CS 305: Computer Networks

Fall 2024

Network Layer – The Control Plane

Tianyue Zheng

Department of Computer Science and Engineering
Southern University of Science and Technology (SUSTech)

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Internet inter-AS routing: BGP

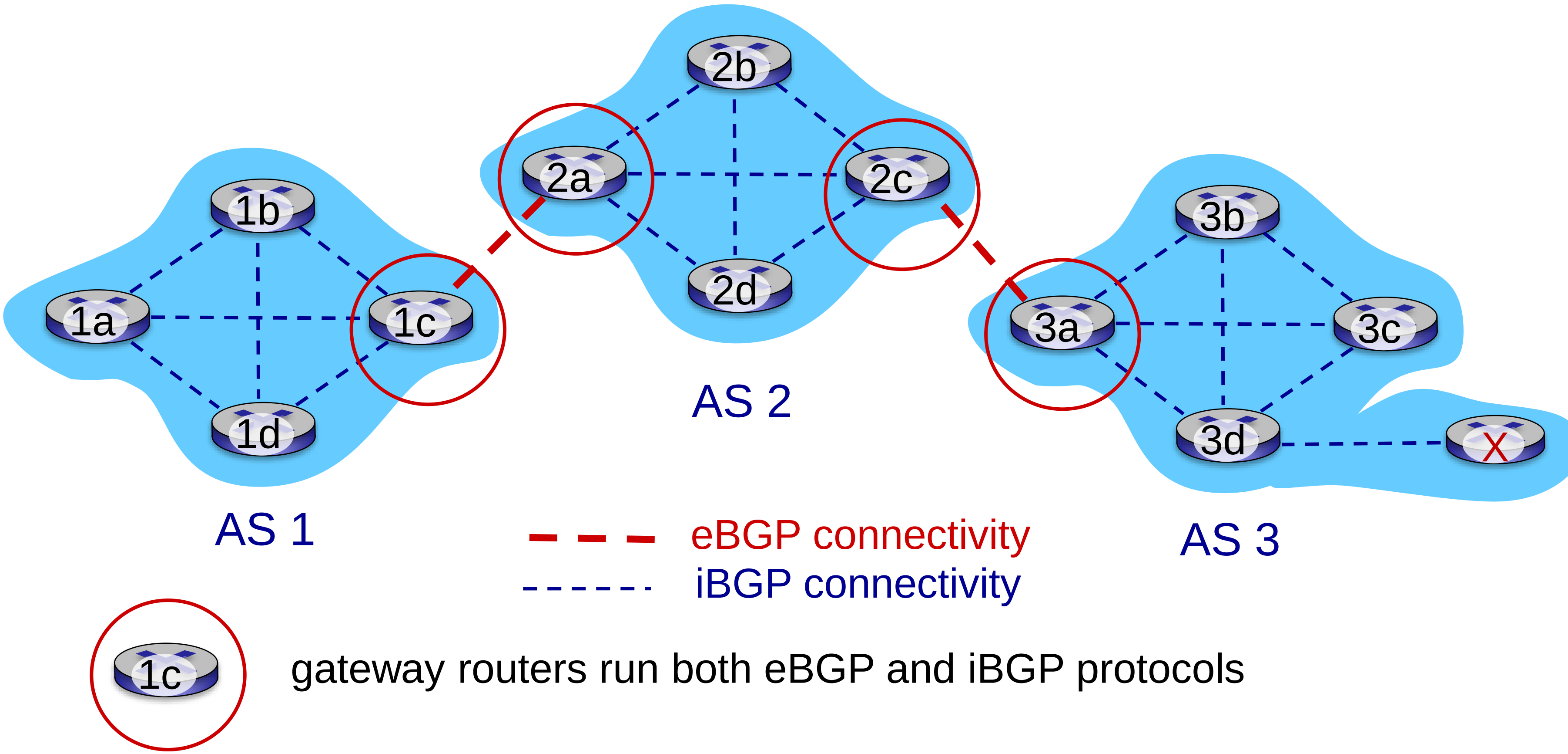
- BGP (Border Gateway Protocol): inter-domain routing protocol
  - "glue that holds the Internet together"
  - Decentralized, asynchronous, distance-vector
- Main functions BGP provides:
  - allows subnet to advertise its existence to rest of Internet: *"I am here"*
    - obtain subnet reachability information from neighboring ASes: eBGP
    - propagate reachability information to all AS-internal routers: iBGP
  - determine "good" routes to other networks based on reachability information and *policy*

Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

BGP basics

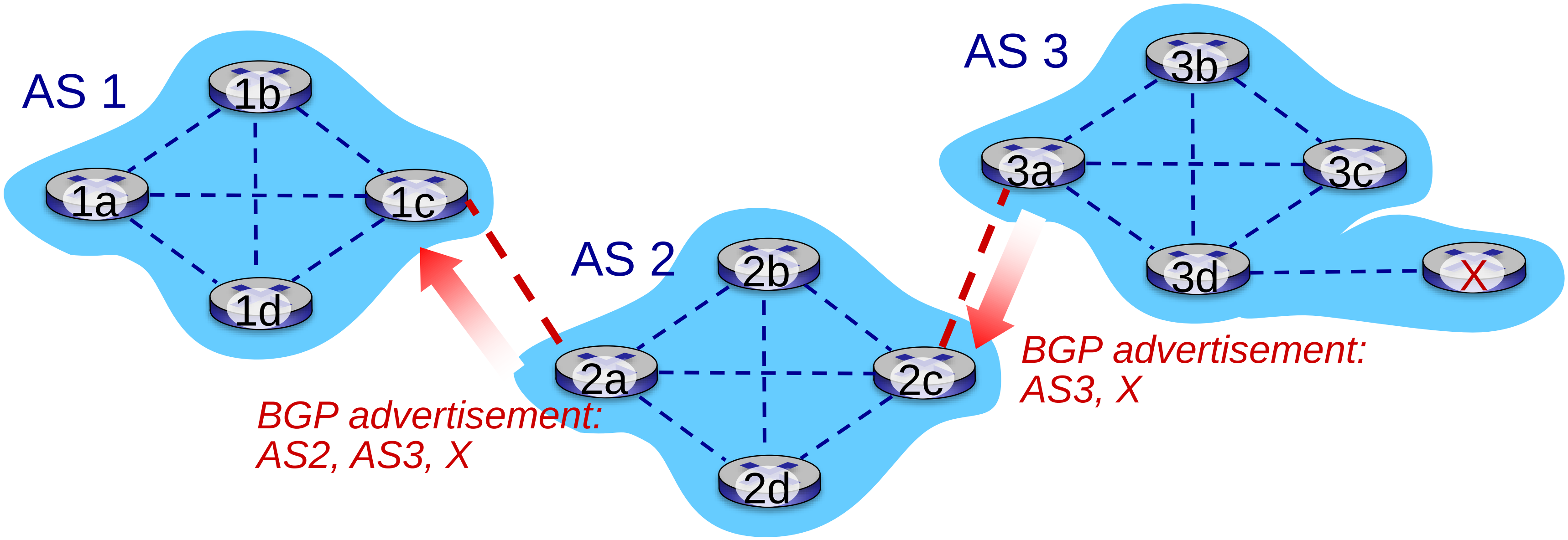
- Each pair of BGP routers (“peers”) exchanges BGP messages over TCP connection:
  - advertising *paths* to destination network prefixes (e.g., X)



eBGP basics

When AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:

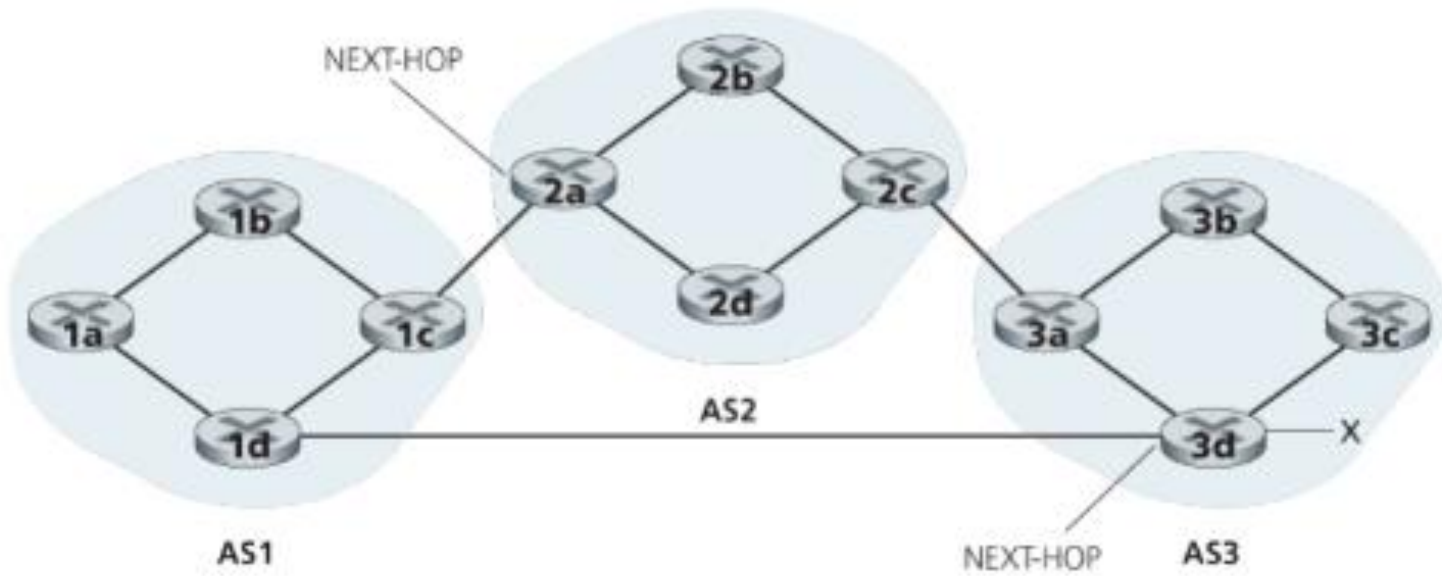
- AS3 *promises* to AS2 it will forward datagrams towards X



Path attributes and iBGP routes

- advertised prefix includes BGP attributes
  - Prefix (destination) + attributes = “route”
- two important attributes:
  - AS-PATH: list of ASes through which the advertisement has passed, e.g., AS2 AS3
    - Advertisement; prevent loops
  - NEXT-HOP: IP address of the router interface that begins the AS-PATH, e.g., IP of the interface of AS2 that begins AS2 AS3

