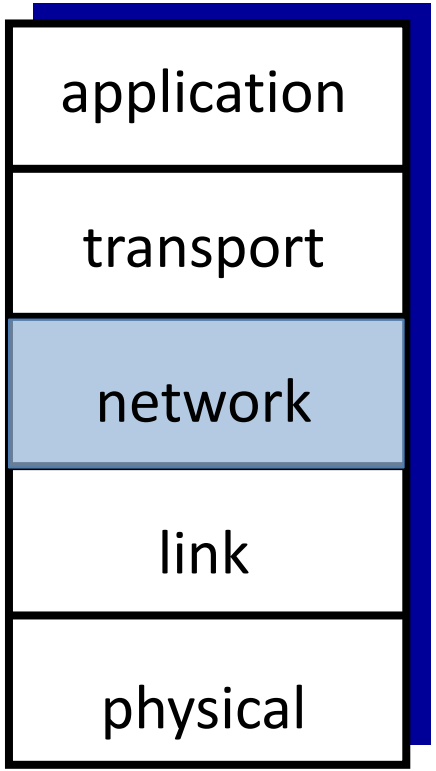
CS 4390: Network Layer

Shuang Hao
University of Texas at Dallas
Fall 2023

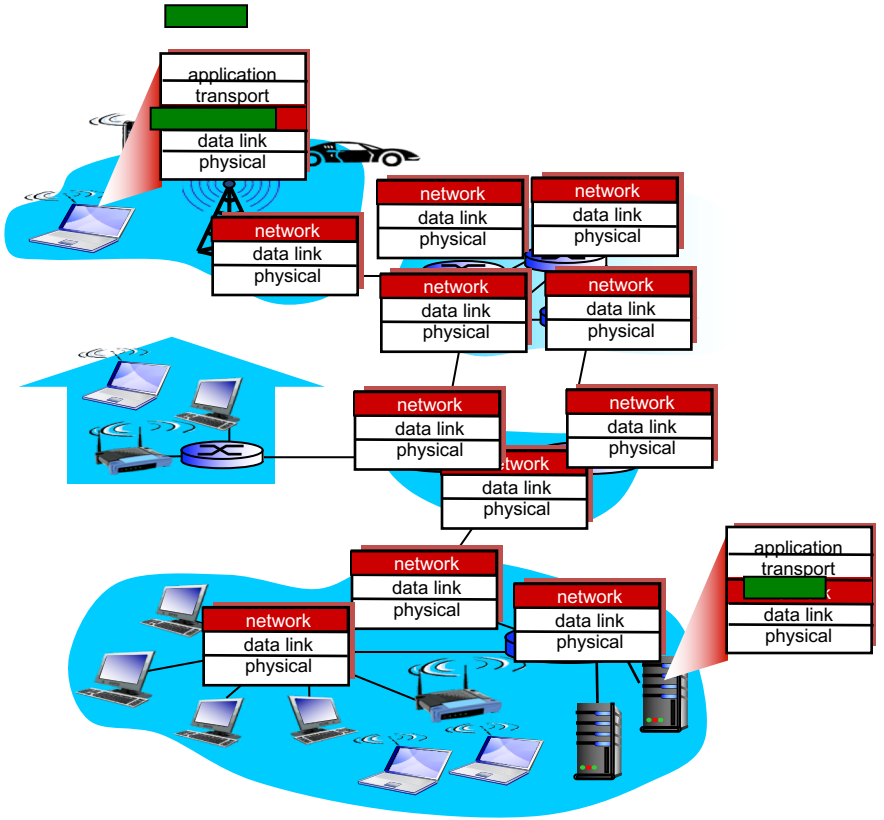
Network Layer

- Our goals
  - understand principles behind network layer services:
    - network layer service models
    - forwarding versus routing
    - how a router works
    - routing (path selection)
    - addressing on the network layer



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, router
  - router examines header fields in all IP datagrams passing through it

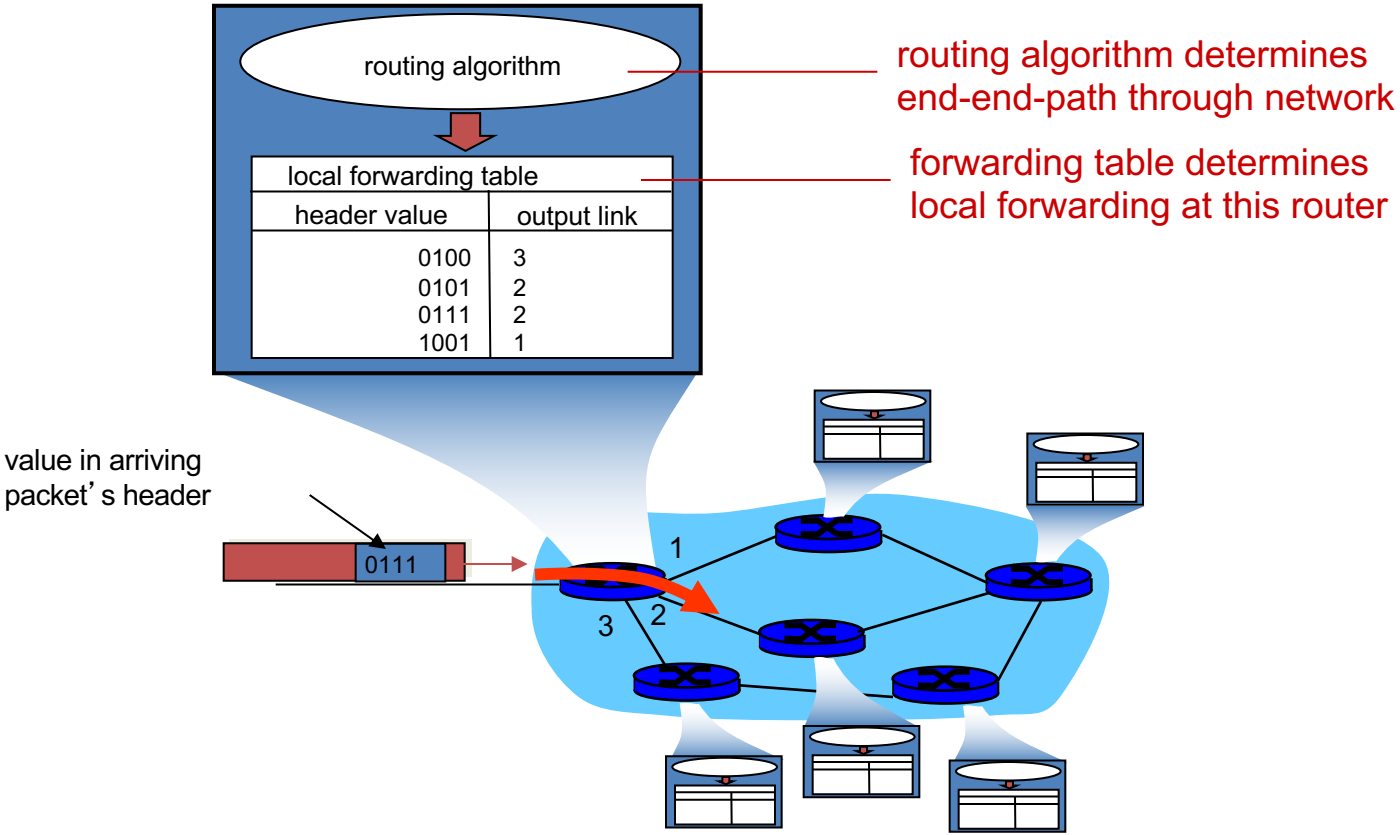


Two Key Network-layer Functions

- Forwarding: move packets from router's input to appropriate router output
- Routing: determine route taken by packets from source to dest.
  - routing algorithms

- Analogy
  - Routing: process of planning trip from source to dest
  - Forwarding: process of getting through single interchange

Interplay Between Routing and Forwarding



Network v.s. Transport Layer Service

- Network layer connection service
  - Between two hosts
- Transport layer connection service
  - between two processes

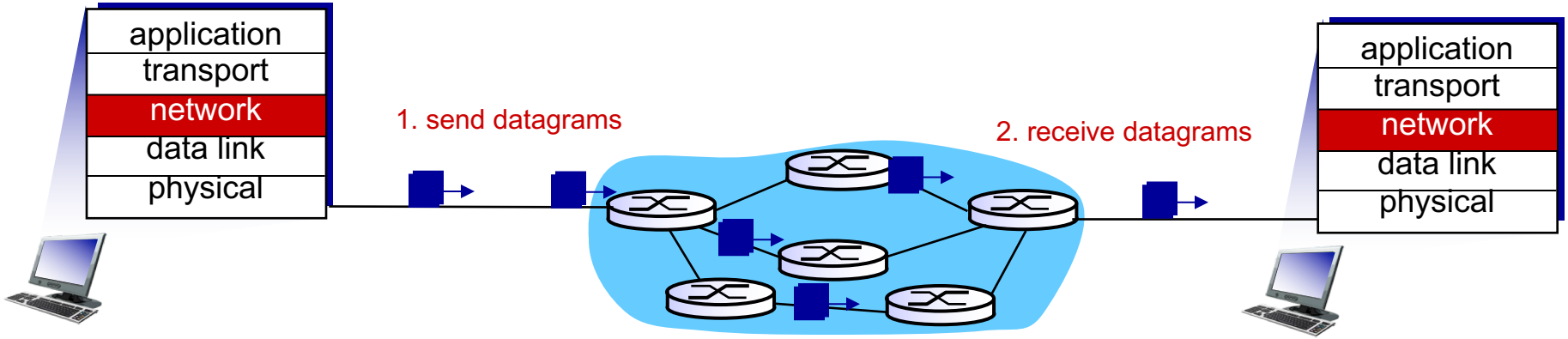
Connection, Connection-less Service

- Datagram network provides network-layer connectionless service
  - Simple inside network, complexity at “edge”

