Network Layer

Network Layer

- Network Layer
  - Routing (Control Plane)
  - Forwarding (Data Plane)
  - Router Architecture Overview

- Data Plane
  - Forwarding
  - Internet Protocol (IP)
  - Generalized Forwarding and SDN (Software Defined Networking)

- Control Plane
  - Routing (Per-Router Control)
    - Algorithms
    - Protocols
    - Policies
  - Software Defined Networking (Logically Centralized Control)

Network Layer

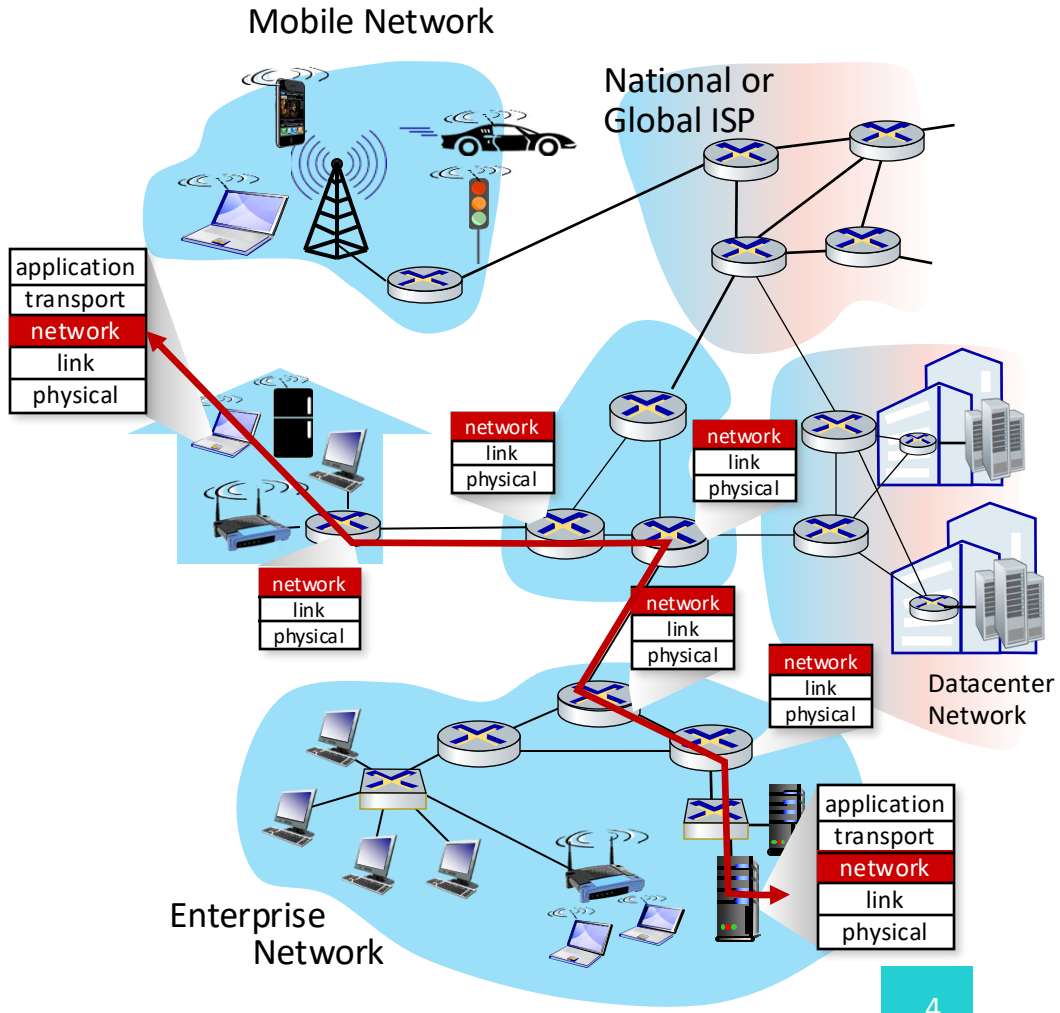
- Transport segment from sending to receiving host
- On sending side encapsulates segments into datagrams
- On receiving side, delivers segments to transport layer
- Network layer protocols in **every** host and router
- Router examines header fields in all IP datagrams passing through it

Network Layer

Network-layer functions

- **Routing:** Determine route taken by packets from source to destination
  - **Routing algorithms**

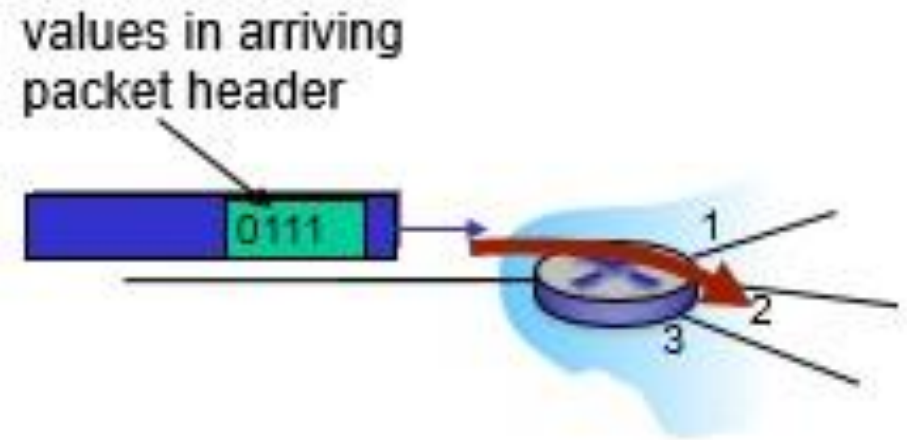
- **Forwarding:** Move packets from router's input to appropriate router output



Network Layer

Data plane

- Local, per-router function (Forwarding function)
- Determines how datagram arriving on router input port is forwarded to router output port

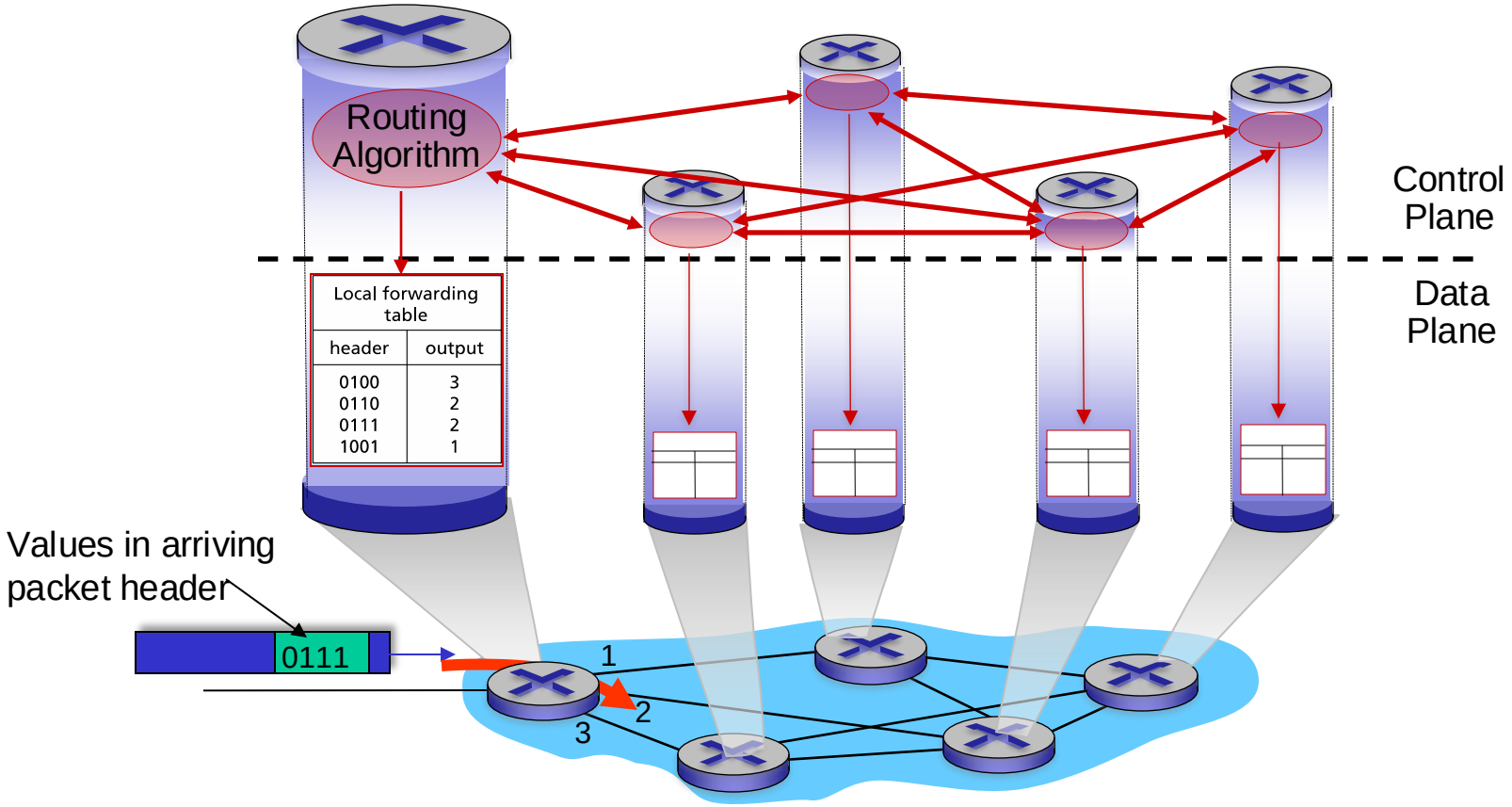


Control plane

- 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

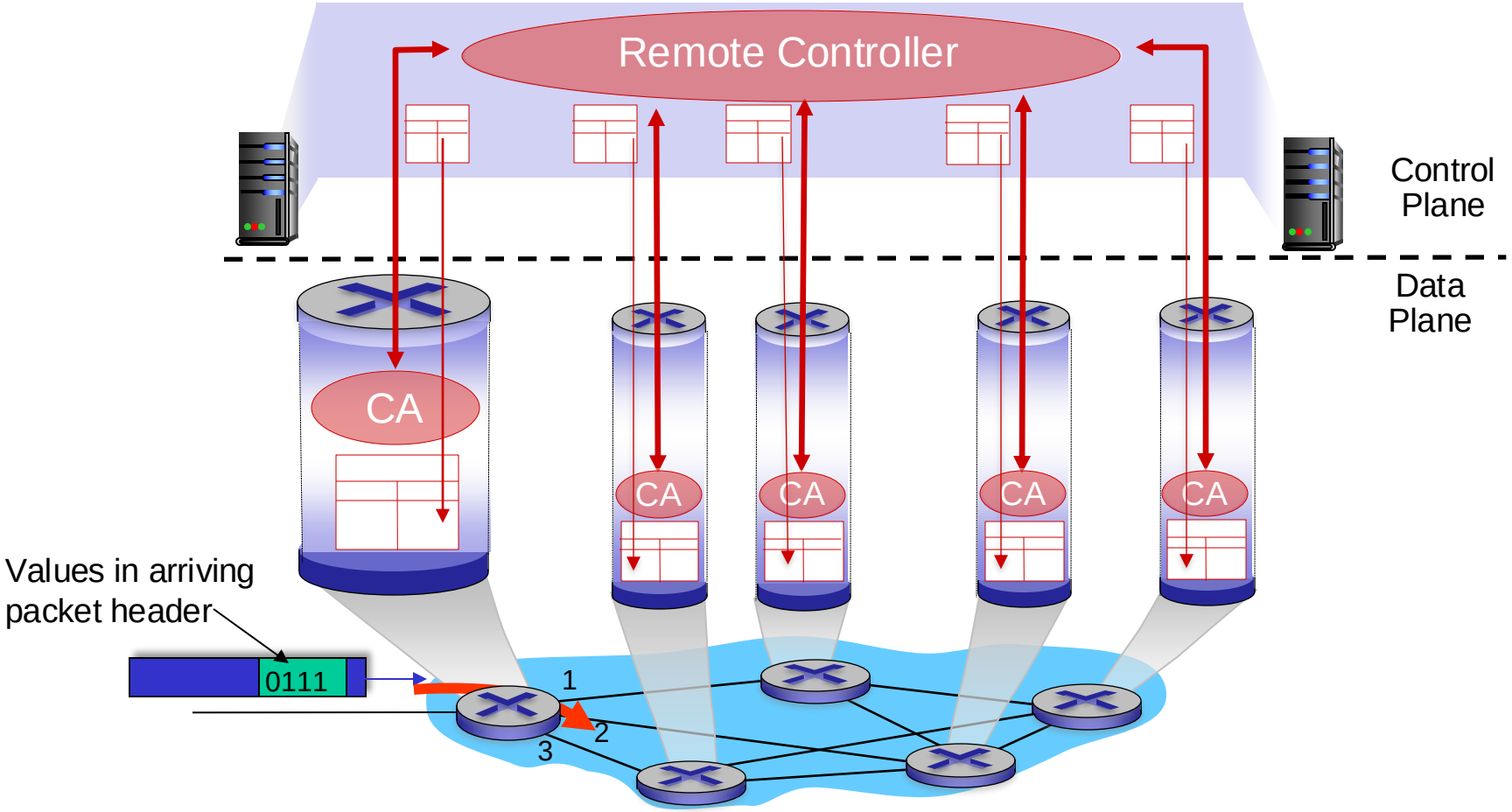
Per-Router Control Plane

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



Logically Centralized Control Plane

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



Network Service Model

Q: What **Service Model** for **Channel** transporting datagrams from sender to receiver?

Example services for individual datagrams:

- Guaranteed delivery
- Guaranteed delivery with less than $40\ msec$ delay

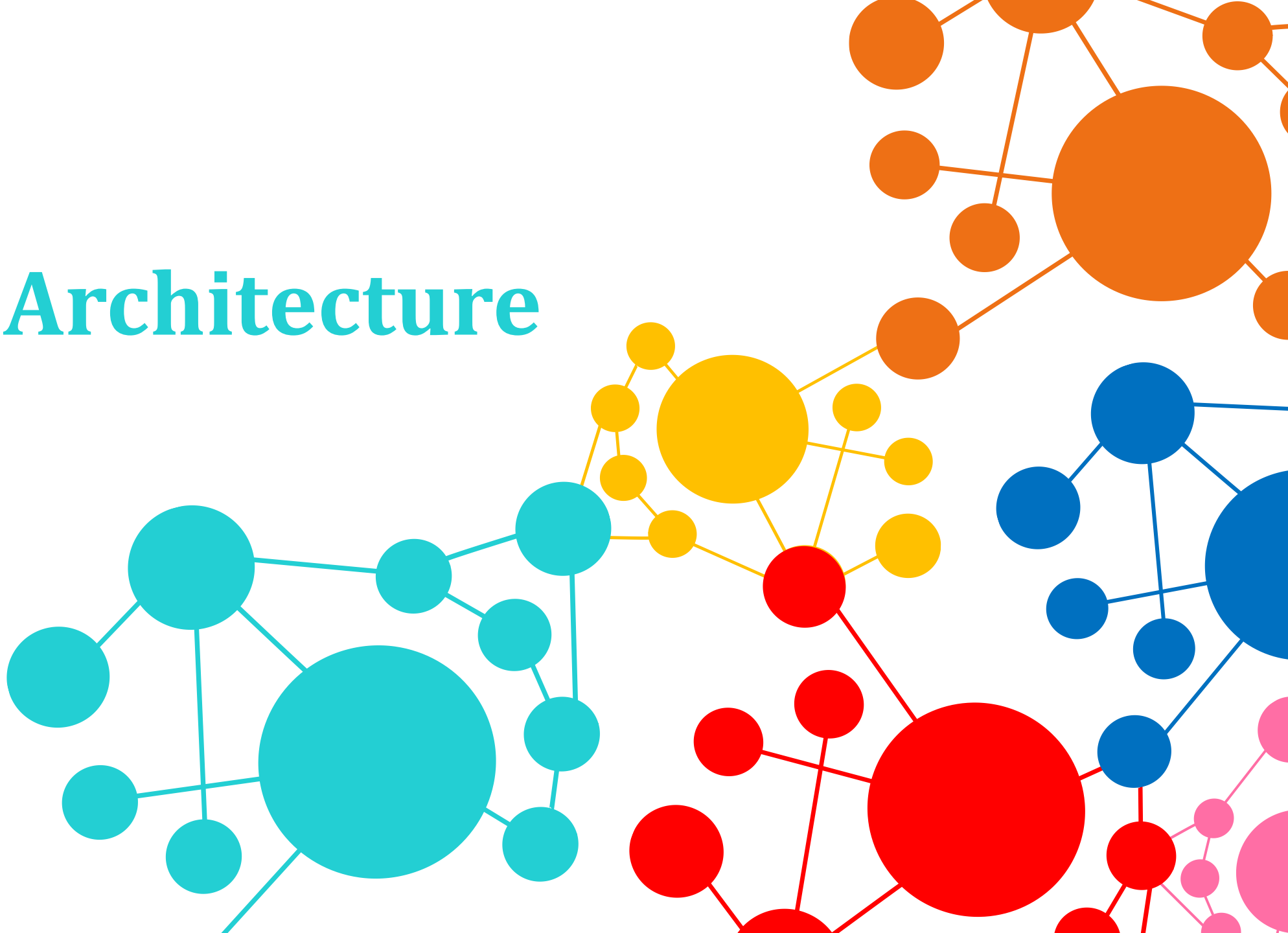
Example services for a flow of datagrams:

- In-order datagram delivery
- Guaranteed minimum bandwidth to flow
- Restrictions on changes in inter-packet spacing

Network Layer Service Models

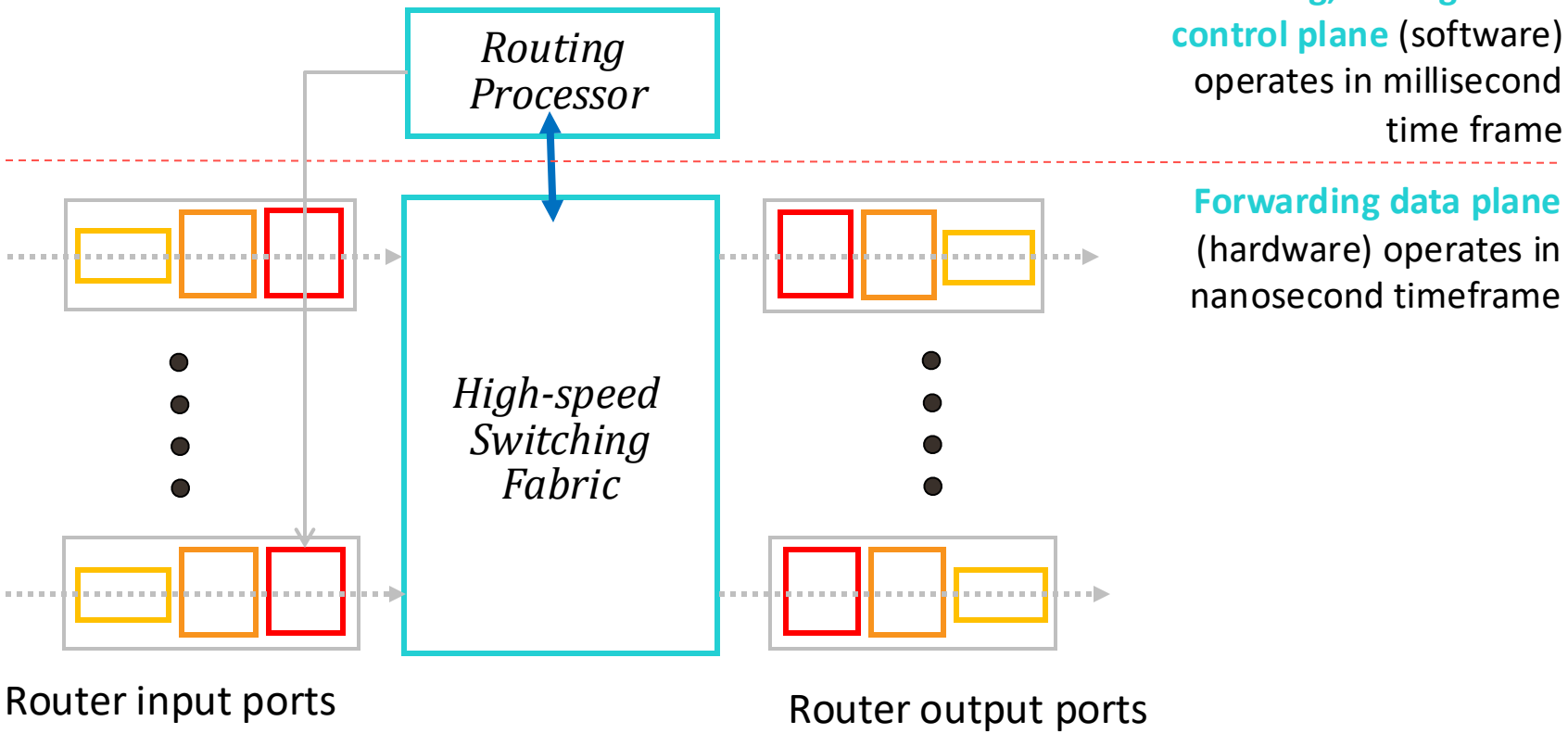
		Guarantees?				
Network Architecture	**Service Model**	**Bandwidth**	**Loss**	**Order**	**Timing**	**feedback**
ATM	CBR	Constant rate	Yes	Yes	Yes	No congestion
ATM	VBR	Guaranteed rate	Yes	Yes	Yes	No congestion
ATM	ABR	Guaranteed minimum	No	Yes	No	Yes
ATM	UBR	None	No	Yes	No	No
Internet	IntServ (RFC 1633)	Yes	Yes	Yes	Yes	Yes
Internet	DiffServ (RFC 2475)	Possible	Possible	Possible	Possible	No
Internet	**Best Effort**	**None**	**No**	**No**	**No**	**No (inferred)**

Router Architecture

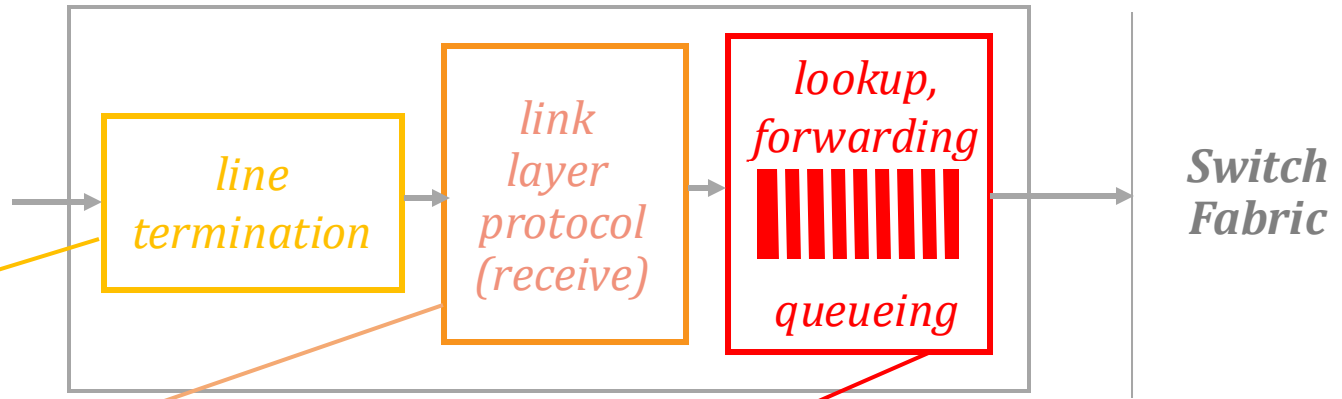


Router Architecture Overview

- High-level view of generic router architecture:



Input Port Functions



Physical layer:
Bit-level reception

Data link layer:
E.g., Ethernet see chapter 5

Decentralized Switching:

- *Using header field values, lookup output port using forwarding table in input port memory ("match plus action")*
- *Goal: complete input port processing at 'line speed'*
- *Queuing: if datagrams arrive faster than forwarding rate into switch fabric*
- *Destination-based forwarding:* forward based only on destination IP address (traditional)
- *Generalized forwarding:* forward based on any set of header field values

Destination-Based Forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: What happens if ranges do not divide up nicely?

Longest Prefix Matching

- **Longest prefix matching**
  - When looking for forwarding table entry for given destination address, use the **longest** address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

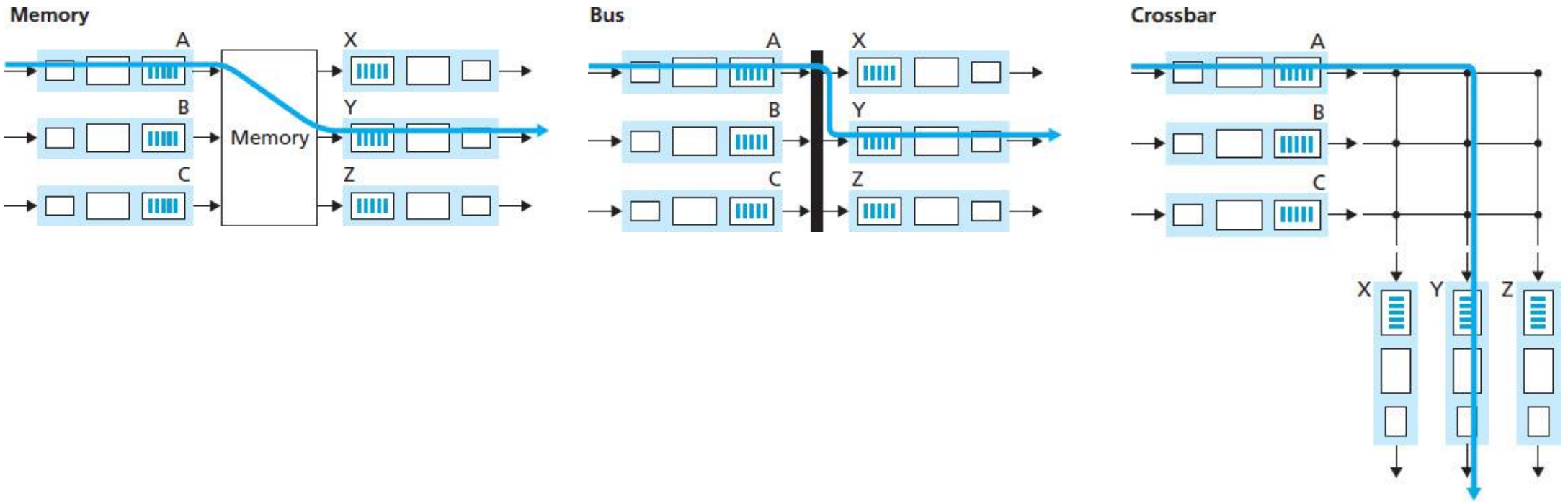
- **Examples**

DA: 11001000 00010111 00010110 10100001 **which interface?**

DA: 11001000 00010111 00011000 10101010 **which interface?**

Switching Fabrics

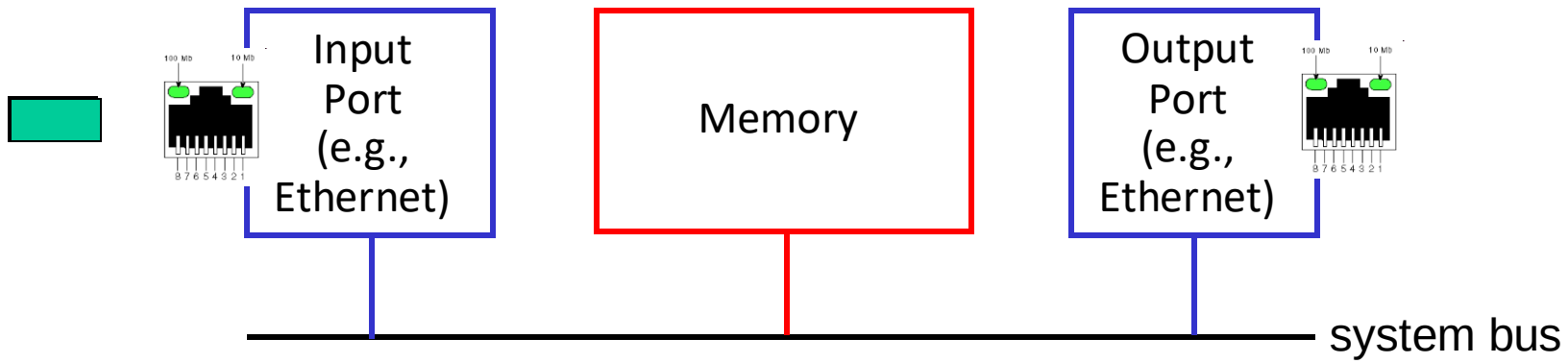
- Transfer packet from input buffer to appropriate output buffer
- Switching rate: Rate at which packets can be transferred from inputs to outputs



Switching via Memory

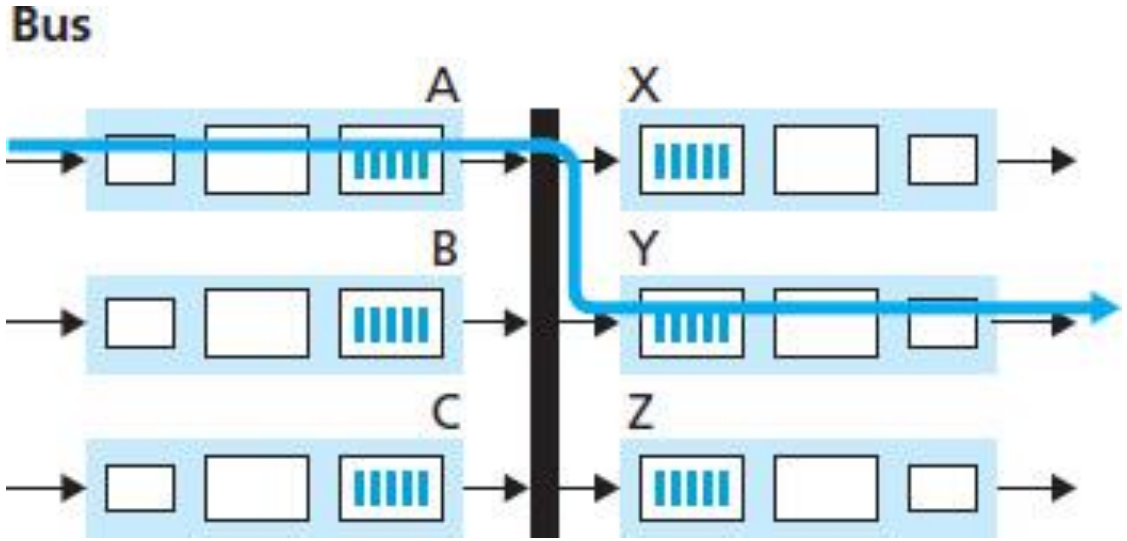
First generation routers

- Traditional computers with switching under direct control of CPU
- Packet copied to system memory
- Speed limited by memory bandwidth (2 bus crossings per datagram)



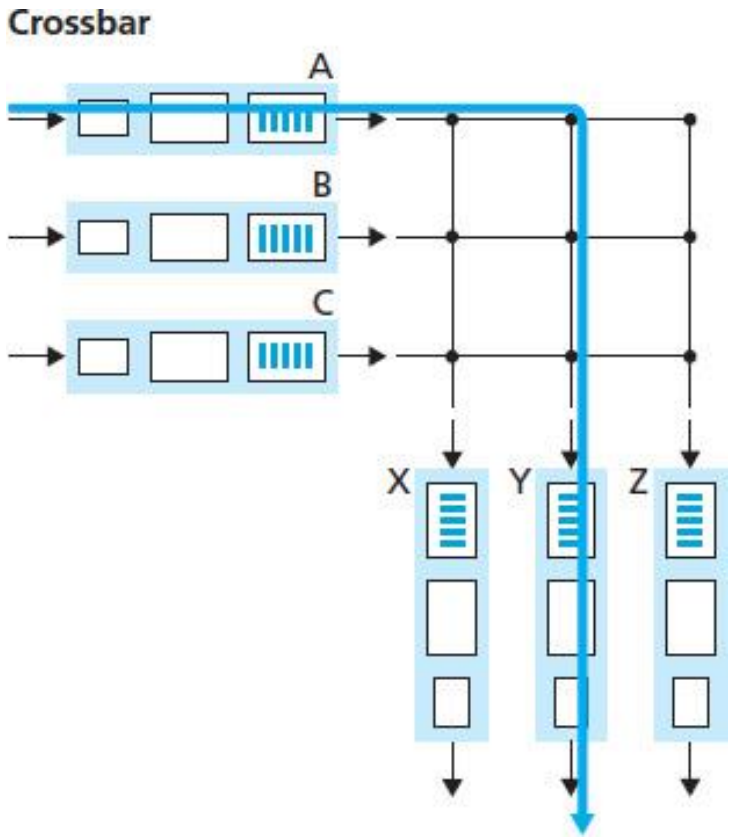
Switching via a Bus

- Datagram from input port memory to output port memory via a shared bus
- **Bus contention:** Switching speed limited by bus bandwidth



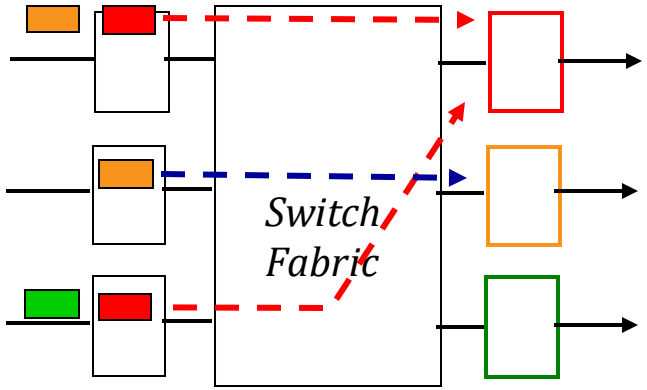
Switching via Interconnection Network

- Overcome bus bandwidth limitations
- Banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- **Advanced design:** Fragmenting datagram into fixed length cells, switch cells through the fabric.