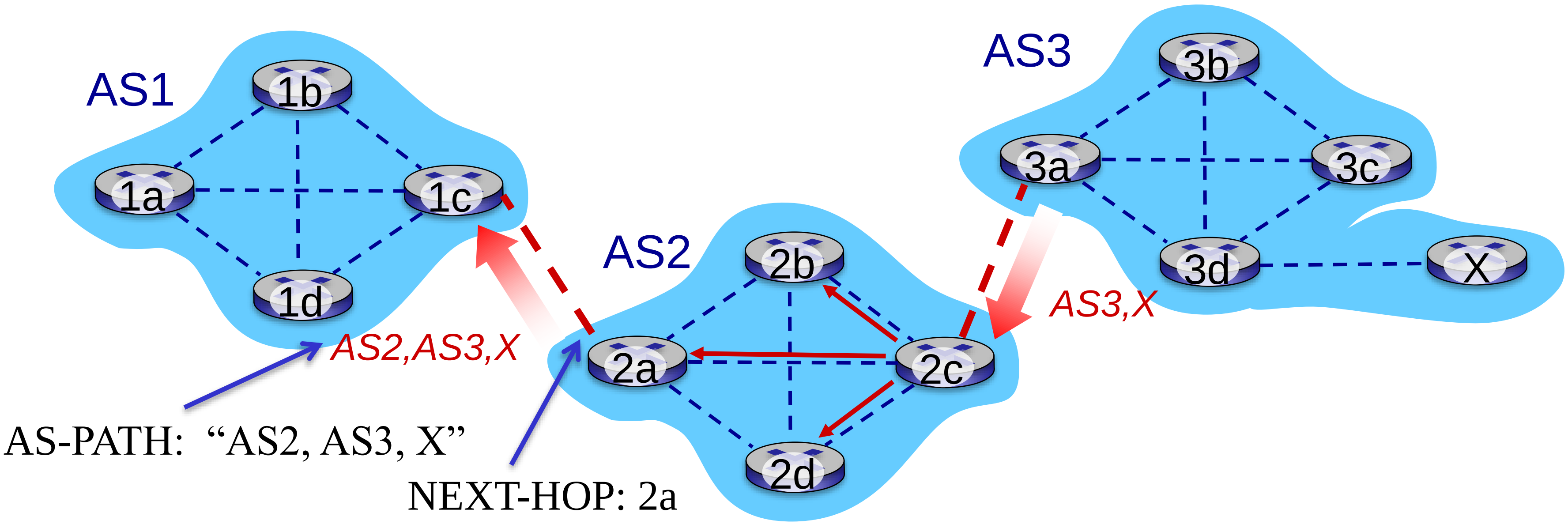
Path attributes and iBGP routes



IP address of leftmost interface for router 2a; AS2 AS3; x

IP address of leftmost interface of router 3d; AS3; x

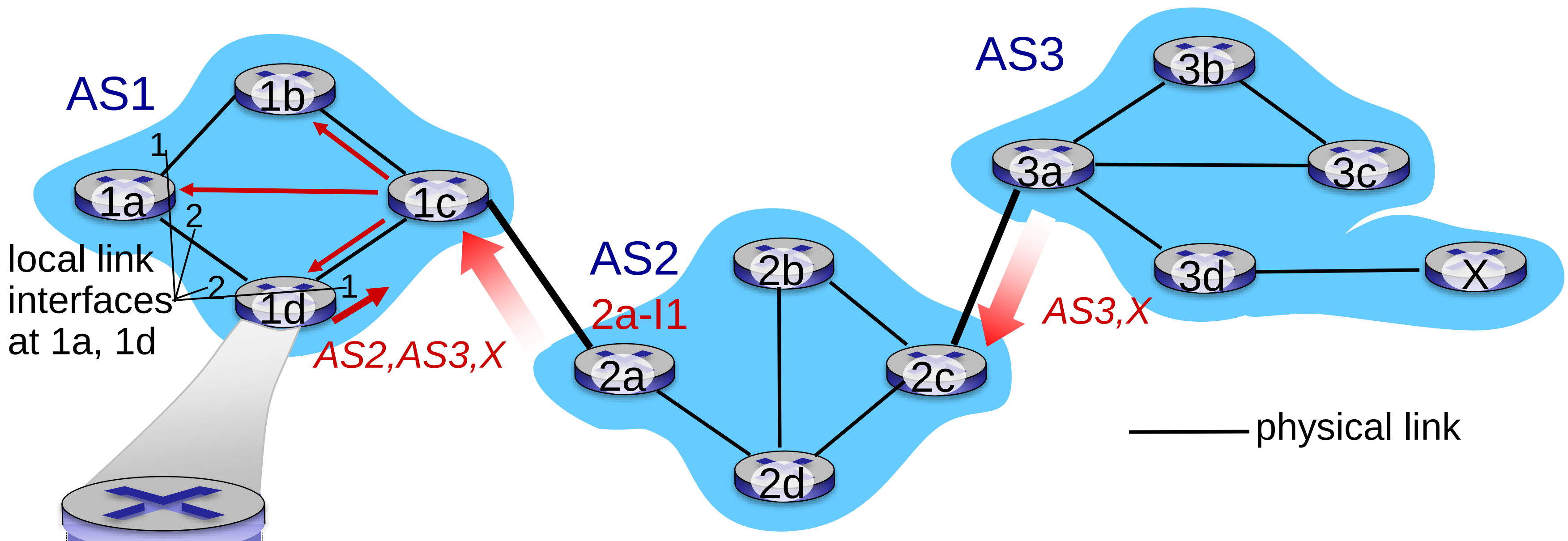
BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

BGP, OSPF, forwarding table entries

Q: how does router set forwarding table entry to distant prefix?



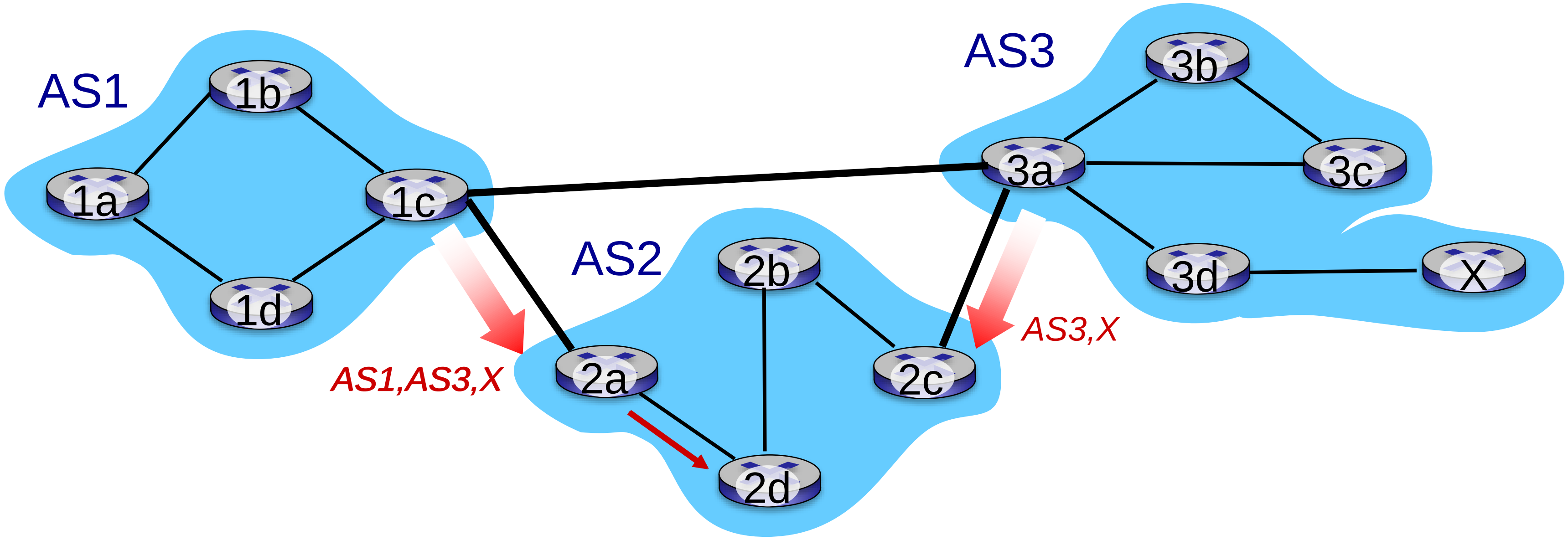
dest	interface
2a-I1	1
X	1
…	…

- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 2a (NEXT-HOP)"
- 1d: to get to 2a-I1, forward over outgoing local interface 1
  - Intra-AS protocol

Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

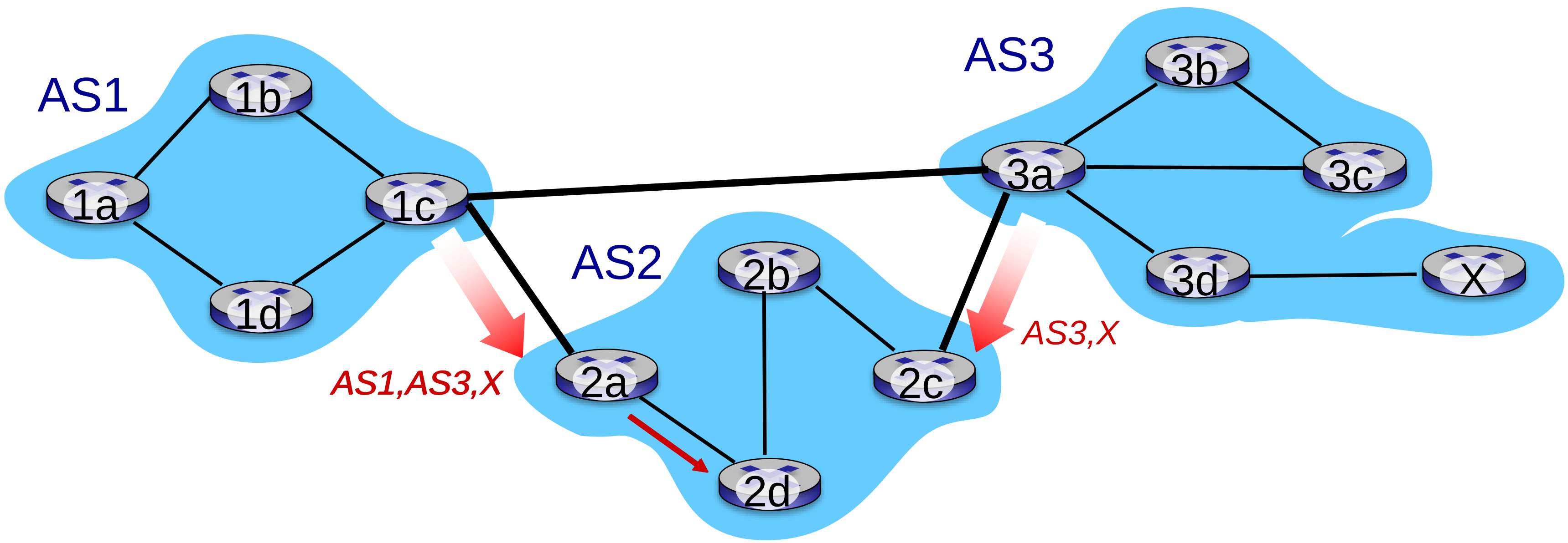
Route selection



A router may learn about multiple paths to destination:

- 2d learns path *AS1,AS3,X* from 1c
- 2d learns path *AS3,X* from 3a

Route selection: Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 1c or 3a
- *hot potato routing:* choose local gateway that has least intra-domain cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!

BGP route selection

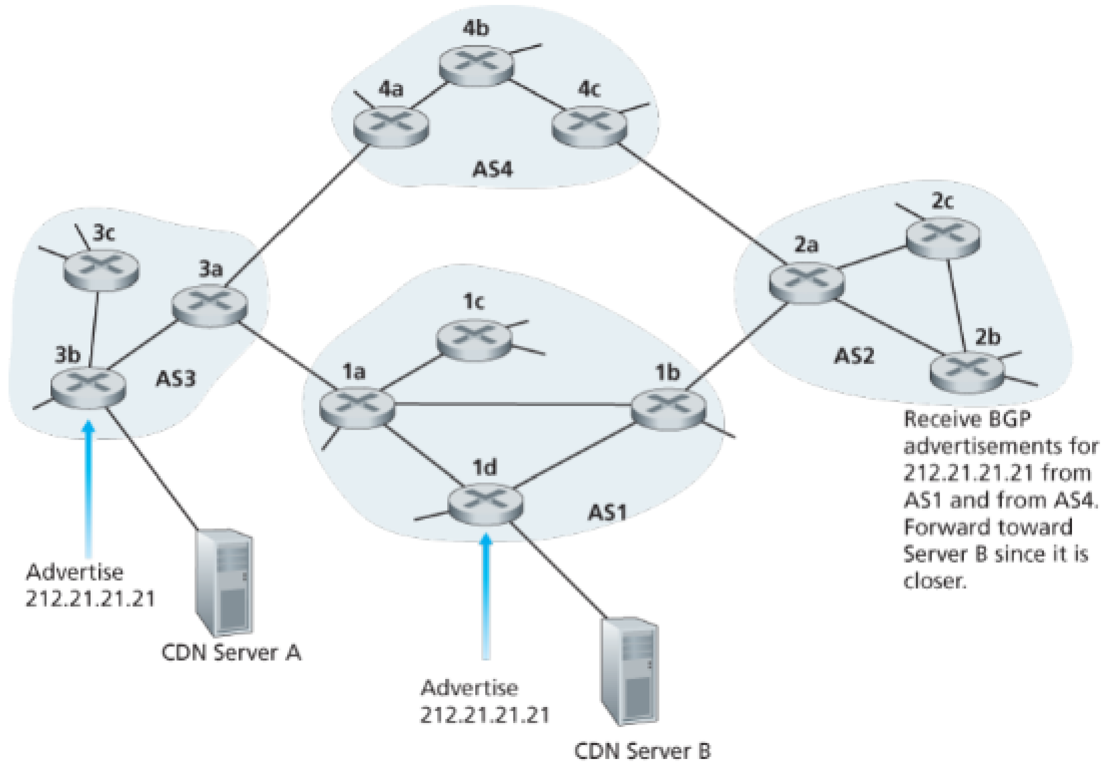
Router may learn about more than one route to destination AS, selects route based on:

