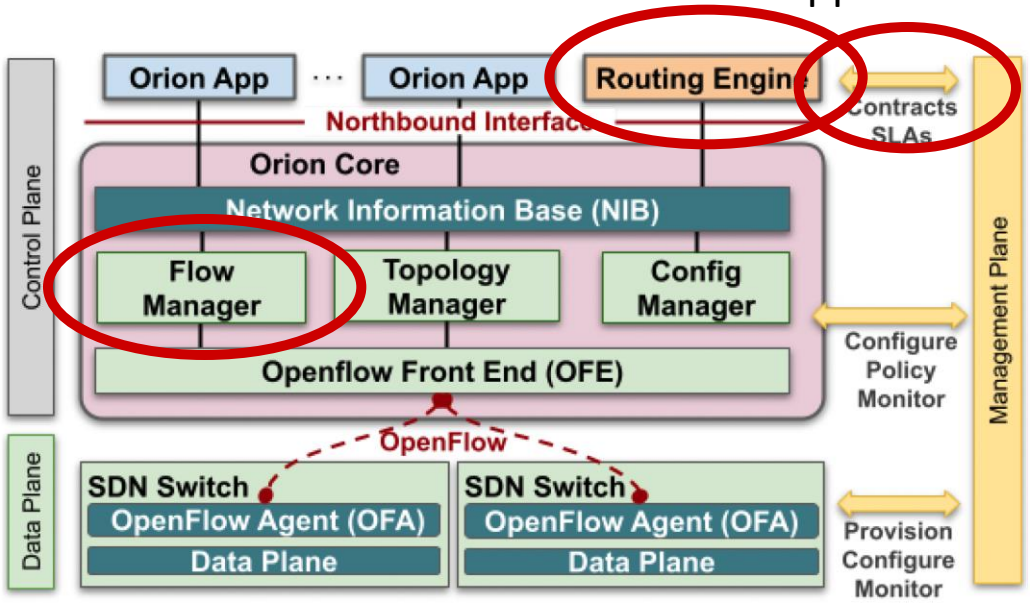
⑥ controller uses OpenFlow to install new tables in switches that need updating

# Google ORION SDN control plane

ORION: Google's SDN control plane (*NSDI'21*): control plane for Google's datacenter (Jupiter) and wide area (B4) networks

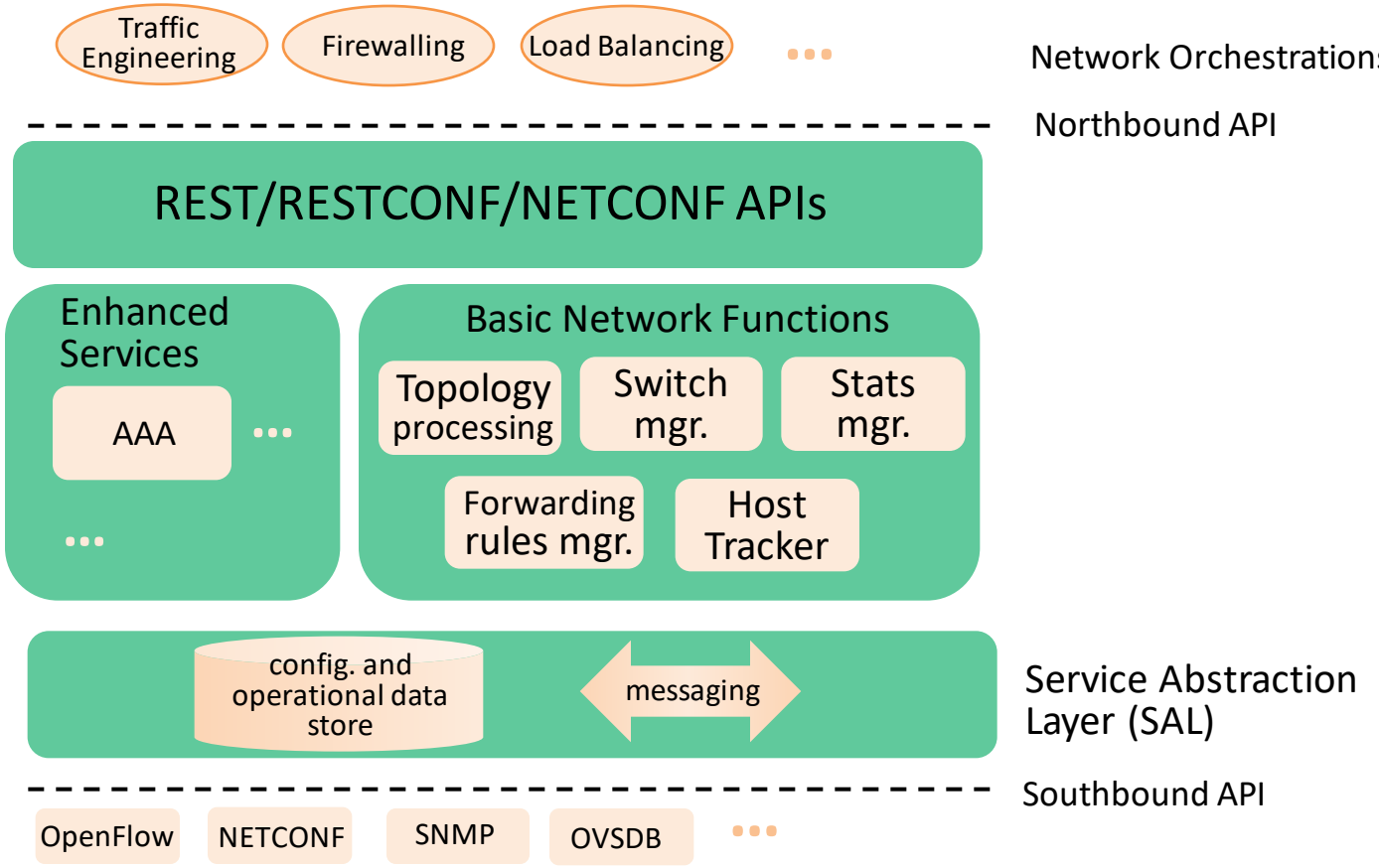
- routing (intradomain, iBGP), traffic engineering: implemented in *applications* on top of ORION core
- edge-edge flow-based controls (e.g., CoFlow scheduling) to meet contract SLAs
- management: pub-sub distributed microservices in Orion core, OpenFlow for switch signaling/monitoring