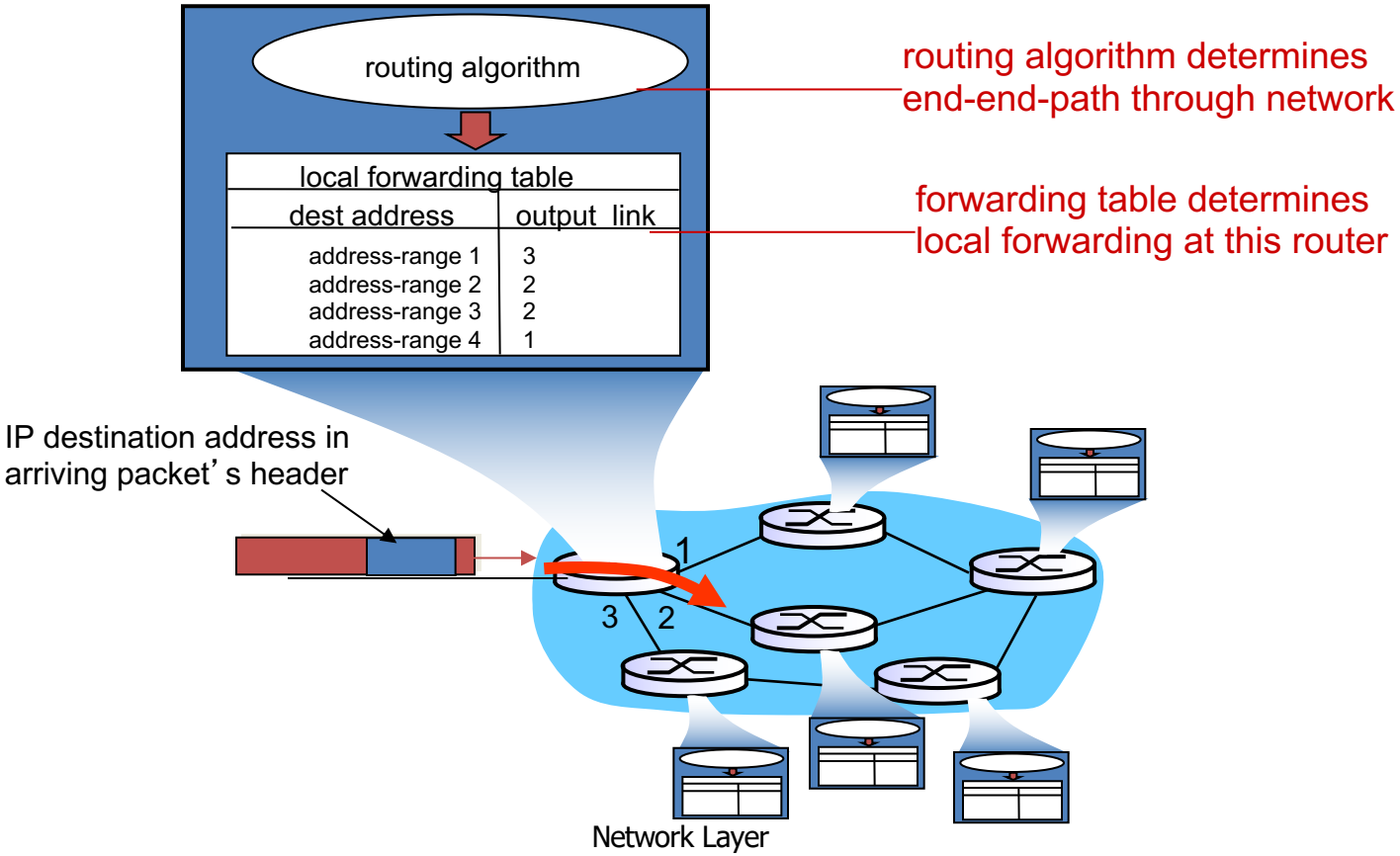
- Virtual-circuit network provides network-layer connection service
  - Complexity inside network

Datagram Networks

- No call setup at network layer
- Routers: no state about end-to-end connections
  - No network-level concept of “connection”
- Packets forwarded using destination host address



Datagram Forwarding Table

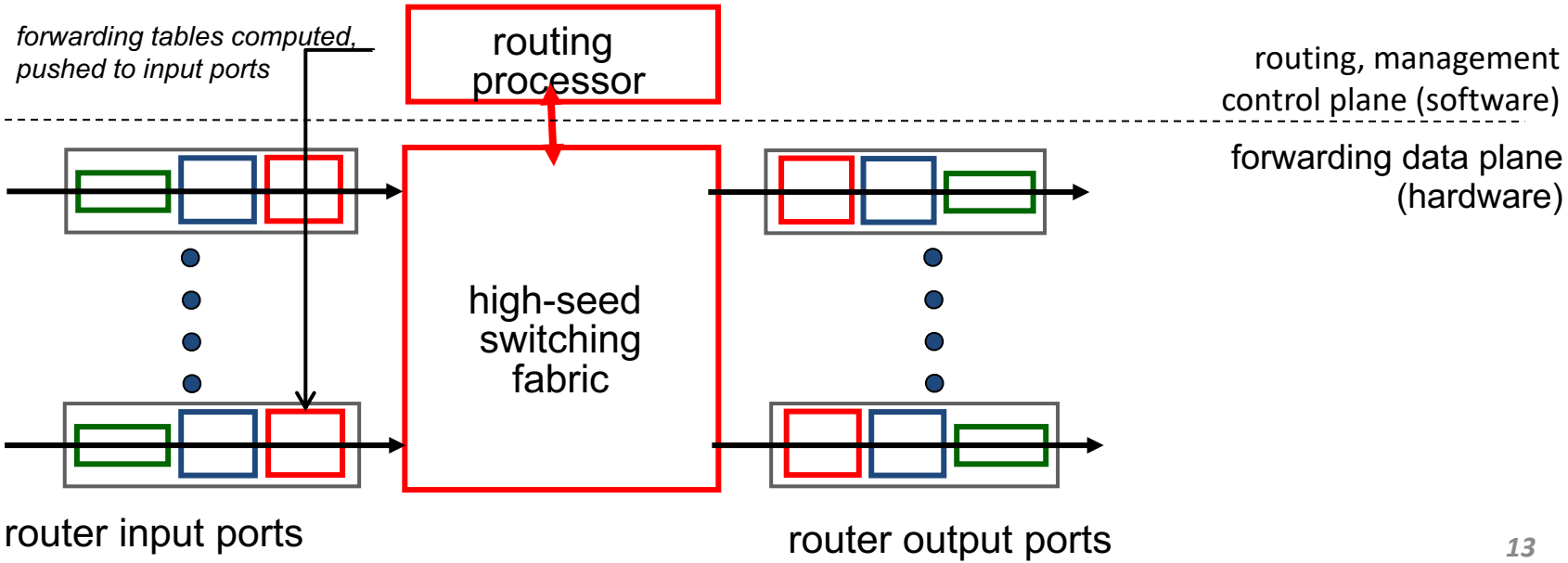


Datagram Forwarding Table

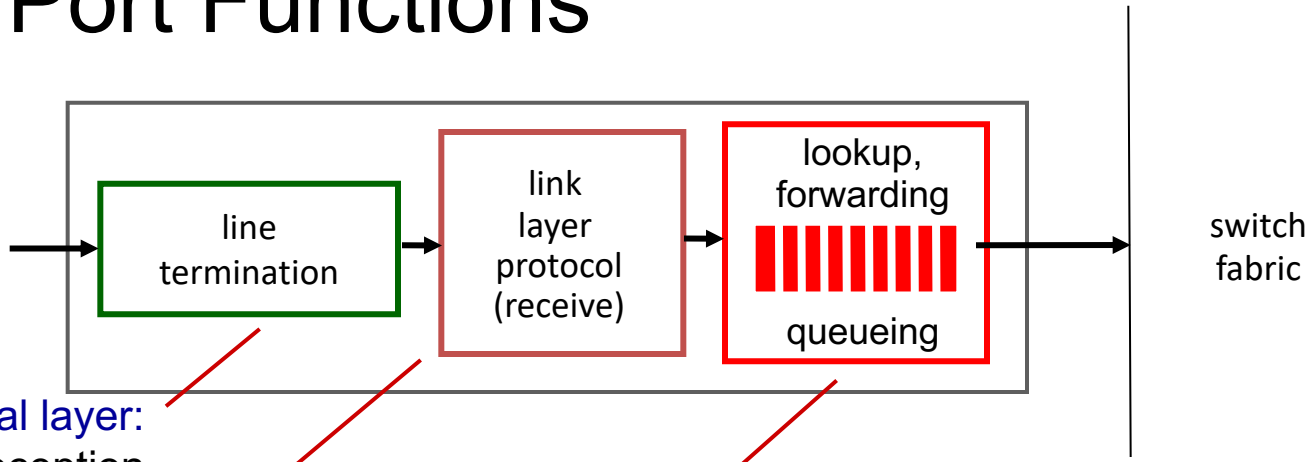
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Router Architecture Overview

- Two key router functions:
  - Run routing algorithms/protocol (RIP, OSPF, BGP)
  - Forwarding datagrams from incoming to outgoing link



Input Port Functions



Physical layer:
bit-level reception

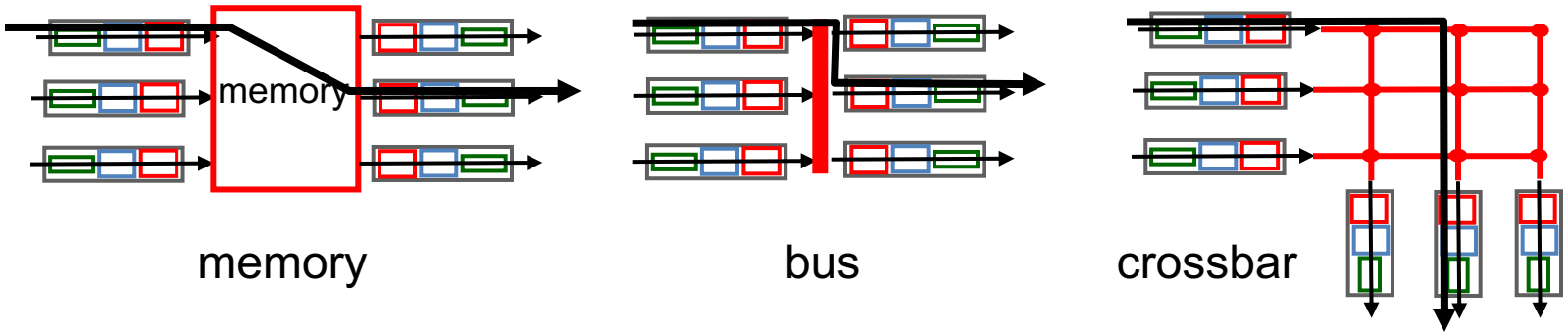
data link layer:
e.g., Ethernet

Switching:

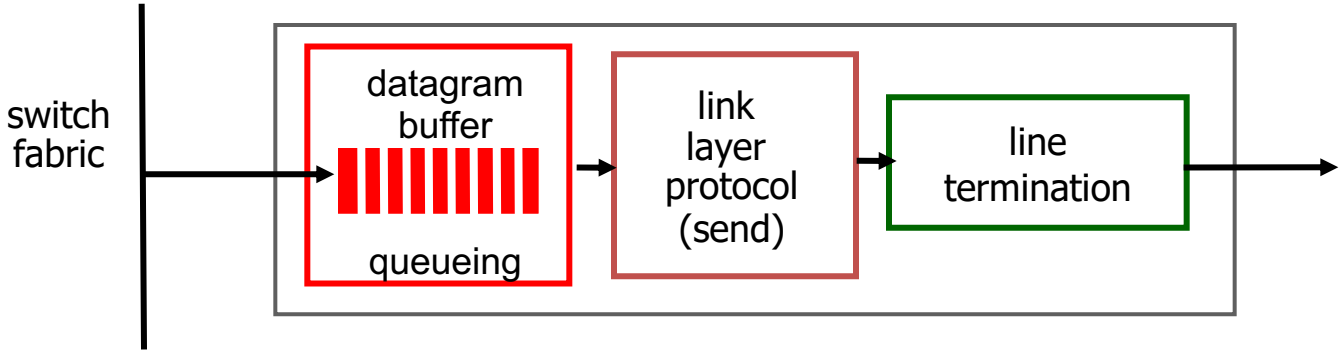
- given datagram destination., lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Switching Fabrics

- Transfer packet from input buffer to appropriate output buffer
- Switching rate: rate at which packets can be transfer from inputs to outputs
  - Often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- Three types of switching fabrics

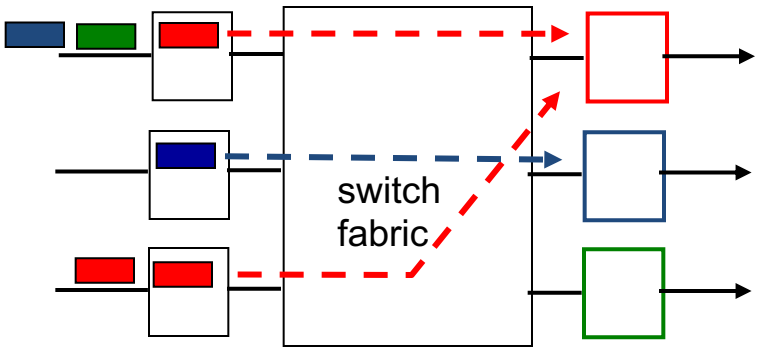


Output ports

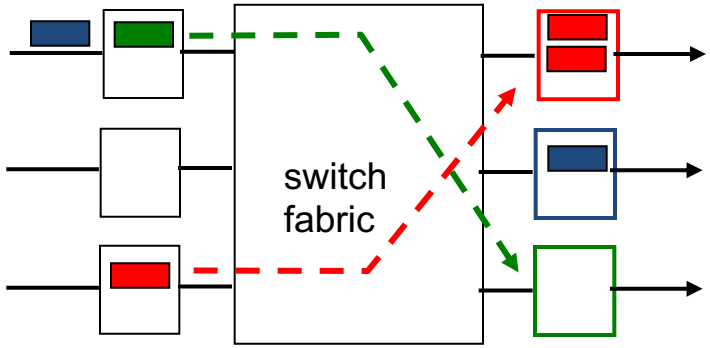


- Buffering required when datagrams arrive from fabric faster than the transmission rate
  - Datagram (packets) can be lost due to congestion, lack of buffers
- Scheduling discipline chooses among queued datagrams for transmission