1. local preference value attribute: policy decision
2. shortest AS-PATH
3. closest NEXT-HOP router: hot potato routing
4. additional criteria

Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

IP-Anycast Service : CDN/DNS



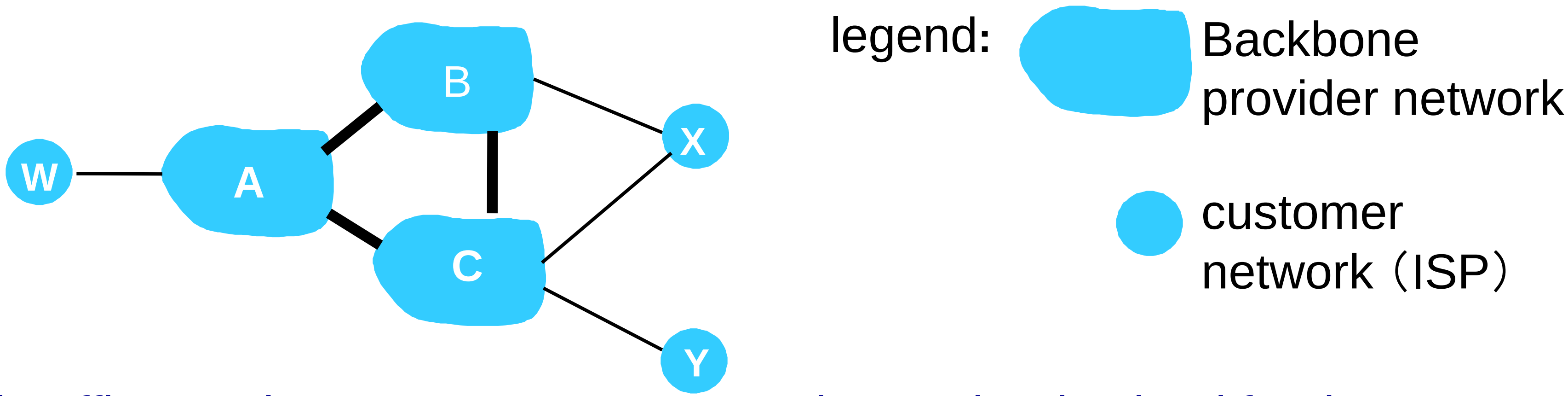
- CDN company assigns the same IP address to each server, and uses standard BGP to advertise this IP address from each server.
- When a BGP router receives multiple route advertisements for this IP address → different paths to the same physical location
- When configuring its routing table, each router will locally use the BGP route-selection algorithm to pick the "best" route to that IP address

Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

  determines whether to *advertise*
  path to other neighboring ASes

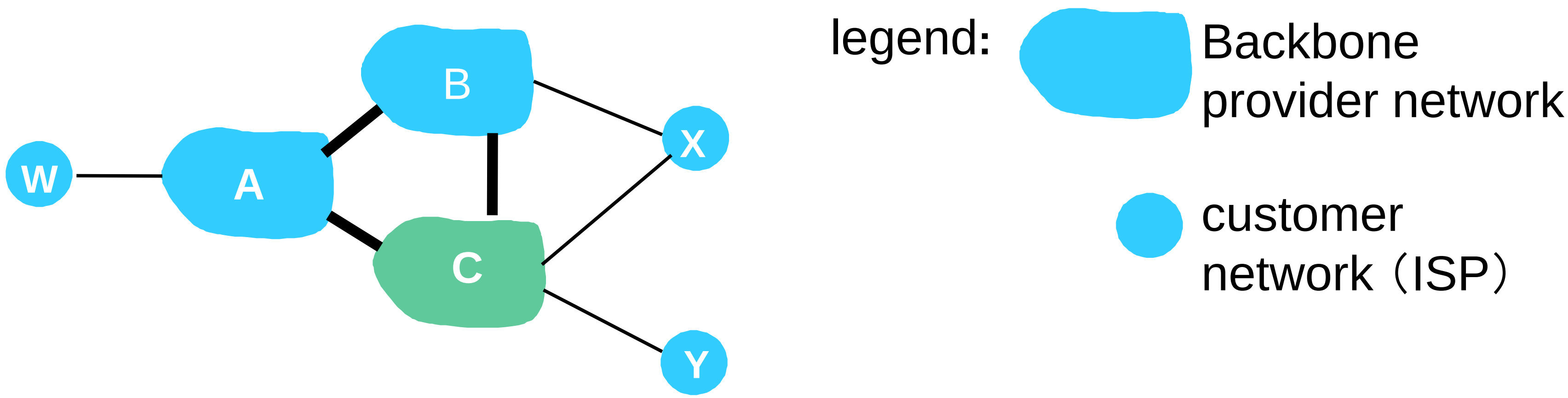
Routing Policy



All traffic entering an ISP access network must be destined for that network, and all traffic leaving an ISP access network must have originated in that network.

- A,B,C are *provider networks*
- X,W,Y are customer (of provider networks)
- X is *dual-homed:* attached to two networks
- *policy to enforce:* X does not want to route from B to C via X
  - .. so X will not advertise to B a route to C
  - i.e., X has no paths to any other destinations except itself

Routing Policy



Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A advertises path Aw to B and to C
- B advertises path BAw to X
- B *chooses not to advertise* BAw to C:
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
  - C does not learn about CBAw path
- C will route CAw (not using B) to get to w

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

performance:

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

scale:

- hierarchical routing saves table size, reduced update traffic

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

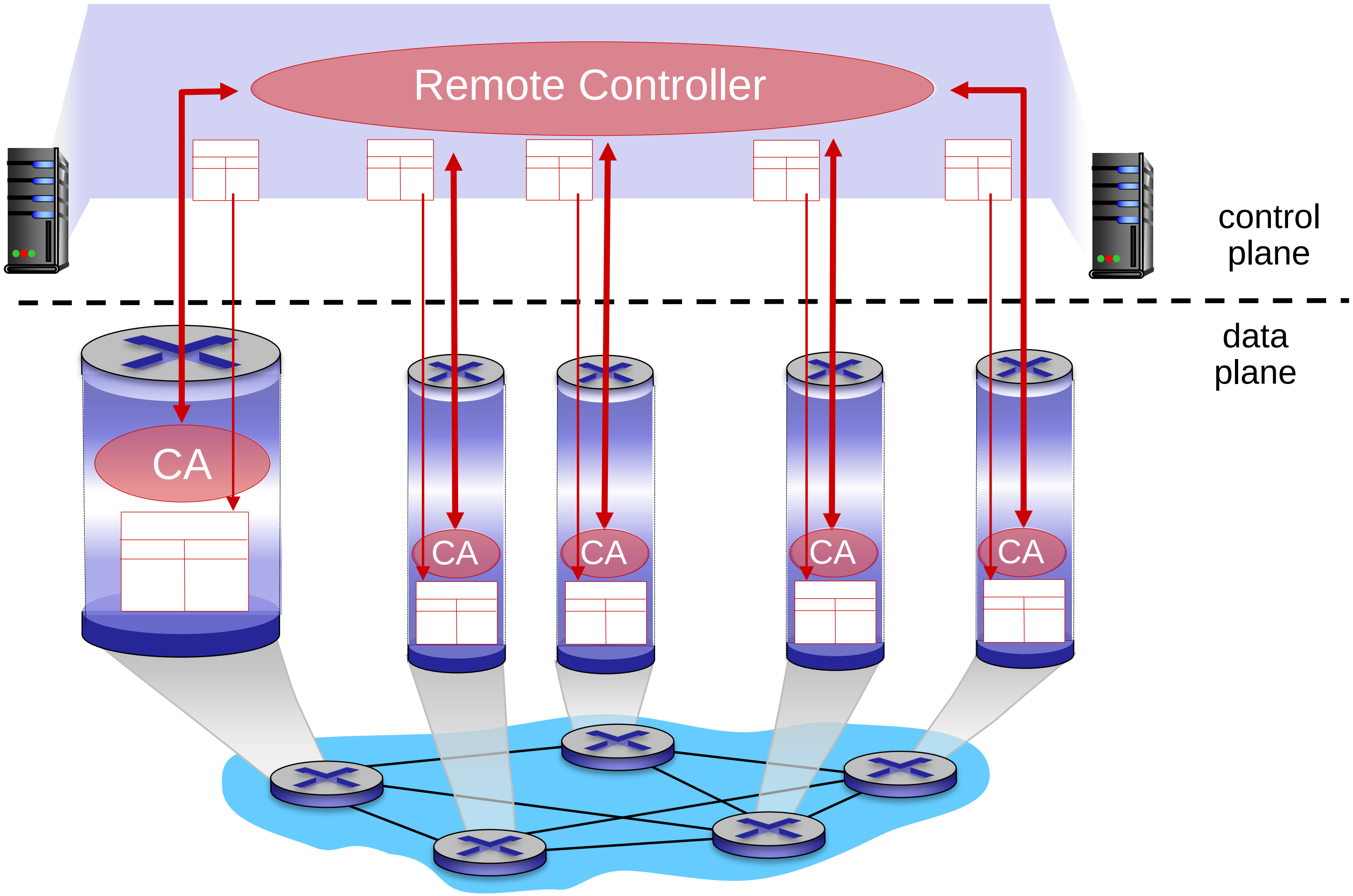
5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Recall: SDN logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

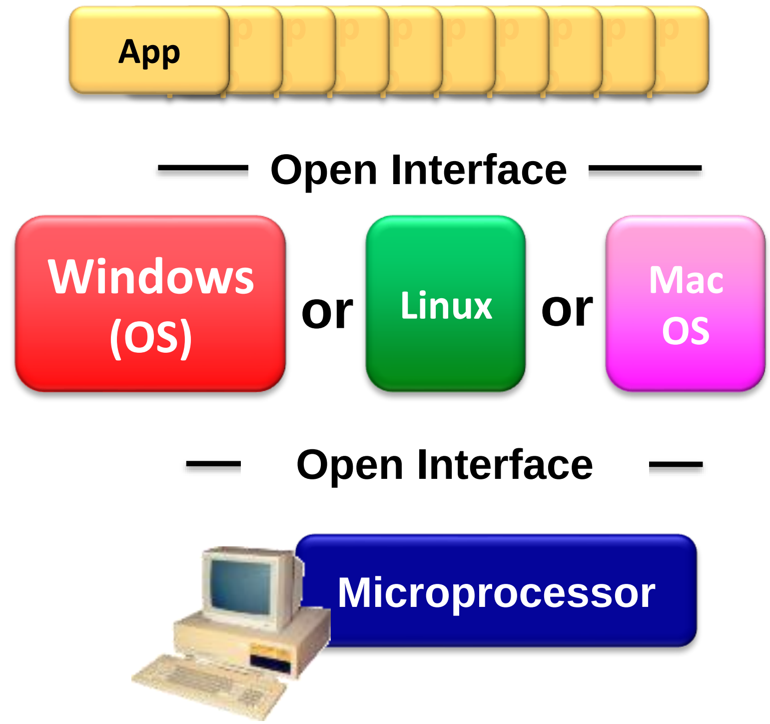


Software defined networking (SDN)

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

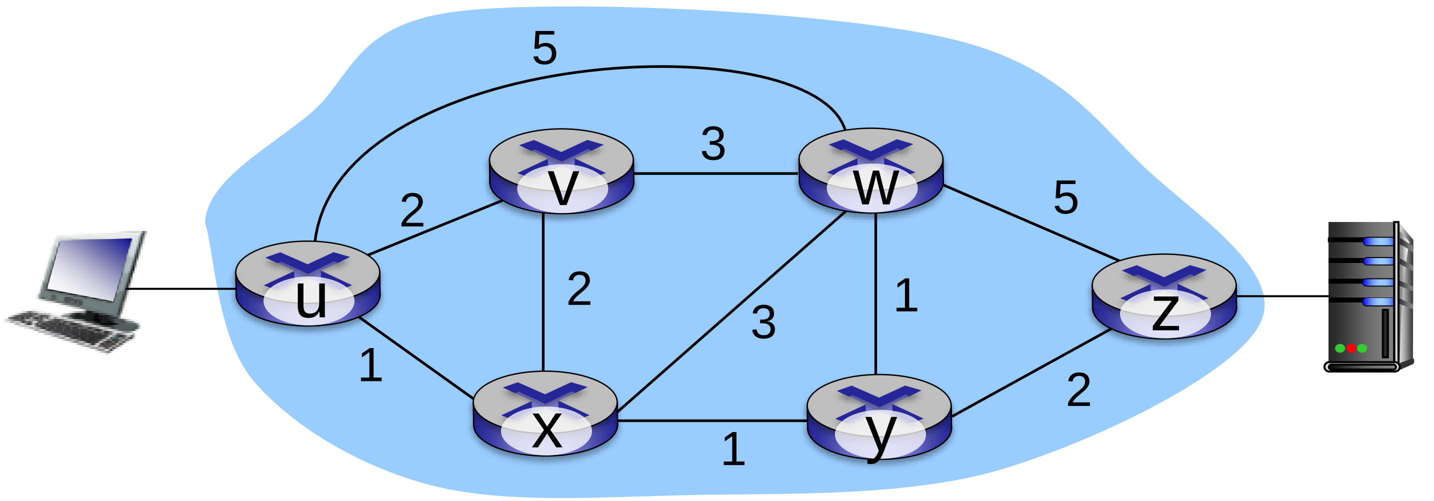
Analogy: mainframe to PC evolution*



Vertically integrated	Horizontal
Closed, proprietary	Open interfaces
Slow innovation	Rapid innovation
Small industry	Huge industry