Orion SDN architecture and core apps




Note: ORION provides *intradomain* services within Google's network

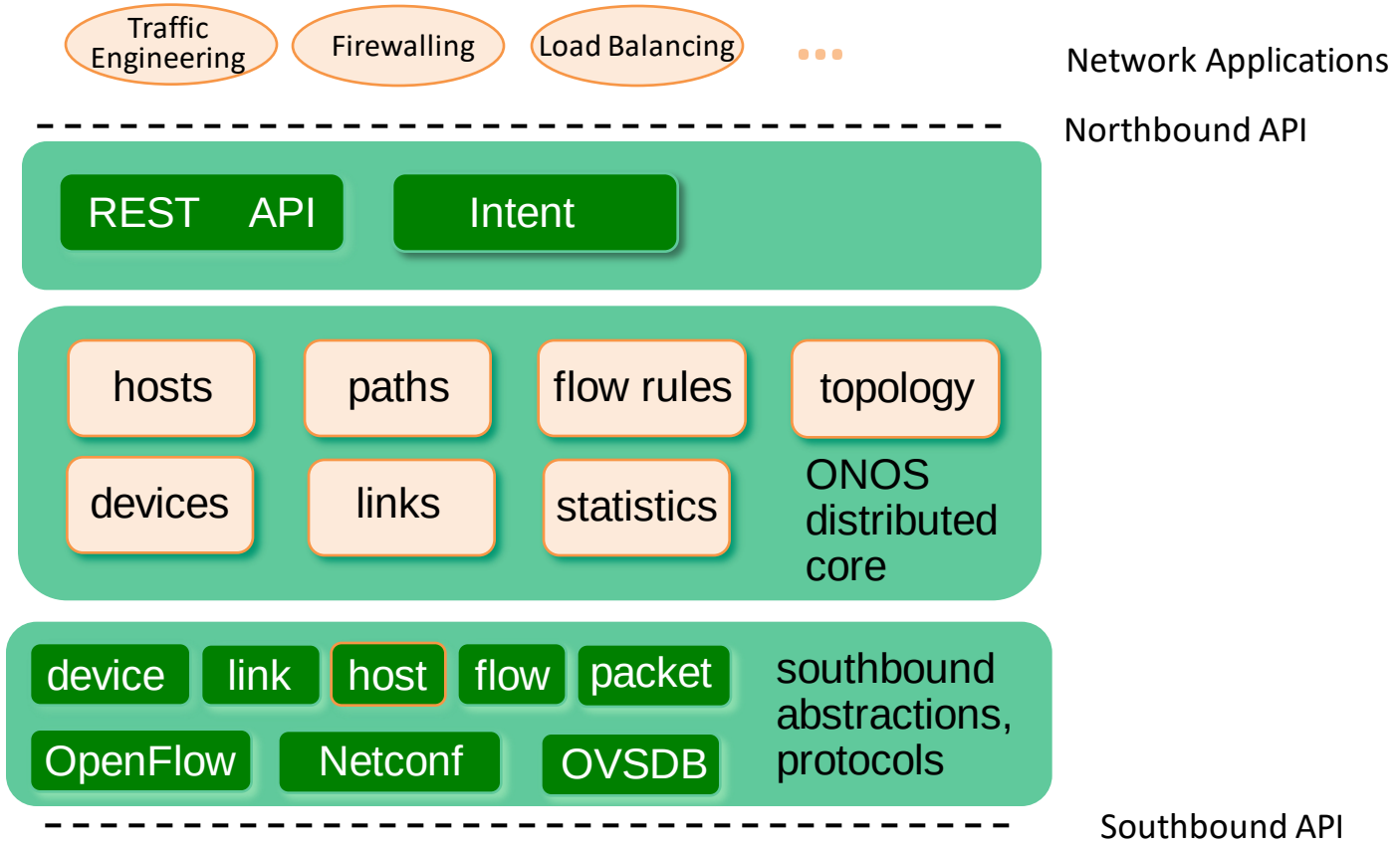
# OpenDaylight (ODL) controller




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

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: beyond a single AS

SDN critical in 5G cellular networks

# SDN and the future of traditional network protocols

SDN-computed versus router-computer forwarding tables:

- just one example of logically-centralized-computed versus protocol computed

one could imagine SDN-computed congestion control:

- controller sets sender rates based on router-reported (to controller) congestion levels

How will implementation of network functionality (SDN versus protocols) evolve?

# Namah Shivaya