Output Port Queueing



at *t*, packets more from input to output

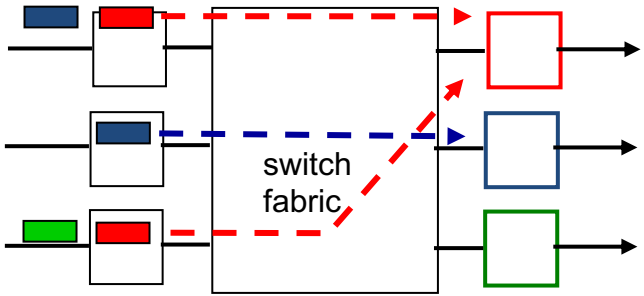


one packet time later

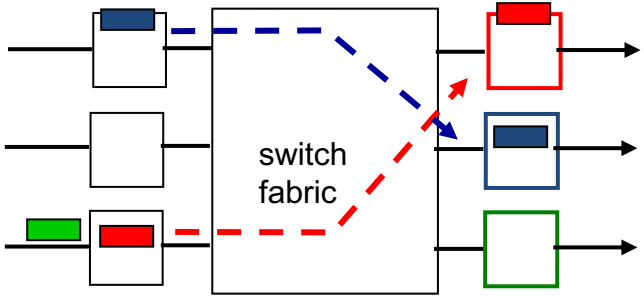
- Buffering when arrival rate via switch exceeds output line speed
- Queueing (delay) and loss due to output port buffer overflow

Input Port Queuing

- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



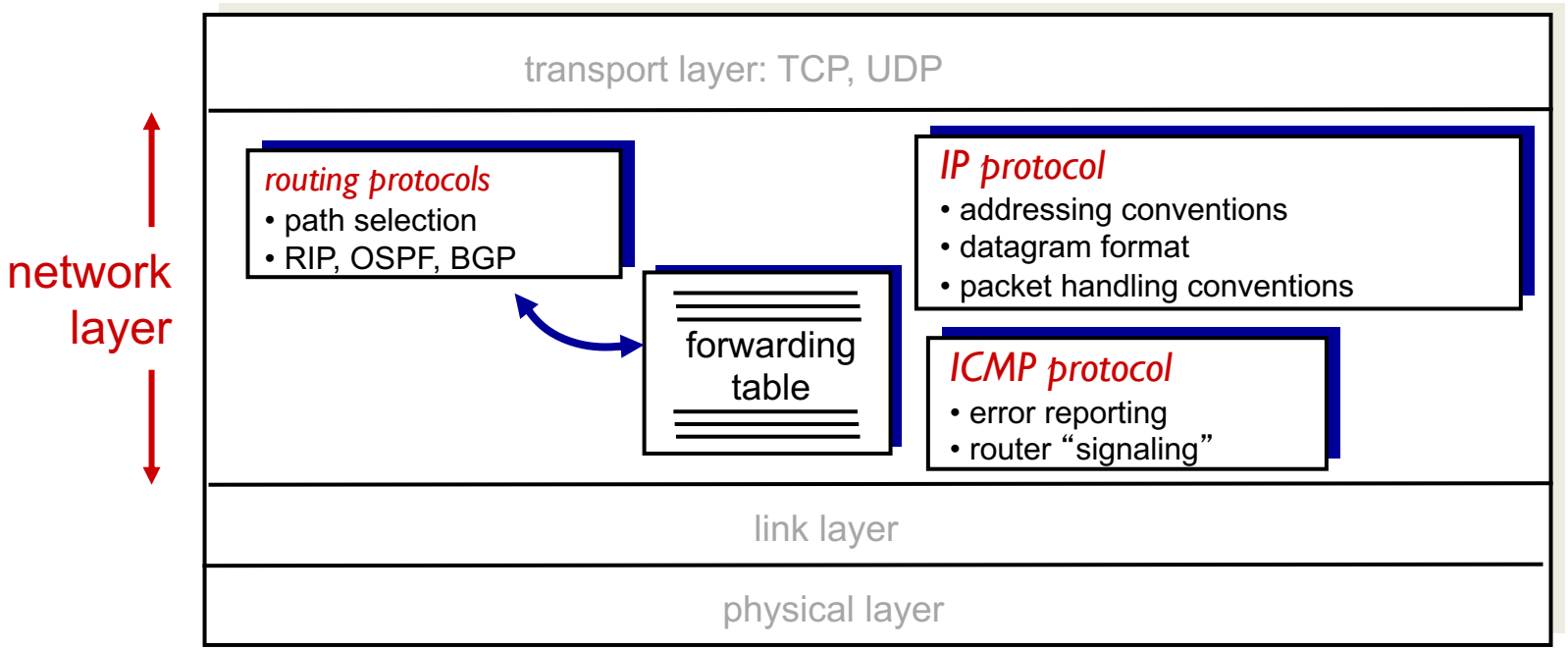
output port contention:
only one red datagram can be transferred.
lower red packet is blocked



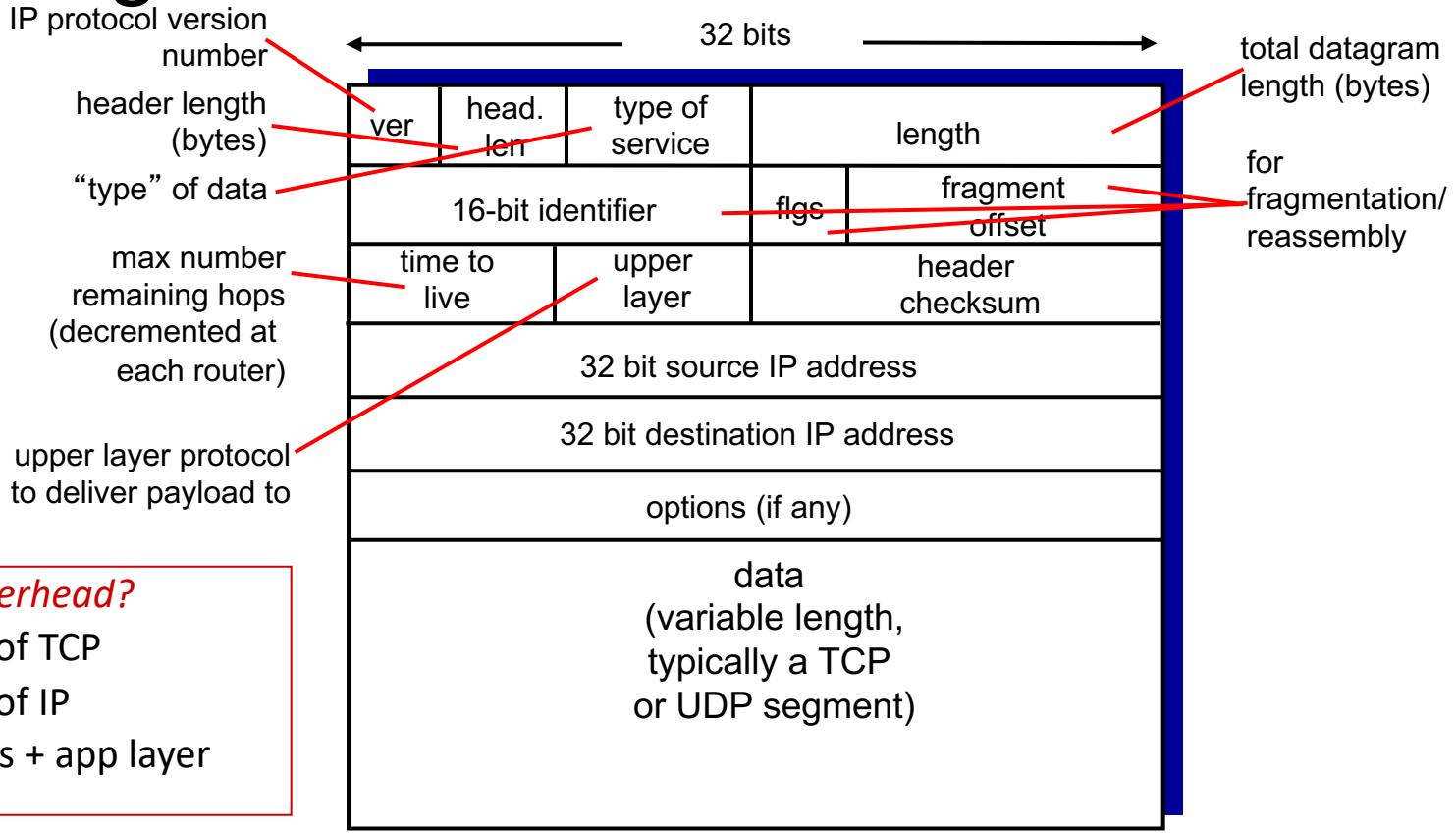
one packet time later: green packet experiences HOL blocking

The Internet Network Layer

- Host, router network layer functions:



IP Datagram Format



how much overhead?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

Recap Where We're At

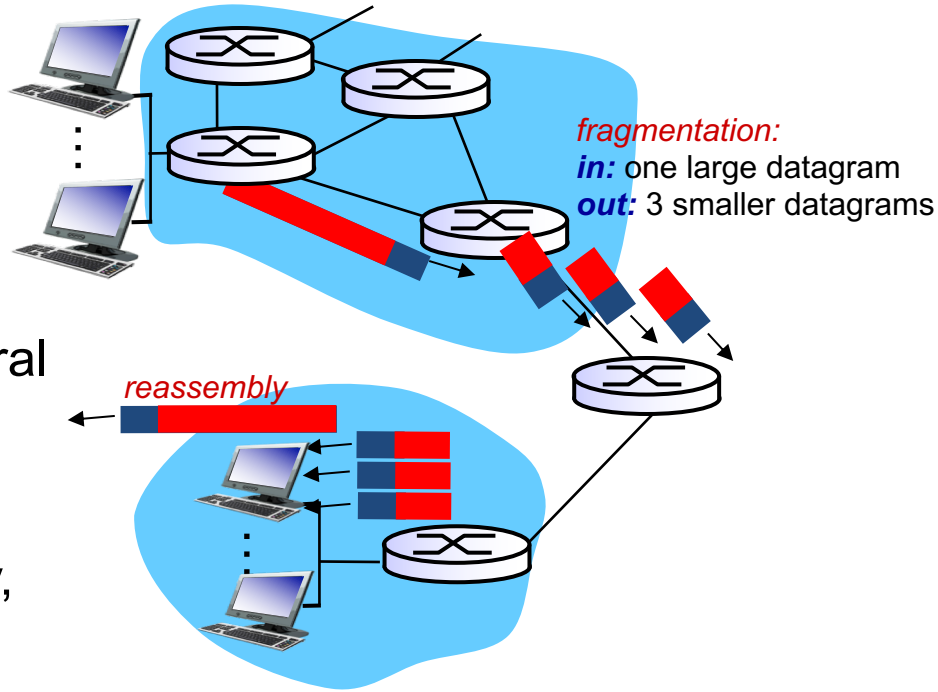
- Network layer
  - Forwarding
  - Router architecture
  - IP packet format