* Slide courtesy: N. McKeown

Traffic engineering: difficult traditional routing

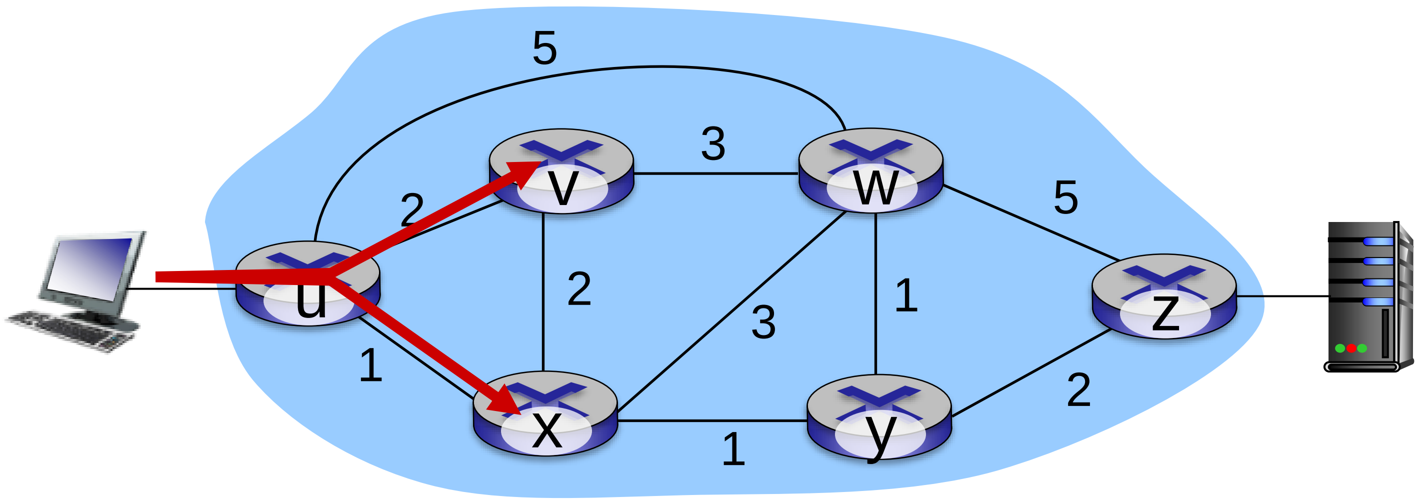


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

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

But the link weights cannot be directly set to certain number

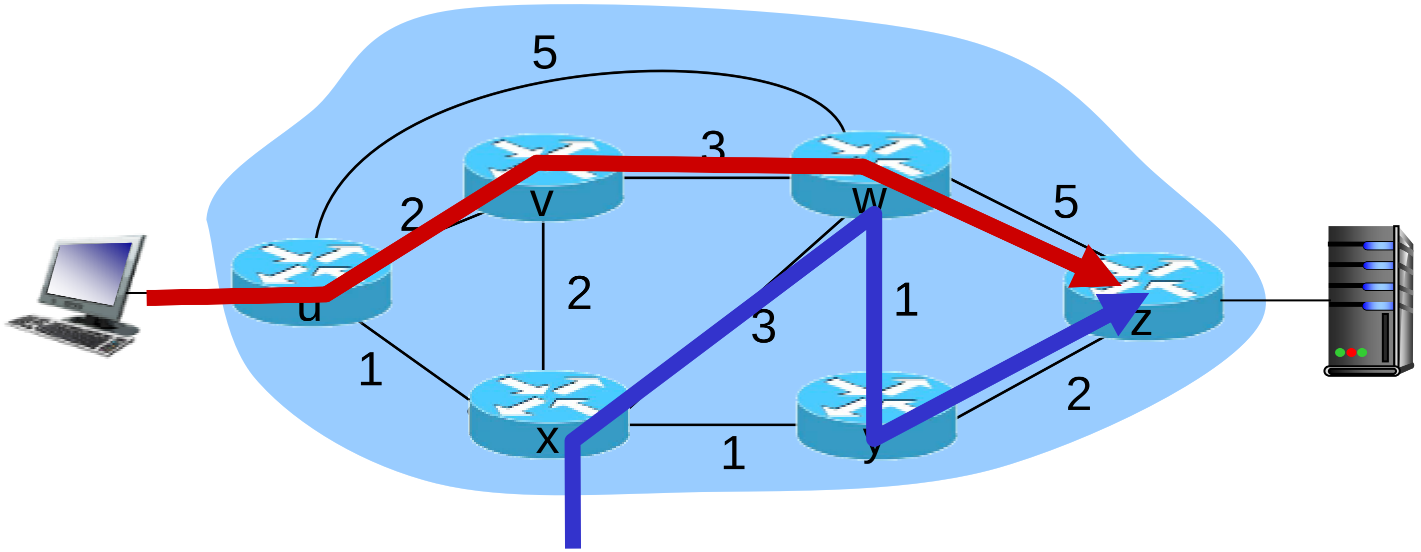
Traffic engineering: difficult



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)

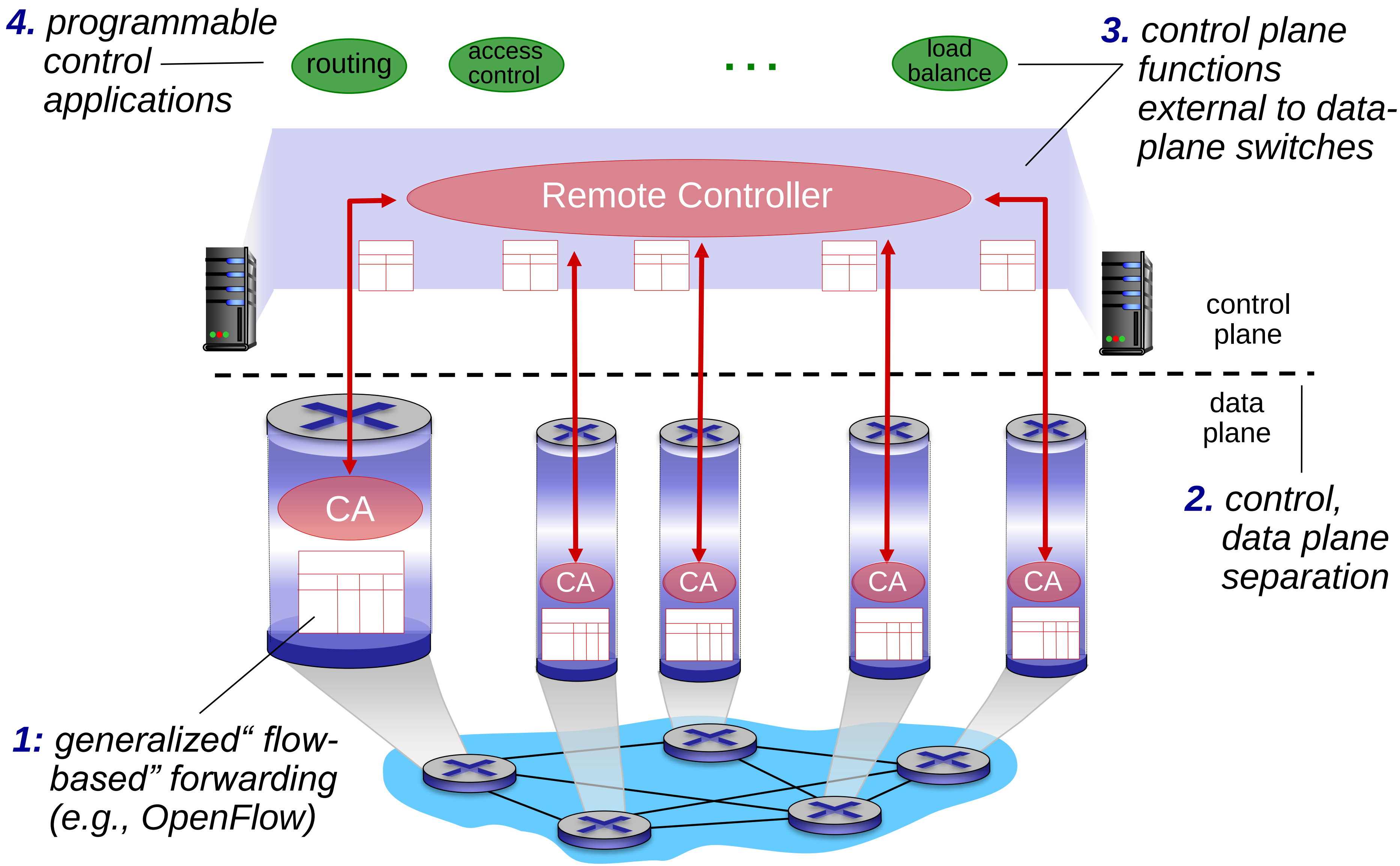
Traffic engineering: difficult



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)

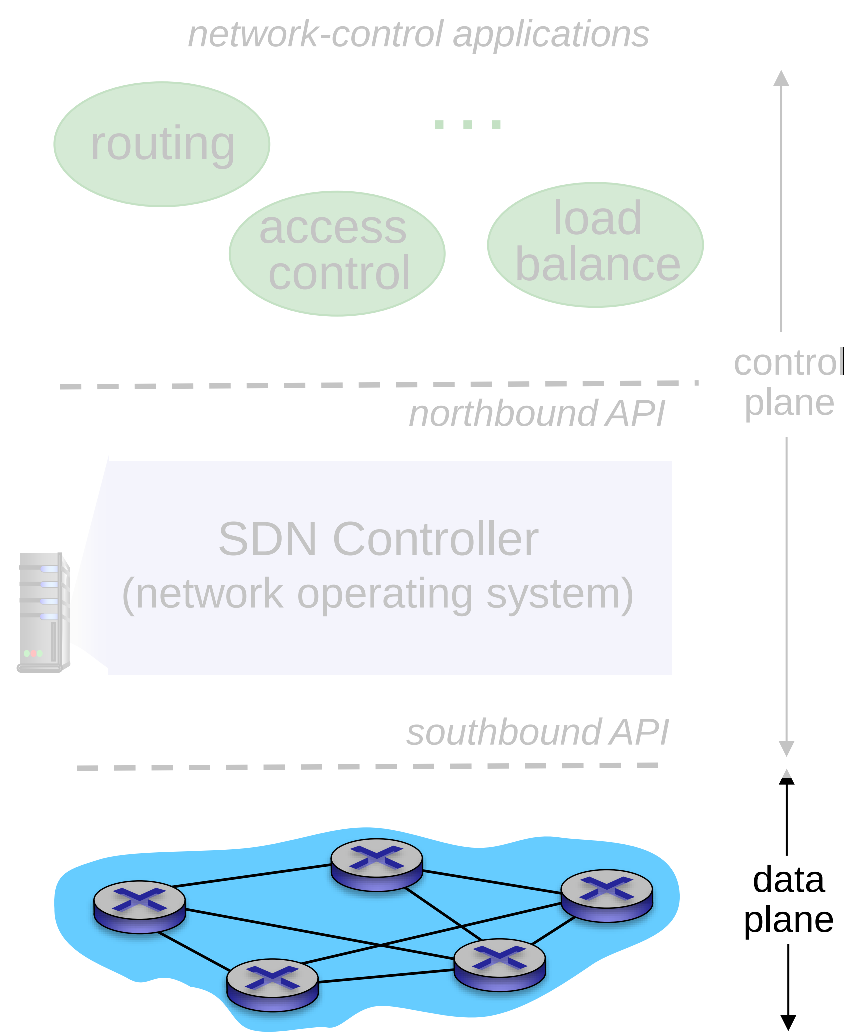
Software defined networking (SDN)