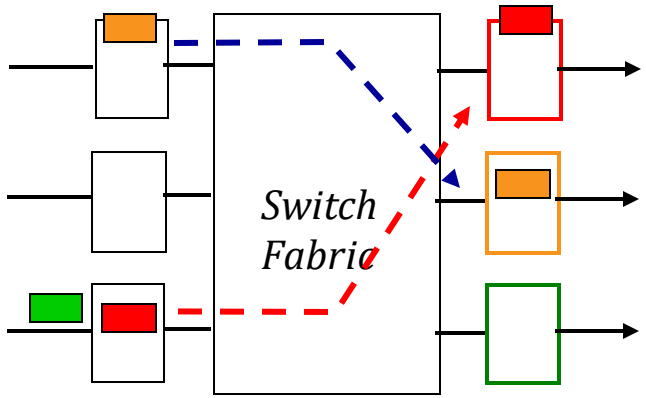


Input Port Queuing

- Fabric slower than input ports combined → Queueing may occur at input queues
  - Queueing delay and loss due to input buffer overflow!
- **Head-of-the-Line (HOL) blocking:** Queued datagram at front of queue prevents others in queue from moving forward



output port contention:
only one red datagram can be transferred.
lower red packet is blocked

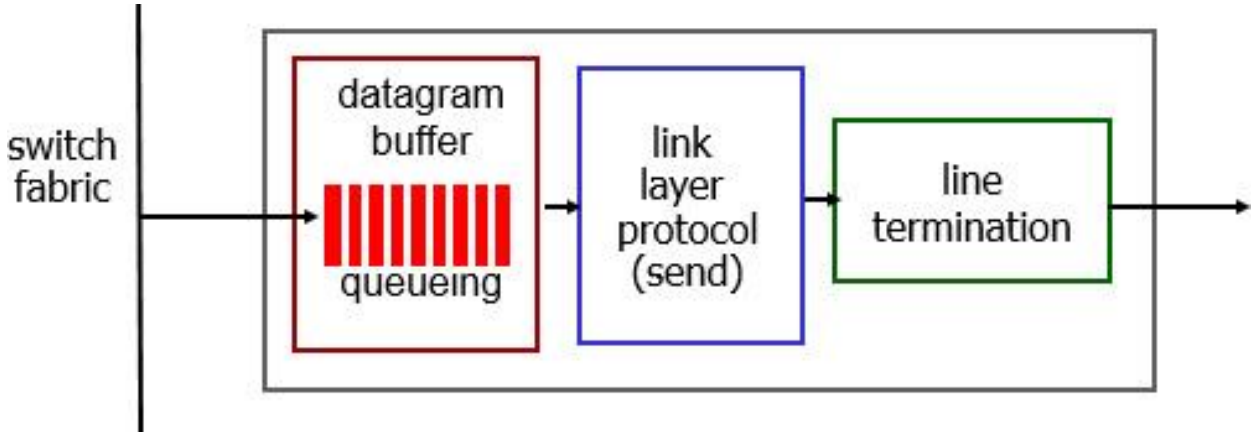


one packet time later: green packet experiences HOL blocking

Output Ports

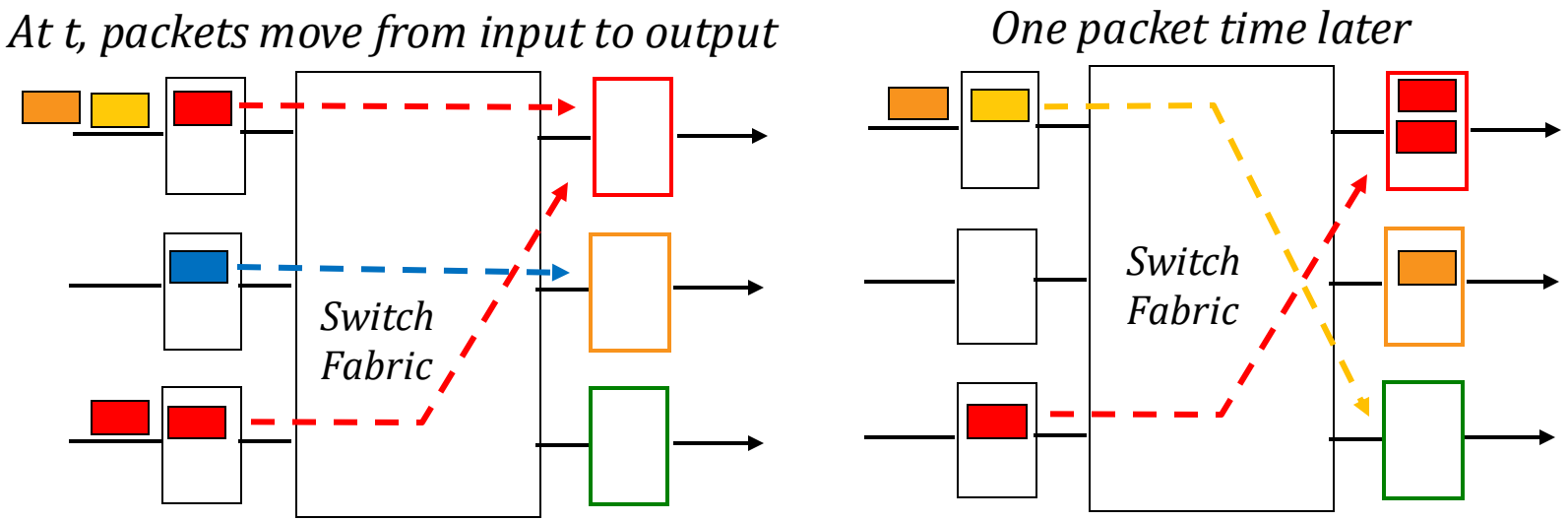
- **Buffering** required when datagrams arrive from fabric faster than the transmission rate

- **Scheduling discipline** chooses among queued datagrams for transmission



Output Port Queueing

- Buffering when arrival rate via switch exceeds output line speed
- **Queueing** (delay) and loss due to output port buffer overflow!

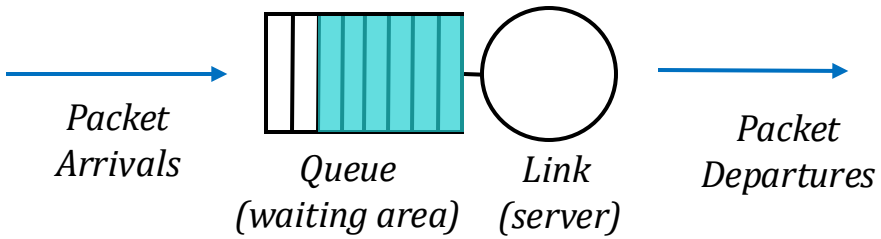


Scheduling Mechanisms

- **Scheduling discipline** chooses among queued the next packet to send on link

- **FIFO (first in first out) scheduling:** Send in order of arrival to queue
  - Real-world example?

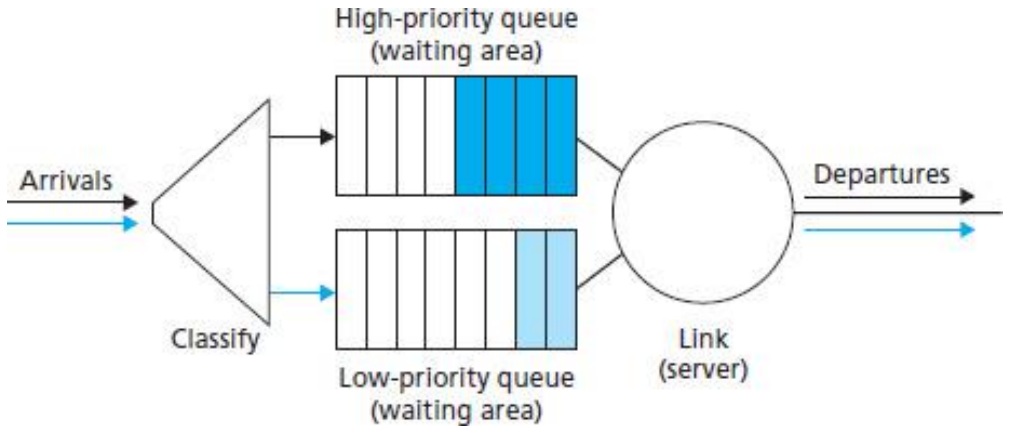
- **Discard policy:** If packet arrives to full queue: Who to discard?
  - **Tail drop:** Drop arriving packet
  - **Priority:** Drop/remove on priority basis
  - **Random:** Drop/remove randomly

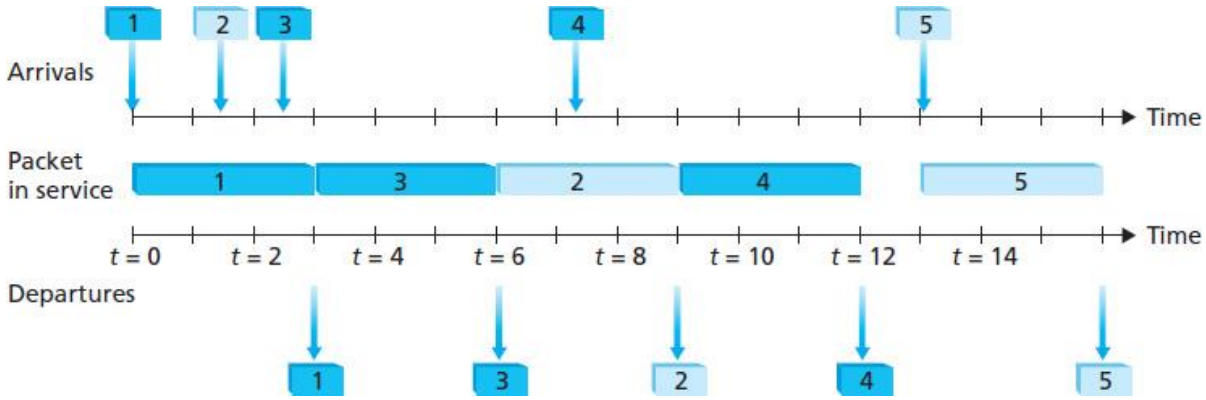


Scheduling Policies: Priority

Priority scheduling

- Multiple **classes** with different priorities: Send highest priority queued packet
- Class may depend on marking or other header info (e.g. IP source & destination & port number)
- Real world example?

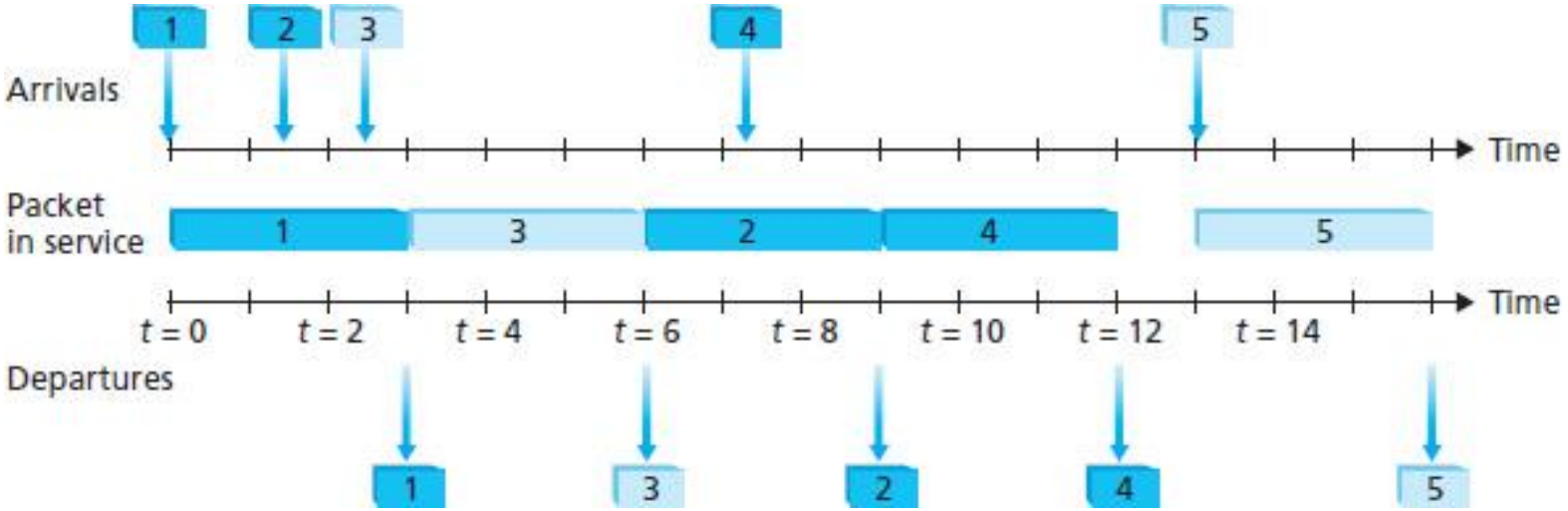




Scheduling Policies: Round Robin

Round Robin (RR) scheduling

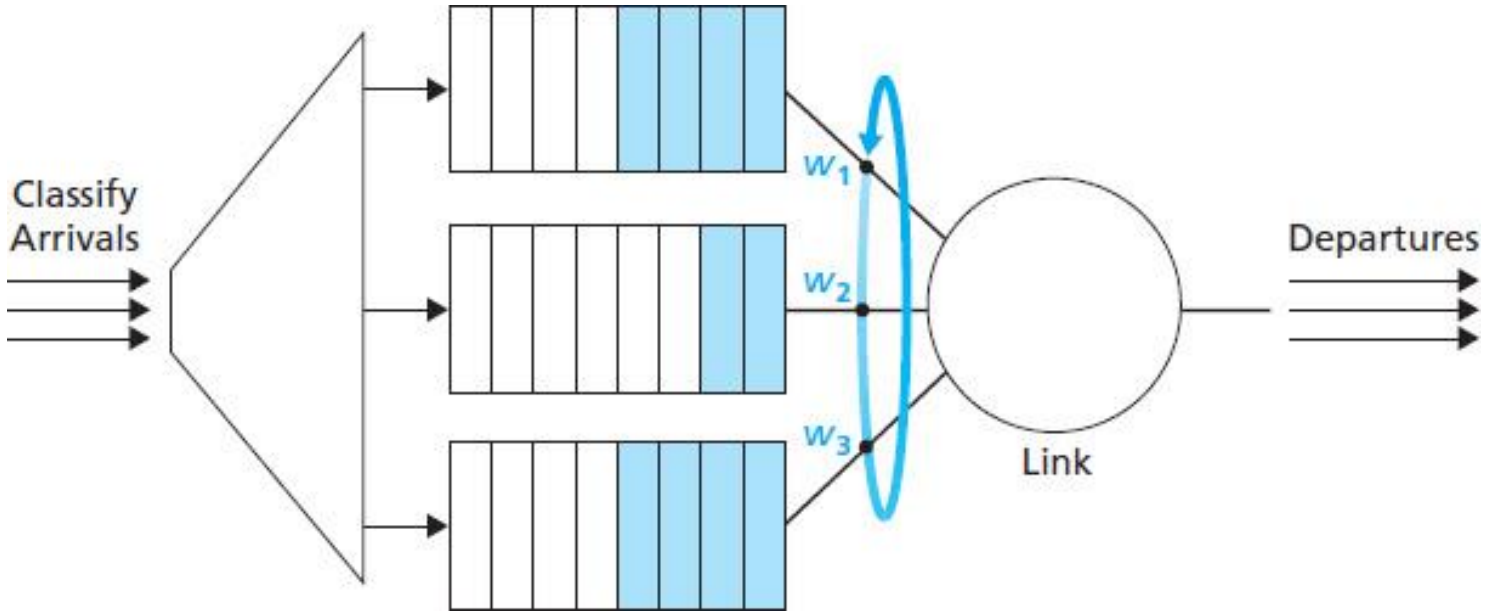
- Multiple classes
- Cyclically scan class queues, sending one complete packet from each class (if available)
- Real world example?



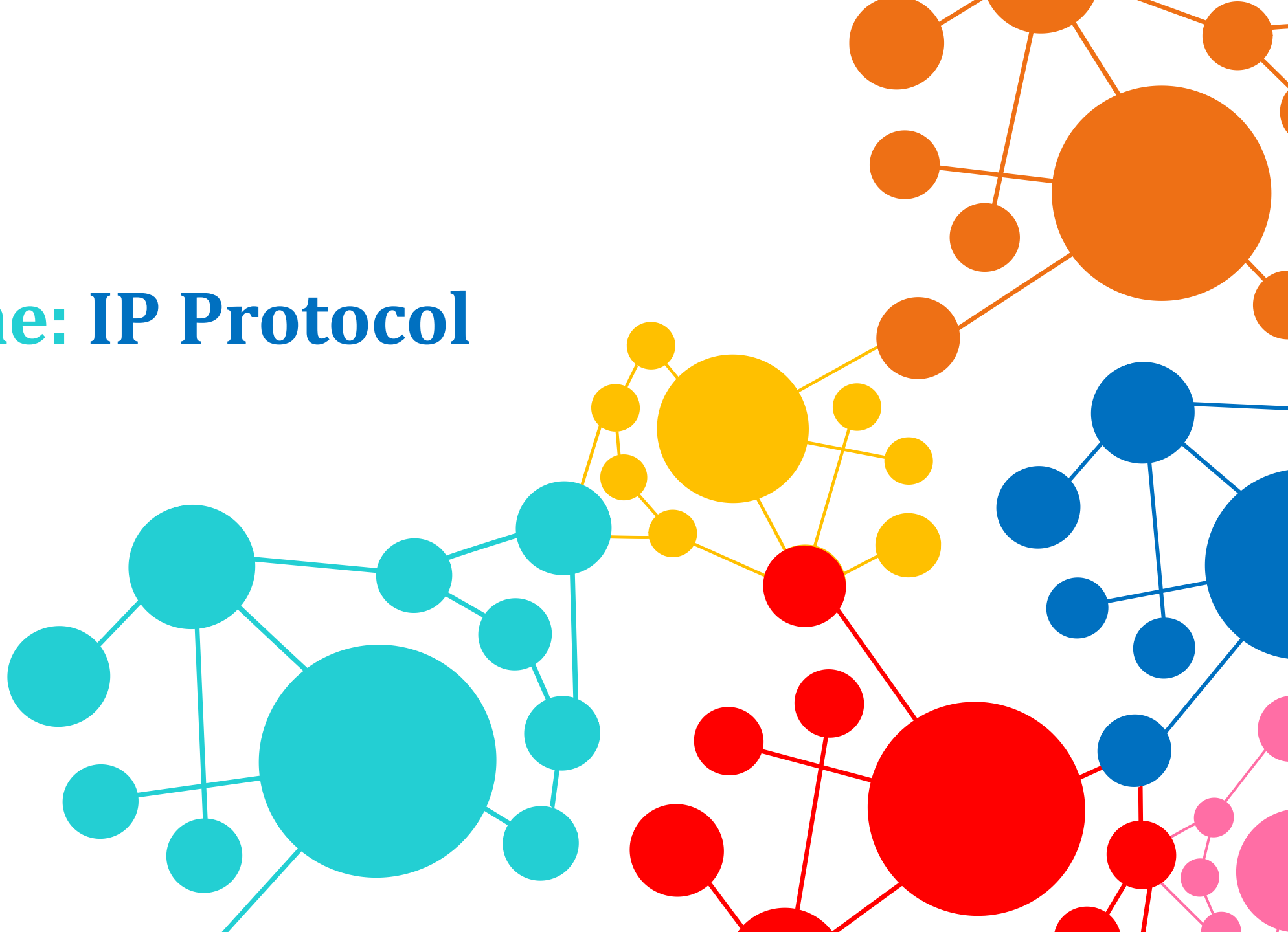
Scheduling Policies: WFQ

Weighted Fair Queuing (WFQ)

- Generalized Round Robin
- Each class gets weighted amount of service in each cycle
- Real-world example?

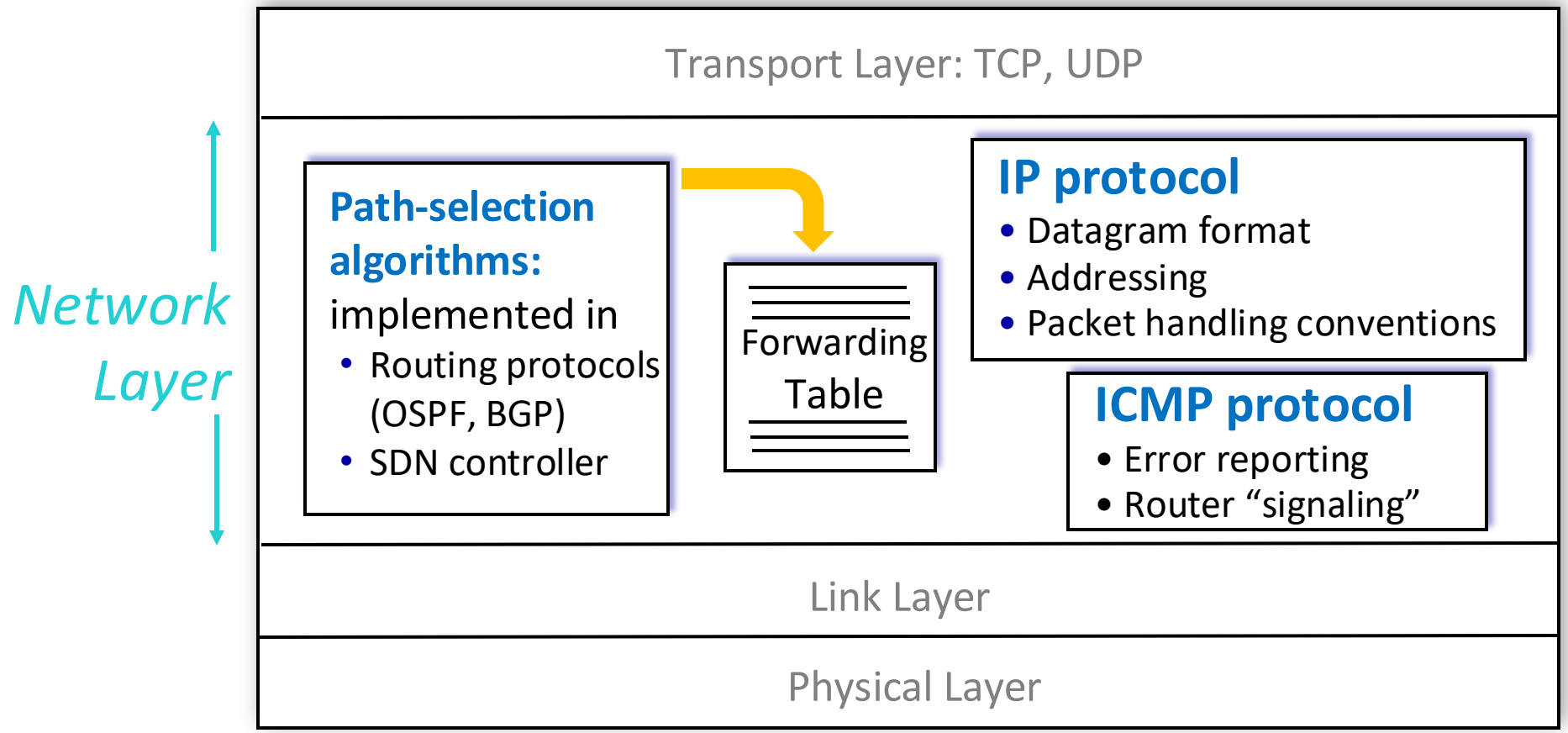


Data Plane: IP Protocol

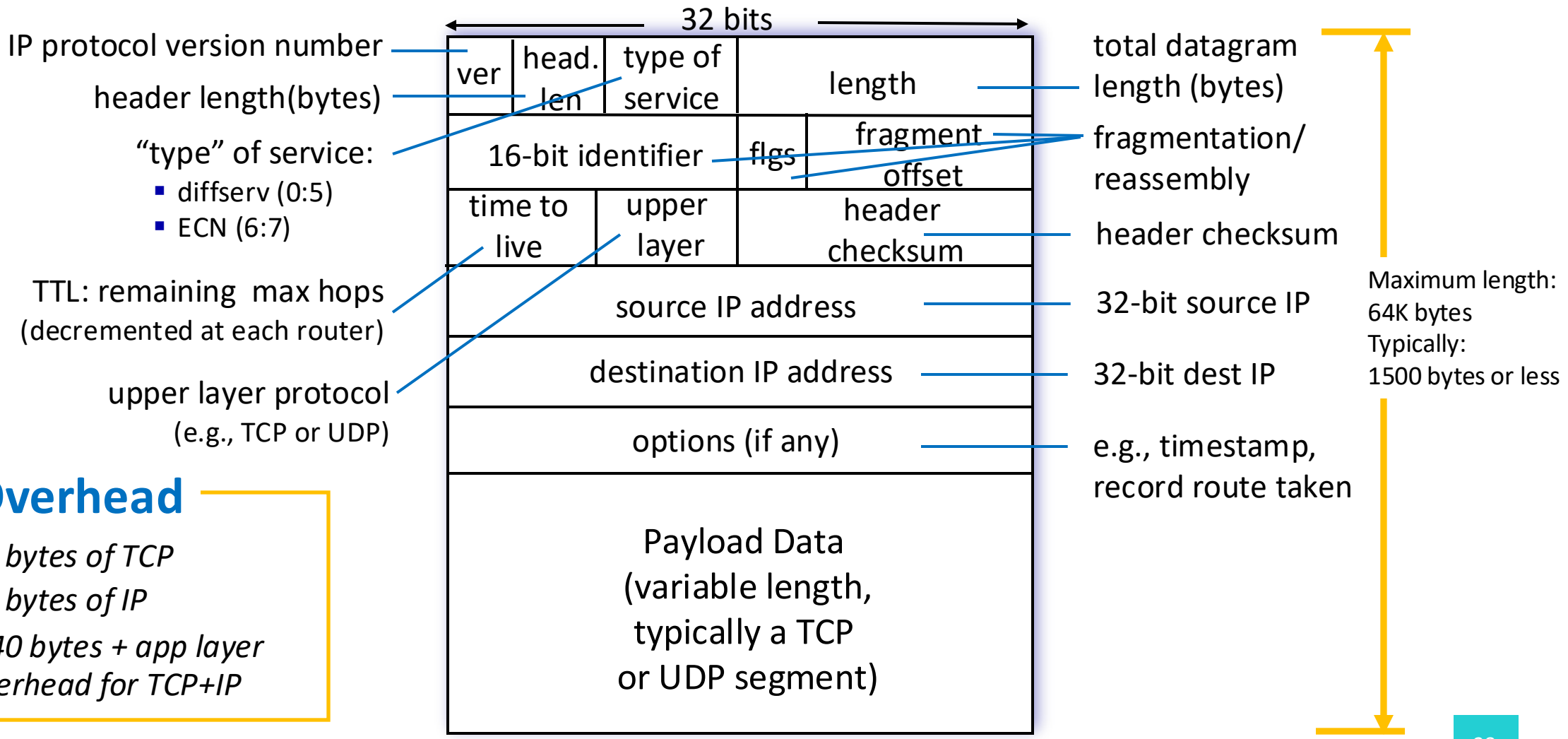


The Internet Network Layer

- Host and router network layer functions



IP Datagram Format



Overhead

20 bytes of TCP

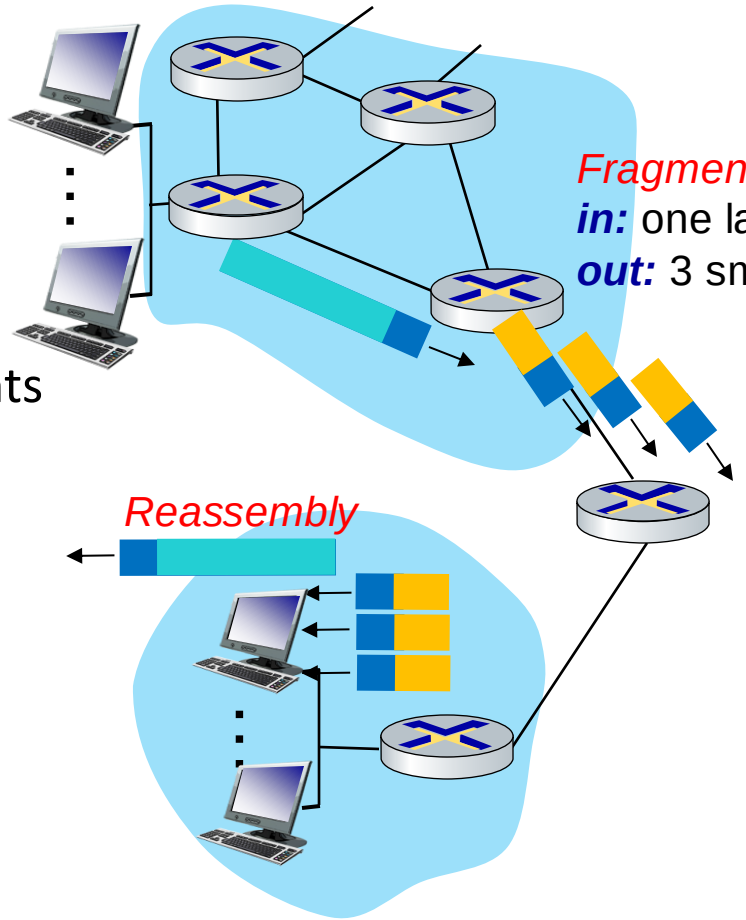
20 bytes of IP

= 40 bytes + app layer overhead for TCP+IP

IP Fragmentation, Reassembly

- Network links have MTU (Max Transfer Size) - largest possible link-level frame
  - Different link types, different MTUs