Outline

- Network layer
  - IP fragmentation
  - IP addressing
  - Longest prefix matching
  - DHCP
  - Hierarchical addressing

IP Fragmentation, Reassembly

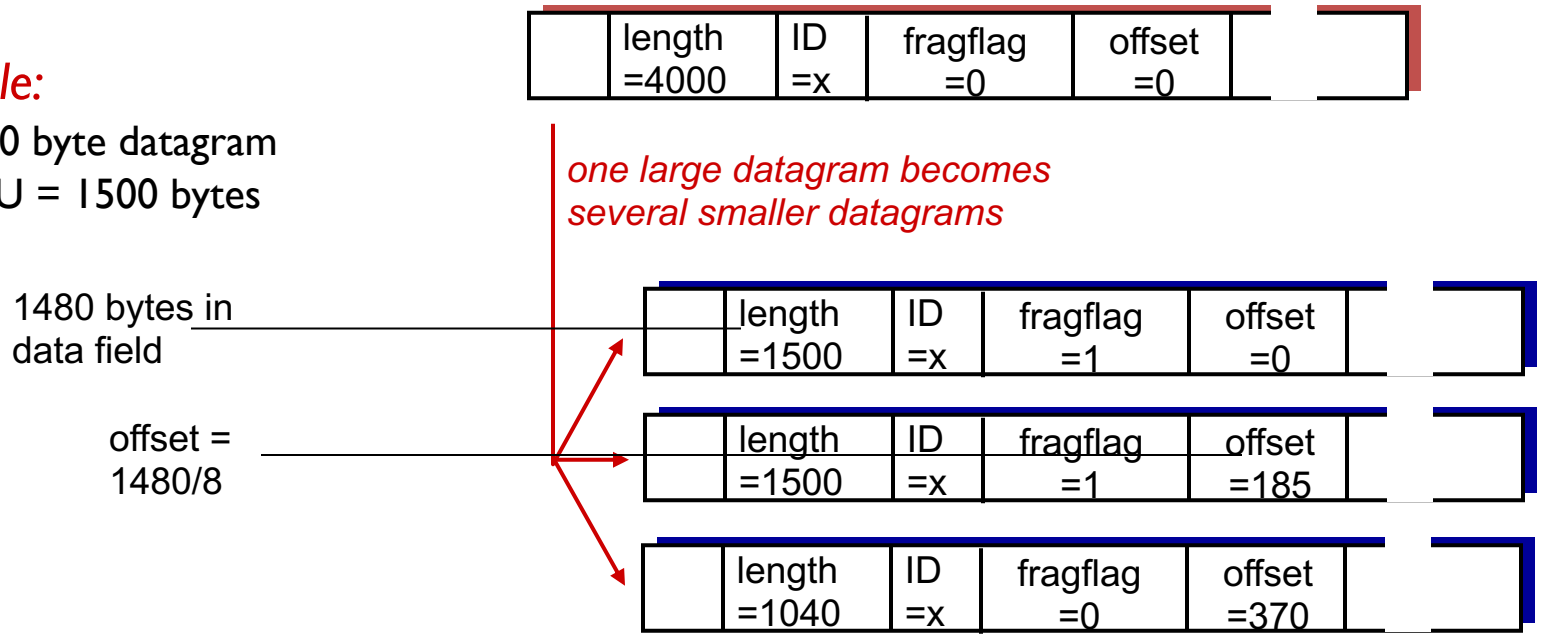
- Network links have MTU (max.transfer size) - largest possible link-level frame
  - Different link types, different MTUs
- Large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP Fragmentation, Reassembly

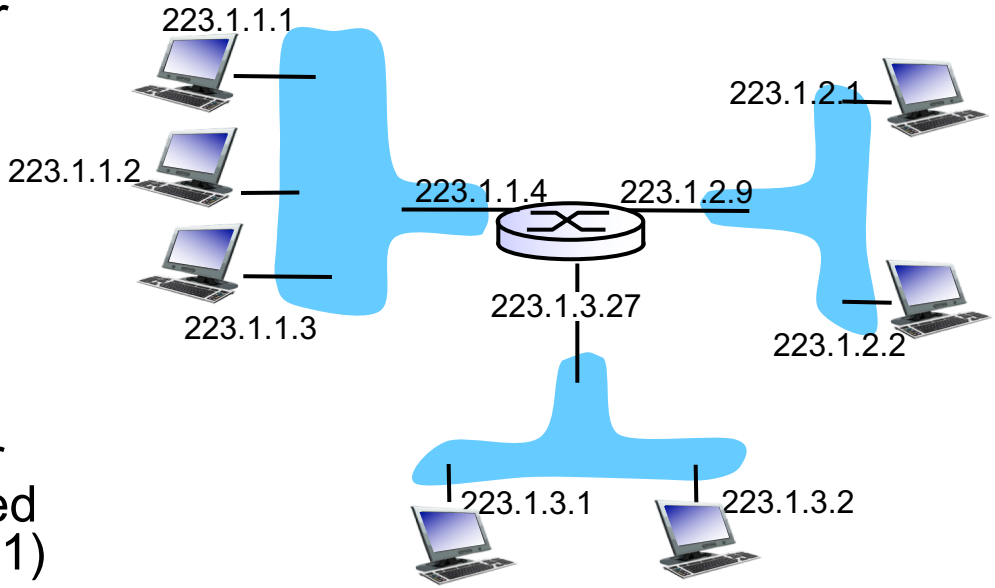
example:

- 4000 byte datagram
- MTU = 1500 bytes



IP Addressing: Introduction

- IP address: 32-bit identifier for host, router interface (dotted-decimal notation)
- Interface: connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- IP addresses associated with each interface



223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Classes

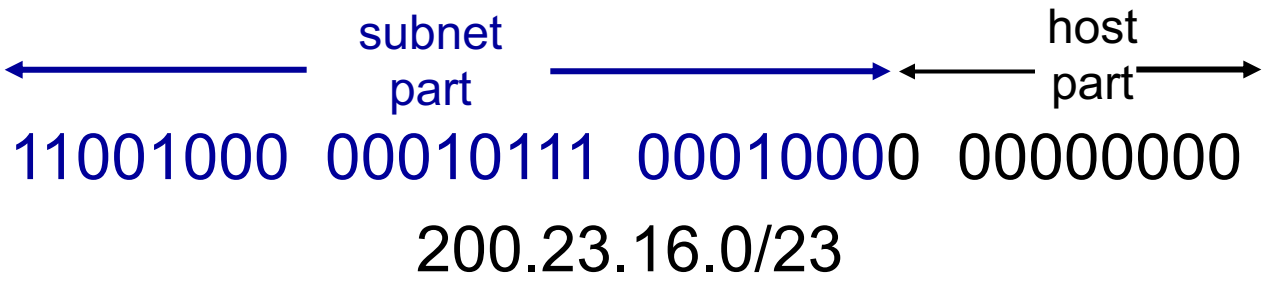
- Classes
  - Class A (0): netid=7 bit (128 networks), hostid=24 bit (16777216 hosts)
  - Class B (10): netid=14 bit (16384 networks), hostid=16 bit (65536 hosts)
  - Class C (110): netid=21 bit (2097152 networks), hostid=8 bit (256 hosts)
  - Class D - Multicast (1110): multicast addresses
  - Class E (1111): reserved or future use

Special Addresses

- As source and destination address
  - Loopback interface: 127.X.X.X (usually 127.0.0.1)
- As destination address
  - All bits set to 1: net-directed broadcast to netid
- Reserved addresses (RFC 1597):
  - 10.0.0.0 - 10.255.255.255
  - 172.16.0.0 - 172.31.255.255
  - 192.168.0.0 - 192.168.255.255

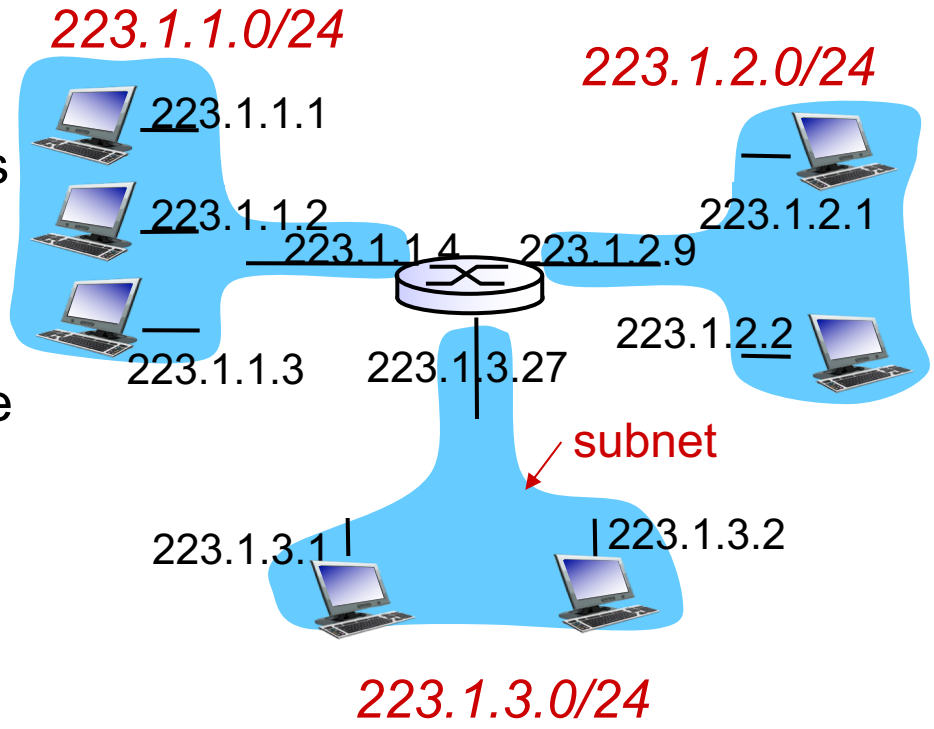
IP Addressing: CIDR

- CIDR: Classless InterDomain Routing
  - subnet portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in subnet portion of address