SDN perspective: data plane switches

Data plane switches

- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) 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)

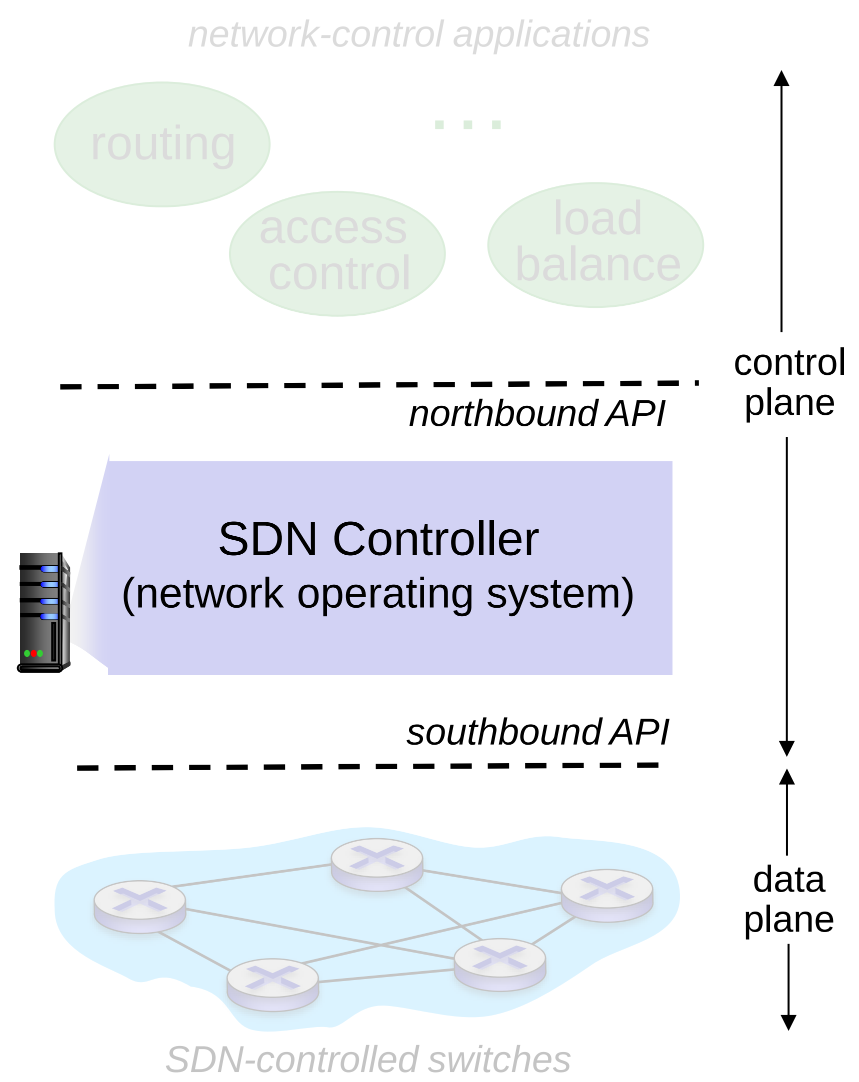


SDN-controlled switches

SDN perspective: SDN controller

SDN controller (network OS):

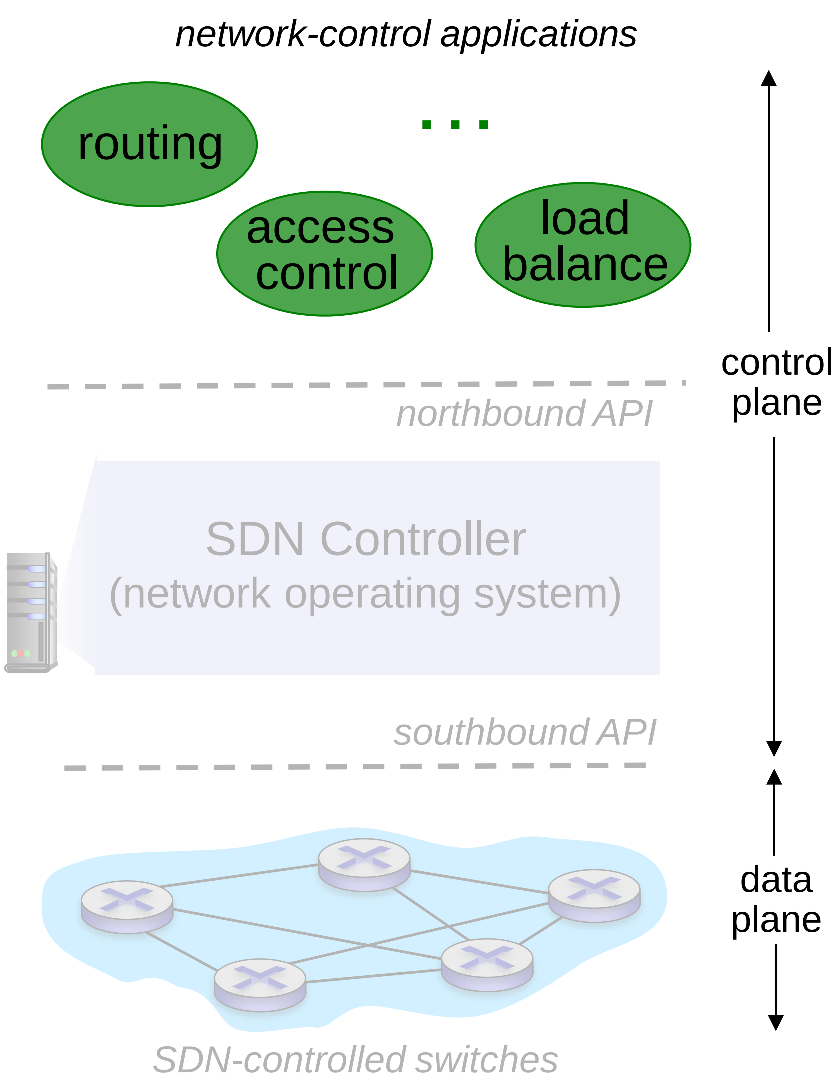
- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



SDN perspective: 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

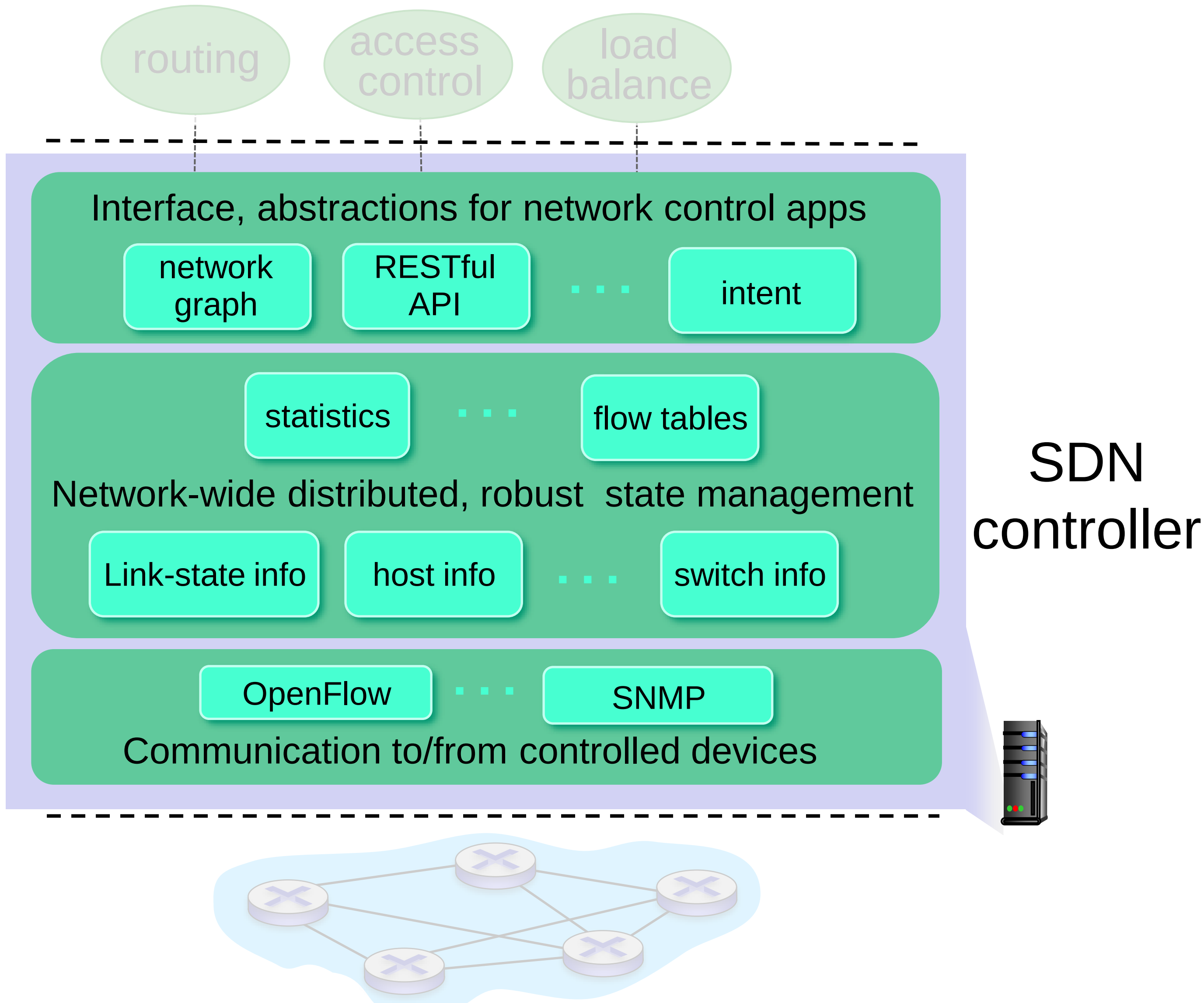


Components of SDN controller

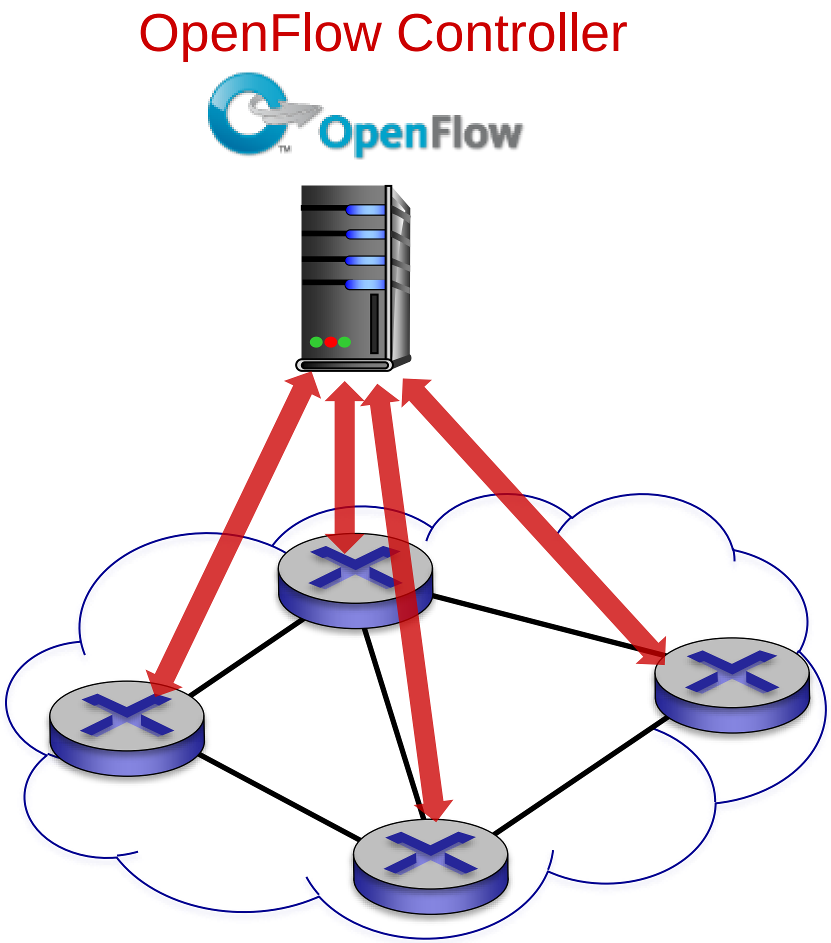
Interface layer to network control apps: abstractions API