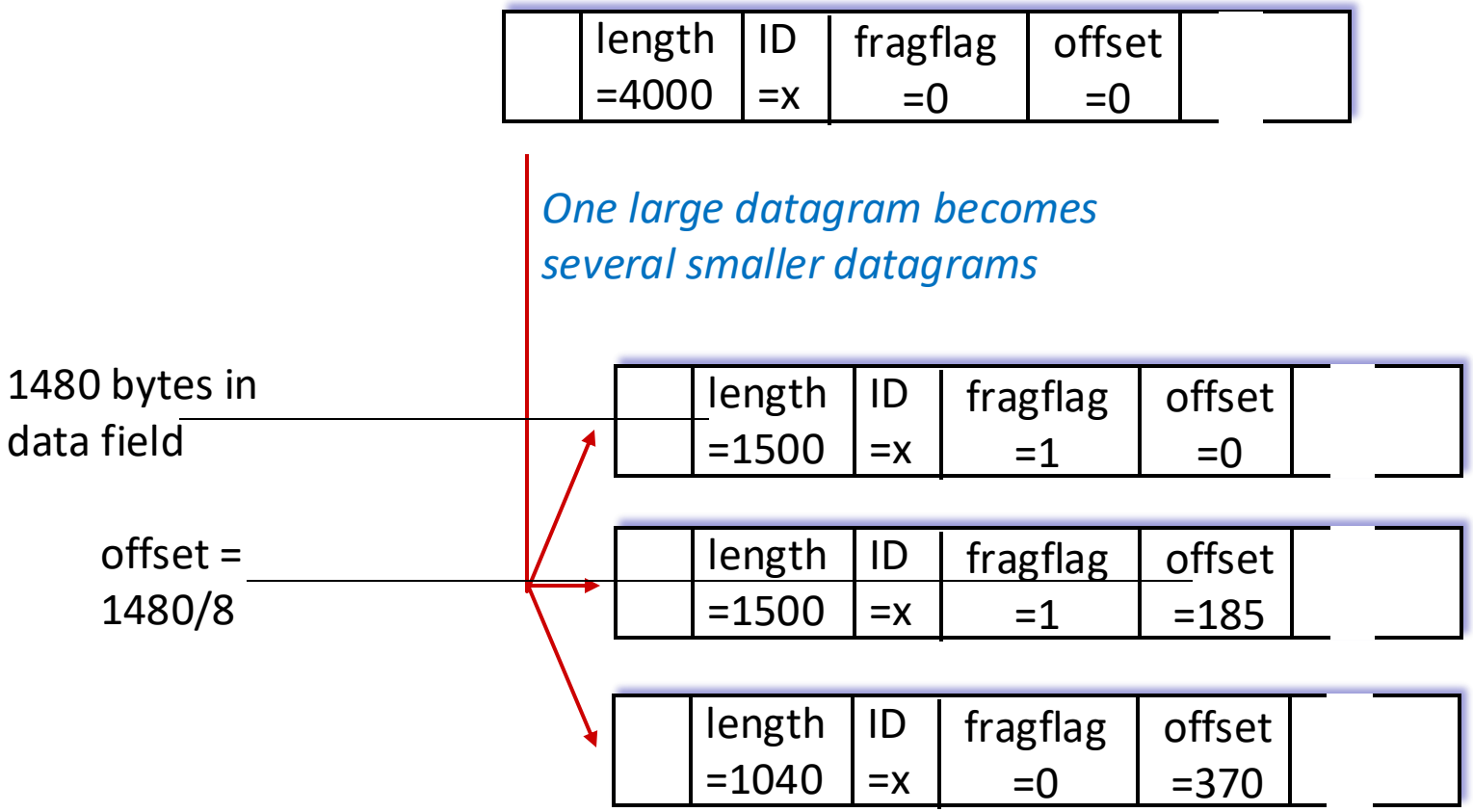
- Large IP datagram divided (fragmented) within net
  - One datagram becomes several datagrams
  - Reassembled only at final destination
  - IP header bits used to identify, order related fragments



IP Fragmentation, Reassembly

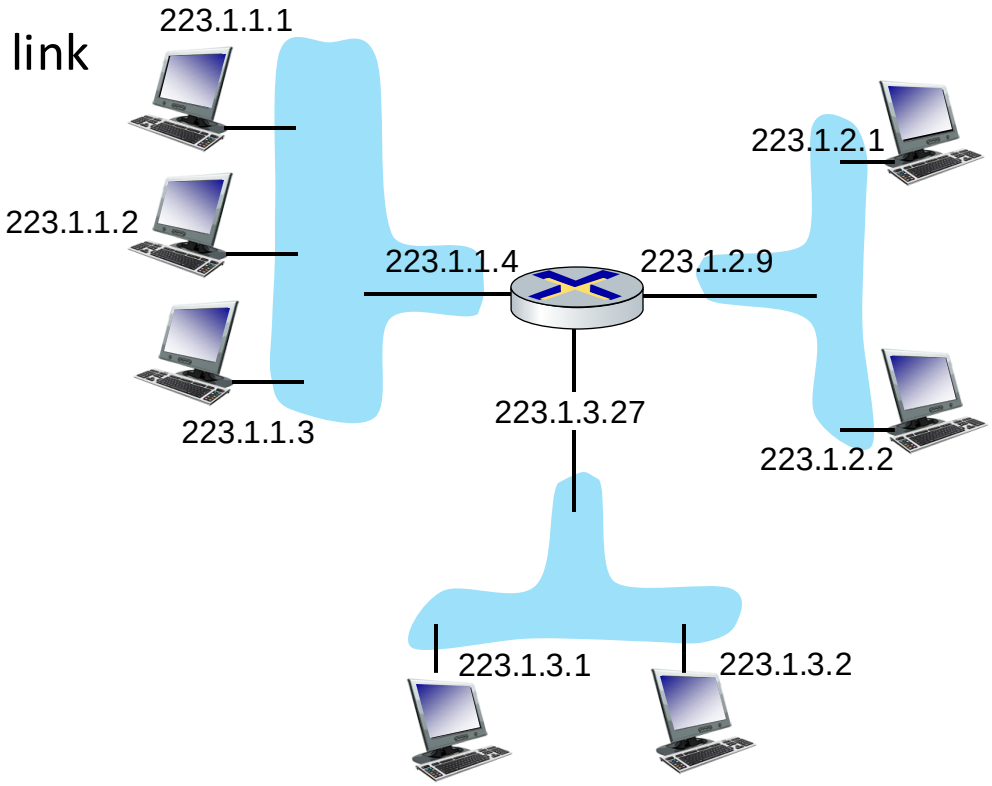
Example:

- 4000 byte datagram
- MTU = 1500 bytes



IP Addressing: Introduction

- **IP address:** 32-bit identifier for host, router **interface**
- **Interface:** Connection between host/router and physical link
  - Routers typically have multiple interfaces
  - Host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- **IP addresses associated with each interface**



Dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

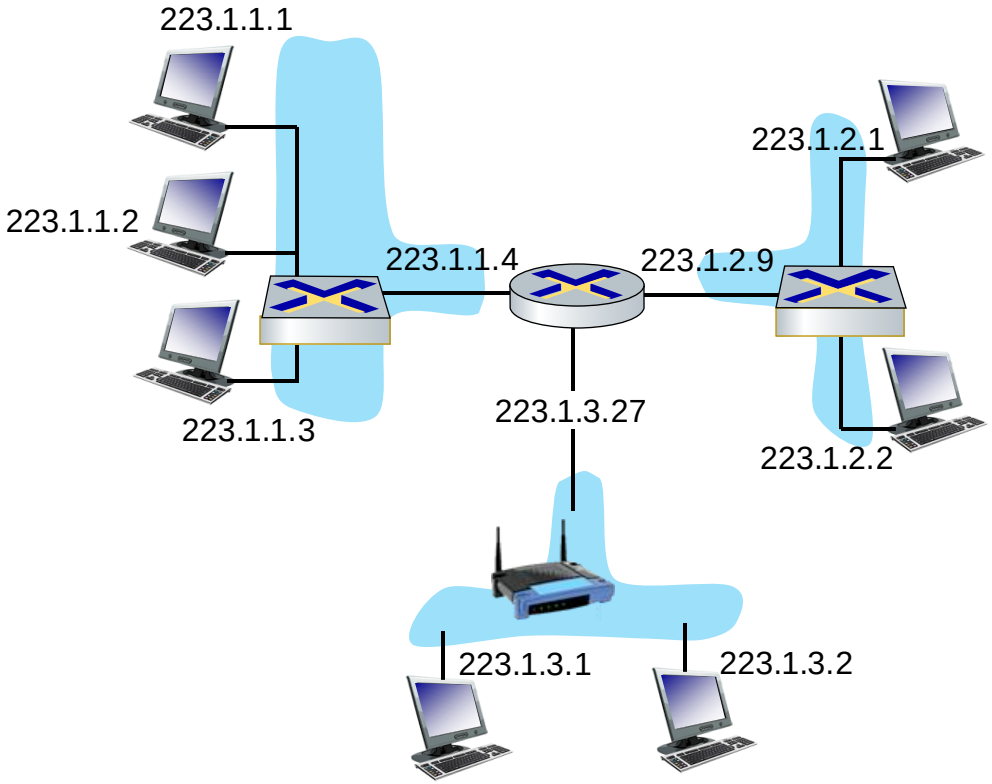
223 1 1 1

IP Addressing: Introduction

Q: How are interfaces actually connected?

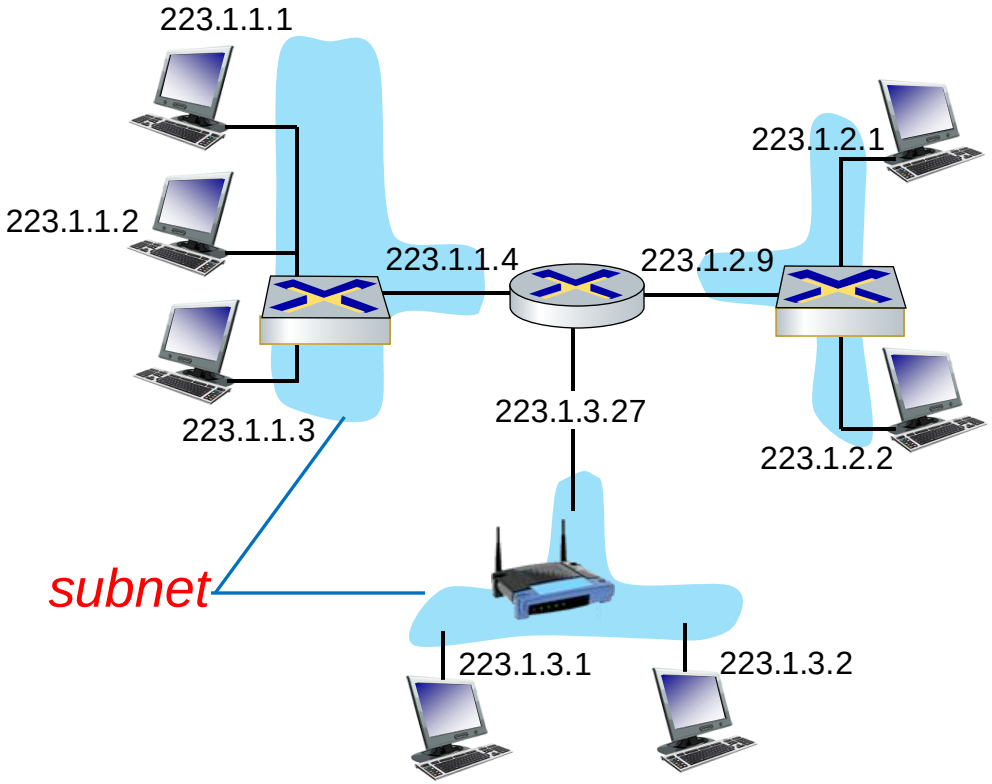
A: We will learn about that in Data Link Layer discussions.

For now: Do **not** need to worry about how one interface is connected to another (with no intervening router)



Subnets

- **IP address**
  - Subnet part - high order bits
  - Host part - low order bits
- **What is a subnet?**
  - Device interfaces with same subnet part of IP address
  - Can physically reach each other **without intervening router**

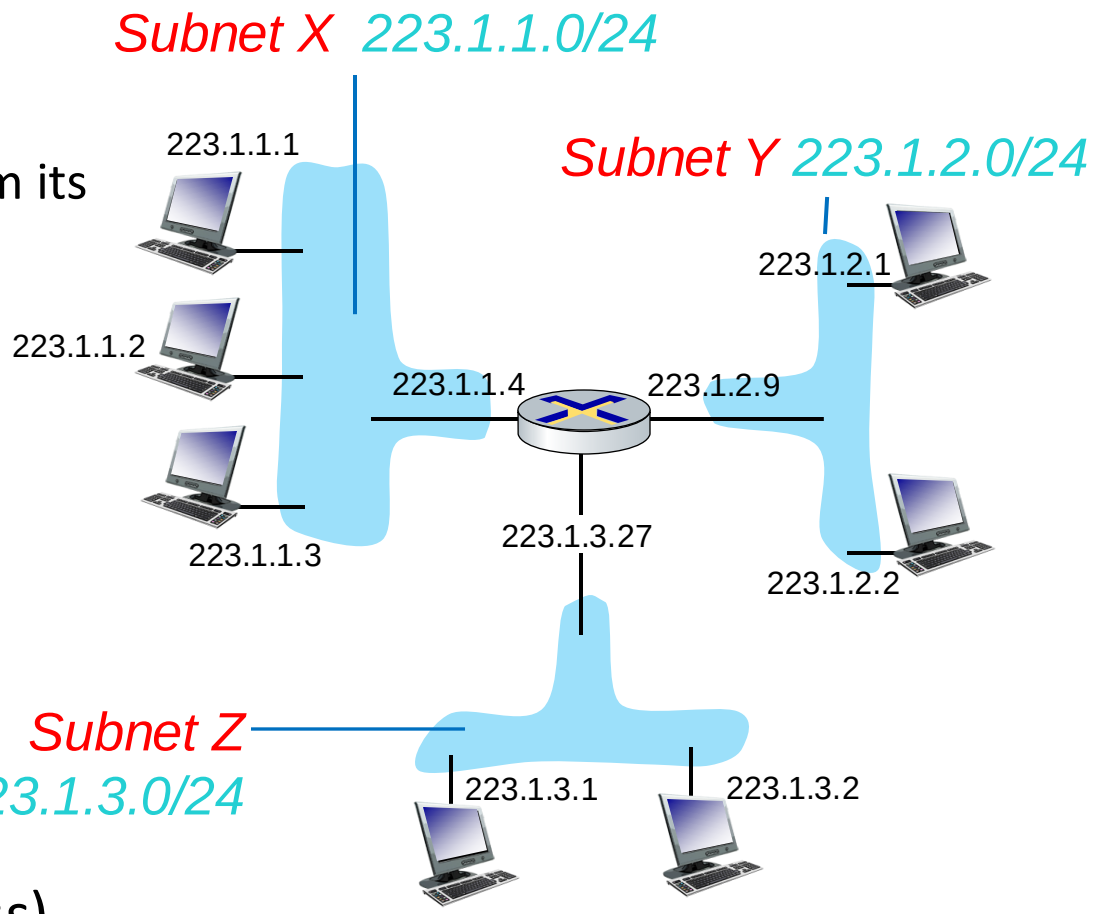


Subnets

Recipe

- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks

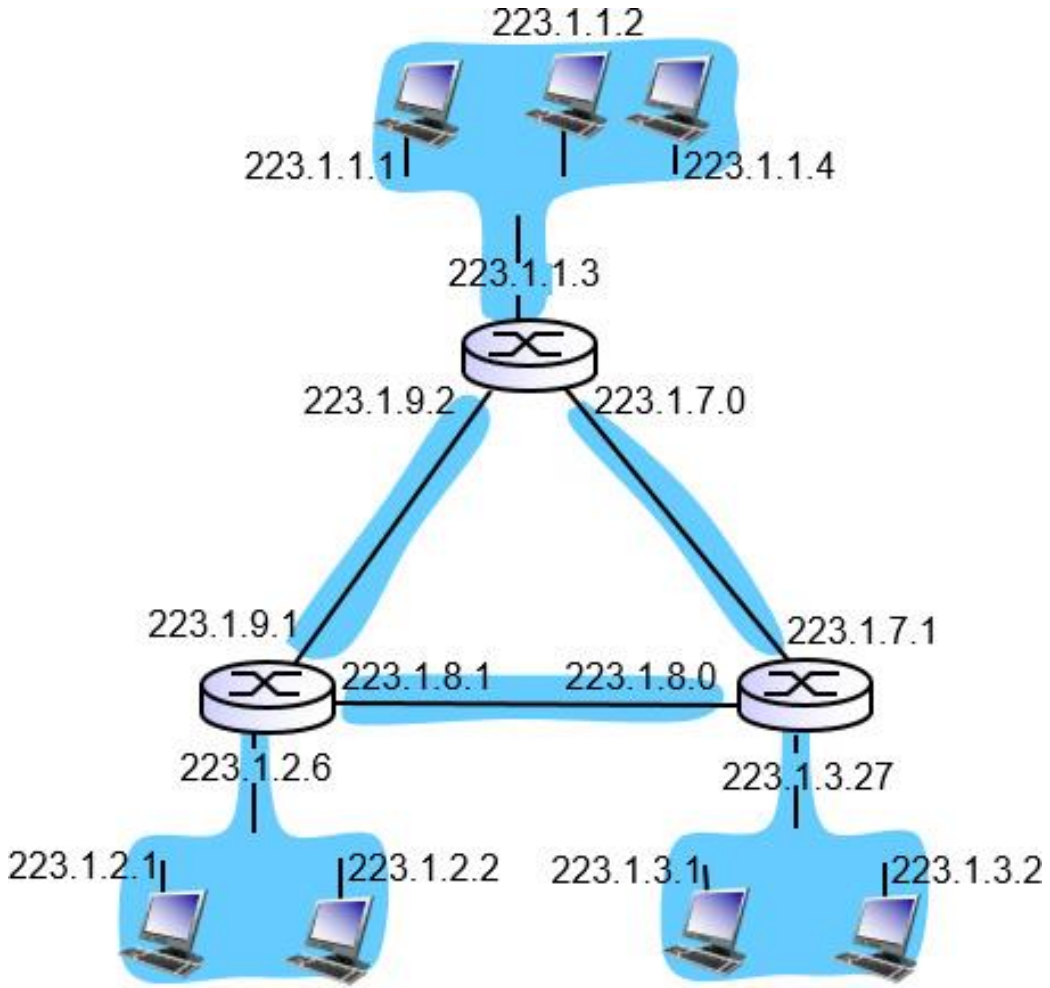
- Each isolated network is called a **subnet**



Subnet Mask: /24
(high-order 24 bits: Subnet part of IP address)

Subnets

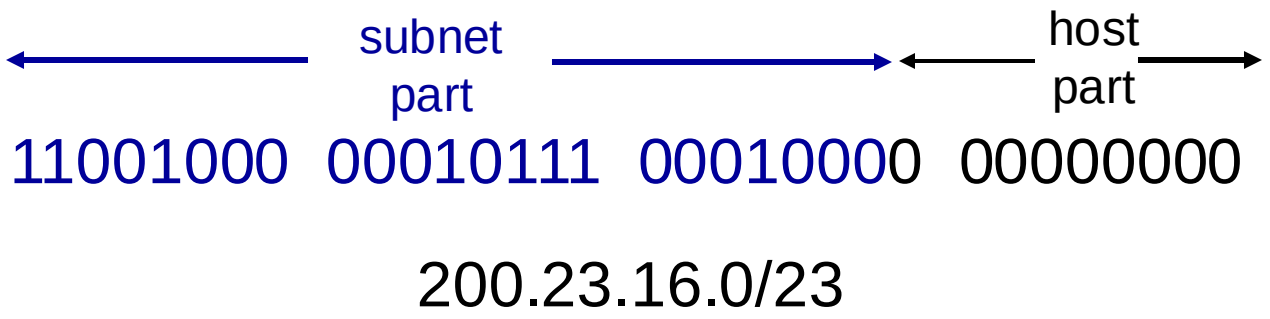
- How many?



IP Addressing: CIDR

CIDR: Classless **I**nter **D**omain **R**outing

- Subnet portion of address of arbitrary length
- Address format **a.b.c.d/x** where **x** is number of bits in subnet portion of address



IP Addresses: How to Get One?

Q: How does a **host** get IP address?

- Hard-coded by system admin in a file
  - Windows: `control-panel->network->configuration->tcp/ip->properties`
  - UNIX: `/etc/rc.config`

- **DHCP: D**ynamic **H**ost **C**onfiguration **P**rotocol
  Dynamically get address from a server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

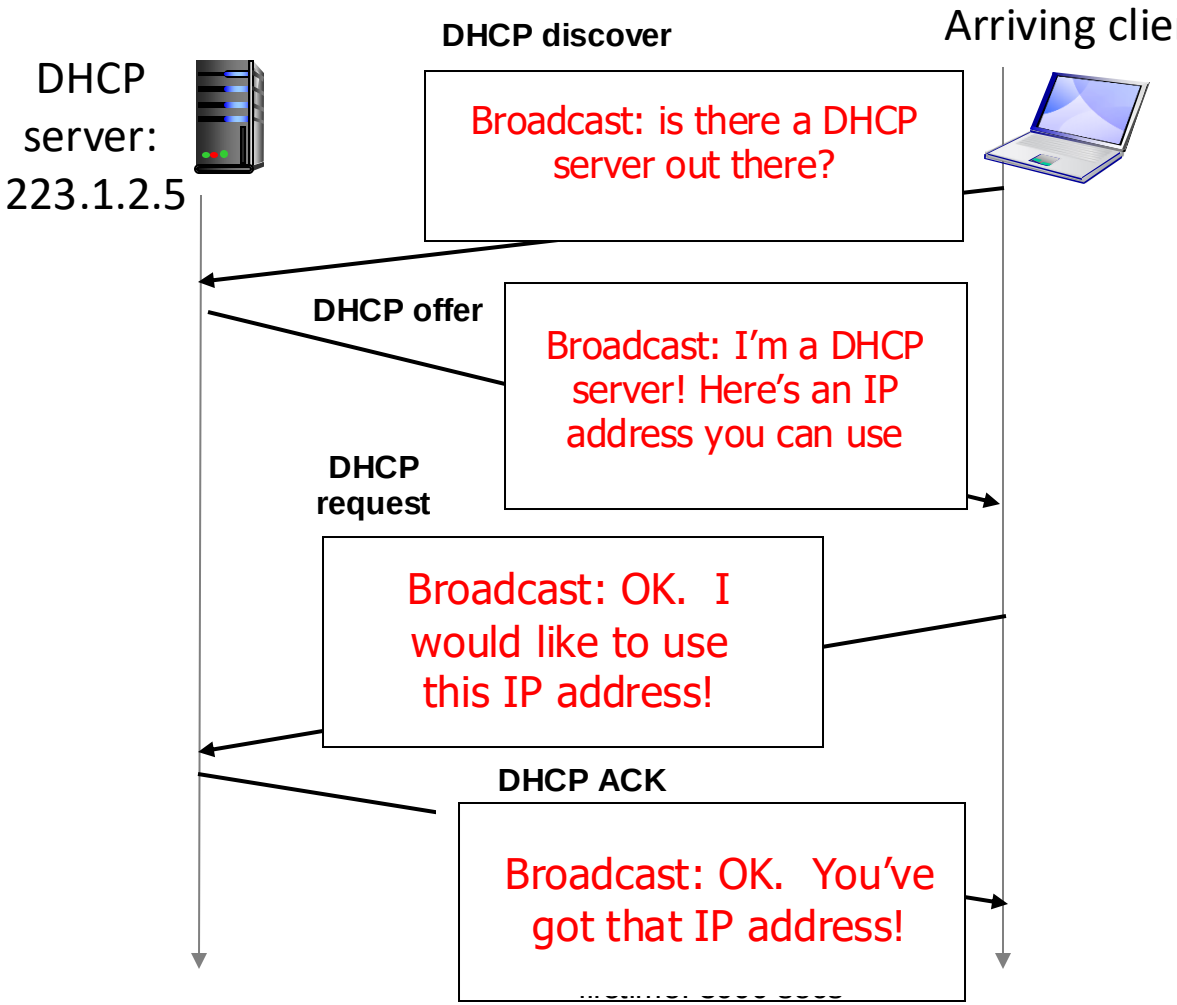
Goal: Allow host to **dynamically** obtain its IP address from network server when it joins network

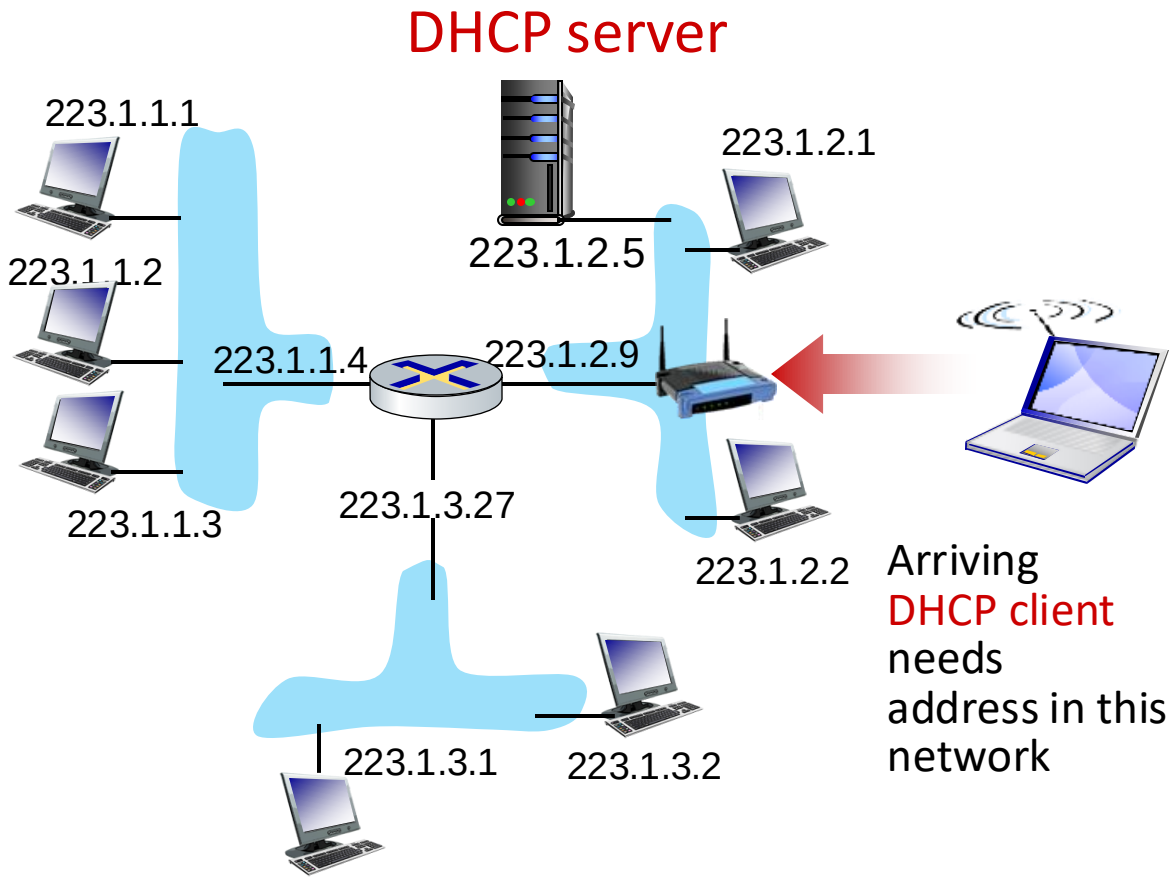
- Can renew its lease on address in use
- Allows reuse of addresses (only hold address while connected/on)
- Support for mobile users who want to join network (more shortly)

DHCP Overview

- Host broadcasts **"DHCP discover"** message [optional]
- DHCP server responds with **"DHCP offer"** message [optional]
- Host requests IP address: **"DHCP request"** message
- DHCP server sends address: **"DHCP ack"** message

DHCP Client-Server Scenario





DHCP: More Than IP Addresses

DHCP can return more than just allocated IP address on subnet:

- Address of first-hop router for client
- Name and IP address of DNS server
- Network mask (indicating network versus host portion of address)

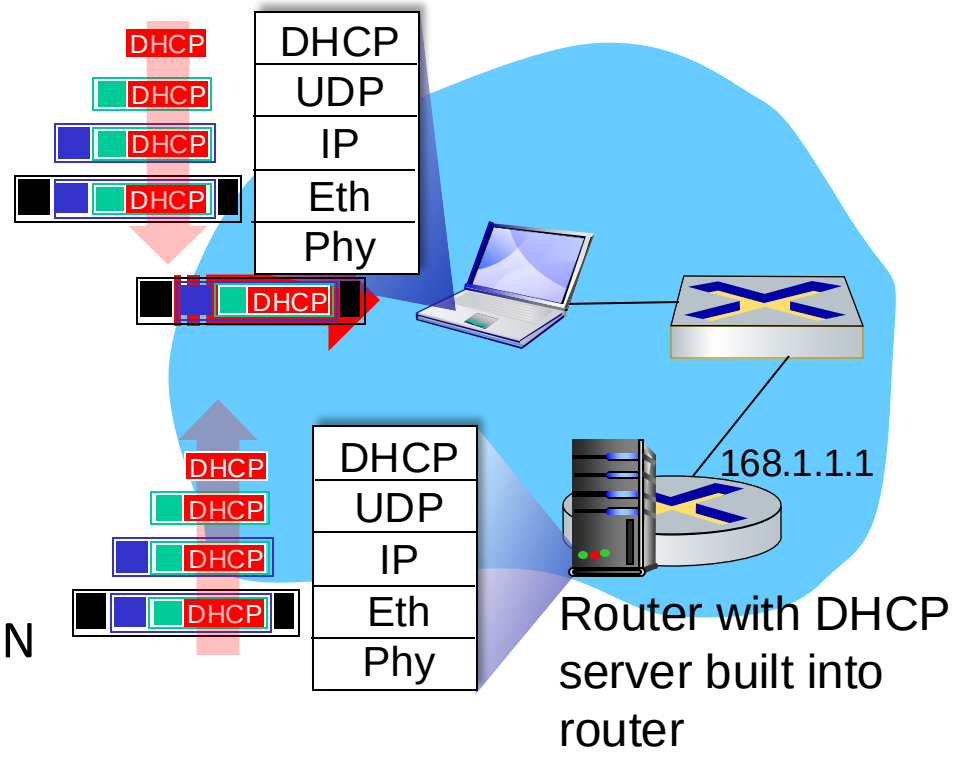
DHCP: Example

- Connecting laptop needs
  - Its IP address
  - Address of first-hop router
  - Address of DNS server

- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet

- Ethernet frame broadcast (destination: FFFFFFFFFFFF) on LAN received at router running DHCP server

- Ethernet demuxed to IP, demuxed to UDP, demuxed to DHCP



DHCP: Example

- DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- Encapsulation of DHCP server, frame forwarded to client, demultiplexing up to DHCP at client
- Client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

DHCP: Wireshark Output (Home LAN)

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server
    ……

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

IP Addresses: How to Get One?