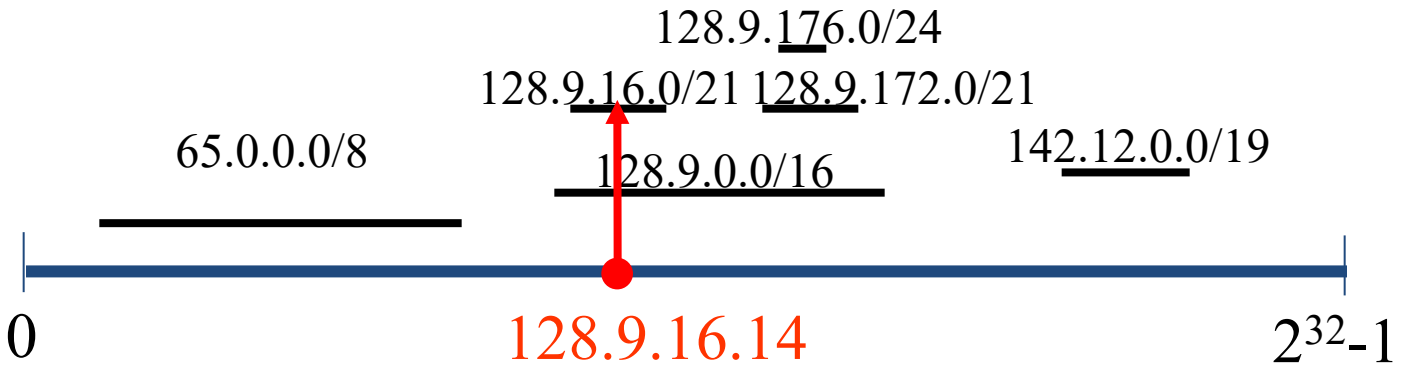
Subnets

- IP address:
  - subnet part - high order bits
  - host part - low order bits
- what's a subnet ?
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router



subnet mask: /24

Longest Prefix Match



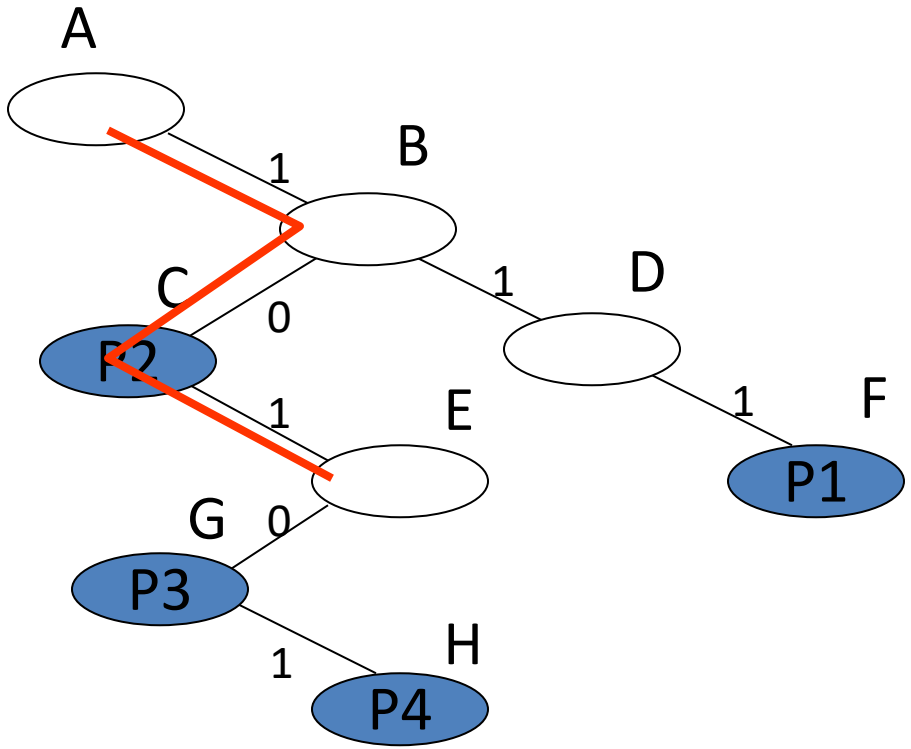
- Routing lookup: Find the longest matching prefix (aka the most specific route) among all prefixes that match the destination address.

Match with Trie

P1	111*	H1
P2	10*	H2
P3	1010*	H3
P4	10101	H4

- find best prefix, spell out address in tree
- last blue node marks longest matching prefix

Lookup 10111



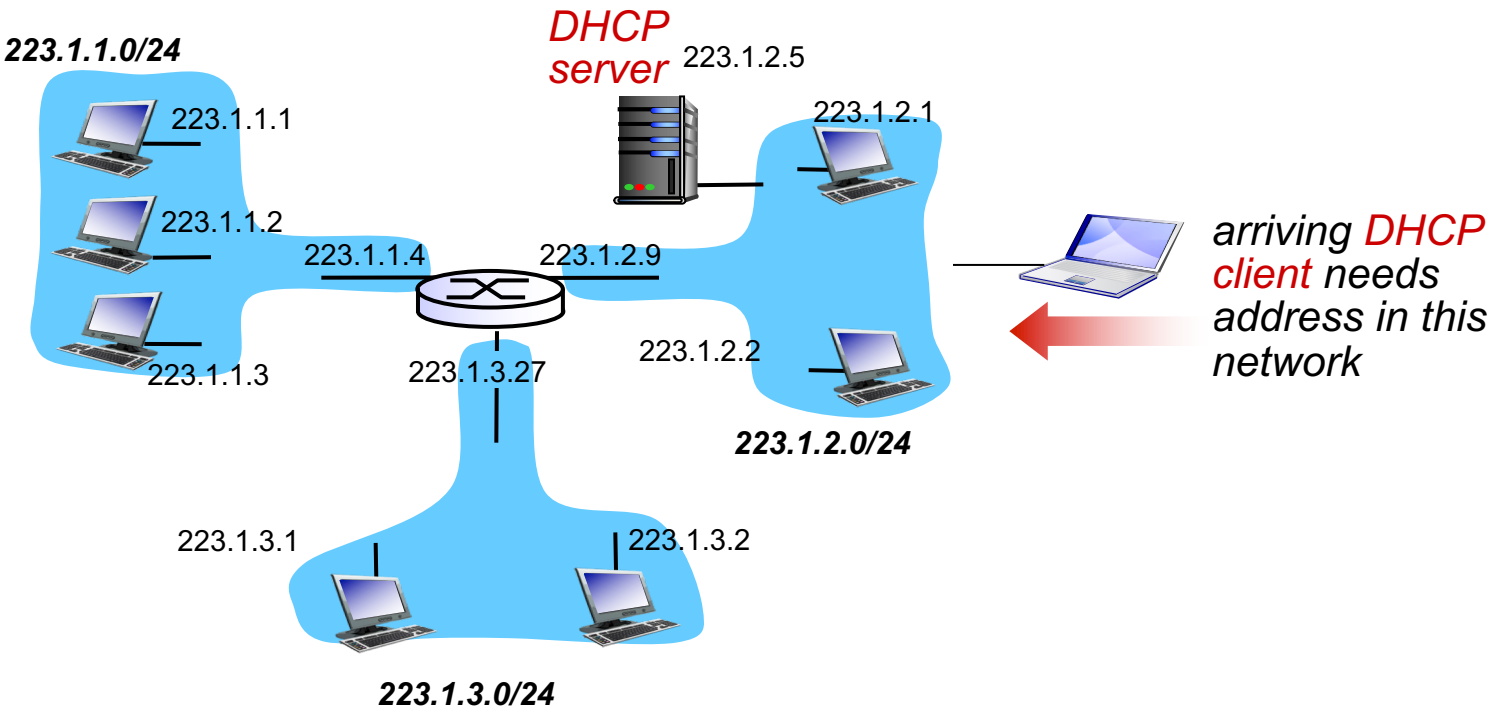
IP Addresses: How to Get One?

- 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: Dynamic Host Configuration Protocol: dynamically get address from as server
    - “plug-and-play”

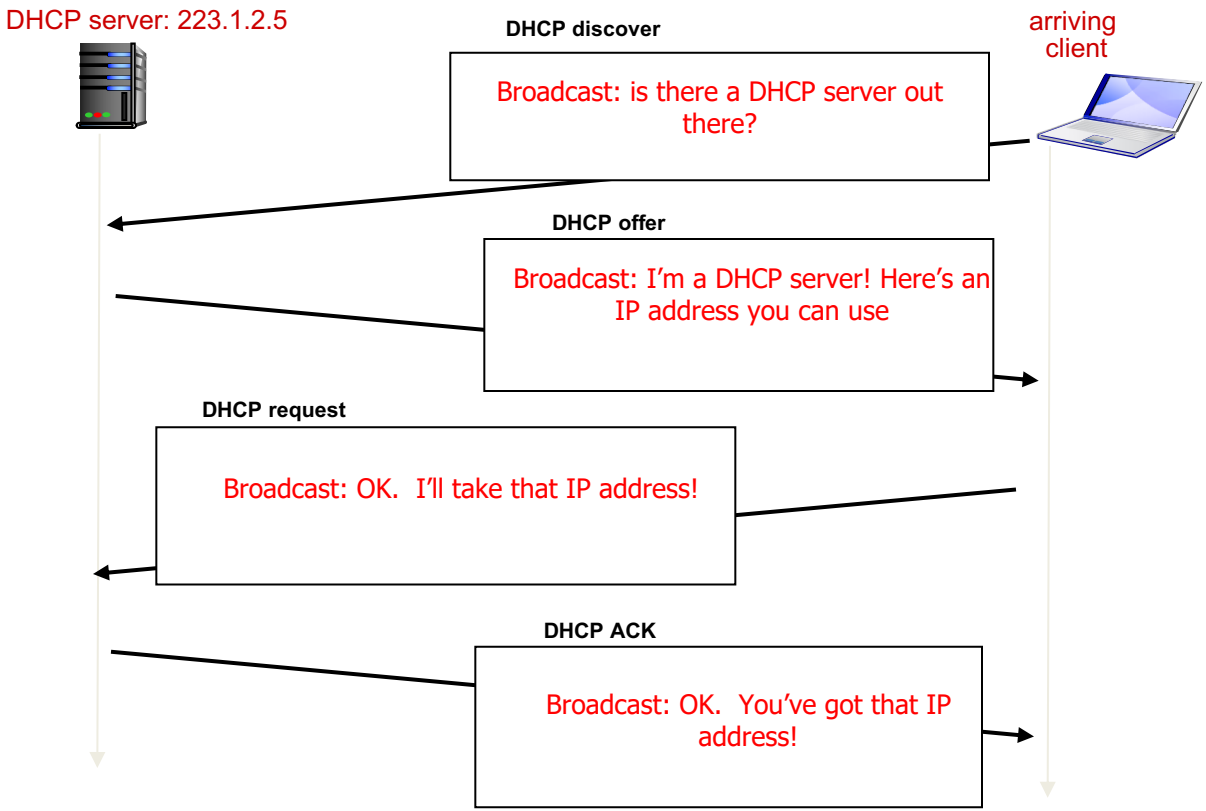
DHCP: Dynamic Host Configuration Protocol

- 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”)
- DHCP overview:
  - Host broadcasts “DHCP discover” msg
  - DHCP server responds with “DHCP offer” msg
  - Host requests IP address: “DHCP request” msg
  - DHCP server sends address: “DHCP ack” msg

DHCP Client-server Scenario



DHCP Client-server Scenario



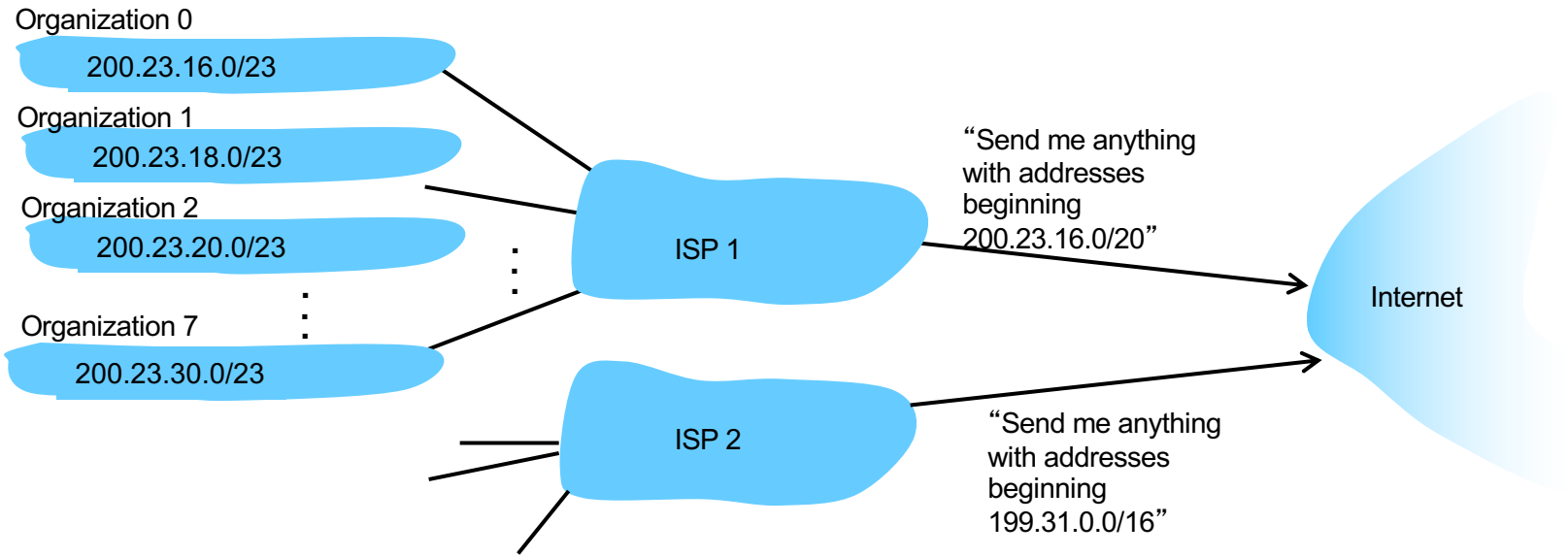
IP Addresses: How to Get One?