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

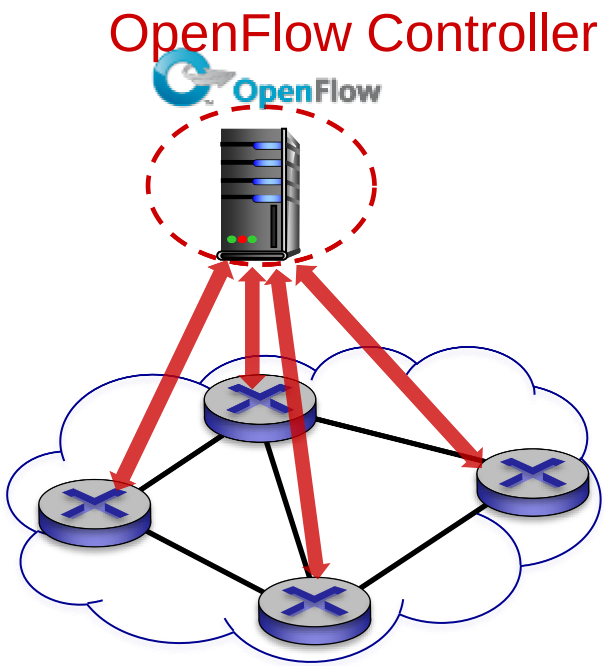


- operates between controller, switch
- TCP used to exchange messages
- OpenFlow messages:
  - controller-to-switch
  - switch to controller

OpenFlow: controller-to-switch messages

Key controller-to-switch messages

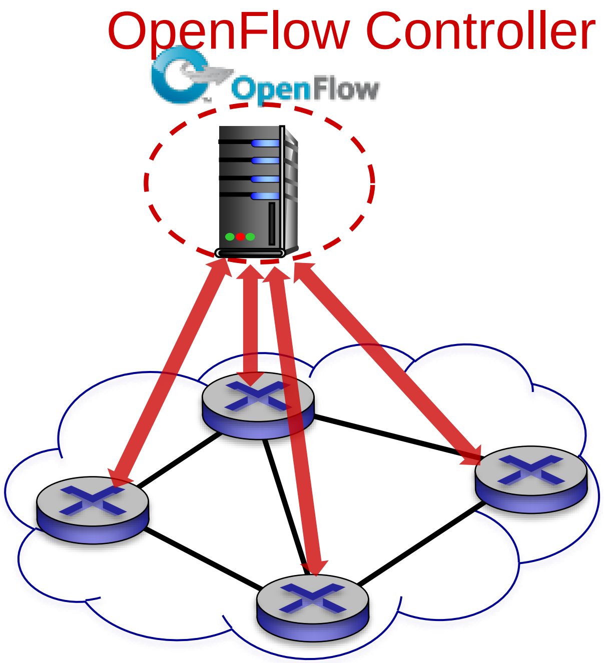
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *Read-state:* collect statistics and counter values from the switch's flow table and ports
- *packet-out:* controller can send this packet out of specific switch port



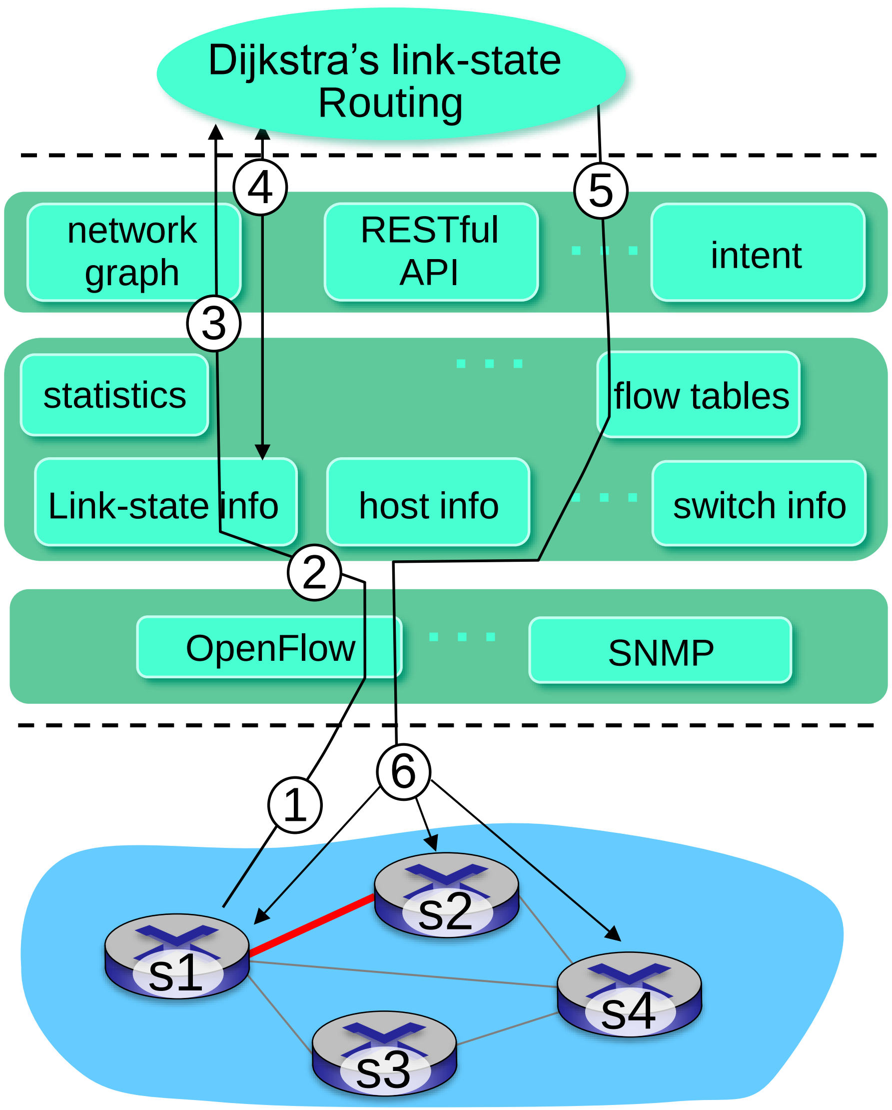
OpenFlow: switch-to-controller messages

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.



SDN: control/data plane interaction example



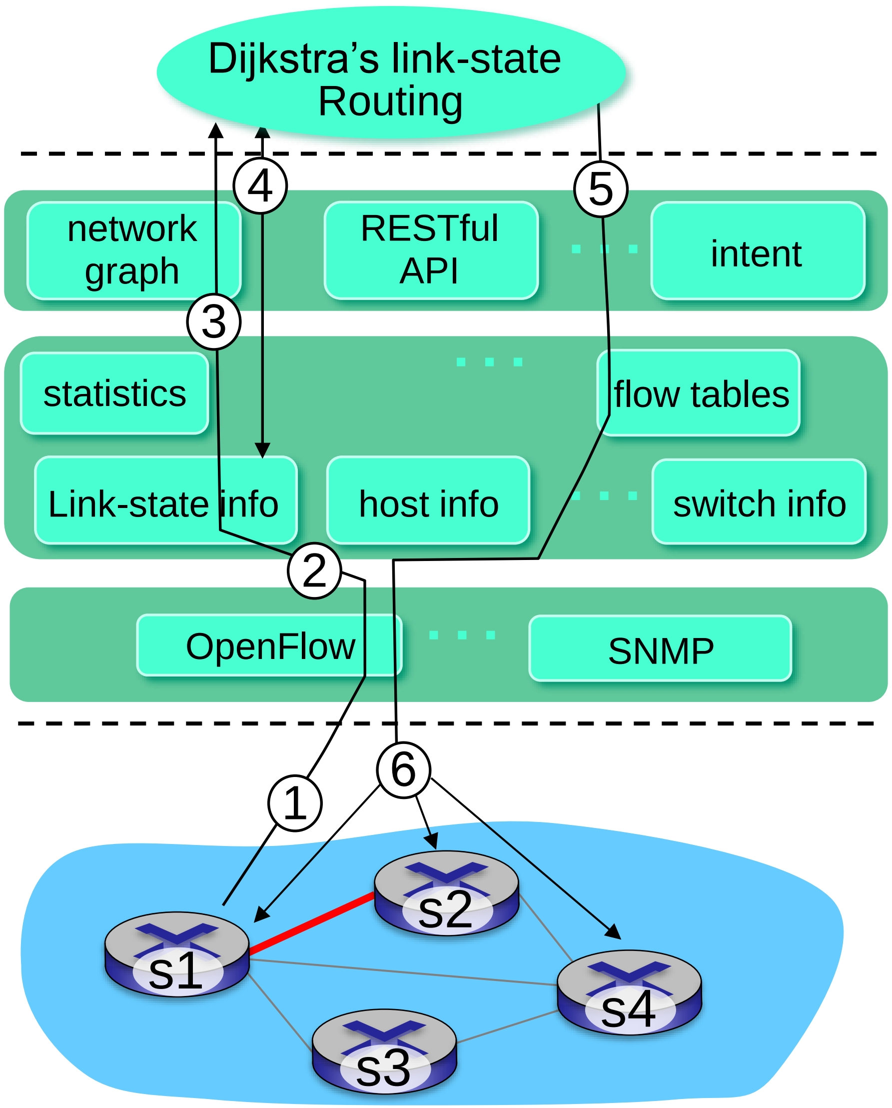
① S1, experiencing link failure using OpenFlow *port-status* message to notify controller

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

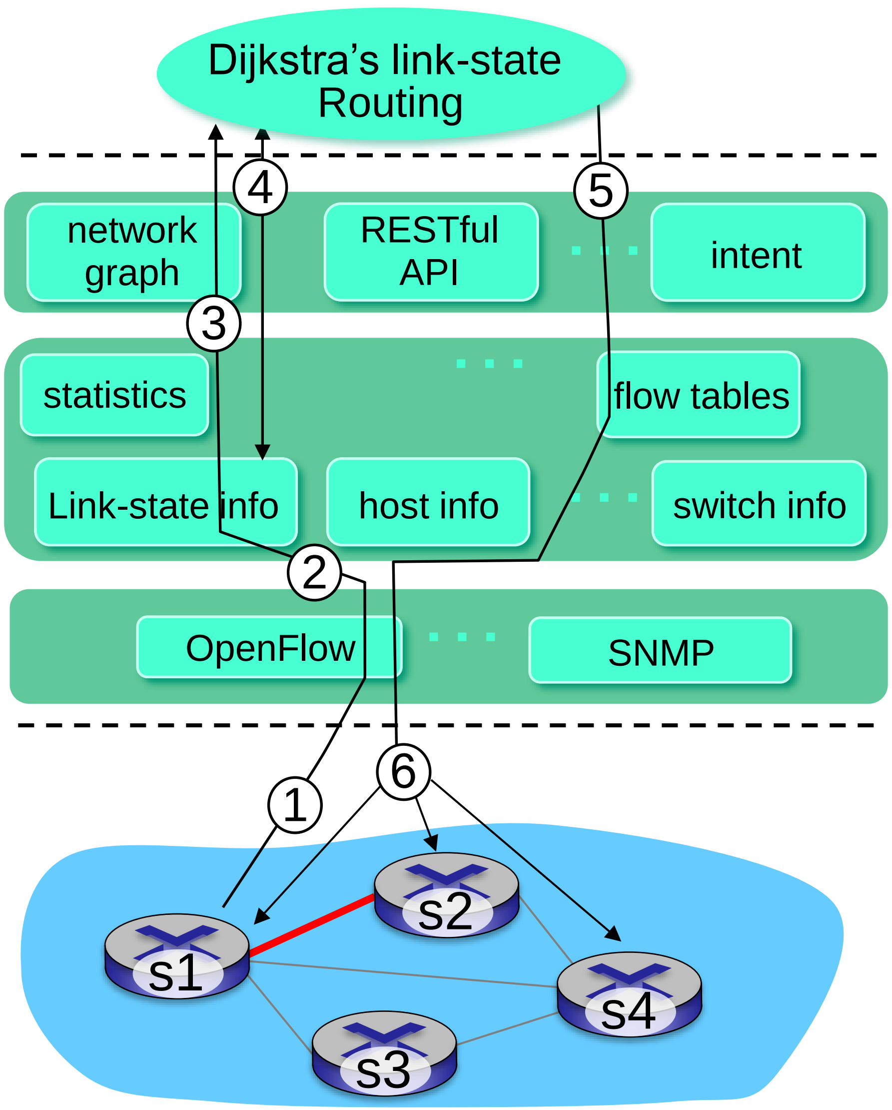
SDN: control/data plane interaction example



Two important differences from the earlier per-router-control scenario:

- Dijkstra's algorithm is executed as a separate application, outside of the packet switches.
- Packet switches send link updates to the SDN controller and not to each other.