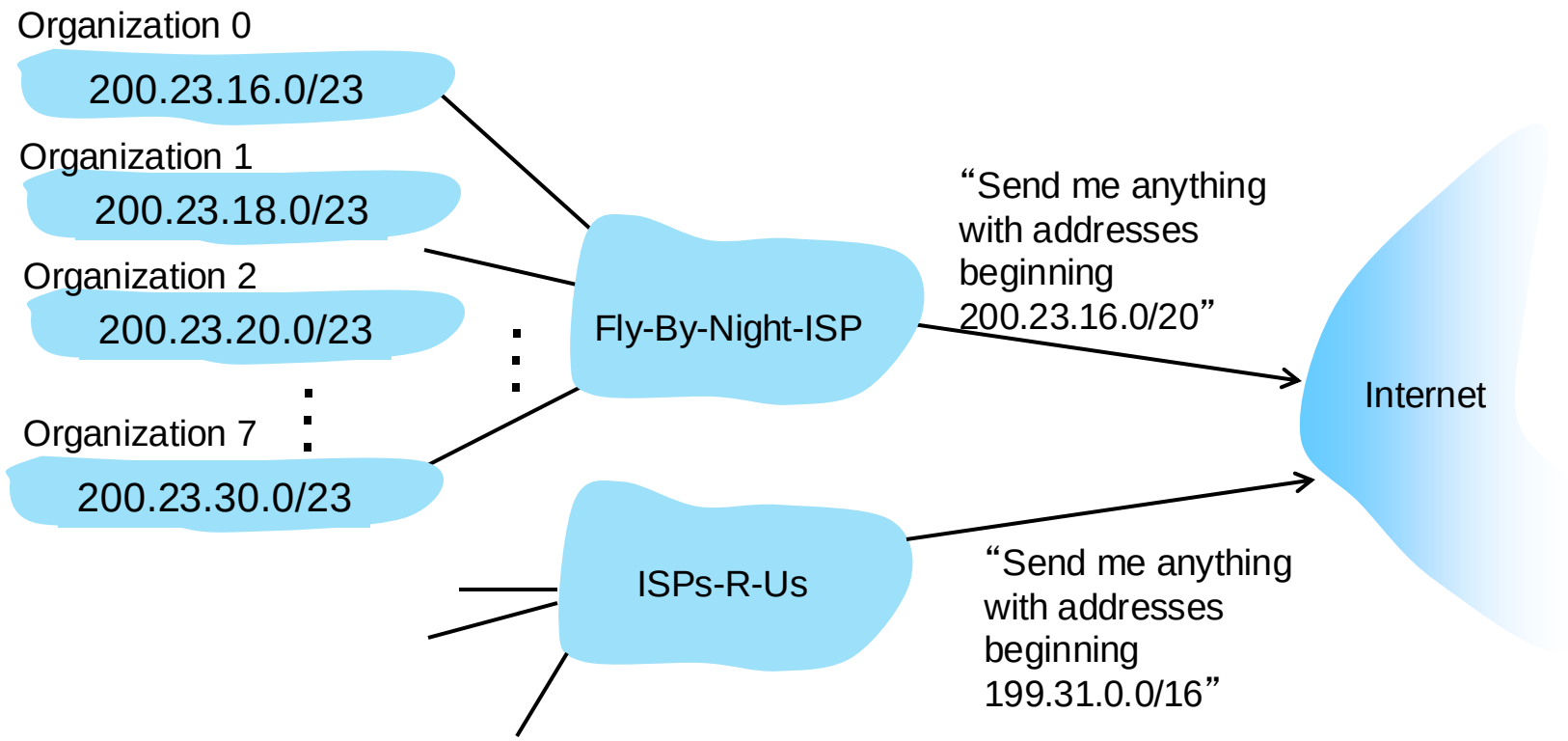
Q: How does **network** get subnet part of IP address?

A: Gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

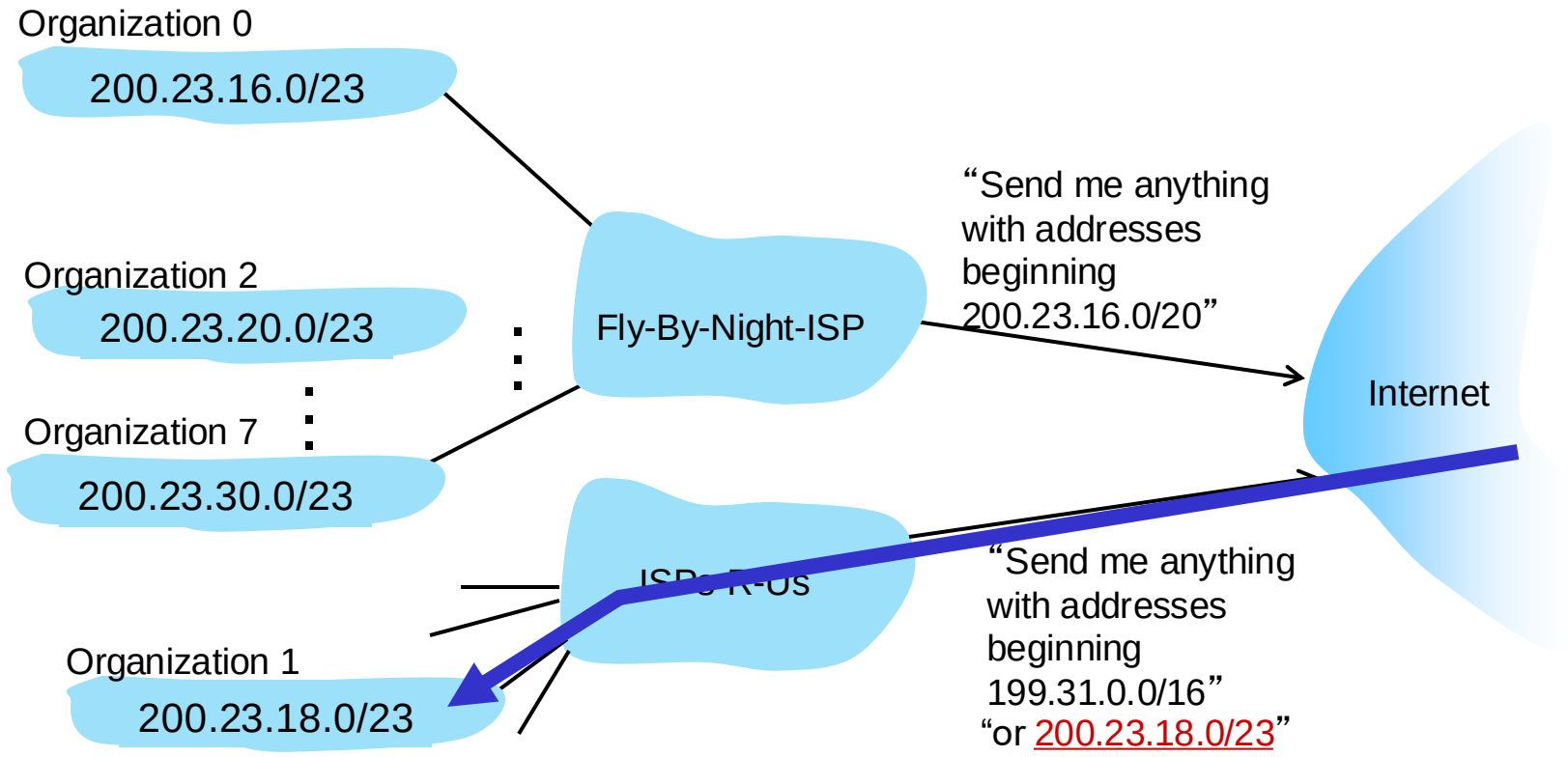
Hierarchical Addressing: Route Aggregation

- Hierarchical addressing allows efficient advertisement of routing information.



Hierarchical Addressing: More Specific Routes

- ISPs-R-Us has a more specific route to Organization 1



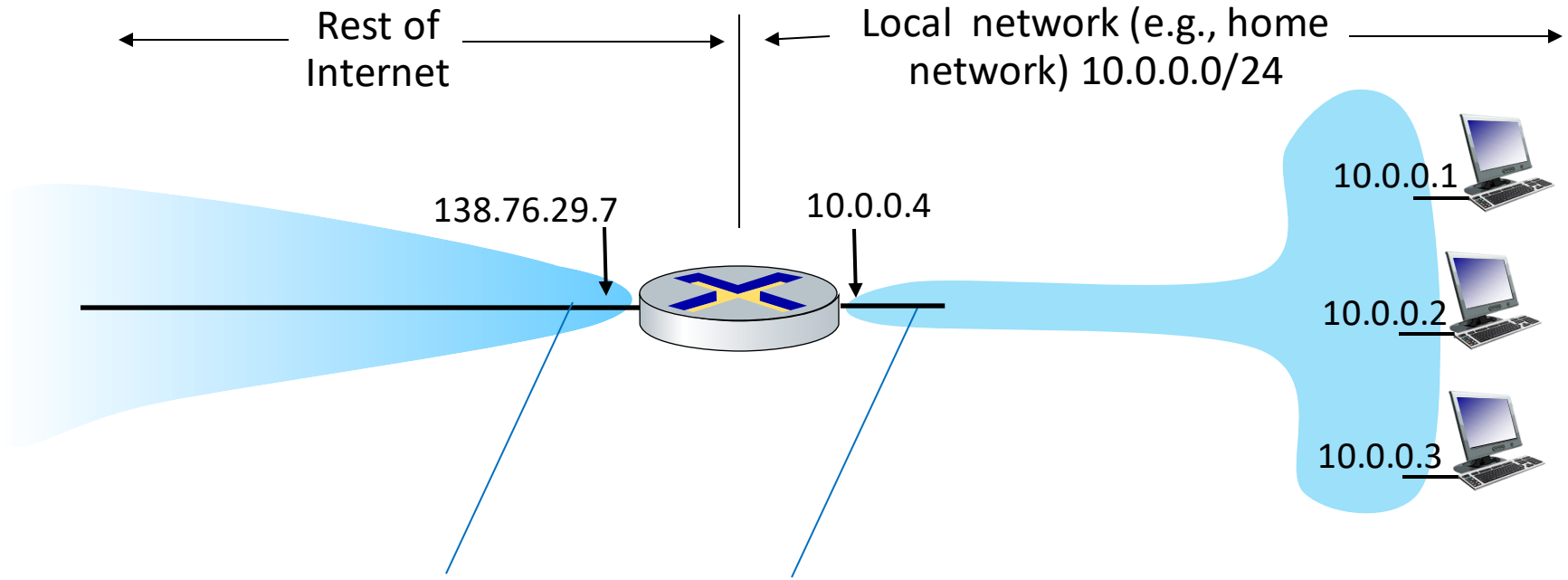
IP Addressing: The Last Word

Q: How does an ISP get block of addresses?

A: ICANN: Internet **C**orporation for **A**ssigned **N**ames and **N**umbers
http://www.icann.org/

- Allocates addresses
- Manages DNS
- Assigns domain names, resolves disputes

NAT: Network Address Translation



All datagrams *leaving* local network have *same* source NAT IP address: 138.76.29.7, but *different* source port numbers

Datagrams with source or destination in this network have 10.0.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation

Motivation

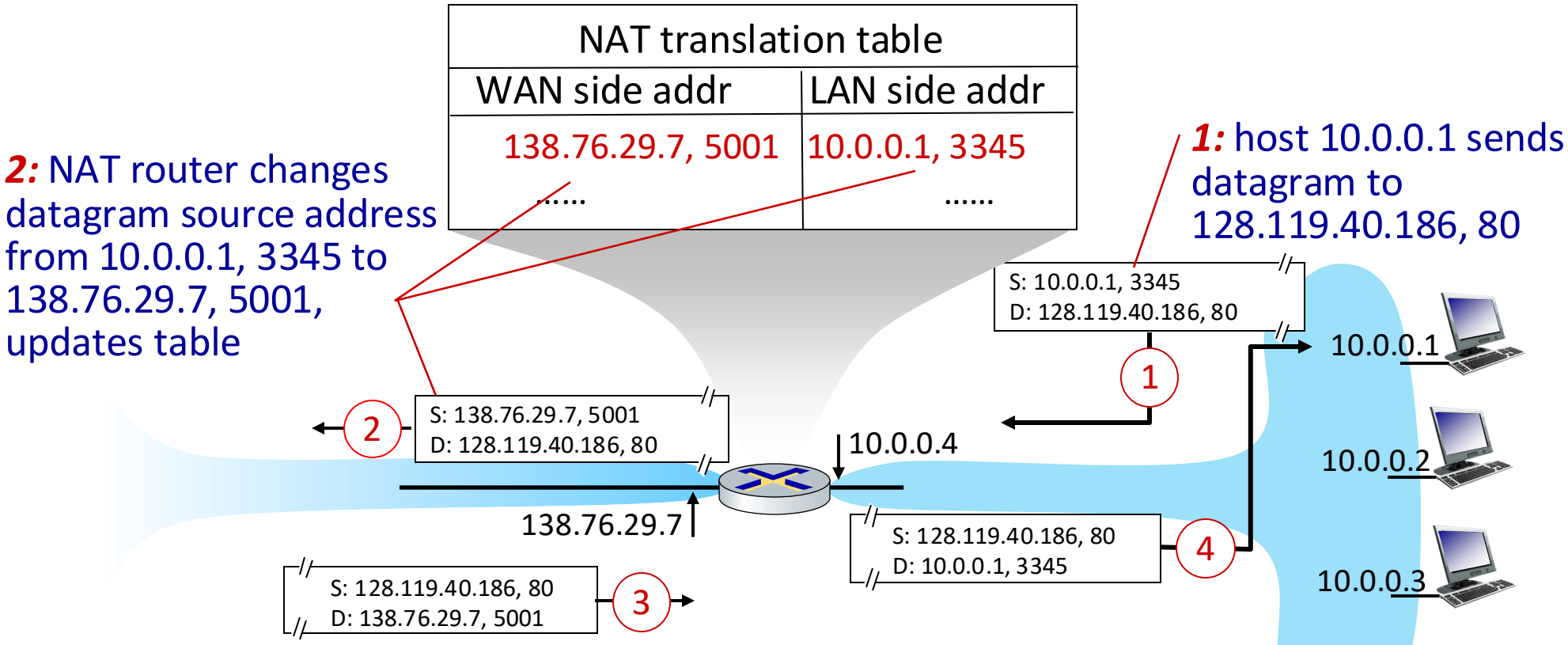
- Local network uses just one IP address as far as outside world is concerned
  - Range of addresses not needed from ISP: just one IP address for all devices
  - Can change addresses of devices in local network without notifying outside world
  - Can change ISP without changing addresses of devices in local network
  - Devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: Network Address Translation

Implementation: NAT router must:

- **Outgoing datagrams:** Replace (source IP address, port number) of every outgoing datagram to (NAT IP address, new port number) . . . remote clients/servers will respond using (NAT IP address, new port number) as destination address
- Remember **(in NAT translation table)** every (source IP address, port number) to (NAT IP address, new port number) translation pair
- **Incoming datagrams:** Replace (NAT IP address, new port number) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



NAT: Network Address Translation

- 16-bit port-number field
  - 60,000 simultaneous connections with a single address!

- NAT is controversial
  - Routers should only process up to layer 3
  - Address shortage should be solved by IPv6
  - Violates end-to-end argument
    - NAT possibility must be taken into account by app designers (e.g., P2P applications)
  - NAT traversal: What if client wants to connect to server behind NAT?

IPv6

- **Initial motivation:** 32-bit address space soon to be completely allocated.

- **Additional motivation**
  - Header format helps speed processing/forwarding
  - Header changes to facilitate QoS

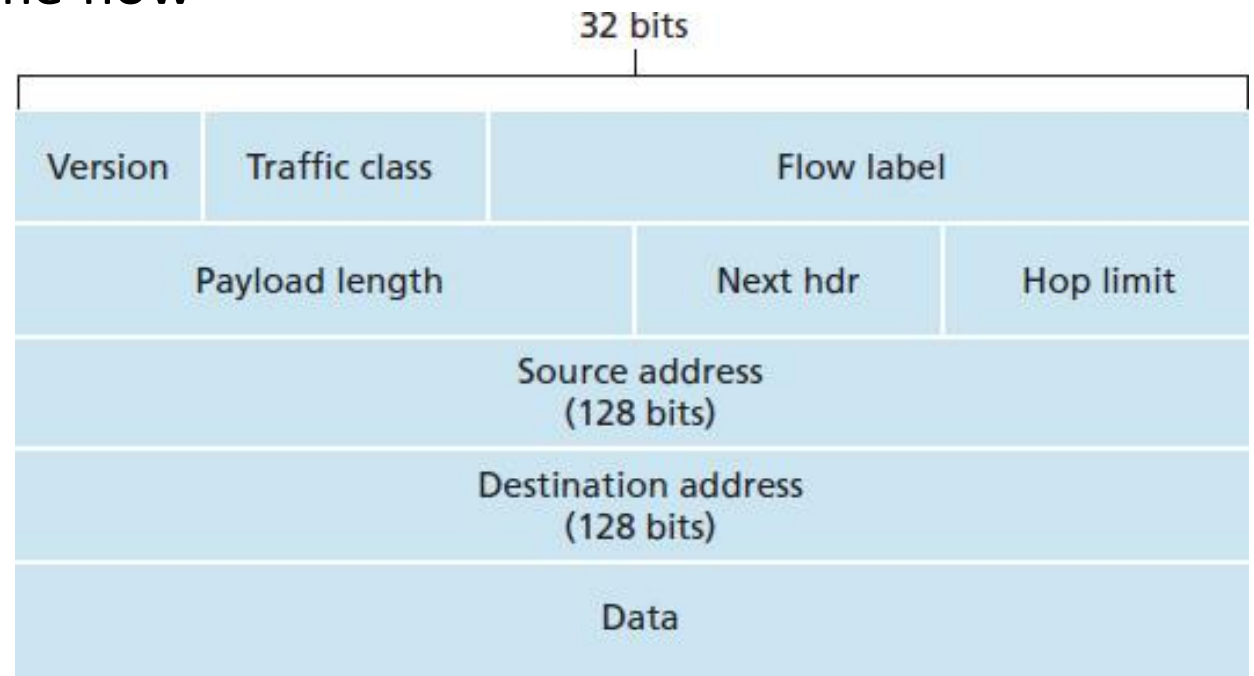
IPv6 datagram format
  - Fixed-length 40 byte header
  - No fragmentation allowed

IPv6 Datagram Format

Priority: Identify priority among datagrams in flow

Flow Label: Identify datagrams in same flow

Next header: Identify upper layer protocol for data



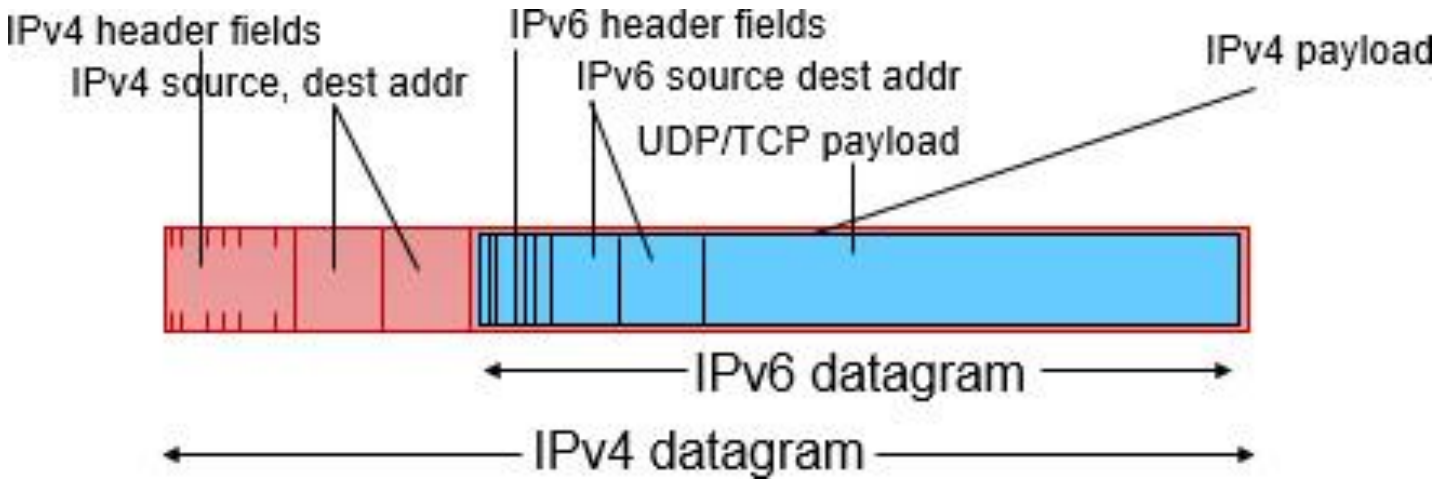
Other Changes from IPv4

- **Checksum:** Removed entirely to reduce processing time at each hop
- **Options:** Allowed, but outside of header, indicated by **Next Header** field
- **ICMPv6:** New version of ICMP
  - Additional message types, e.g. Packet Too Big
  - Multicast group management functions

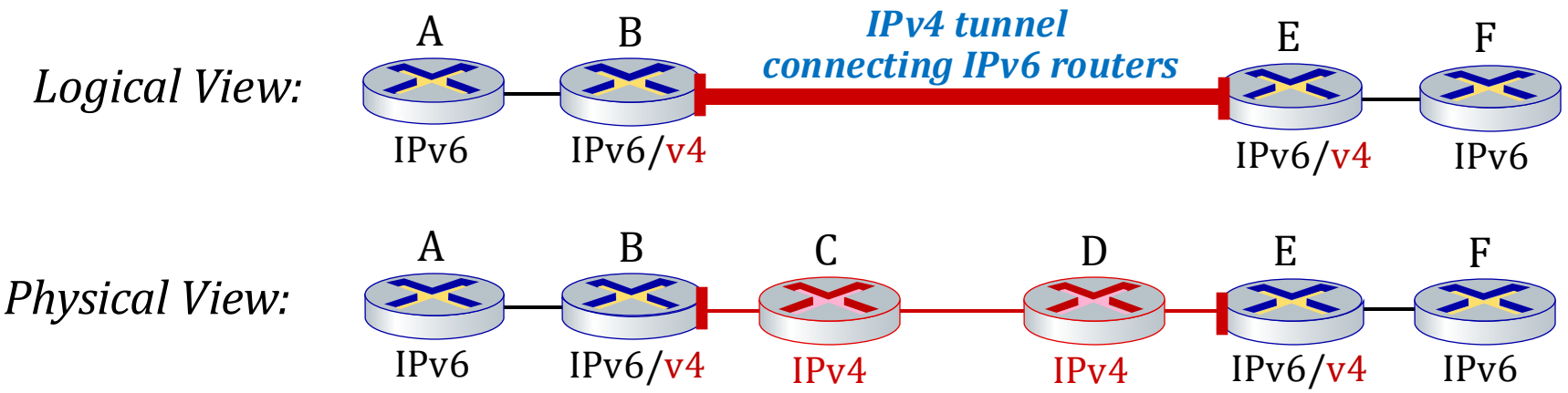
Transition from IPv4 to IPv6

- Not all routers can be upgraded simultaneously
  - How will network operate with mixed IPv4 and IPv6 routers?

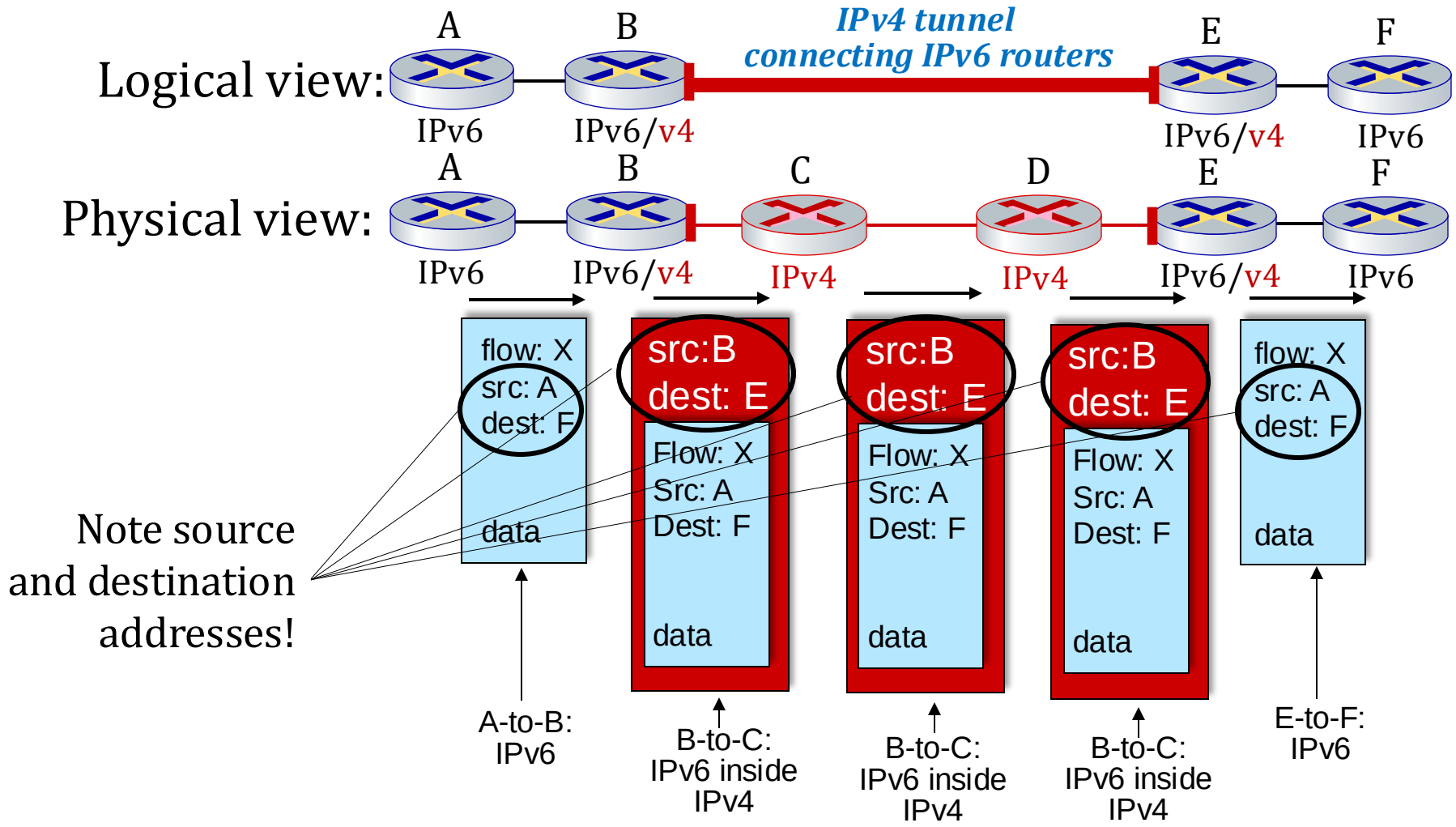
- **Tunneling:** IPv6 datagram carried as **payload** in IPv4 datagram among IPv4 routers



Tunneling


Logical View:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
Physical View:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6

Tunneling


Logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
Physical view:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6
Note source
and destination
addresses!
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

IPv6: Adoption

- Google: 8% (2016) of clients access services via IPv6
  The number is at 32% (2020)

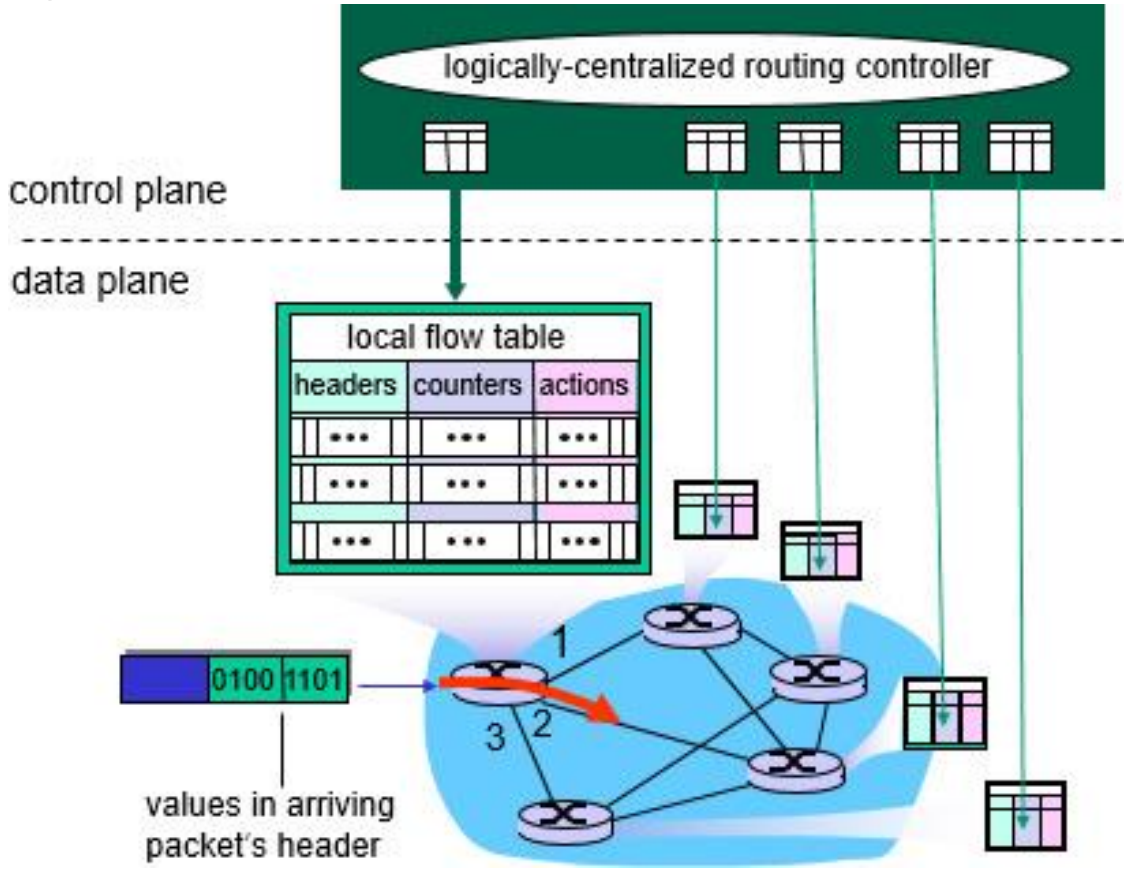
- NIST (National Institute of Standards and Technology): 1/3 of all US government domains are IPv6 capable

- **Long time for deployment and use**
  - 20 years and counting!
  - Think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
  - **Why?**

Data Plane: SDN

Generalized Forwarding & SDN

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



OpenFlow Data Plane Abstraction

- **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:** Number of bytes & number of packets