- How does network get subnet part of IP addresses?
  - 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 0001**000**0	00000000	200.23.16.0/23
Organization 1	11001000 00010111 0001**001**0	00000000	200.23.18.0/23
Organization 2	11001000 00010111 0001**010**0	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 0001**111**0	00000000	200.23.30.0/23

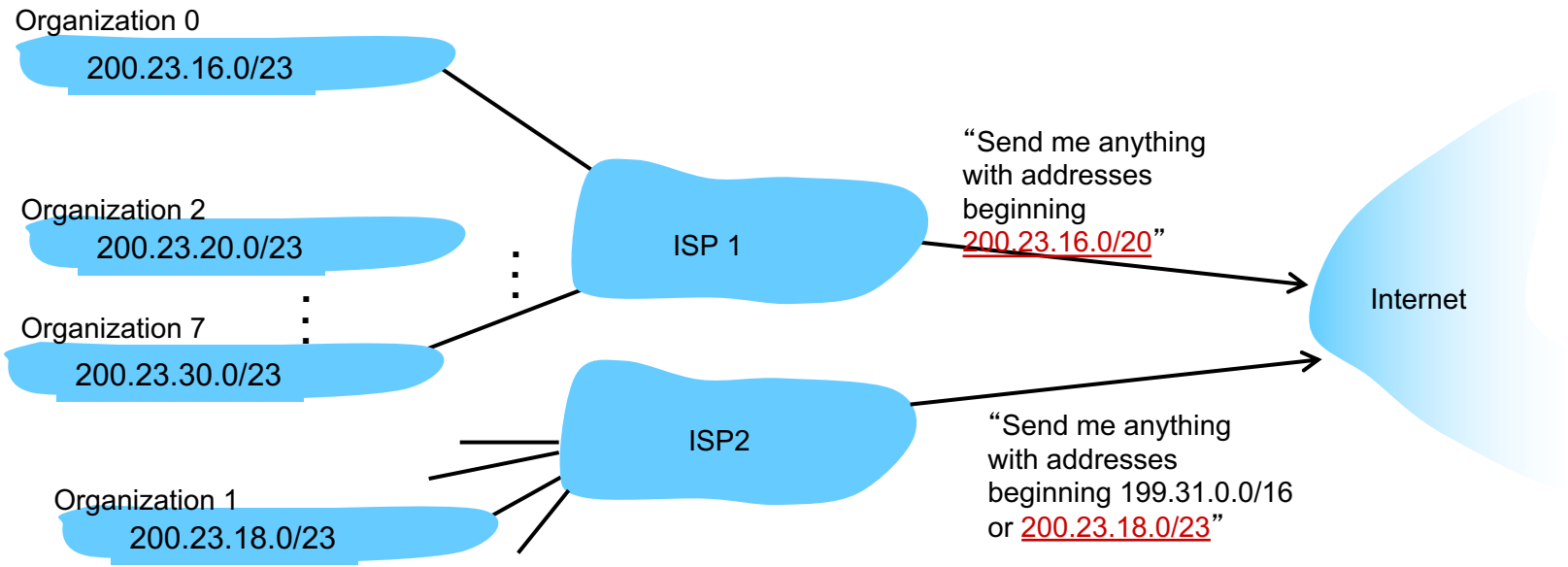
Hierarchical Addressing: Route Aggregation

- Hierarchical addressing allows efficient advertisement of routing information



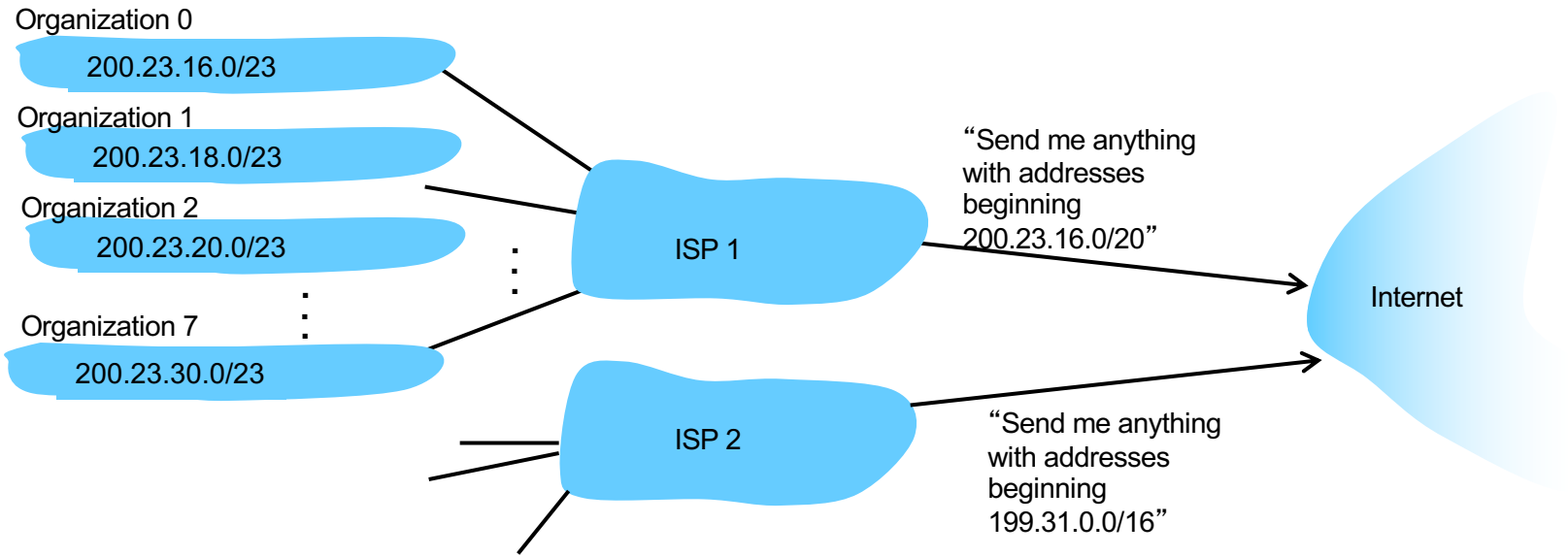
Hierarchical Addressing: More Specific Routes

- ISP 2 has a more specific route to Organization 1



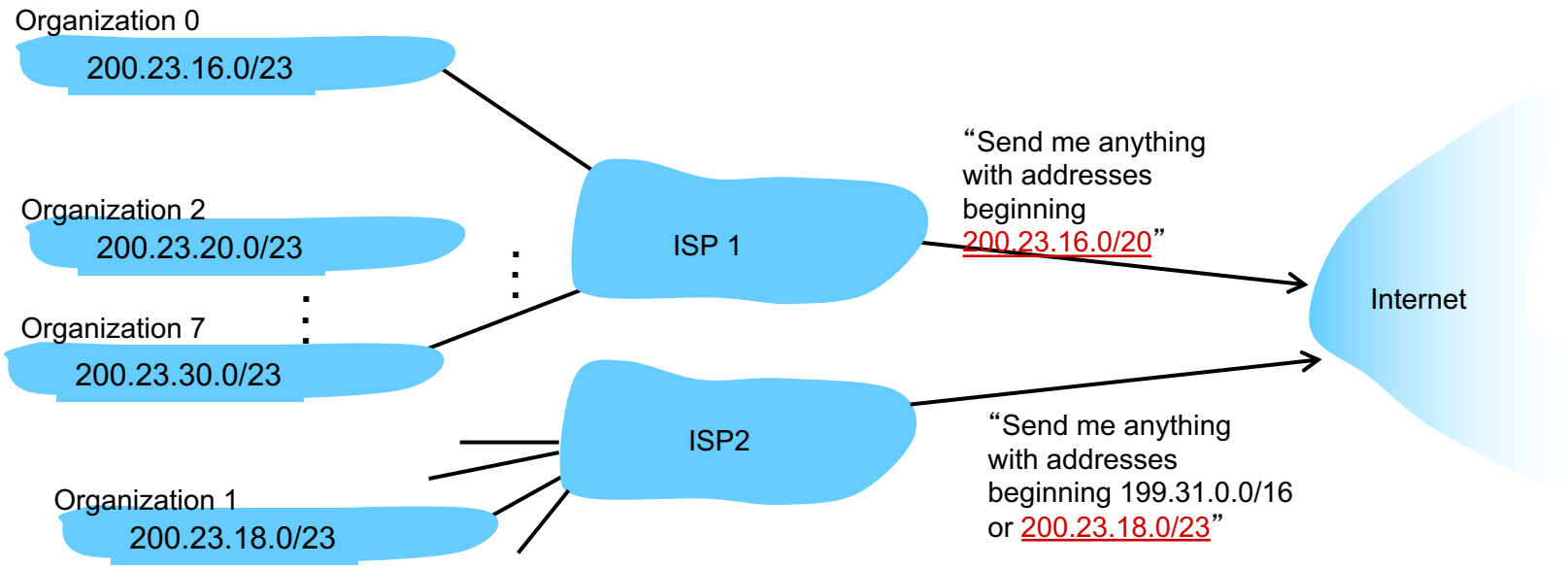
Hierarchical Addressing: Route Aggregation

- Hierarchical addressing allows efficient advertisement of routing information



Hierarchical Addressing: More Specific Routes

- ISP 2 has a more specific route to Organization 1



IP Addresses: How to Get A Block?