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

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message:
  - Type + code + the header and the first 8 bytes of IP datagram causing error

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

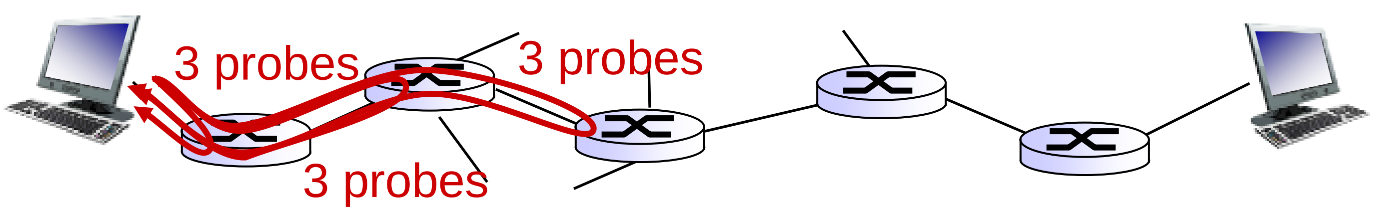
Traceroute and ICMP

- 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



Chapter 5: summary

we've learned a lot!

- approaches to network control plane
  - per-router control (traditional)
  - logically centralized control (software defined networking)
- traditional routing algorithms
  - implementation in Internet: OSPF, BGP
- SDN controllers
  - implementation in practice: ODL, ONOS
- Internet Control Message Protocol
- (network management)

next stop: link layer!

Chapter 6: Link layer and LANs

our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- instantiation, implementation of various link layer technologies

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

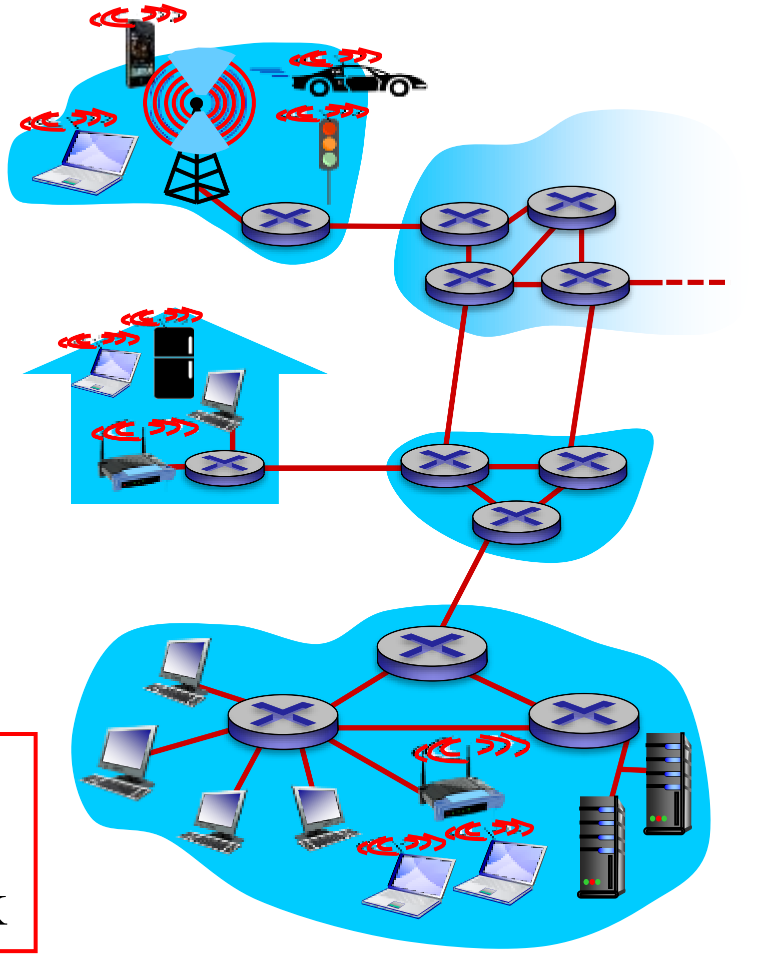
6.6 data center networking

6.7 a day in the life of a web request

Link layer: introduction

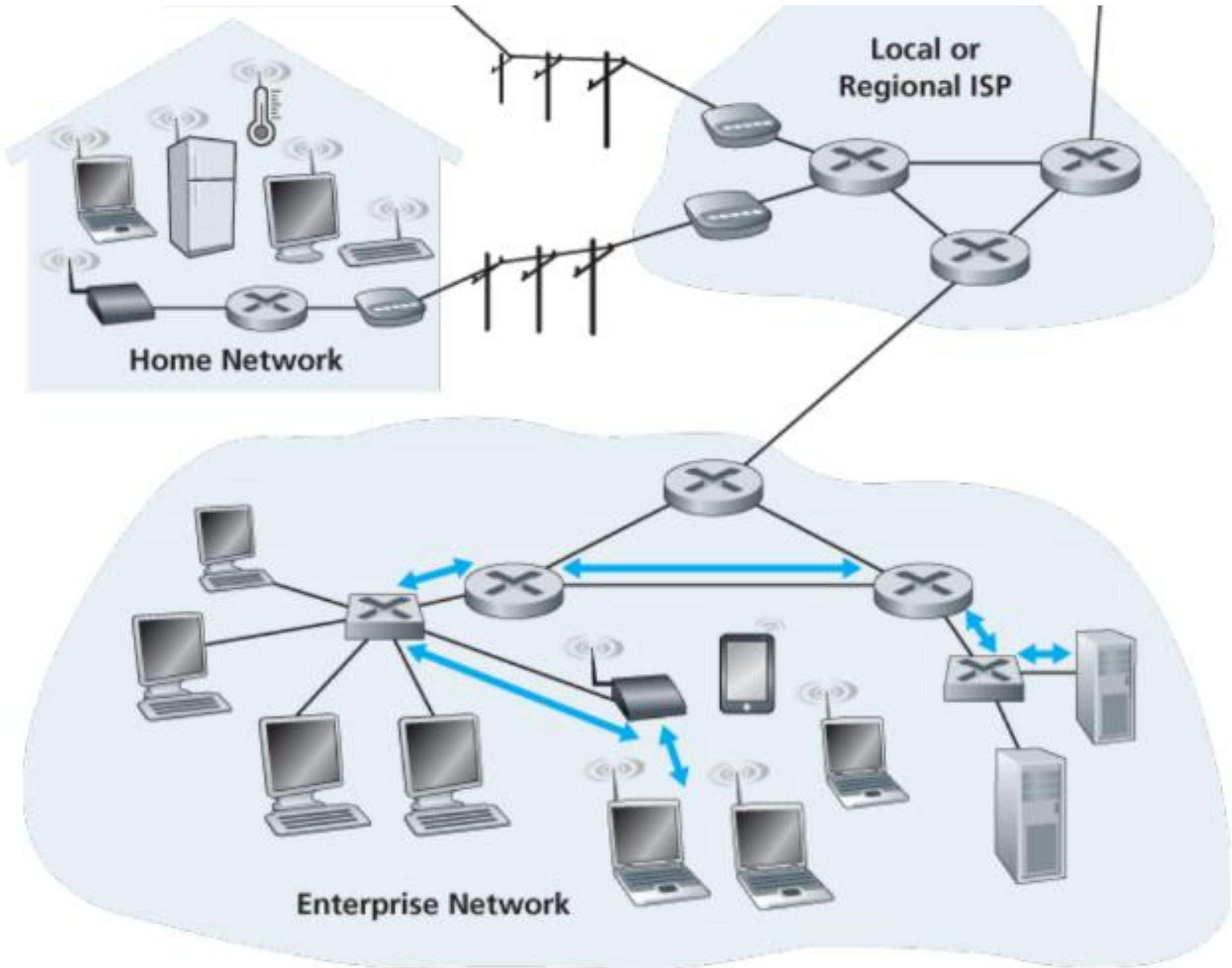
terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
- layer-2 packet: frame, encapsulates datagram



link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link

Link layer: introduction



Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, PPP on intermediate links, 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide rdt over link

transportation analogy:

- trip from SUSTech to Tsinghua
  - metro: SUSTech to SZ North
  - High speed train: SZ North to Beijing West
  - taxi: Beijing West to Tsinghua
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

Link layer services

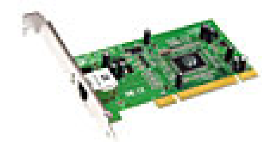
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, destination
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?

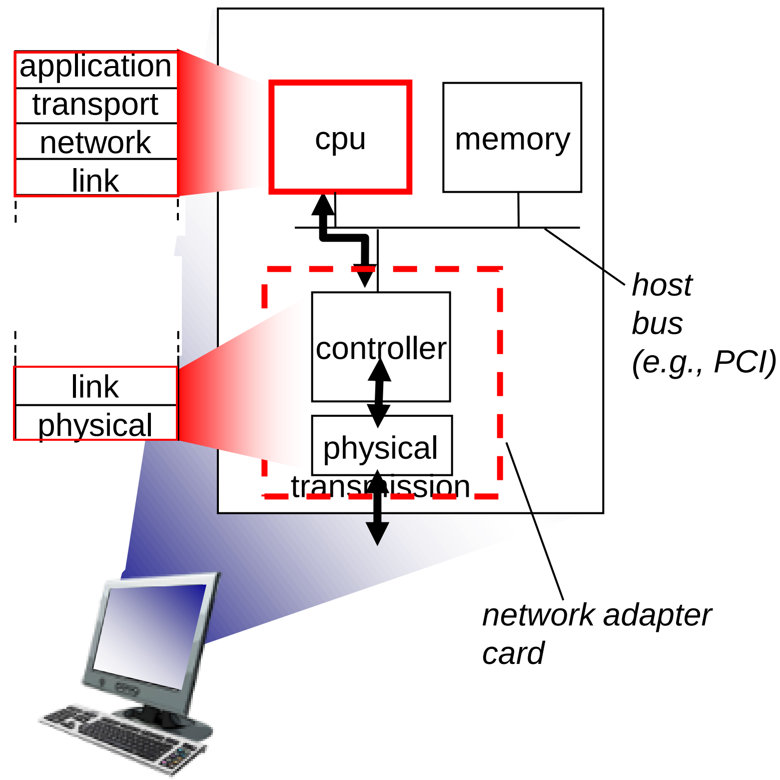
Link layer services (more)

- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

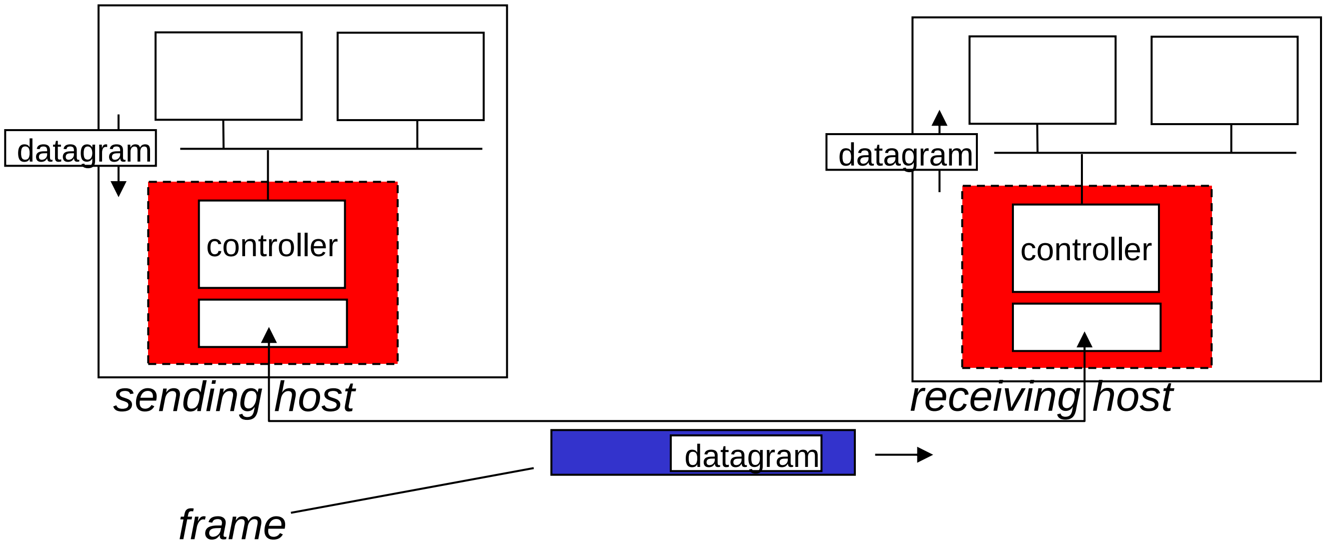
Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware