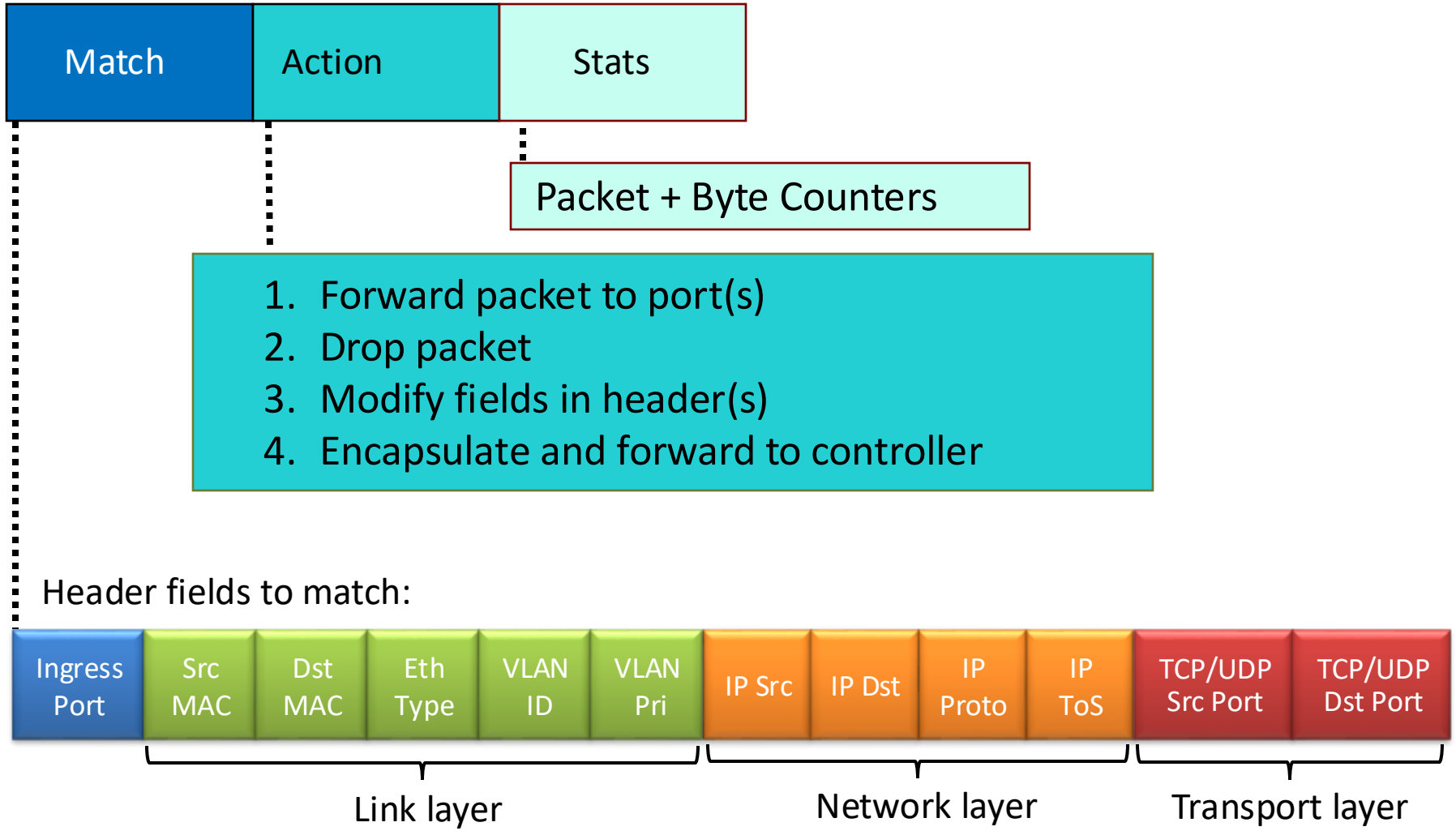
- Flow table in a router (computed and distributed by controller) define router's **match+action** rules

OpenFlow Data Plane Abstraction

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

* : wildcard

OpenFlow: Flow Table Entries



Example

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	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	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port number)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1

Example

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port3

Layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

Open Flow Abstraction

- **Match+action:** Unifies different kinds of devices

- Router
  - **Match:** Longest destination IP prefix
  - **Action:** Forward out a link

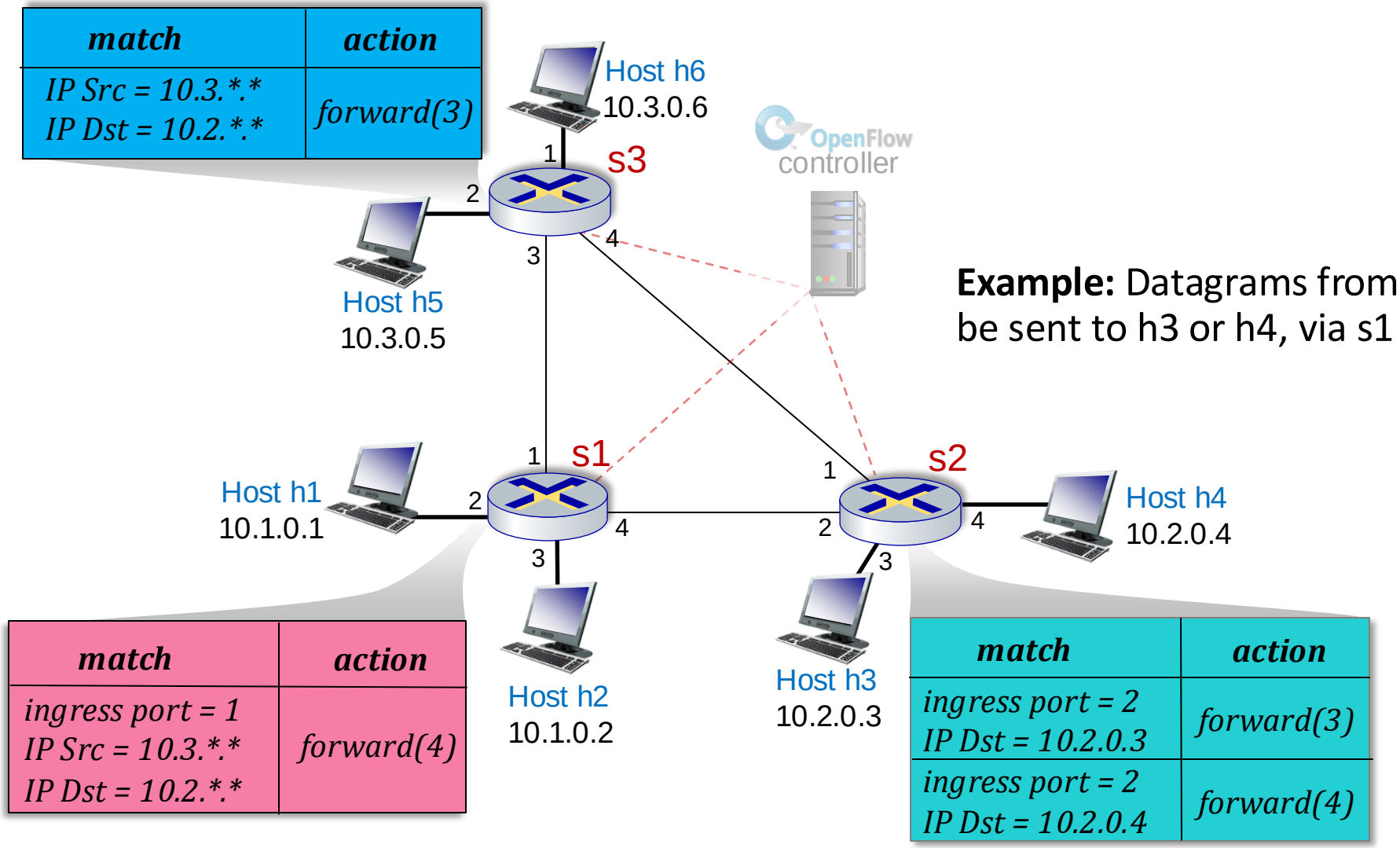
- Switch
  - **Match:** Destination MAC address
  - **Action:** Forward or flood

Open Flow Abstraction

- 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

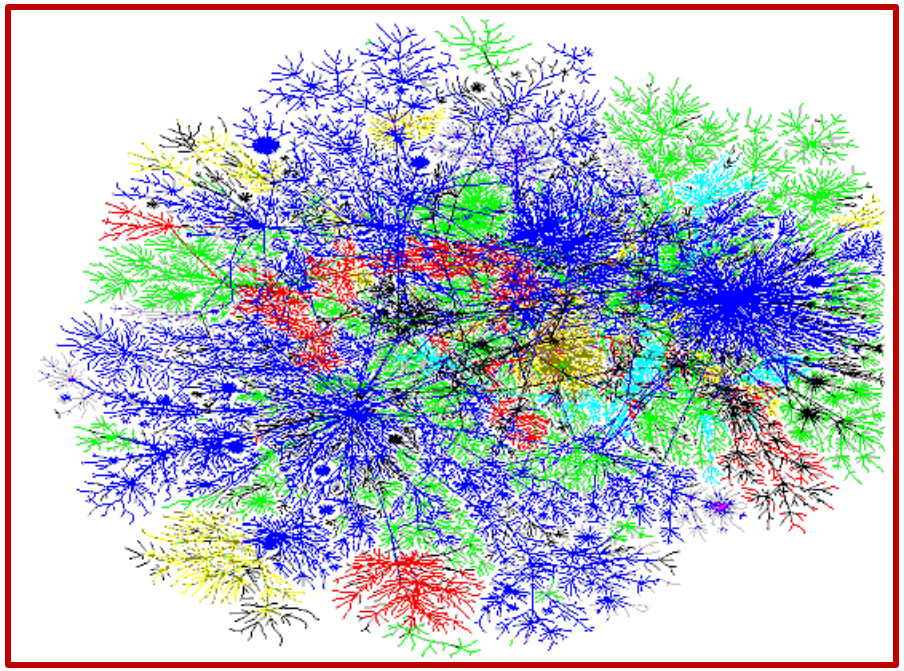
Control Plane: Routing

Network Layer

- Two main functionalities
  - Routing
    - Determine the route taken by packets from source to destination
    - Happens in Control Plane
    - Network-wide logic
      - Per-router control
      - Centralized control
  - Forwarding
    - Moving packet from input to the appropriate output at each device (router) along the path
    - Happens in Data Plane
    - Local function at each device

Control Plane

"The process of determining systematically how to forward messages toward the destination node based on its address is called **Routing**."



Definition from Computer Networks: A System Approach (Larry L. Patterson, and Bruce S. Davie)

Control Plane: Routing

- **Router:** “A device that is connected to two or more networks is called a router”

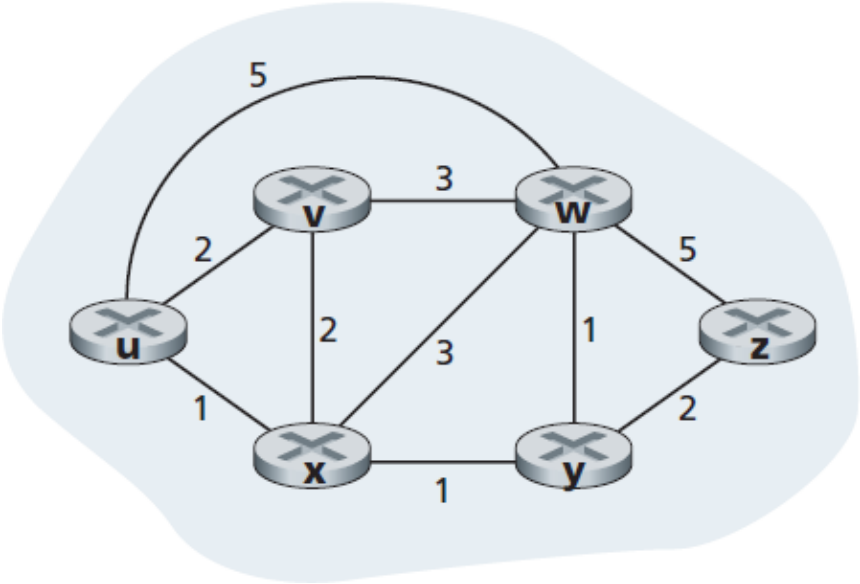
- **Routing**
  - **Traditional Routing (Per-Router Control)**
    - Static
    - Dynamic
      - **Routing Algorithms**
        - Distance Vector
        - Link State
      - Routing Protocols
      - Routing Policies
  - **Software Defined Networking (Logically Centralized Control)**

Routing Protocols

Routing protocol goal: Determine good paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- **Path**: Sequence of routers packets will traverse in going from given initial source host to given final destination host
- **Good**: Least cost, fastest, least congested
- **Routing**: A top-10 networking challenge!

Graph Abstraction of the Networks



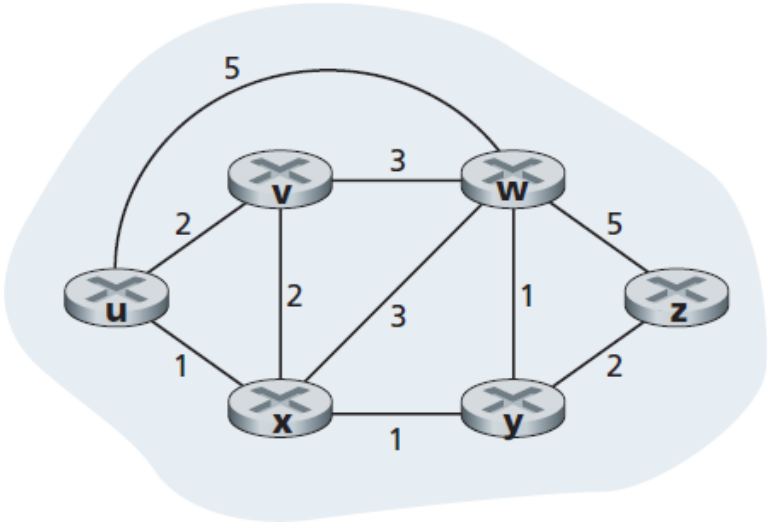
Graph: G = (N, E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Side Note: Graph abstraction is useful in other network contexts
e.g., P2P, where **N** is set of peers and **E** is set of TCP connections

Graph Abstraction: Costs



$c(x,x') = \text{cost of link } (x,x') \text{ e.g., } c(w,z) = 5$

- Cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

$$\text{cost of path } (x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$$

Key Question: What is the least-cost path between u and z?
Routing Algorithm: Algorithm that finds that least cost path

Routing Algorithms: Distance Vector

- Each router only knows directly connected neighbors
- Algorithm: **Bellman-Ford**

$D_x(y)$: **Cost of least cost path from** x **to** y

$Cost(x,y)$: **Cost of a direct link from** x **to** y

Least cost path from ***x*** to ***y*** is the minimum of cost path to ***y*** through all direct neighbors of ***x***

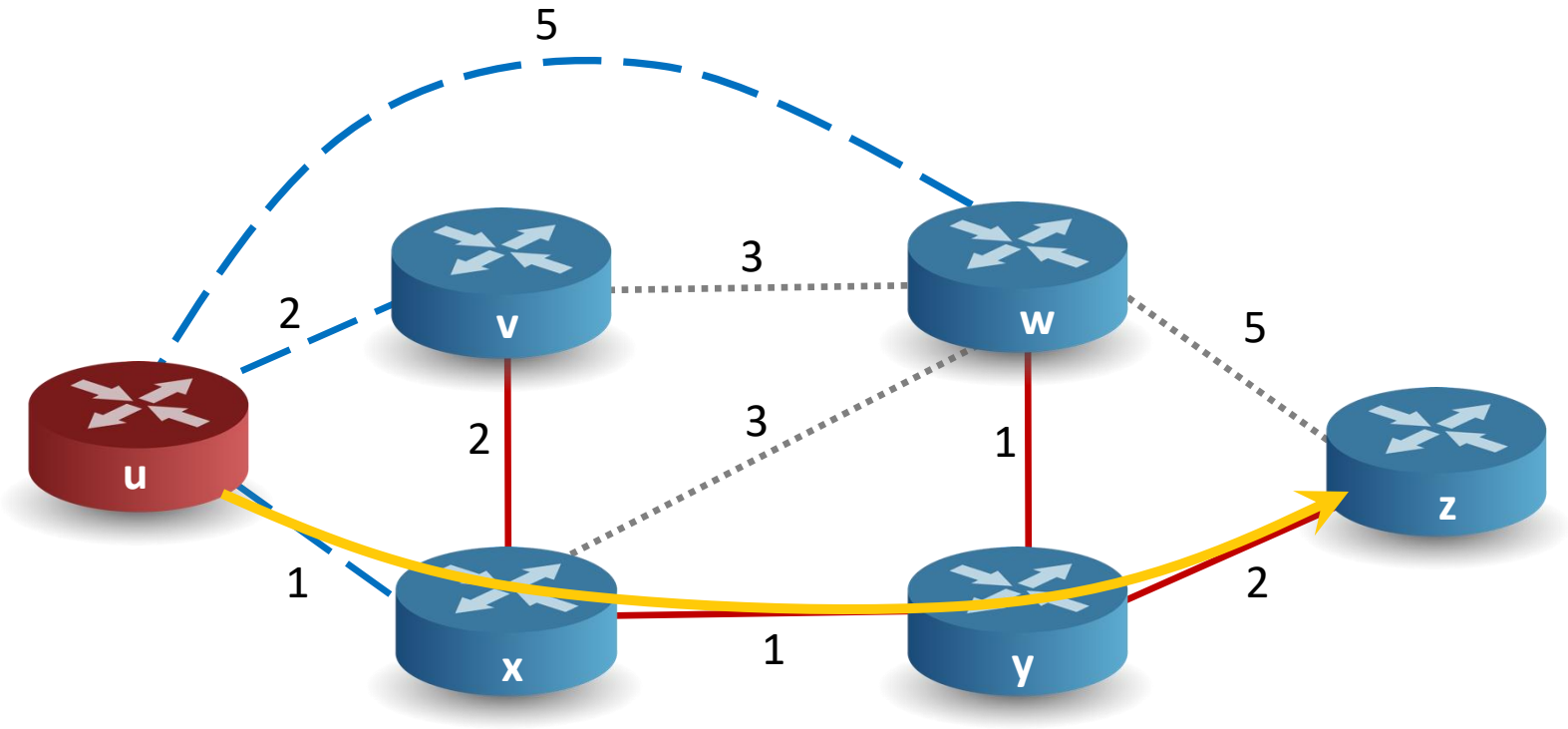
$$D_x(y) = min_v \{ Cost(x,v) + D_v(y) \}$$

Routing Algorithms: Distance Vector

$Cost(u,x)=1$
$Cost(u,v)=2$
$Cost(u,w)=5$

$D_x(z) = 3$
$D_v(z) = 5$
$D_w(z) = 3$

$D_u(z) = min\ \{\mathbf{Cost(u,x)+ D_x(z)},$
$Cost(u,v)+ D_v(z),$
$Cost(u,w)+ D_w(z)\}$
$= min\ \{\mathbf{1+3},$
$2+5,$
$5+3\}$
$= 4$



The ***next hop*** to send packets to **z** is *x* with cost **4**.
Does not keep the knowledge of the path after the next hop.

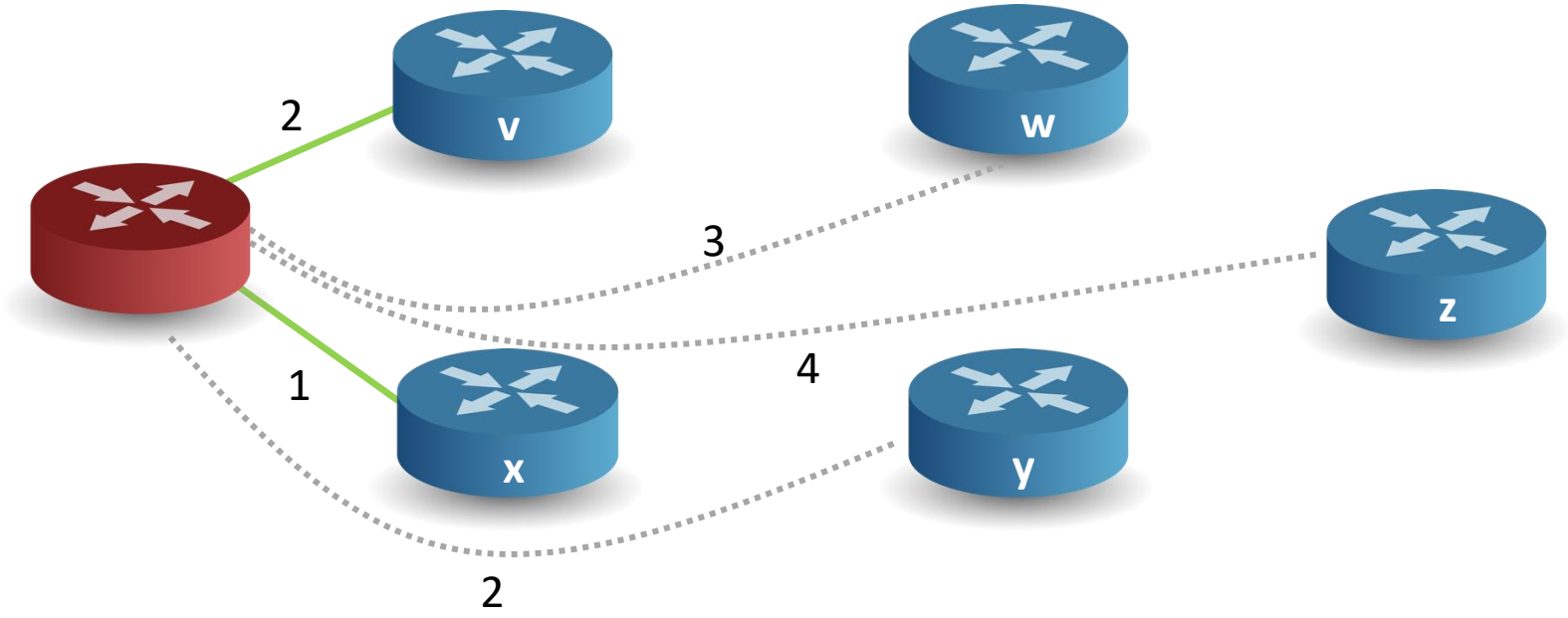
Routing Algorithms: Distance Vector

Bellman-Ford

Resulting forwarding table at u:

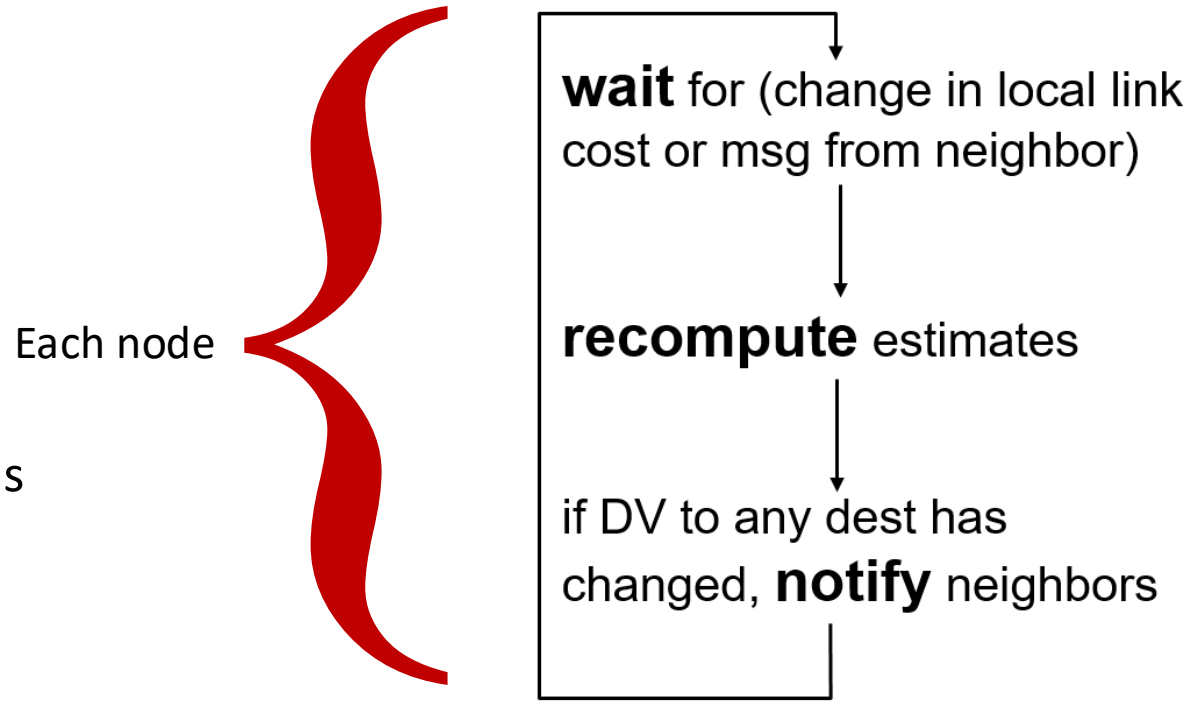
Destination	Link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Resulting path information at u:



Routing Algorithms: Distance Vector Analysis

- Iterative
  - Each local iteration cause by
    - Local link change
    - Update message from a neighbor
- Asynchronous
  - Iterations not in sync among the nodes
- Distributed
  - Each node notifies neighbors only if changes



Routing Algorithms: Distance Vector Analysis

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

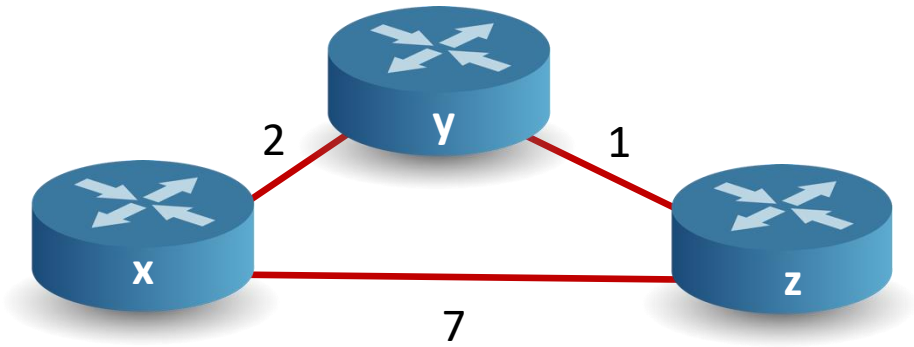
node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

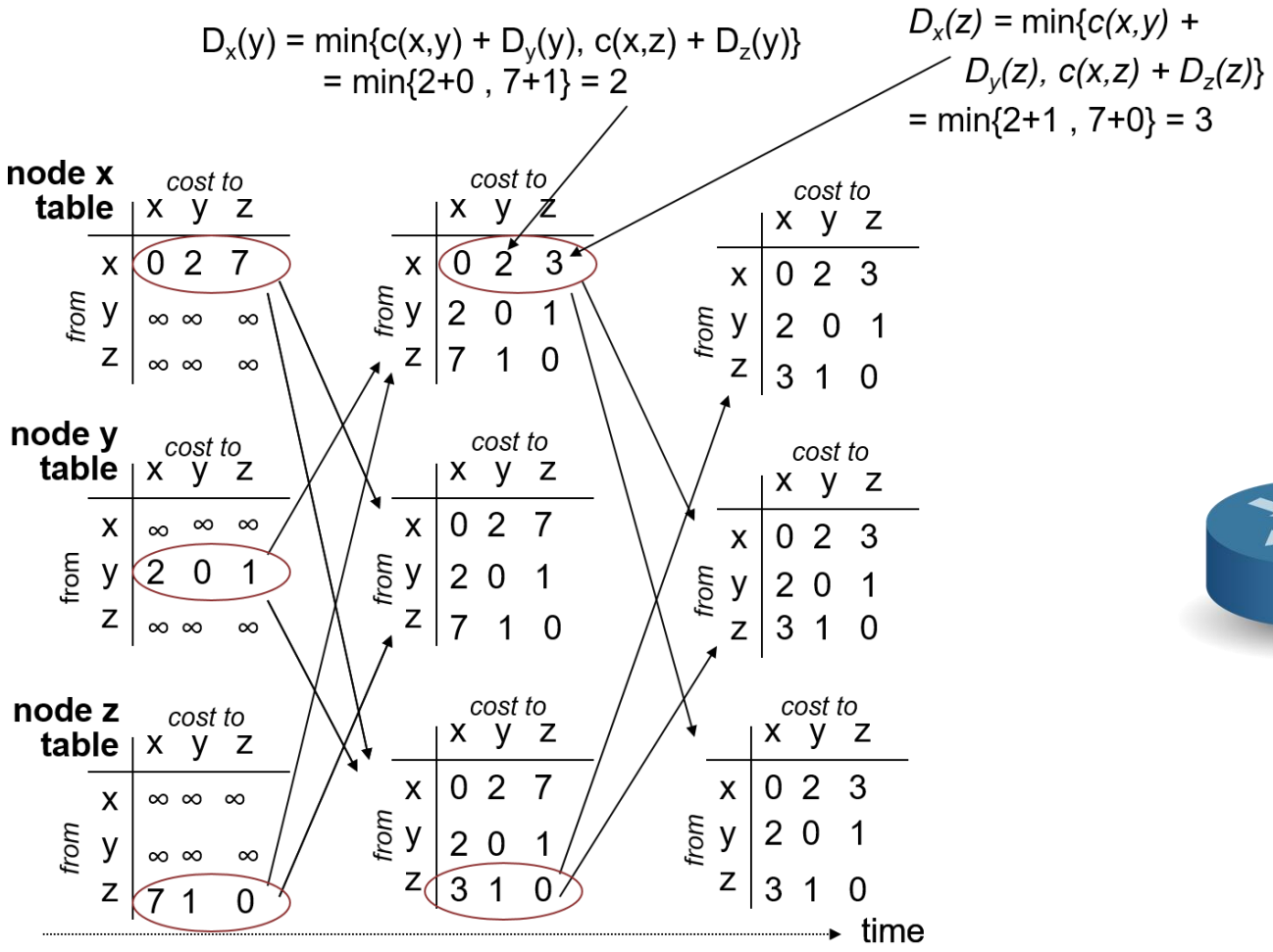
node z table

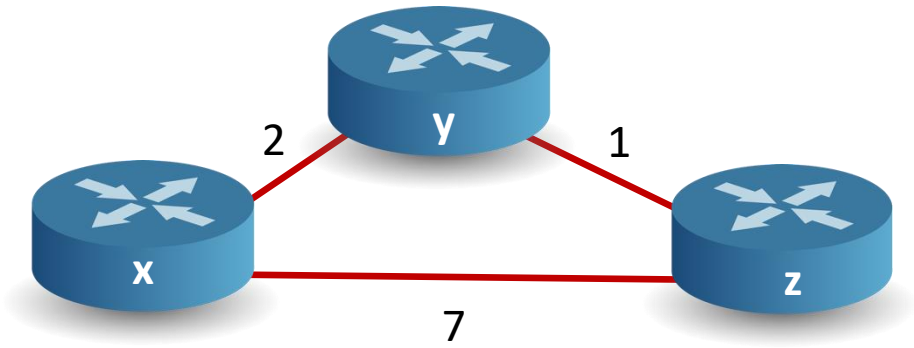
from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



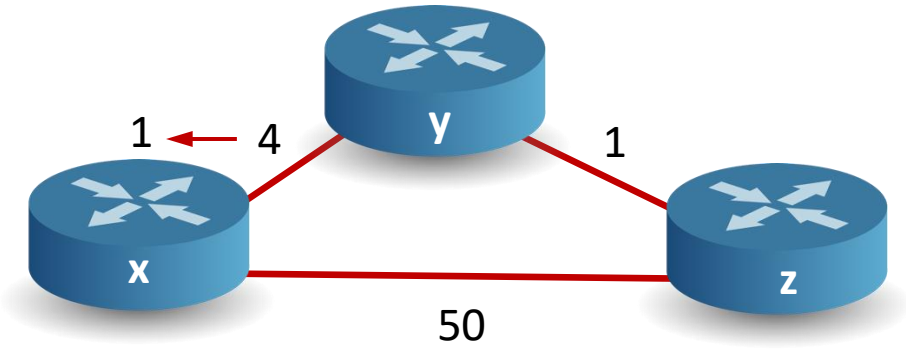
Routing Algorithms: Distance Vector Analysis





Routing Algorithms: Distance Vector Analysis

- Link cost changes and link failures
  - Good news traverses fast
  - Bad news traverses slow
    - Count to infinity
    - Poisoned Reverse
      - If Z routes through Y to get to X
        - Z tells Y that Z's distance to X is infinite
      - Will this completely solve count to infinity problem? ***No. Will not solve the loops involving three or more nodes***



Routing Algorithms: Link State

- All routers have complete topology of the network & link cost info
- Algorithm: **Dijkstra's shortest path**