- Q: How does an ISP get block of addresses?
  - ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/
    - allocates addresses
    - manages DNS
    - assigns domain names, resolves disputes

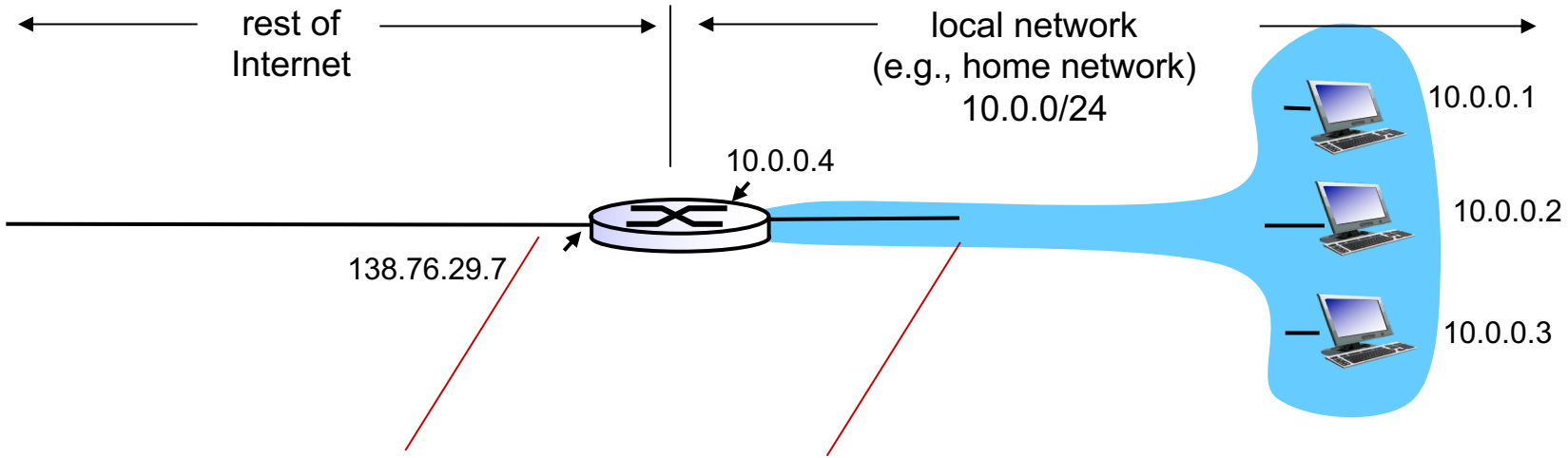
Recap Where We're At

- Network layer
  - IP fragmentation
  - IP addressing
  - Longest prefix matching
  - DHCP
  - Hierarchical addressing

Outline

- Network layer
  - NAT (Network Address Translation)
  - ICMP
  - IPv6
  - Routing algorithms

NAT: Network Address Translation



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

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

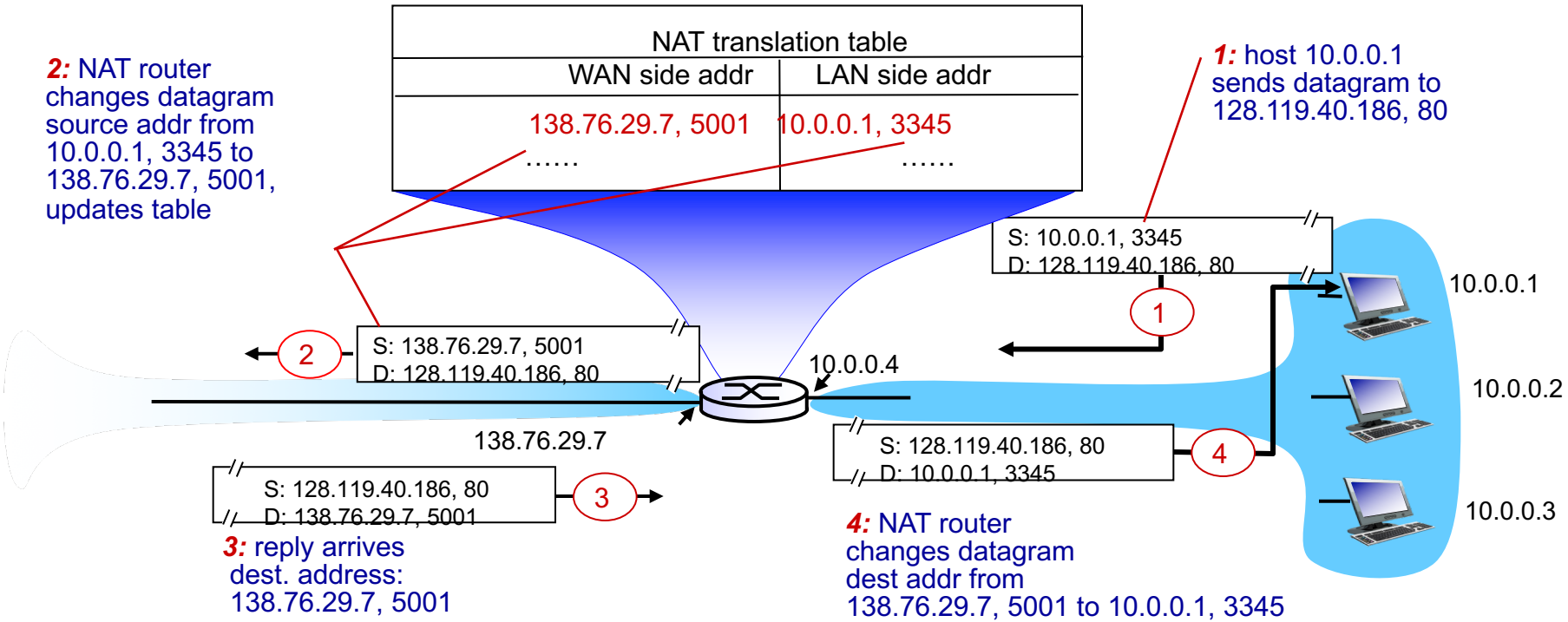
NAT: Network Address Translation

- 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 network 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 #) of every outgoing datagram to (NAT IP address, new port #)
    - . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr
  - remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
  - incoming datagrams: replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



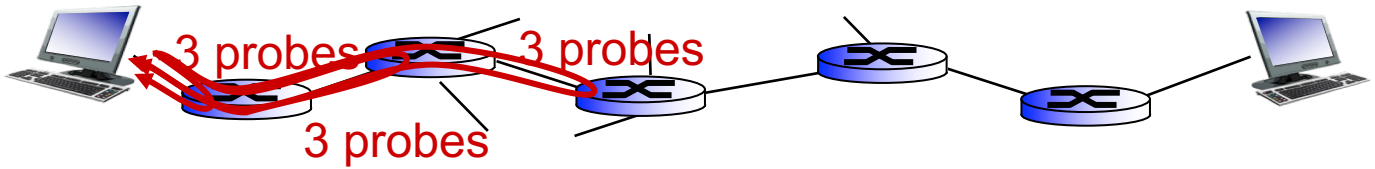
ICMP: Internet Control Message protocol

- 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 plus 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 dest
  - first set has TTL =1
  - second set has TTL=2, etc.
- When nth set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
  - when ICMP messages arrives, source records RTTs



IPv6: Motivation

- Initial motivation: 32-bit address space is completely allocated.
  - Other technical benefits (format not compatible with IPv4)

- IPv6 packet format
  - Fixed-length 40 byte header
  - 128 bits of IPv6 addresses

IPv6 Address

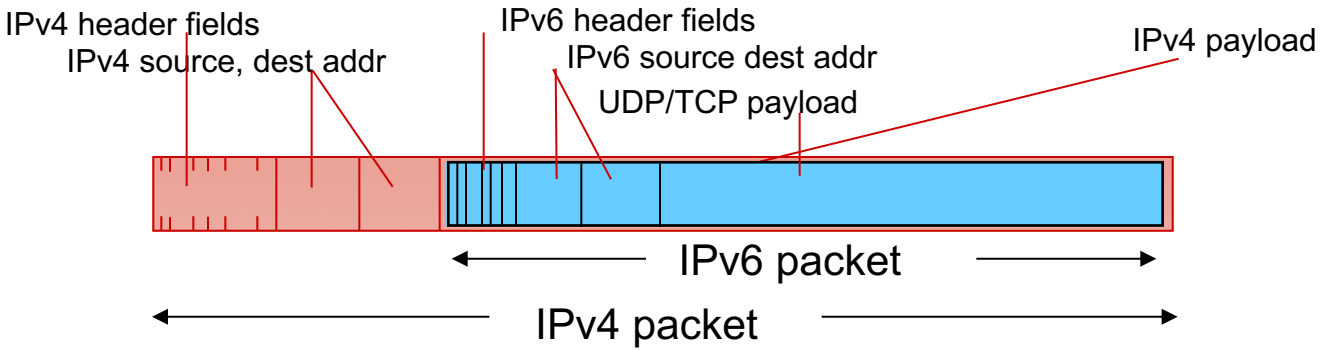
- IPv6 address representation
  - Full, 32 nibbles
    2001:0db8:0000:0000:0000:0000:0000:0001
  - Short, zeroes omitted
    2001:db8::1

Transition from IPv4 to IPv6

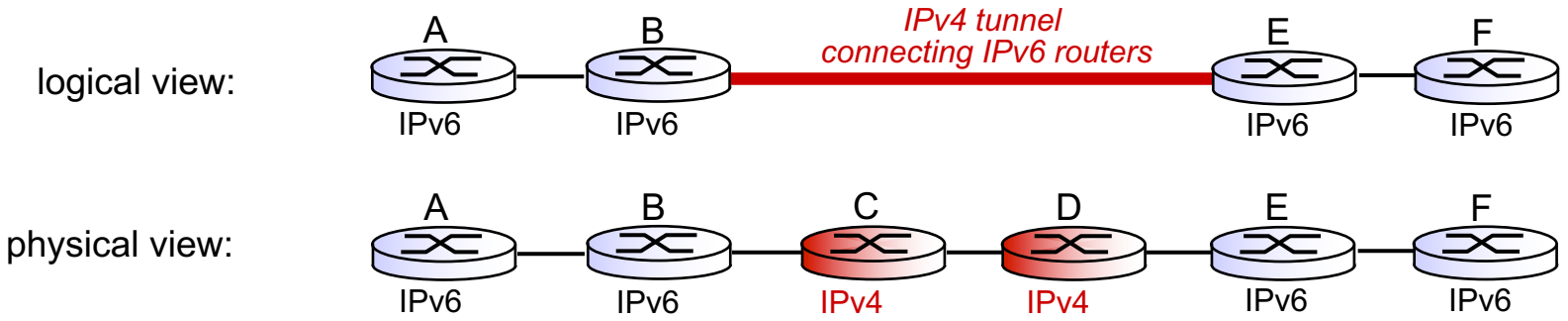
- Not all routers can be upgraded simultaneously
  - No “flag days”
  - Network operate with mixed IPv4 and IPv6 routers
- Tunneling: IPv6 packet carried as payload in IPv4 packet among IPv4 routers