Adaptors communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagram, passes to upper layer at receiving side

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

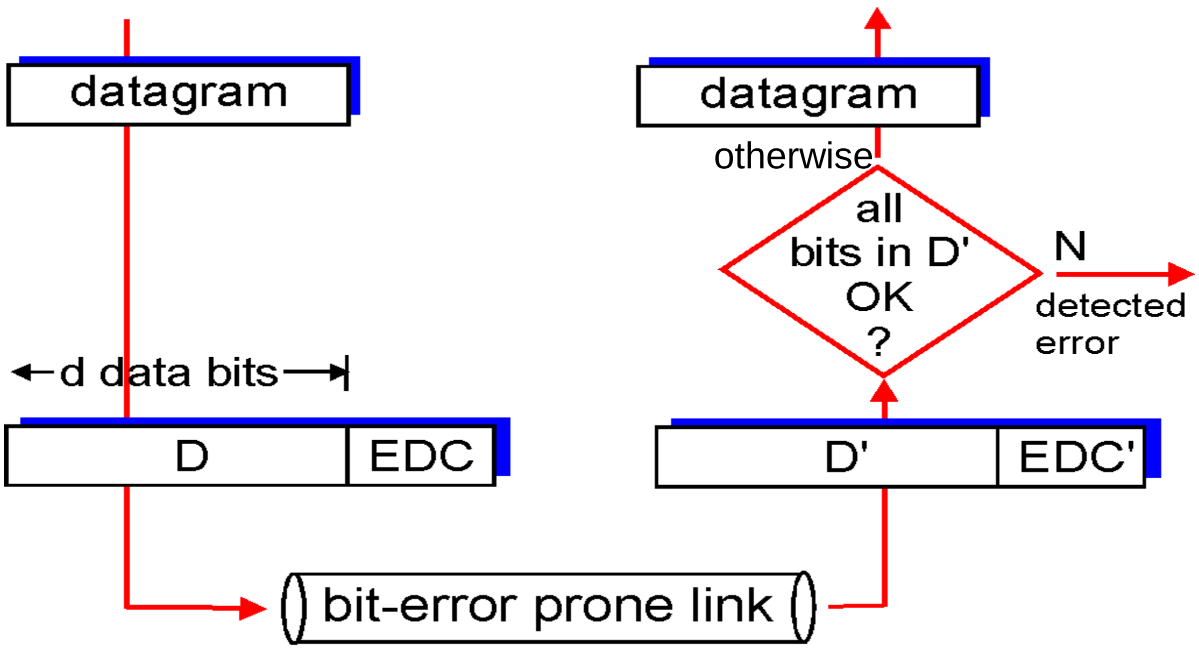
6.6 data center networking

6.7 a day in the life of a web request

Error detection

EDC= Error Detection and Correction bits
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction, but larger overhead

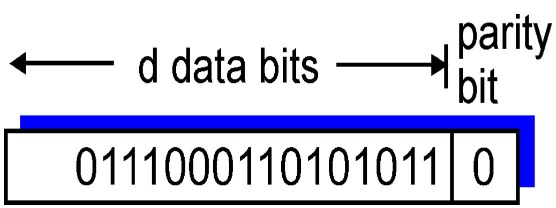


- Parity checks
- Check-summing methods
- Cyclic-redundancy check

Parity checking

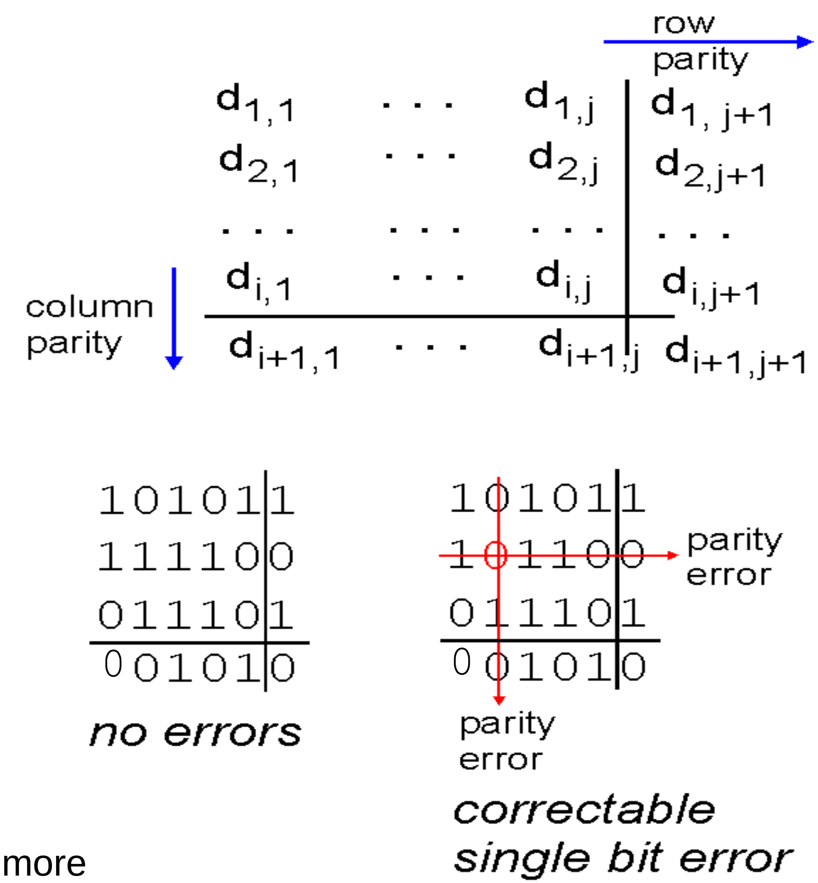
single bit parity:

- *d*etect single bit errors
- Even parity scheme
- Odd parity scheme



two-dimensional bit parity:

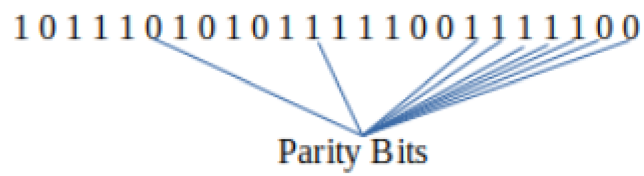
- detect and correct single bit errors



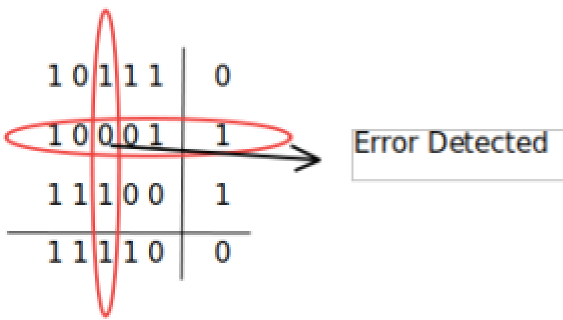
* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Parity checking

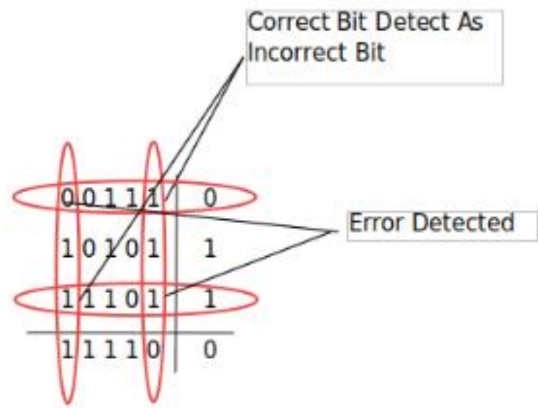
1 0 1 1 1	0
1 0 1 0 1	1
1 1 1 0 0	1
1 1 1 1 0	0



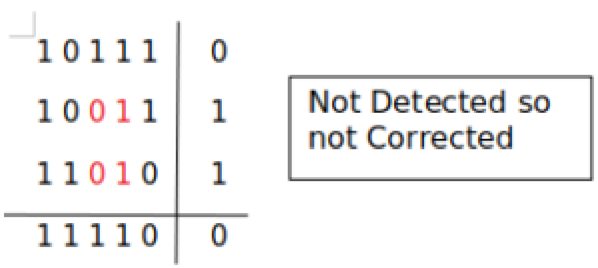
Case 1: a bit is in error.



Case 2: two bits are in error.



Case 3: error not detected



Many other cases …

Internet checksum (review)

goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

Cyclic redundancy check

- more powerful error-detection coding
- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all consecutive bit errors of r bits or less
- widely used in practice (Ethernet, 802.11 WiFi, ATM)

1011 XOR 0101 = 1110
1001 XOR 1101 = 0100

1011 - 0101 = 1110
1001 - 1101 = 0100

← d bits → ← r bits →

D: data bits to be sent	*R:* CRC bits

bit pattern

$D * 2^r$ XOR R

mathematical formula

Cyclic redundancy check

All CRC calculations are done in modulo-2 arithmetic without carries in addition or borrows in subtraction.

- This means that addition and subtraction are identical, and
- both are equivalent to the bitwise exclusive-or (XOR) of the operands.

```
1011 XOR 0101 = 1110        1011 - 0101 = 1110
1001 XOR 1101 = 0100        1001 - 1101 = 0100
```

Multiplication and division are the same as in base-2 arithmetic, except that any required addition or subtraction is done without carries or borrows.

```
            10001  remainder 101
      ___________
10011|100100110
      10011
          10110
          10011
            101
```

CRC example

want:

$D \cdot 2^r$ XOR $R = nG$

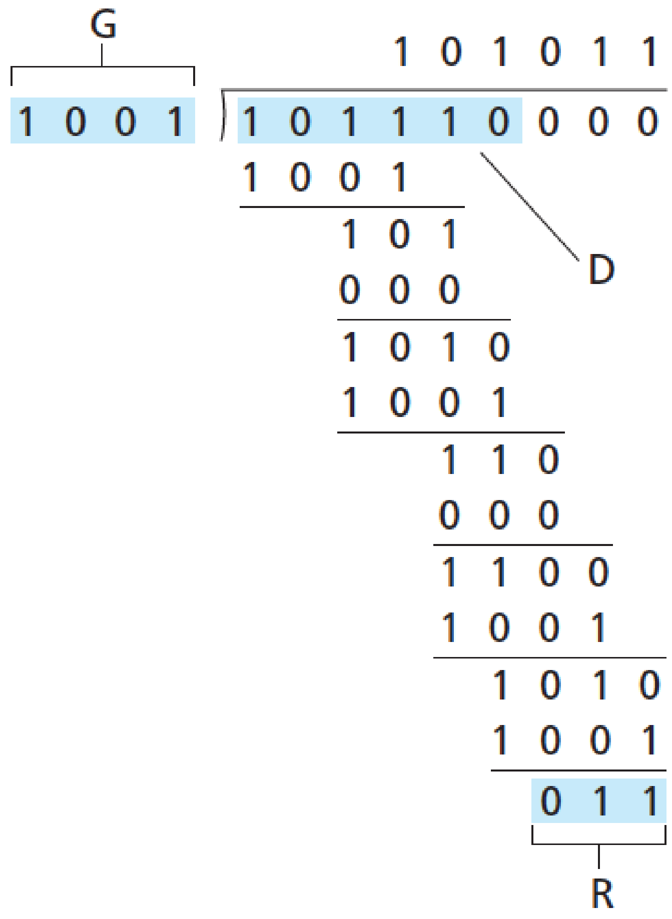
equivalently:

$D \cdot 2^r = nG$ XOR R

equivalently:

if we divide $D \cdot 2^r$ by G, want remainder R to satisfy:

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/