Initialization

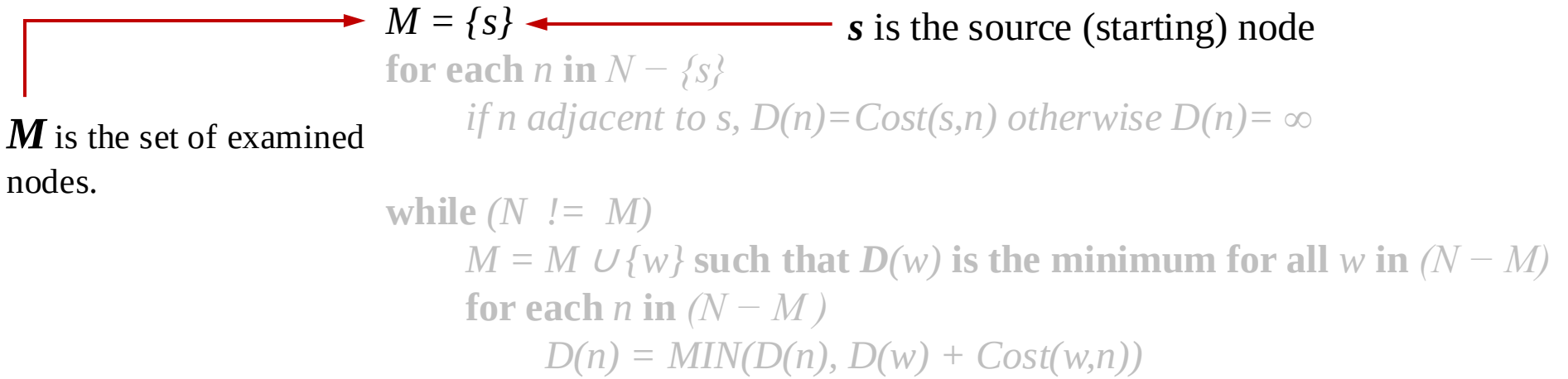
$M = \{s\}$
for each n **in** $N - \{s\}$
    if n adjacent to s, $D(n) = Cost(s,n)$ *otherwise* $D(n) = \infty$

Computation Loop

while $(N \; != \; M)$
    $M = M \cup \{w\}$ **such that** $D(w)$ **is the minimum for all** w **in** $(N - M)$
    for each n **in** $(N - M)$
        $D(n) = MIN(D(n), D(w) + Cost(w,n))$

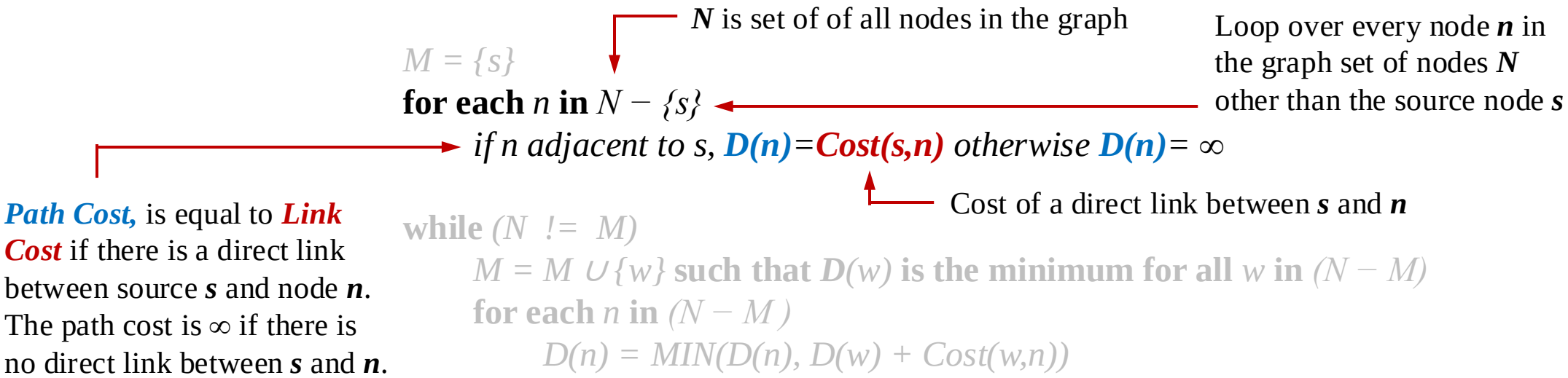
Routing Algorithms: Link State

- All routers have complete topology of the network & link cost info
- Algorithm: **Dijkstra's shortest path**



Routing Algorithms: Link State

- All routers have complete topology of the network & link cost info
- Algorithm: **Dijkstra's shortest path**



Routing Algorithms: Link State

- All routers have complete topology of the network & link cost info
- Algorithm: **Dijkstra's shortest path**

```
M = {s}
for each n in N − {s}
    if n adjacent to s, D(n)=Cost(s,n) otherwise D(n)= ∞

while (N != M)
    M = M ∪{w} such that D(w) is the minimum for all w in (N − M)
    for each n in (N − M )
        D(n) = MIN(D(n), D(w) + Cost(w,n))
```

While still there are nodes in the graph we have not examined

Routing Algorithms: Link State

- All routers have complete topology of the network & link cost info
- Algorithm: **Dijkstra's shortest path**

```
M = {s}
for each n in N − {s}
    if n adjacent to s, D(n)=Cost(s,n) otherwise D(n)= ∞

while (N != M)
    M = M ∪{w} such that D(w) is the minimum for all w in (N − M)
    for each n in (N − M )
        D(n) = MIN(D(n), D(w) + Cost(w,n))
```

For each node ***w***, add ***w*** with minimum cost from ***s*** among all remaining nodes to the set ***M***

Routing Algorithms: Link State

- All routers have complete topology of the network & link cost info
- Algorithm: **Dijkstra's shortest path**

```
M = {s}
for each n in N − {s}
    if n adjacent to s, D(n)=Cost(s,n) otherwise D(n)= ∞

while (N != M)
    M = M ∪{w} such that D(w) is the minimum for all w in (N − M)
    for each n in (N − M )
        D(n) = MIN(D(n), D(w) + Cost(w,n))
```

For each node ***n***, if the cost of reaching ***n*** from ***s*** (current source node) through ***w*** is less than currently stored cost, update the cost to the path cost **going through node *w***

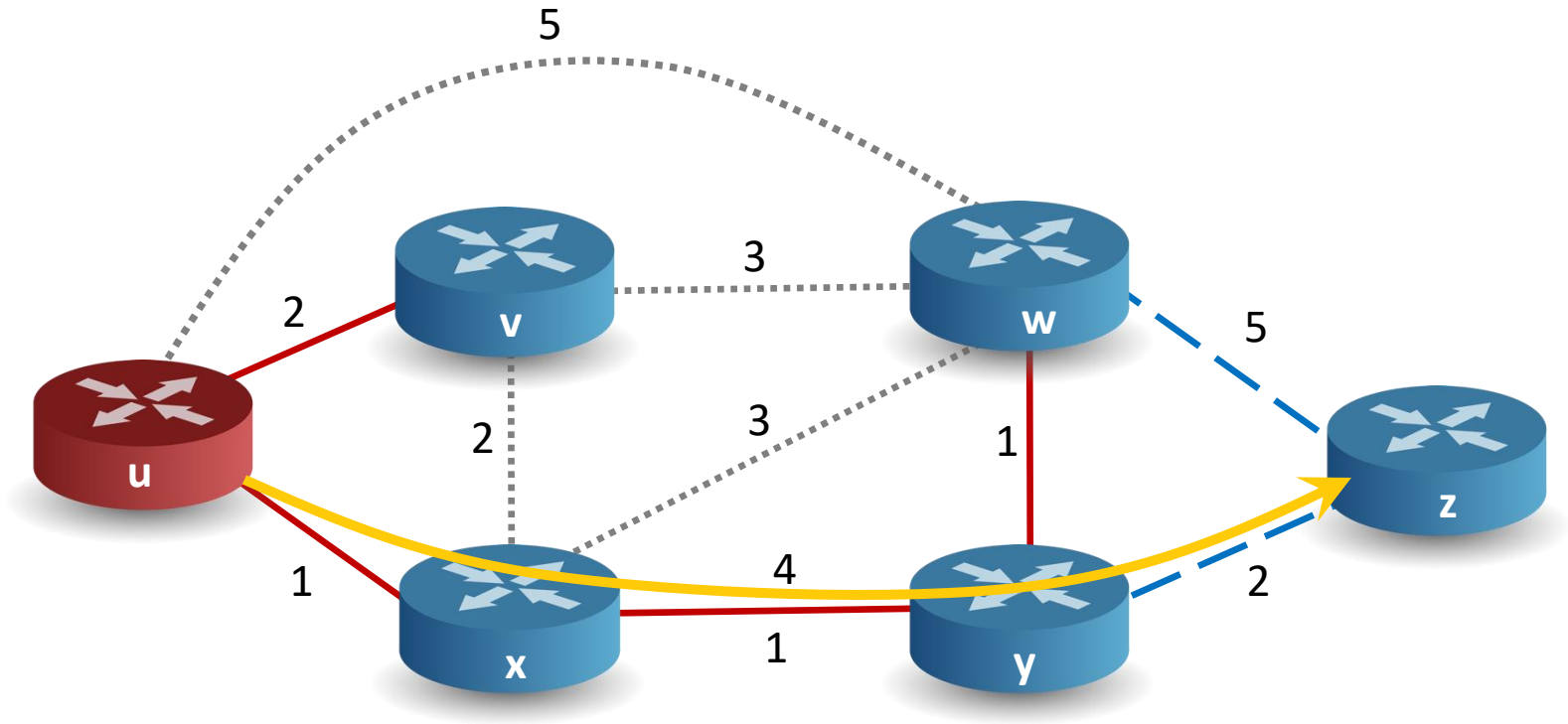
Routing Algorithms: Link State

Dijkstra's shortest path

Min cost calculation. Cost updates from ***every other node in the network***, not just the neighbors.

D(x) = 1 with path {u,x}
D(y) = 2 with path {u,x,y}
D(v) = 2 with path {u,v}
D(w) = 3 with path {u,x,y,w}

D(z) = min
{ D(x) + Cost(x,z),
D(y)+ Cost(y,z)
D(v) + Cost(v,z),
D(w)+ Cost(w,z),
}
= min { 1+ ∞,
2+ 2,
2 + ∞,
3+ 5 } = 4



The ***path*** from u to z goes through ***{u,x,y,z}*** with cost ***4***.

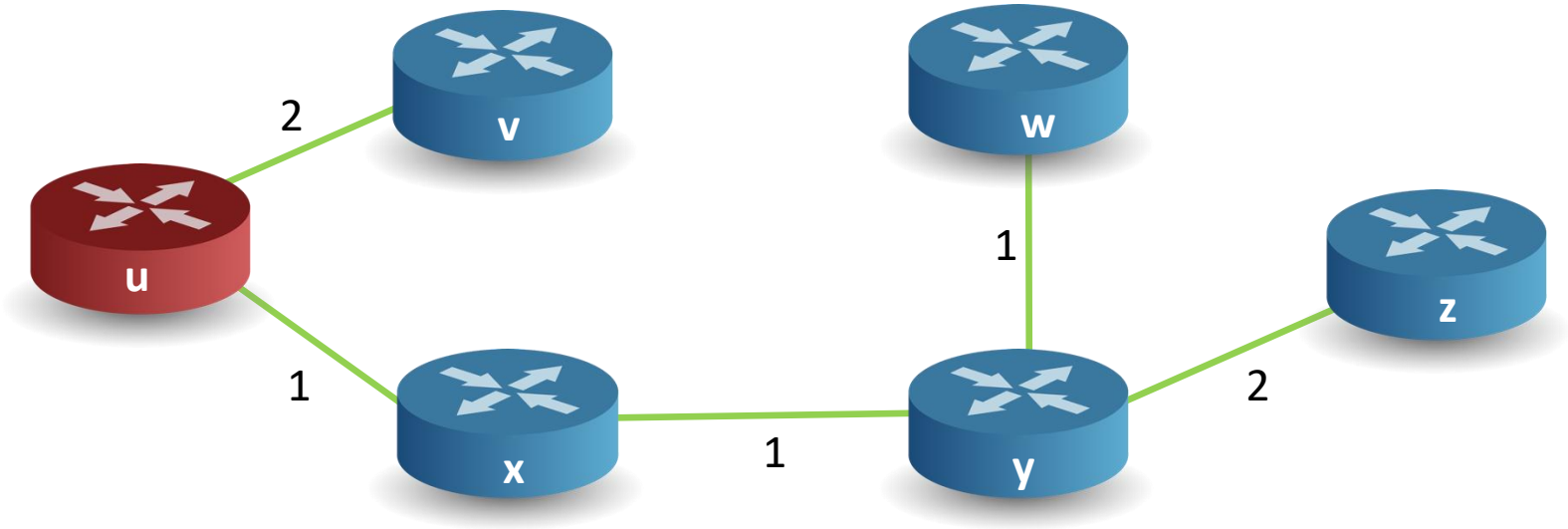
Routing Algorithms: Link State Analysis

Dijkstra's shortest path

Resulting forwarding table at u:

Destination	Link
v	*(u,v)*
x	*(u,x)*
y	*(u,x)*
w	*(u,x)*
z	*(u,x)*

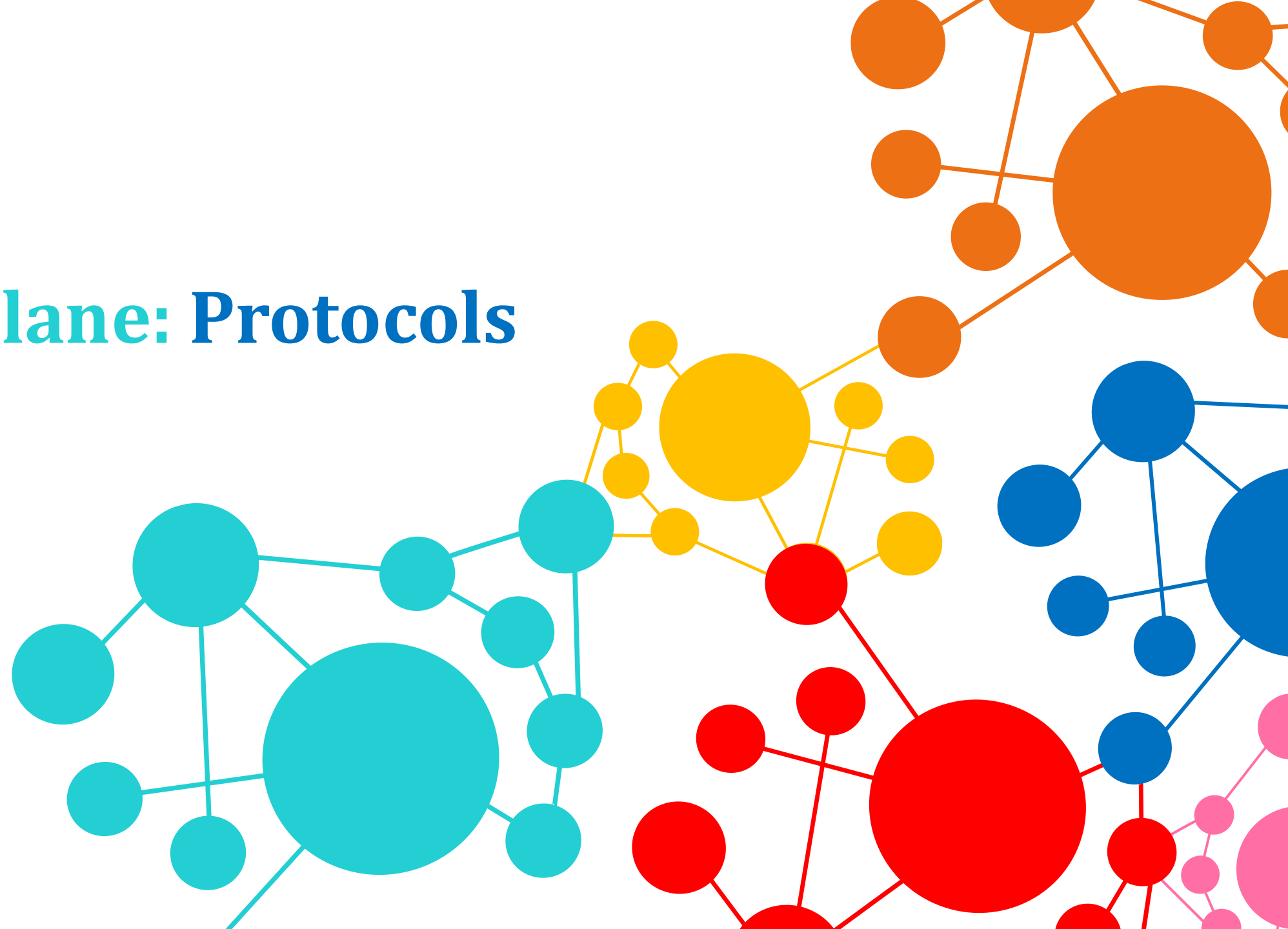
Resulting shortest-path tree from u:



Routing Algorithms: Distance Vector or Link State?

- Message Exchange
  - **LS**: Exchange with all nodes in the network $O(|N||E|)$
  - **DV**: Exchange with neighbors only
- Convergence
  - **LS**: Compute path from each node to every other node ($O(N^2)$)
  - **DV**: Decide next directly connected node from each node to every other node
    - Routing Loops & Count to Infinity Problem
- Robustness
  - **LS**: Error on the incorrect advertised link
  - **DV**: Error propagates through the network
- Store
  - **LS**: All paths
  - **DV**: Next hop

Control Plane: Protocols



Routing & Scalability

Our routing study thus far: **Idealized**

- All routers identical
- Flat Network
- **Not** true in practice

Scale: With billions of destinations

- Can not store all destinations in routing tables
- Routing table exchange would swamp links

Administrative autonomy

- Internet: Network of networks
- Each network admin may want to control routing in its own network

Internet Approach to Scalable Routing

- Aggregate routers into regions known as **autonomous systems (AS)** or **domains**

- **Intra-AS routing**
  - Routing among hosts and routers in the same AS (network)
  - All routers in AS must run **same** intra-domain protocol
  - Routers in **different** AS can run **different** intra-domain routing protocol
  - Gateway router: At **edge** of its own AS, has link(s) to router(s) in other ASes

- **Inter-AS routing**
  - Routing among ASes
  - Gateways perform inter-domain routing (as well as intra-domain routing)

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

Scale

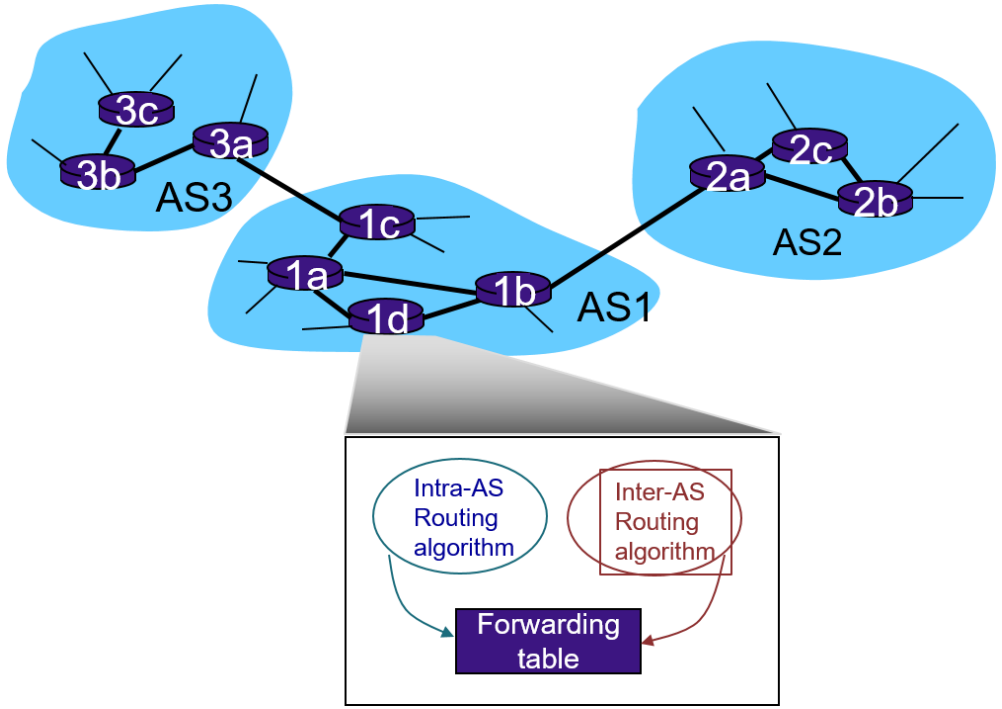
- Hierarchical routing saves table size, reduced update traffic

Performance

- Intra-AS: Can focus on performance
- Inter-AS: Policy may dominate over performance

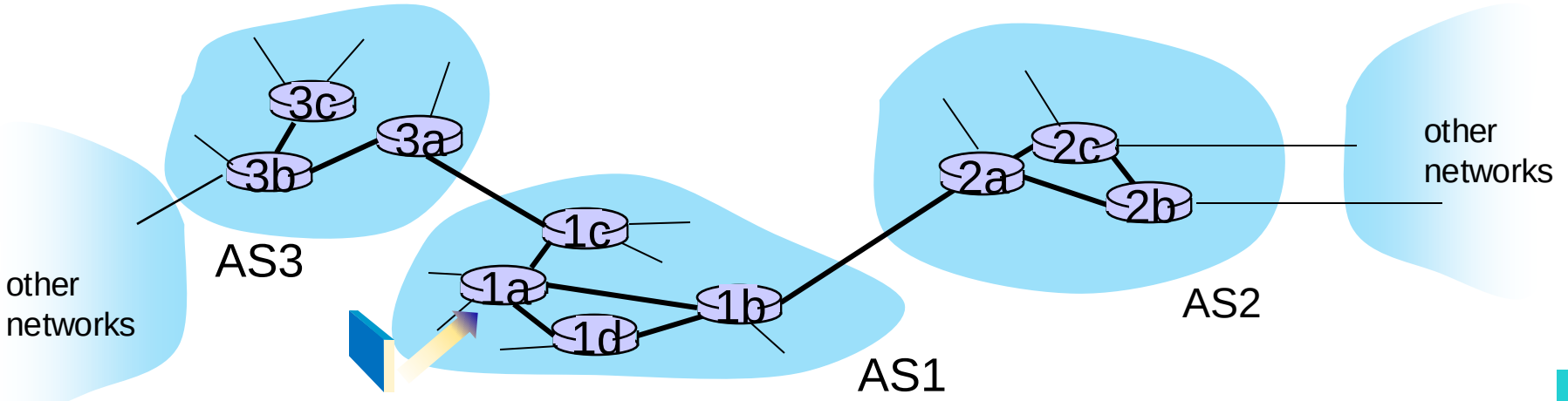
Interconnected ASes

- Forwarding table configured by both intra-AS and inter-AS routing algorithm
  - Intra-AS routing determine entries for destinations within AS
  - Inter-AS & intra-AS determine entries for external destinations



Inter-AS Tasks

- Suppose router in AS1 receives datagram destined outside of AS1
  - Router should forward packet to gateway router
    - Which gateway router?
- **AS1 must**
  - Learn which destinations are reachable through AS2 and which destinations through AS3
    - **Job of inter-AS routing**
  - Propagate reachability info to all routers in AS1



Intra-AS Routing

- Also known as **Interior Gateway Protocols (IGP)**

- Most common intra-AS routing protocols
  - **RIP**: Routing Information Protocol
  - **OSPF**: Open Shortest Path First
    (IS-IS protocol similar to OSPF)
  - **IGRP**: Interior Gateway Routing Protocol
    (Cisco proprietary for decades, until 2016)

OSPF (Open Shortest Path First)

- Open: Publicly available

- Uses link-state algorithm
  - Link state packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra

- Router floods OSPF link-state advertisements to all other routers in **entire** AS
  - Carried in OSPF messages **directly over IP** (rather than TCP or UDP)
  - Link state: For each attached link

- **IS-IS routing protocol**: Similar to OSPF

OSPF: Advanced Features

- **Security***:* All OSPF messages authenticated (to prevent malicious intrusion)

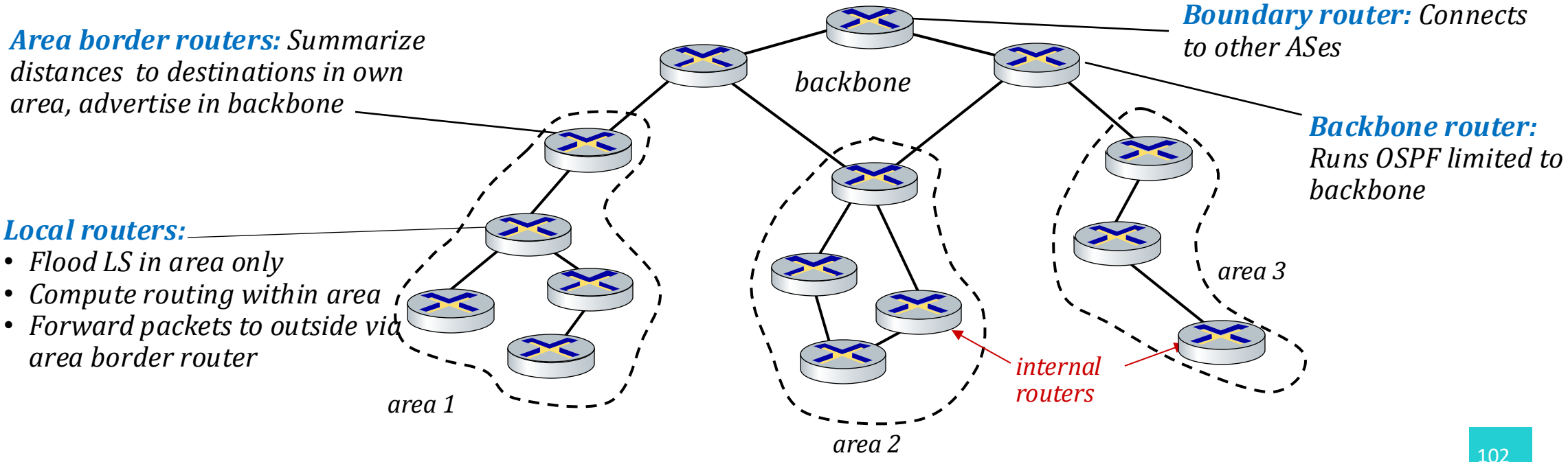
- **Multiple** same-cost **paths** allowed (only one path in RIP)
- For each link multiple cost metrics for different **TOS**
    - Example: Satellite link cost set low for best effort ToS & high for real-time ToS

- Integrated unicast and **multicast** support
    - Multicast OSPF (MOSPF) uses same topology database as OSPF

- **Hierarchical** OSPF in large domains.

Hierarchical OSPF

- **Two-level hierarchy:** Local area, backbone.
  - Link-state advertisements only in area
  - Each node has detailed area topology, only know direction (shortest path) to nets in other areas.



Control Plane: Policies

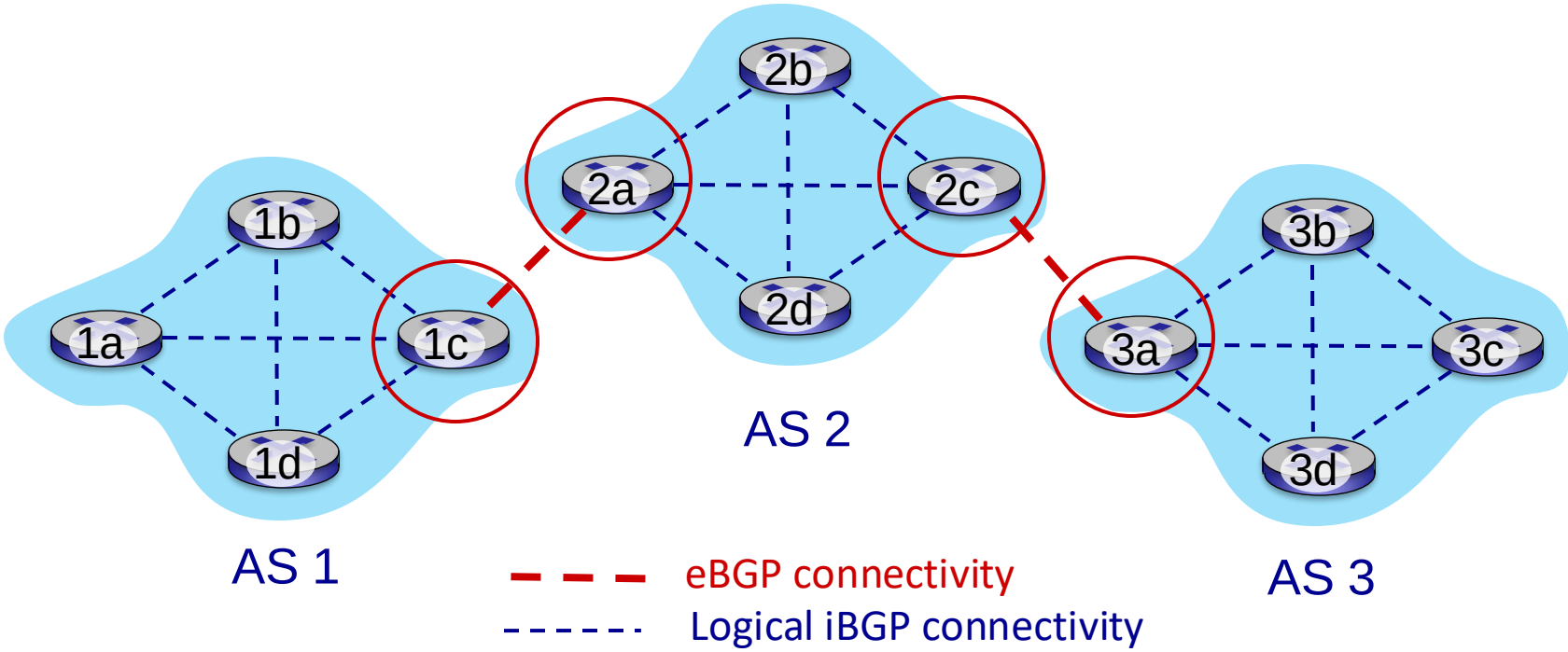
Internet Inter-AS Routing: BGP

- **BGP (Border Gateway Protocol):** The de facto inter-domain routing protocol
  - Glue that holds the Internet together

- BGP provides each AS a means to
  - **eBGP:** Obtain subnet reachability information from neighboring ASes
  - **iBGP:** Propagate reachability information to all AS-internal routers.
  - Determine **good** routes to other networks based on **reachability** information and **policy**

- Allows subnet to advertise its existence to rest of Internet: **I am here!**

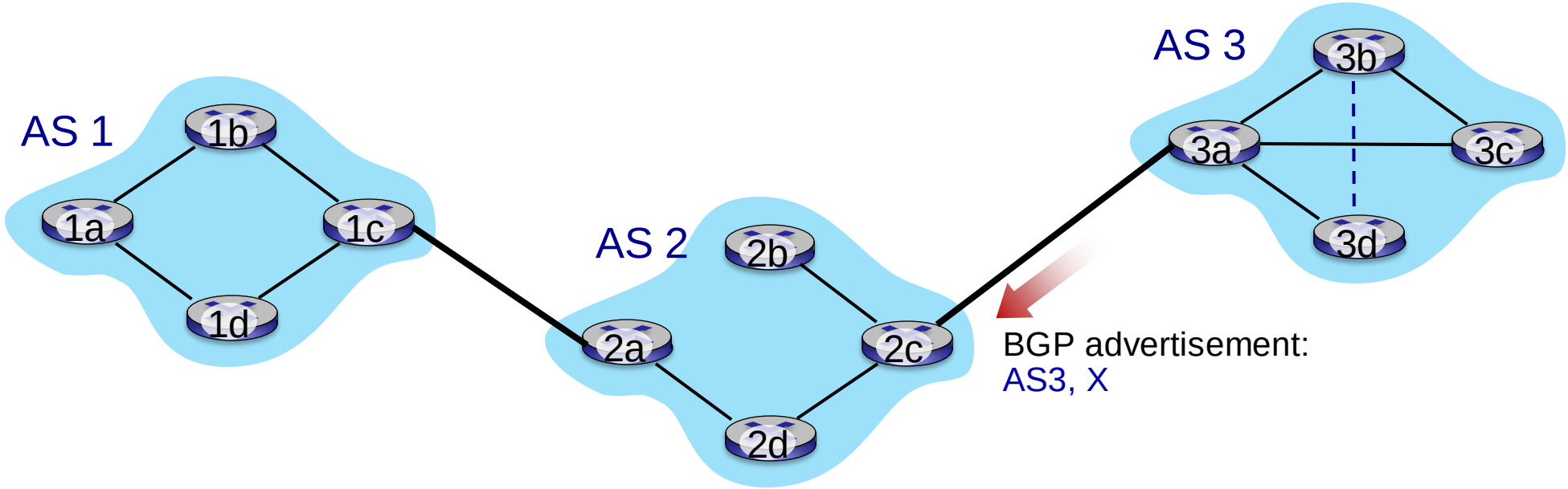
eBGP & iBGP Connections



gateway routers run both eBGP and iBGP protocols

BGP Basics

- **BGP session:** Two BGP routers (peers) exchange BGP messages over semi-permanent TCP connection:
  - Advertising **paths** to different destination network prefixes (BGP is a **path vector** protocol)
- 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 & BGP Routes

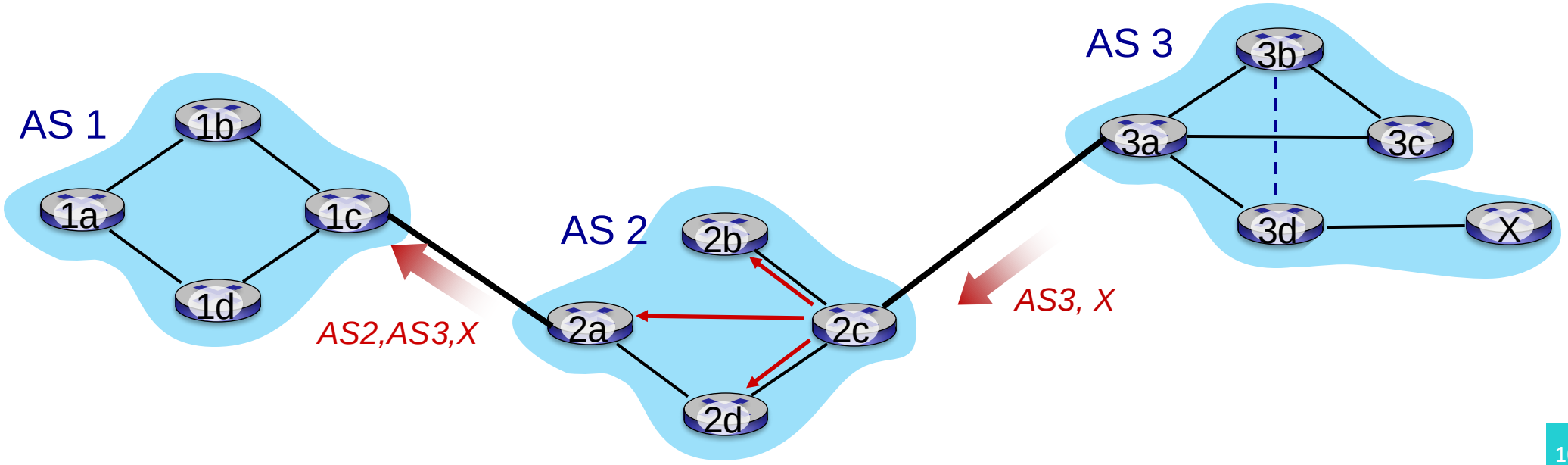
- Advertised prefix includes BGP attributes
  - Prefix + attributes = **route**

- Two important attributes
  - **AS-PATH:** List of ASes through which prefix advertisement has passed
  - **NEXT-HOP:** Indicates specific internal-AS route to next-hop AS

- **Policy-based routing**
  - Gateway receiving route advertisement uses **import policy** to accept/decline path (e.g. never route through AS Y)
  - AS policy also determines whether to **advertise** path to other neighboring ASes

BGP Path Advertisement

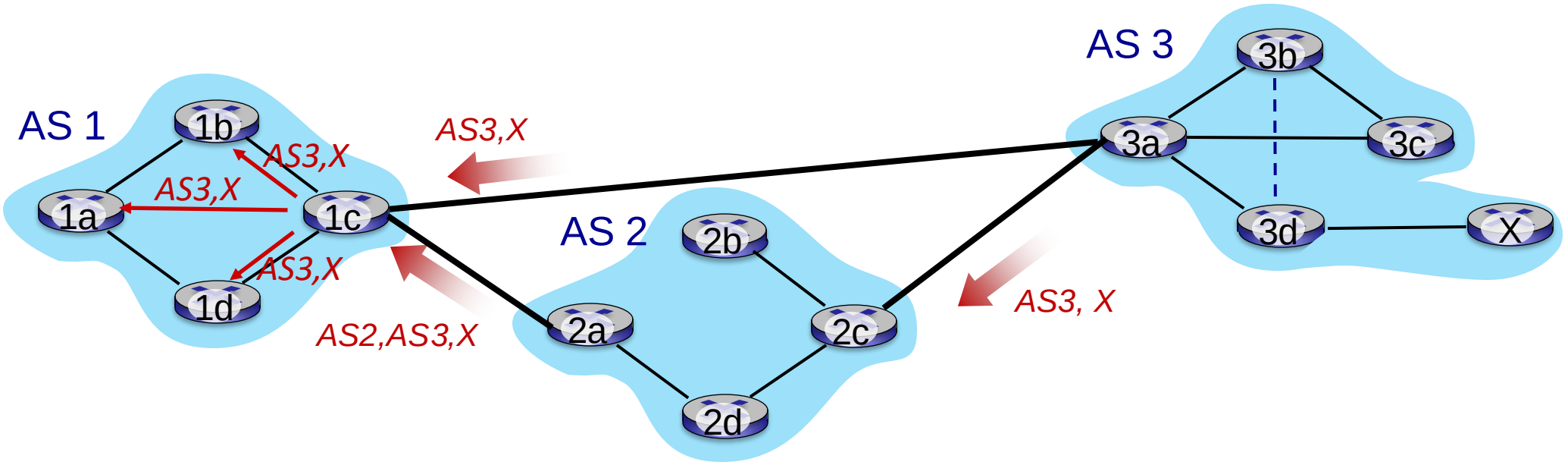
- AS2 router 2c receives path advertisement **AS3, X** (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3, X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path **AS2, AS3, X** to AS1 router 1c



BGP Path Advertisement

Gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path **AS2, AS3, *X*** from 2a
- AS1 gateway router 1c learns path **AS3, *X*** from 3a
- Based on **policy**, AS1 gateway router 1c chooses path **AS3, *X*, and advertises path within AS1 via iBGP**



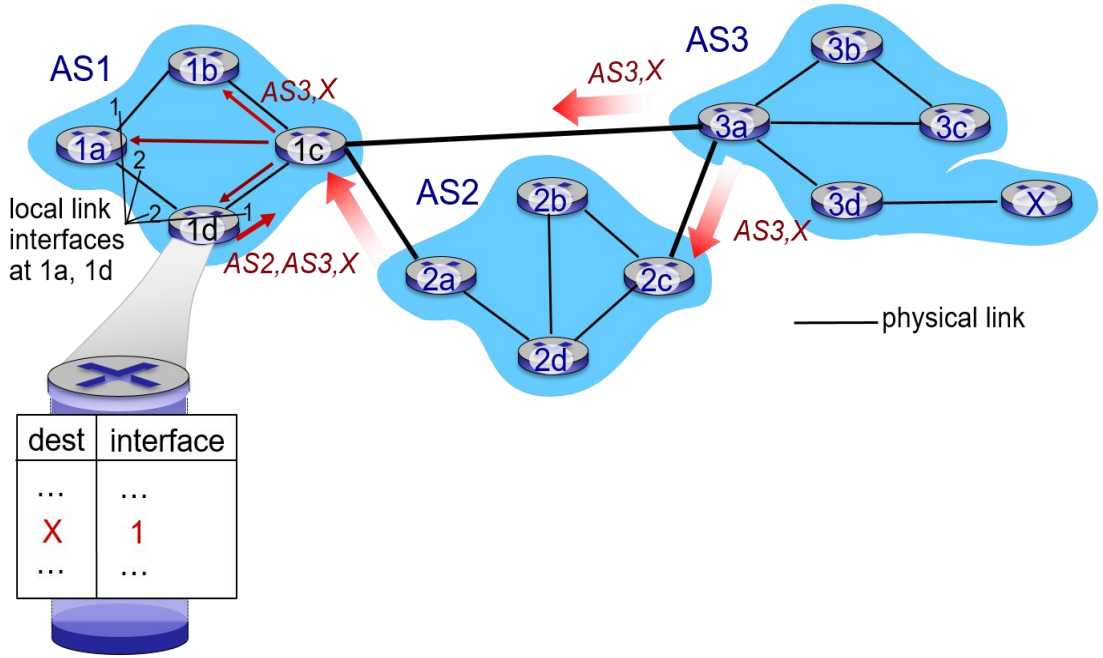
BGP Messages

- BGP messages exchanged between peers over TCP connection

- BGP messages
  - **Open:** Opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - **Update:** Advertises new path (or withdraws old)
  - **Keepalive:** Keeps connection alive in absence of **Updates**; also ACKs **Open** request
  - **Notification:** Reports errors in previous message; also used to close connection

BGP, OSPF, Forwarding Table Entries

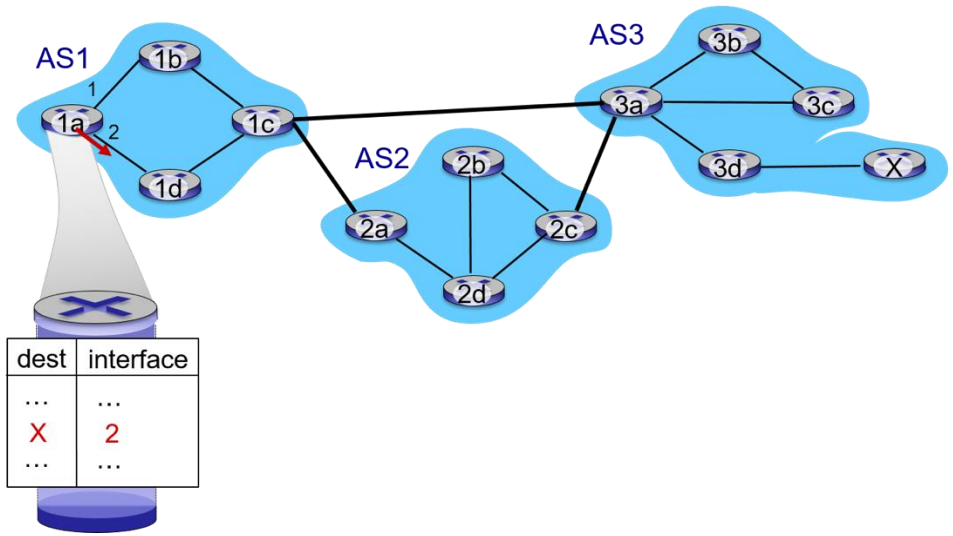
- **Q: How does router set forwarding table entry to distant prefix?**



- Recall: 1a, 1b, 1d learn about destination X via iBGP from 1c: path to X goes through 1c
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP, OSPF, Forwarding Table Entries

- **Q: How does router set forwarding table entry to distant prefix**
  - Recall: 1a, 1b, 1d learn about destination X via iBGP from 1c: Path to X goes through 1c
  - 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
  - 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

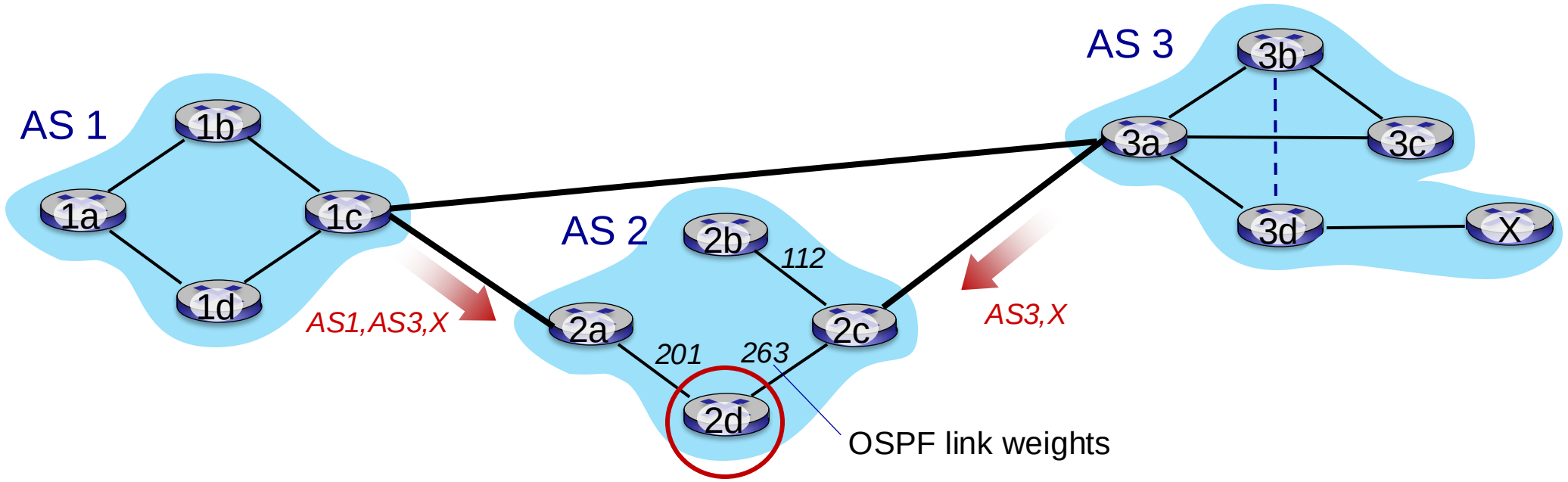


BGP Route Selection

- Router may learn about more than one route to destination AS
  It selects route based on
    - Local preference value attribute: **Policy decision**
    - Shortest AS-PATH
    - Closest NEXT-HOP router: **Hot potato routing**
    - Additional criteria

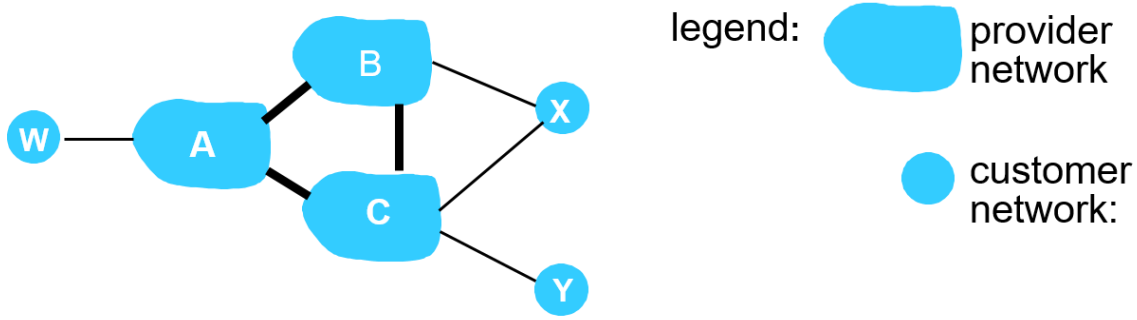
Hot Potato Routing

- 2d learns (via iBGP) it can route to X via 2a or 2c
- **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!



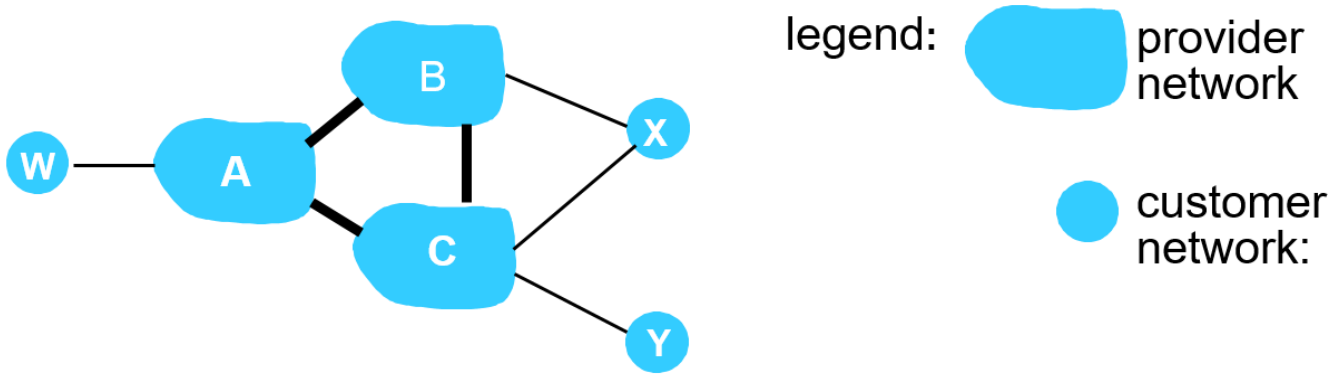
BGP: Achieving Policy Via Advertisements

- **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 **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



BGP: Achieving Policy Via Advertisements

- **Suppose an ISP only wants to route traffic to/from its customer networks (Does not want to carry transit traffic between other ISPs)**
  - A,B,C are **provider networks**
  - X,W,Y are customers (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



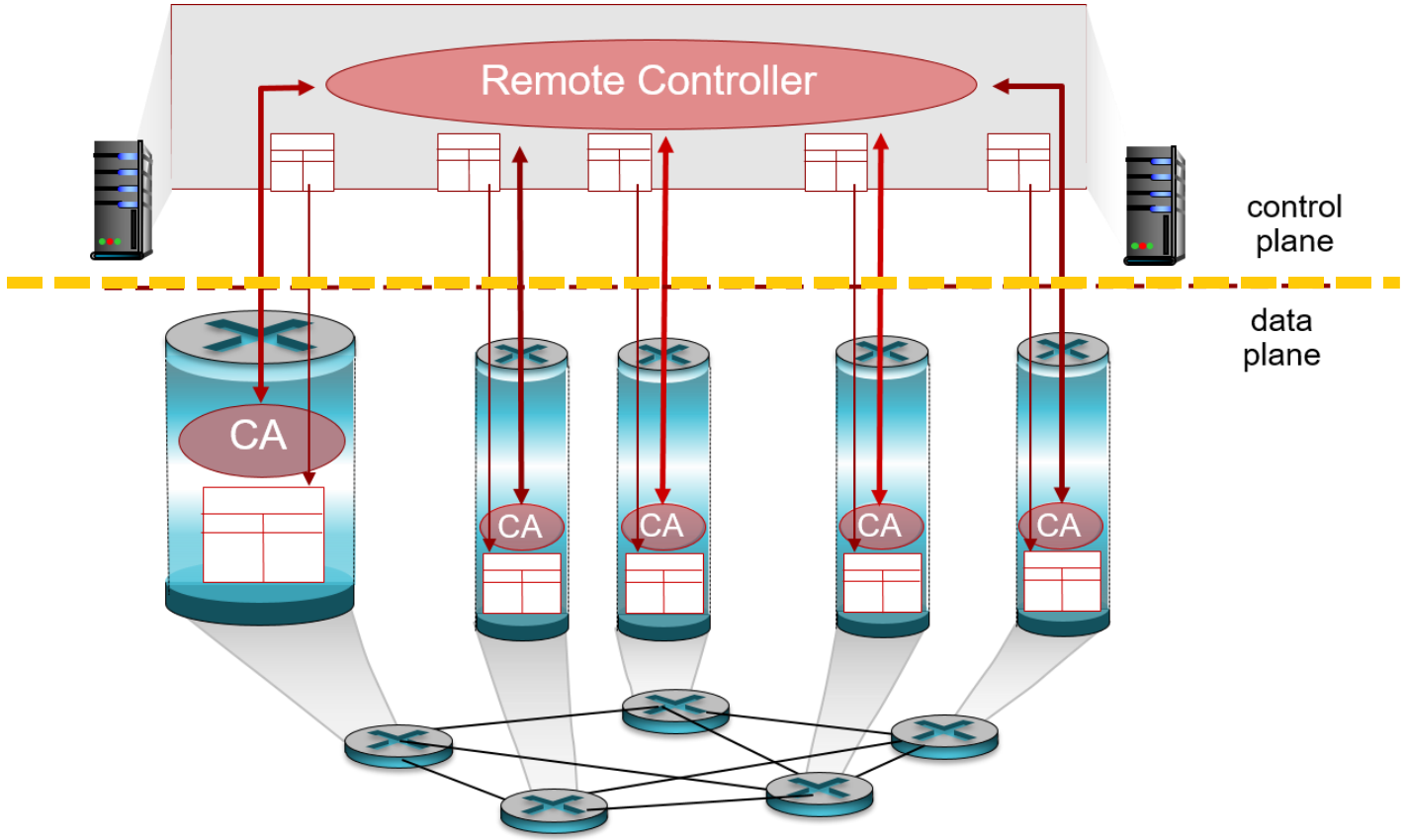
Control Plane: SDN

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

Logically Centralized Control

- 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?

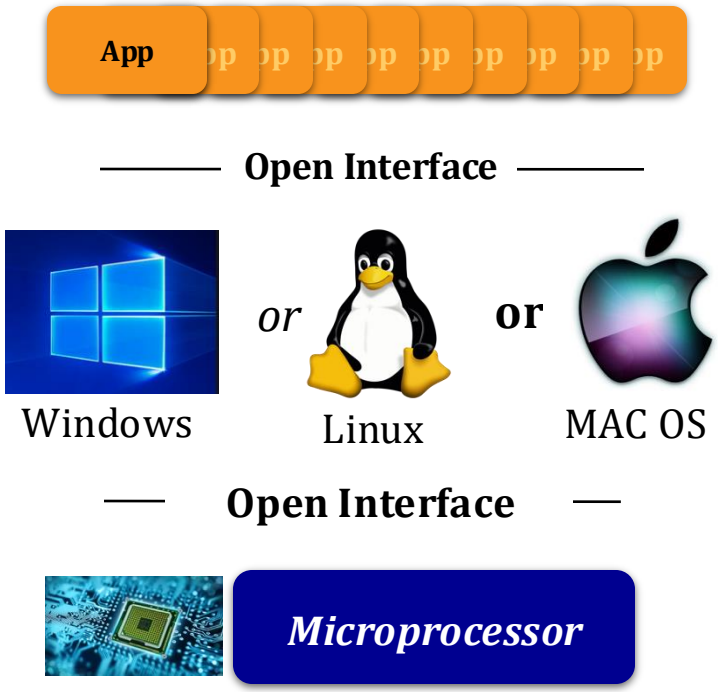
- Easier network management
  - Avoid router misconfigurations
  - Greater flexibility of traffic flows

- 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

Vertically integrated
Closed, proprietary
Slow innovation
Small industry



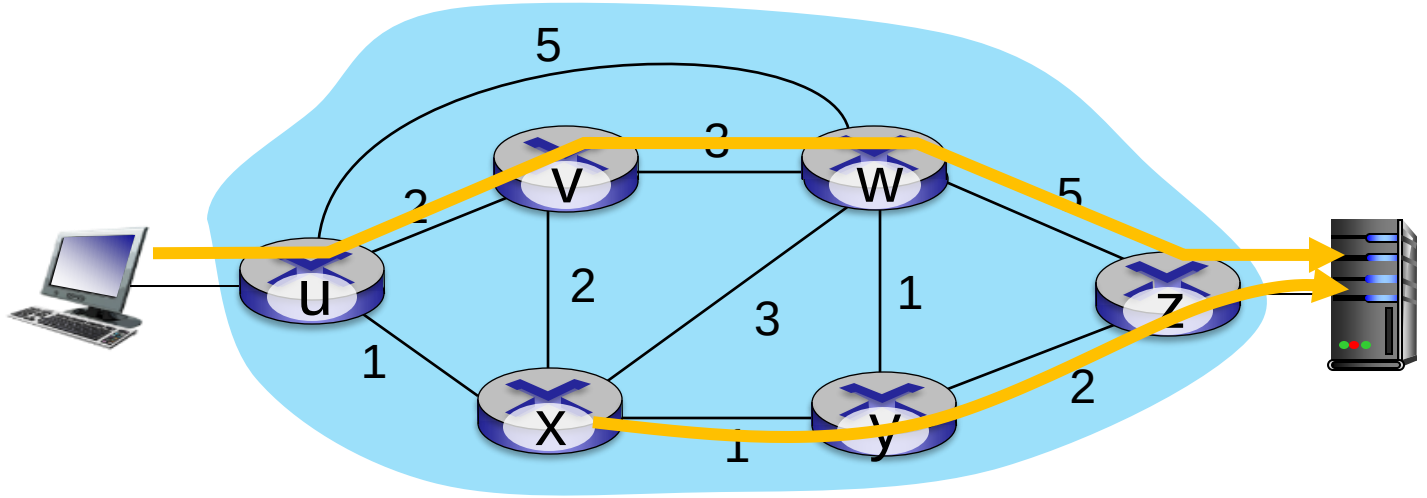
Horizontal
Open interfaces
Rapid innovation
Huge industry

Traffic Engineering

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

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

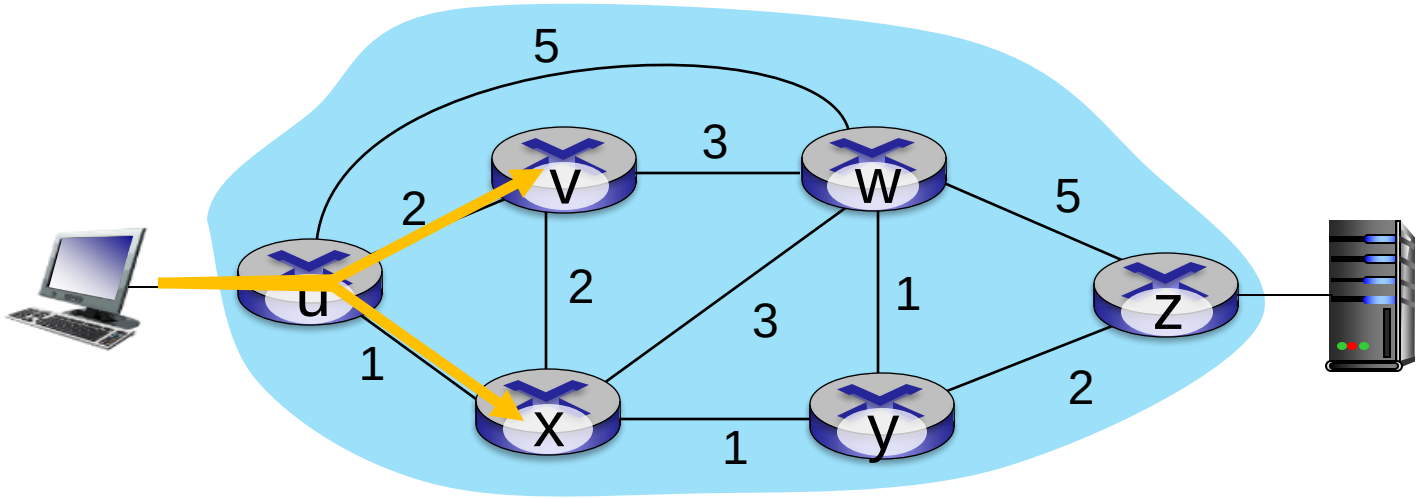
Link weights are only control knobs: wrong!



Traffic Engineering

Q: What if network operator wants to split u-to-z traffic along uvwz **and** uxyz (load balancing)?

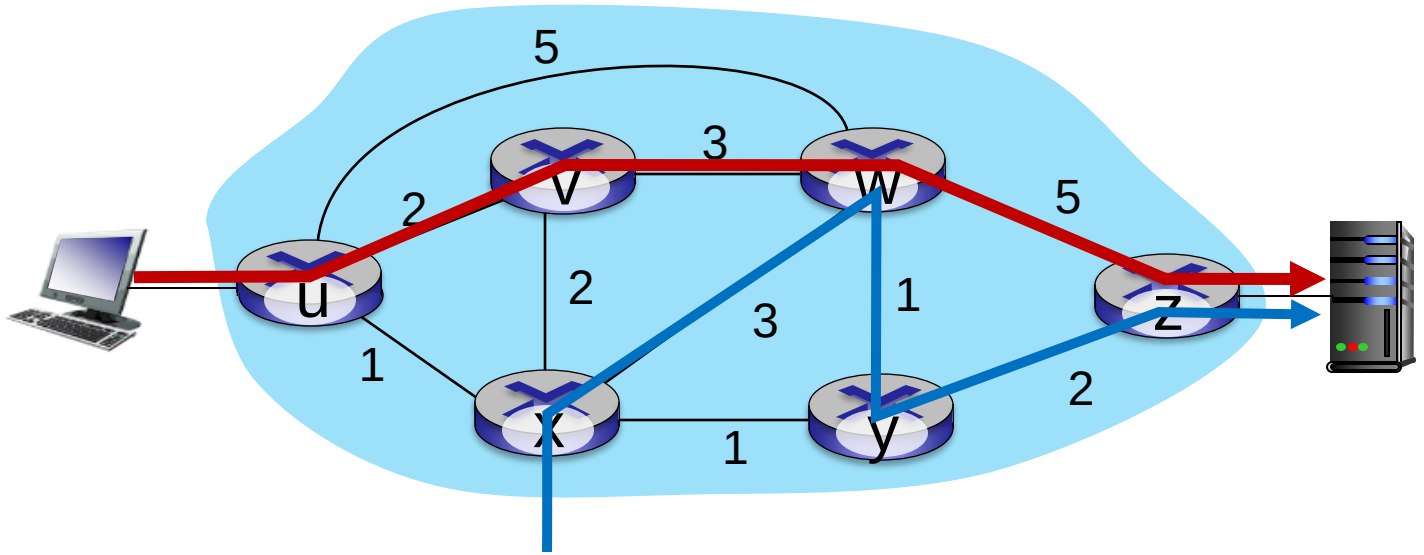
A: Cannot do it (or need a new routing algorithm)



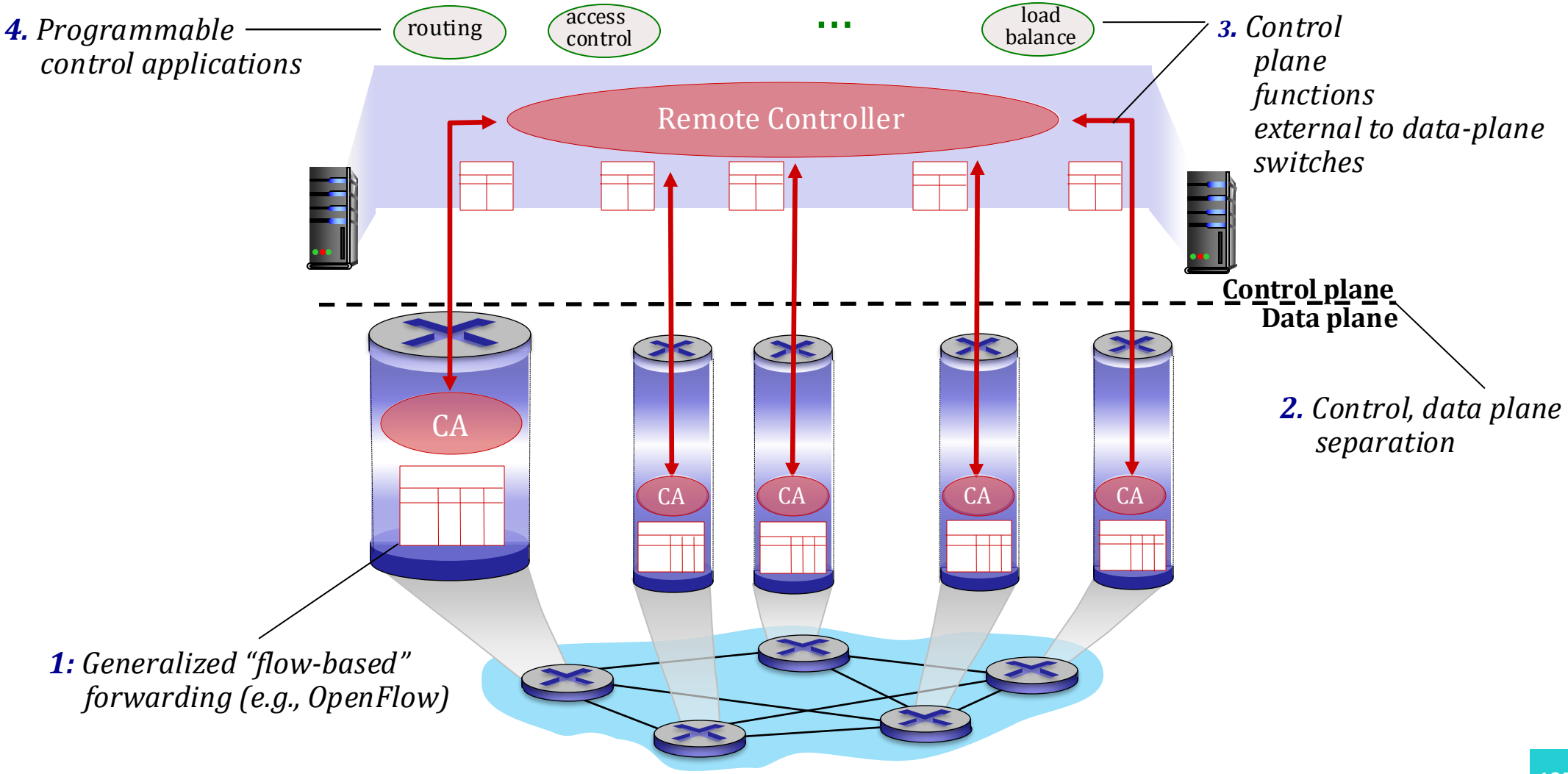
Traffic Engineering

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

A: Cannot do it (with destination based forwarding, and LS, DV routing)



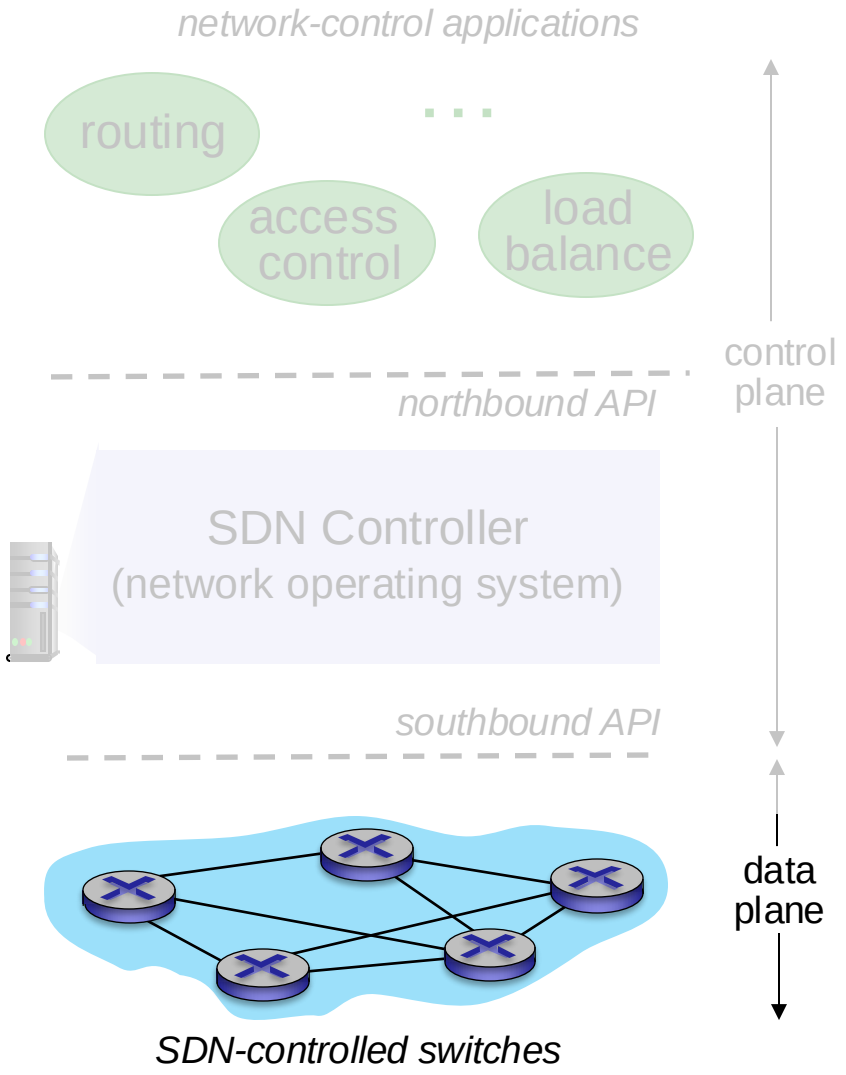
Software Defined Networking (SDN)



SDN Perspective: Data Plane Switches

Data Plane Switches

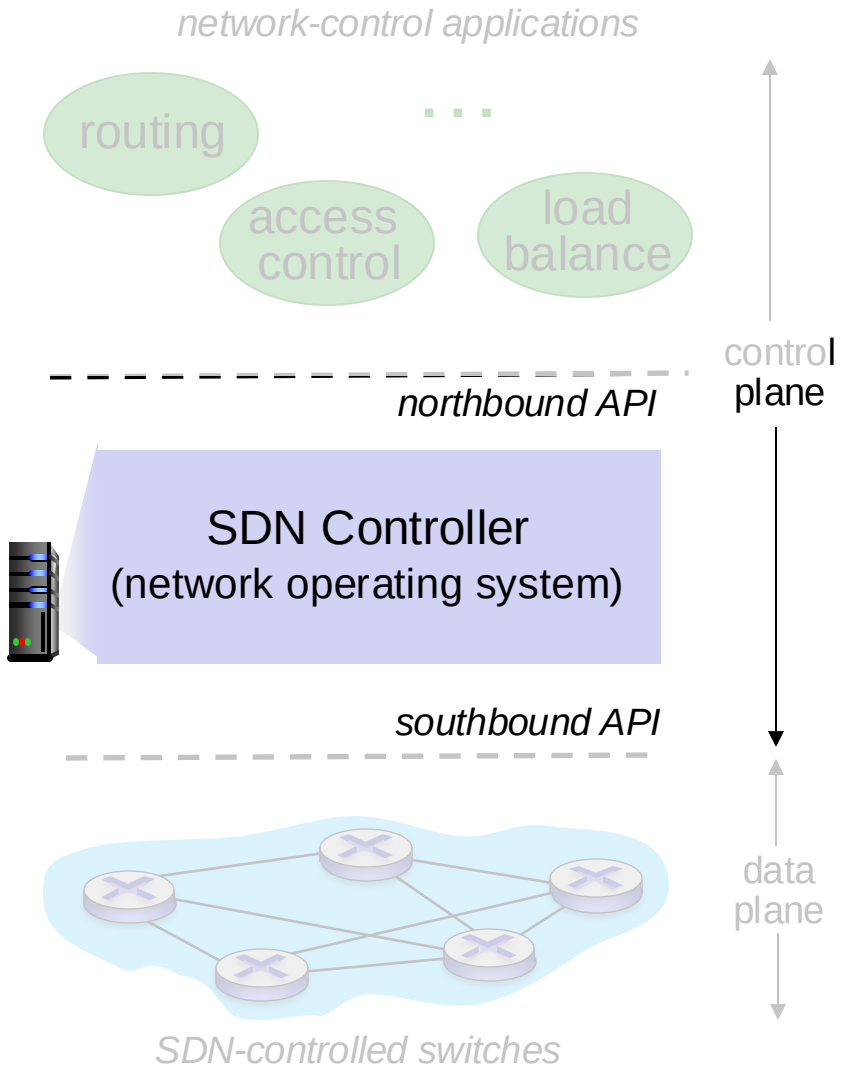
- Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- Switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
  - Defines what is controllable and what is not
- Protocol for communicating with controller (e.g.,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



SDN Perspective: SDN Controller

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

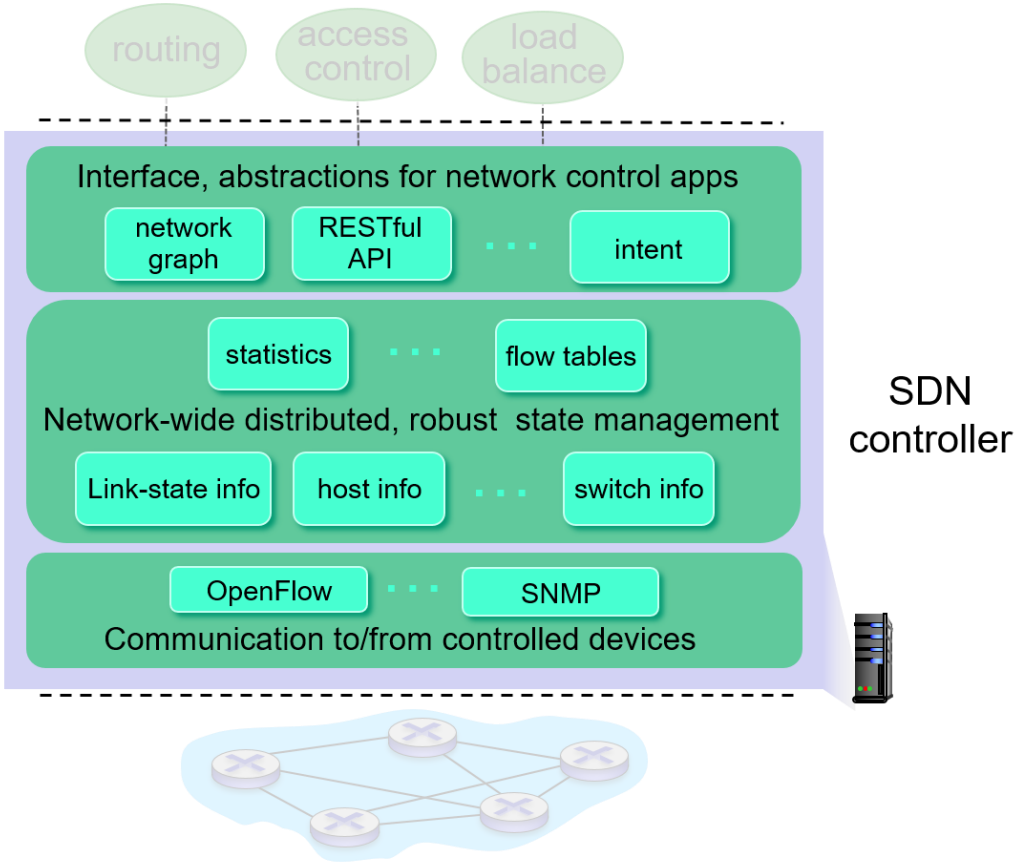


Components of SDN Controller

Interface layer to network control apps: Abstractions API

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

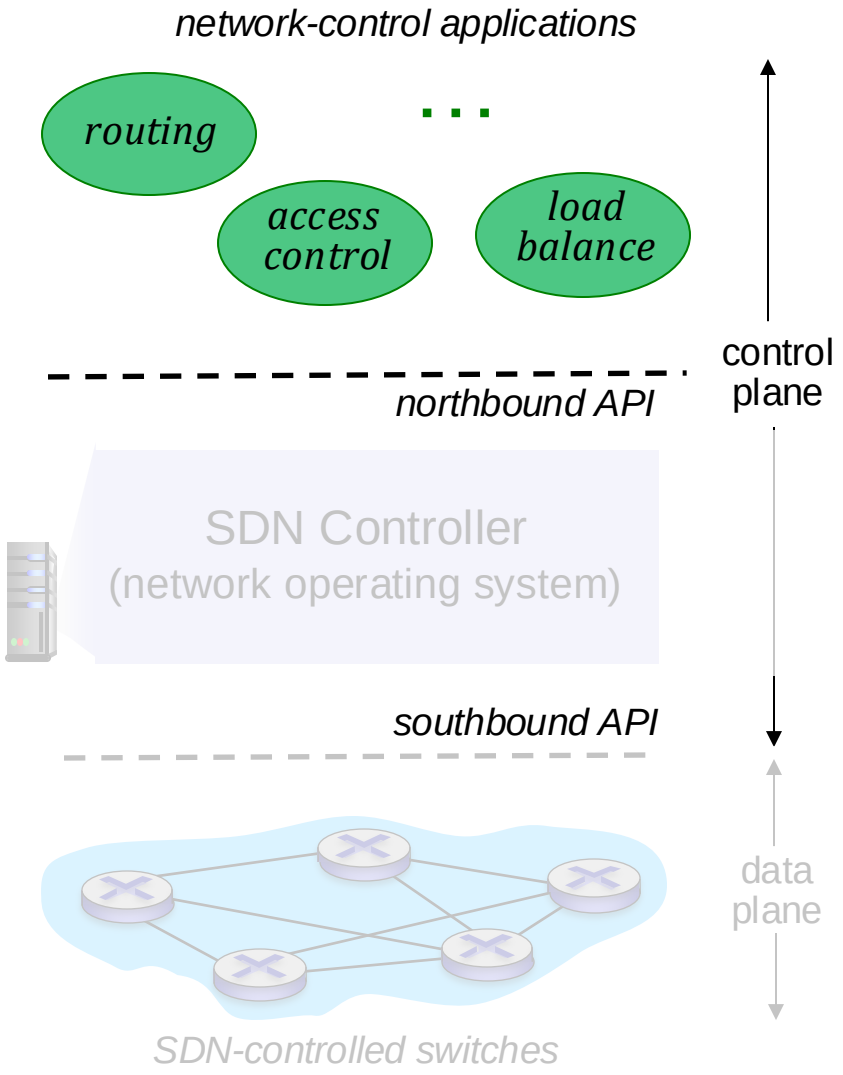
Communication layer: Communicate between SDN controller and controlled switches



SDN Perspective: Control Applications

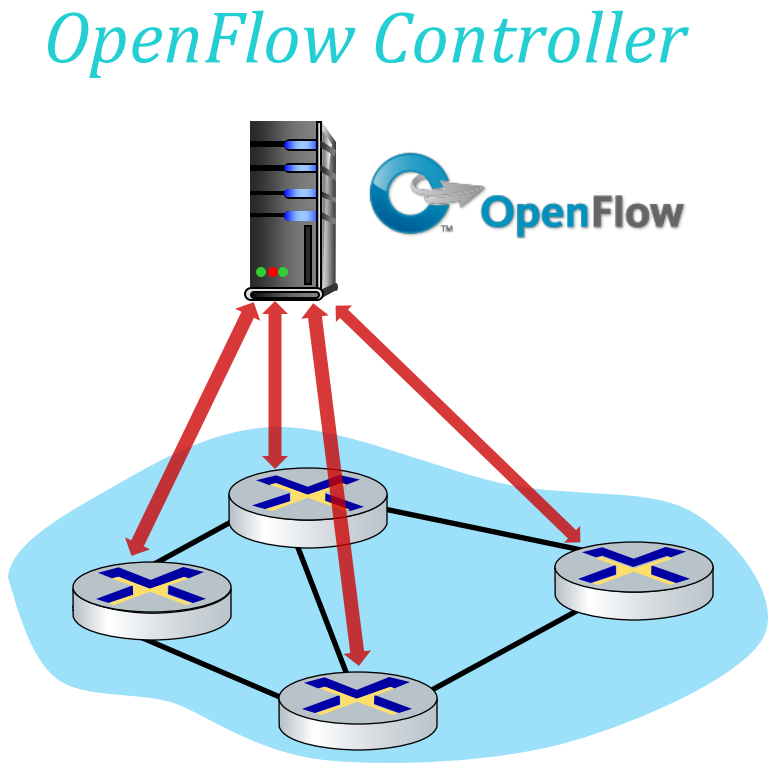
Network-control apps

- Brains of control: Implement control functions using lower-level services, API provided by SDN controller
- **Unbundled:** Can be provided by third party: Distinct from routing vendor, or SDN controller



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

Key controller-to-switch messages

- **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
  - Message contains the packet

OpenFlow: Switch-to-Controller Messages

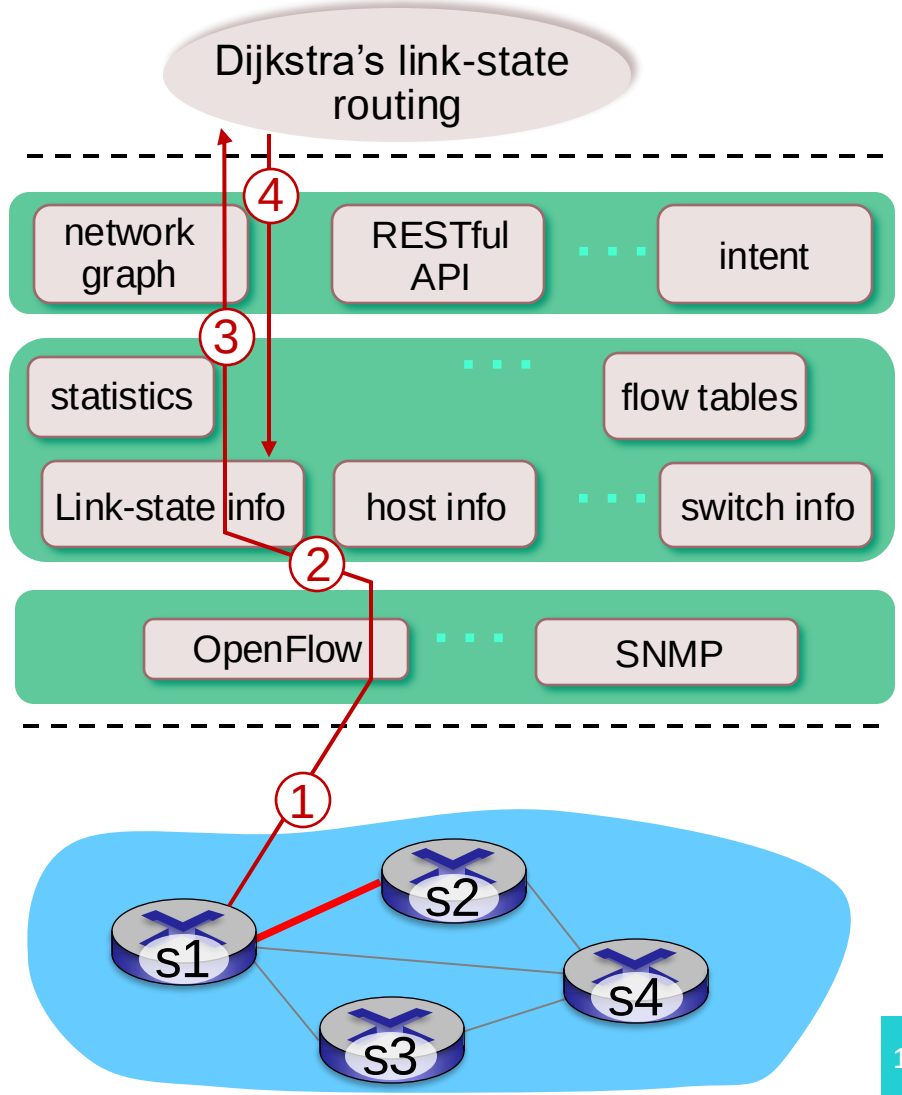
Key switch-to-controller messages

- **Packet-in:** Transfer packet (and its control) to controller.
- **Flow-removed:** Flow table entry deleted at switch
- **Port status:** Inform controller of a change on a port.

Fortunately, network operators do not program switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

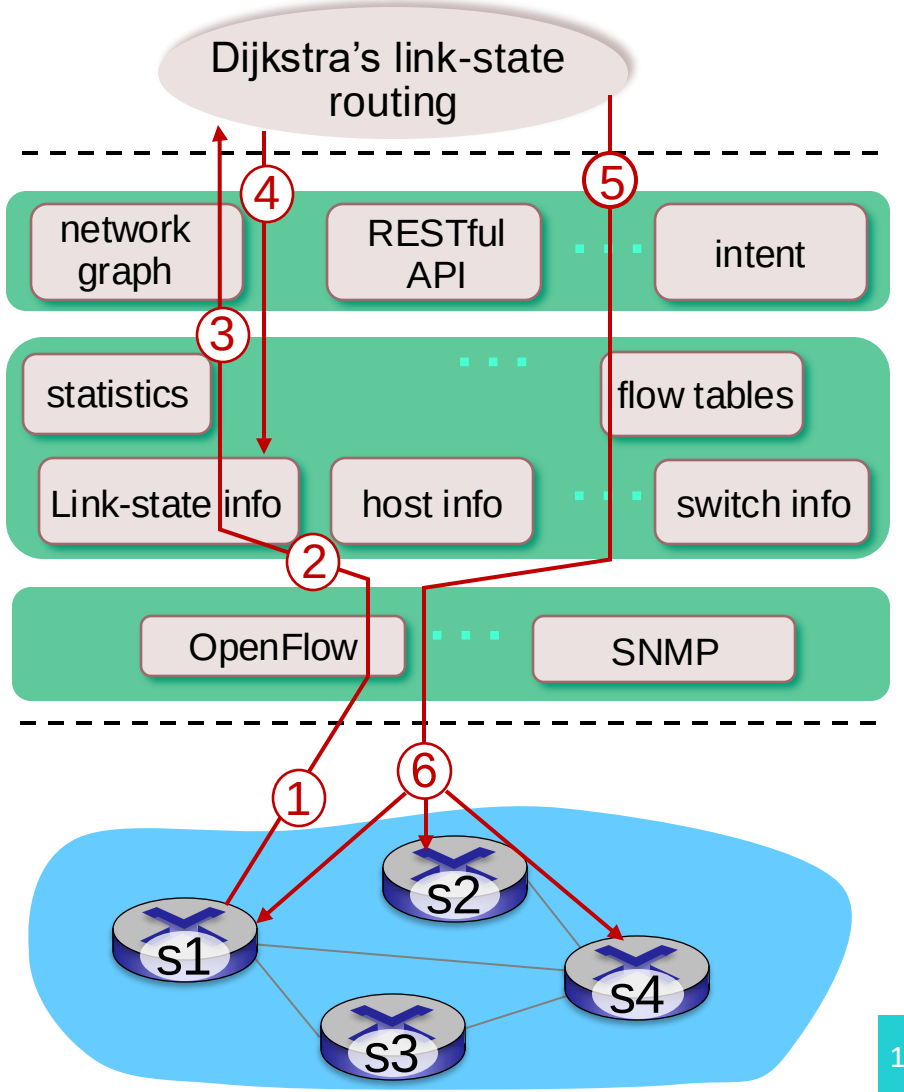
SDN: Control/Data Plane Interaction Example

- S1, experiencing link failure using OpenFlow port status message to notify controller
- SDN controller receives OpenFlow message, updates link status info
- Dijkstra's routing algorithm application has previously registered to be called whenever link status changes. It is called.
- Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes



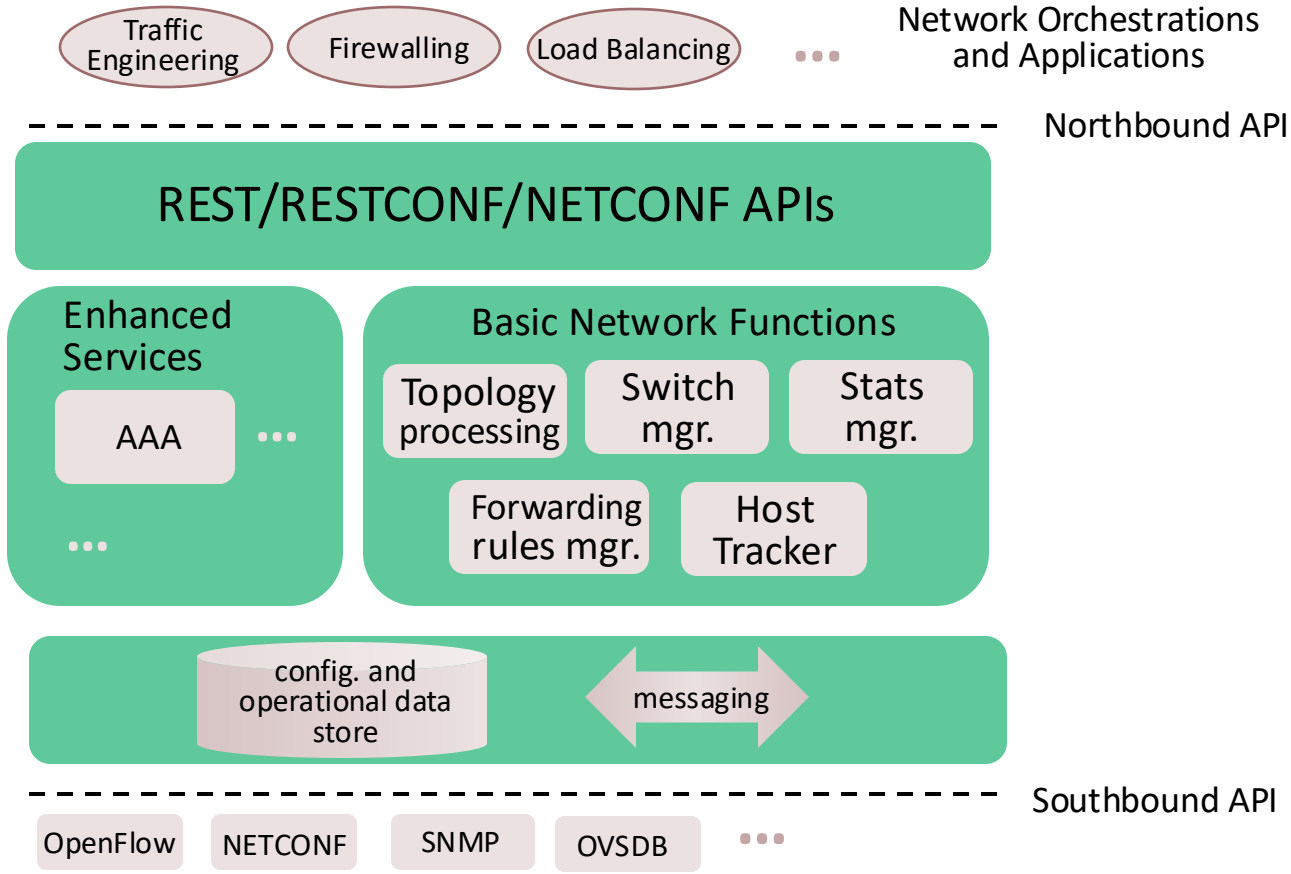
SDN: Control/Data Plane Interaction Example

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



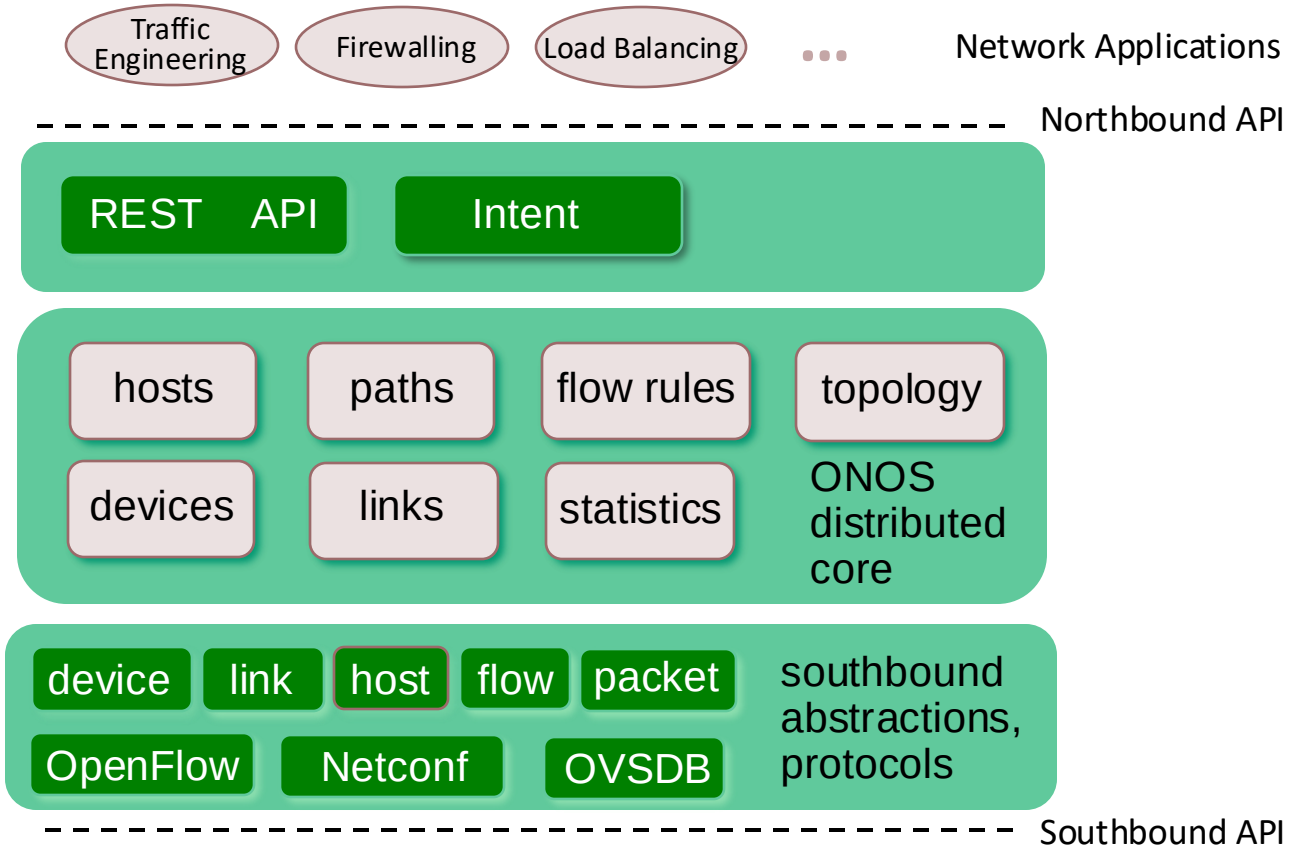
Open Daylight Controller

- 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

- 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

Summary

- Network Layer
  - Routing (Control Plane)
  - Forwarding (Data Plane)
  - Router Architecture Overview

- Data Plane
  - Forwarding
  - Internet Protocol (IP)
  - Generalized Forwarding and SDN (Software Defined Networking)

- Control Plane
  - Routing (Per-Router Control)
    - Algorithms
    - Protocols
    - Policies
  - Software Defined Networking (Logically Centralized Control)

Acknowledgements

- The following materials have been used in preparation of this presentation:

  [1] Textbook and (edited) Slides: Computer Networking: A Top-Down Approach
  James Kurose, Keith Ross
  7th and 8th Edition, Pearson
  http://gaia.cs.umass.edu/kurose_ross/

  [2] Reference: Computer Networks: A Systems Approach
  https://www.systemsapproach.org/book.html

- Recommended Additional Resources:

  [1] Interactive Exercises (Chapters Four and Five)
  http://gaia.cs.umass.edu/kurose_ross/interactive/