Transition from IPv4 to IPv6

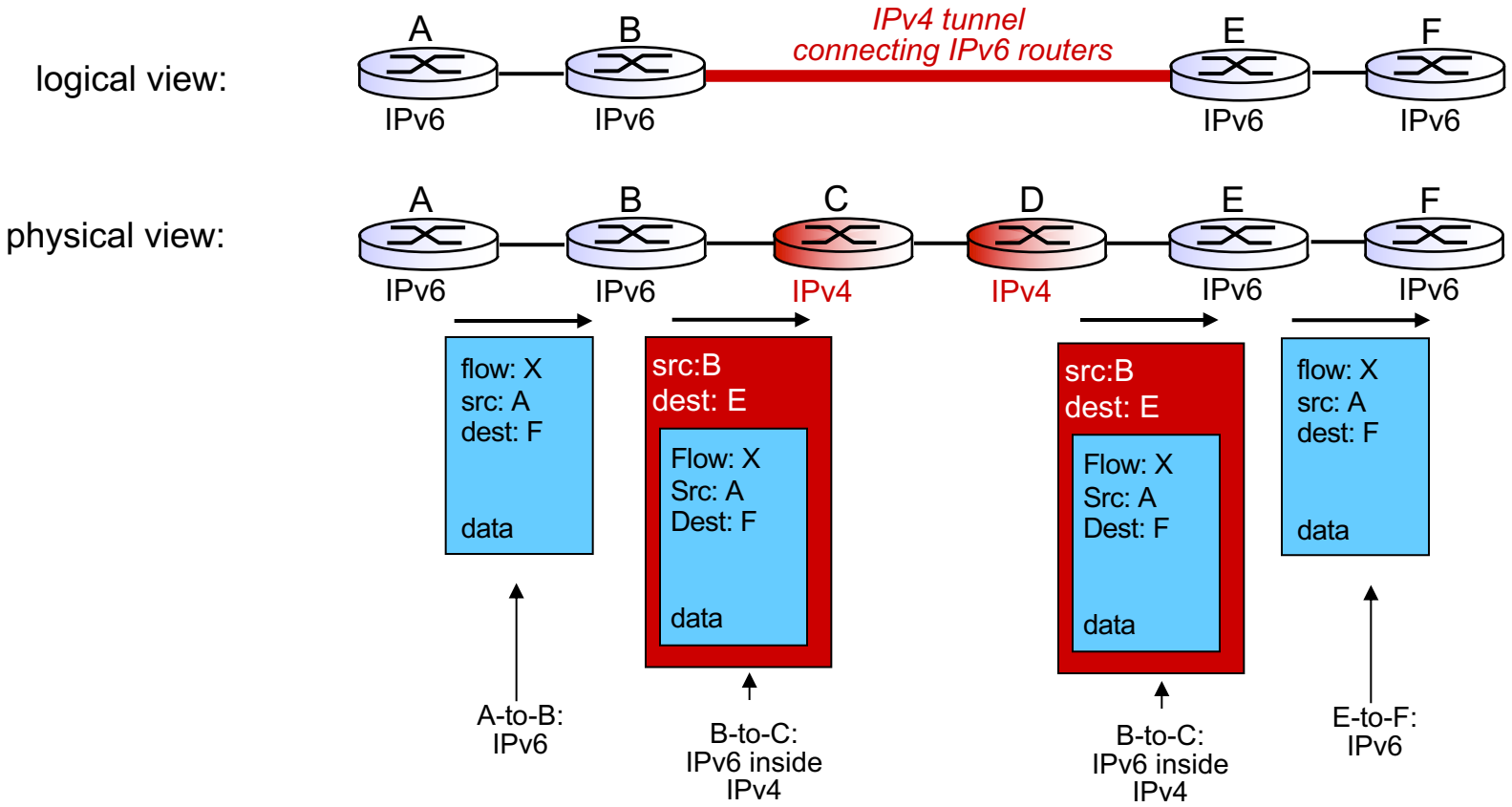
- Not all routers can be upgraded simultaneously
  - No “flag days”
  - Network operate with mixed IPv4 and IPv6 routers
- Tunneling: IPv6 packet carried as payload in IPv4 packet among IPv4 routers



Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling

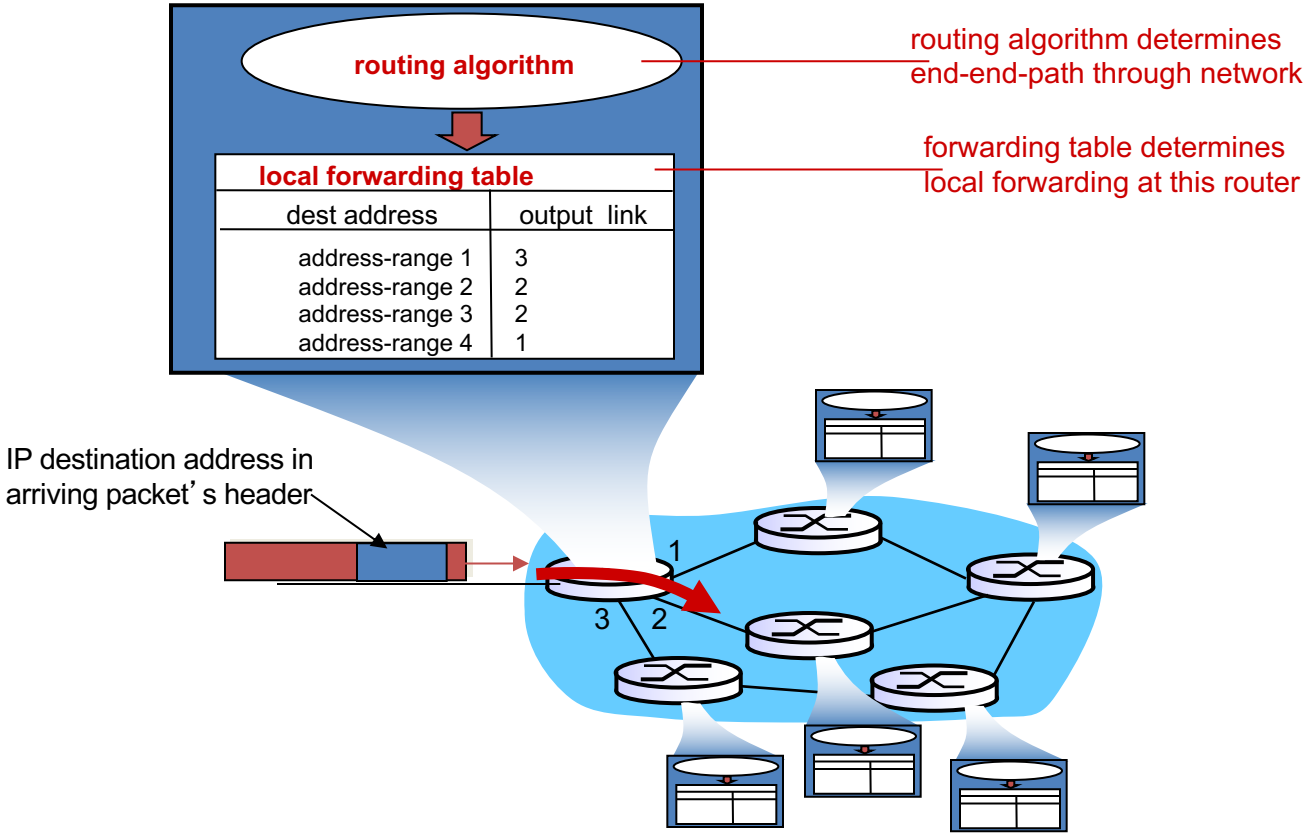

logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

IPv6 Adoption

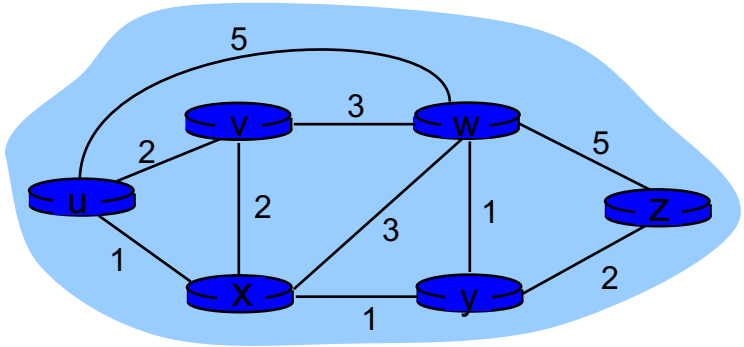
- Industry adoption
  - 40% traffic to Google is from IPv6 (2022)
  - Amazon EC2 provides IPv6 addresses

- Long time for deployment
  - 20 years and counting

Interplay between Routing, Forwarding



Graph Abstraction

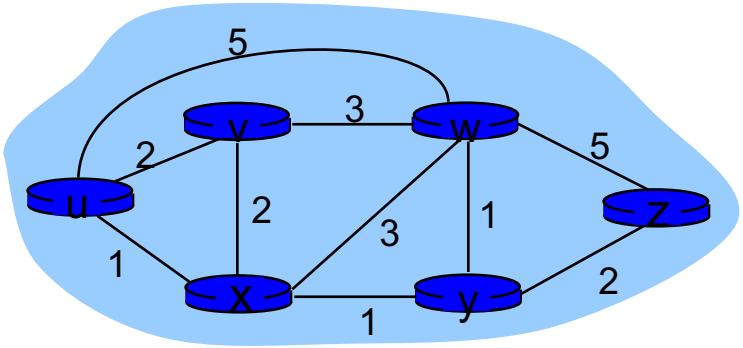


graph: G = (N,E)

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) }

Graph Abstraction: Costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

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 Algorithm Classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- "link state" algorithms

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Recap Where We're At

- Network layer
  - NAT (Network Address Translation)
  - ICMP
  - IPv6
  - Routing algorithms

Outline

- Routing algorithms
  - Link-state algorithm
  - Distance vector algorithms

A Link-State Routing Algorithm

Dijkstra's algorithm

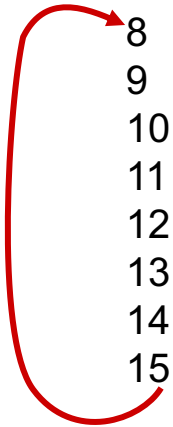
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - Provides *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.' s

notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

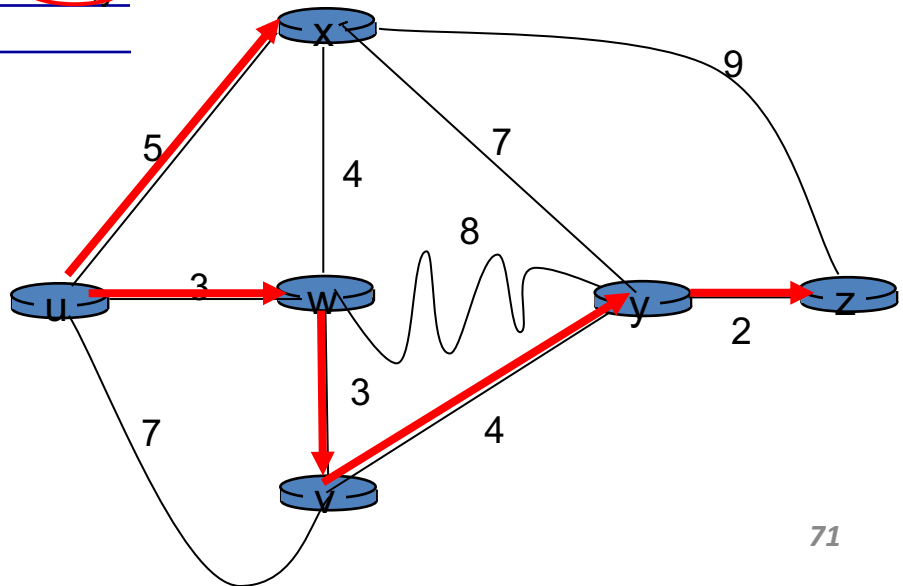


Dijkstra's Algorithm: Example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



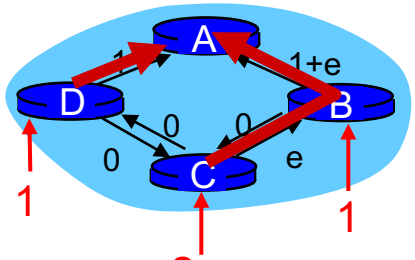
Dijkstra's Algorithm, Discussion

algorithm complexity: n nodes

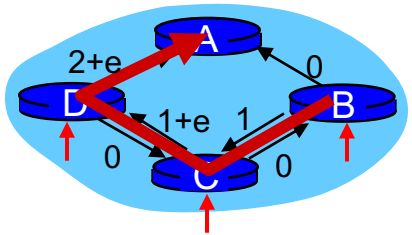
- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$

oscillations possible:

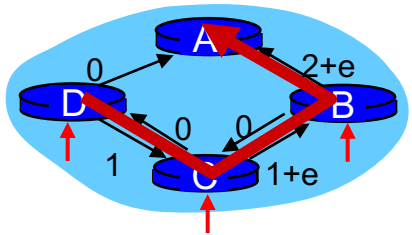
- e.g., suppose link cost equals amount of carried traffic:



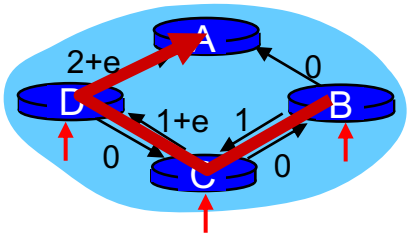
initially



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs

Distance Vector Algorithm

Bellman-Ford equation (dynamic programming)

let

$d_x(y)$:= cost of least-cost path from x to y

then

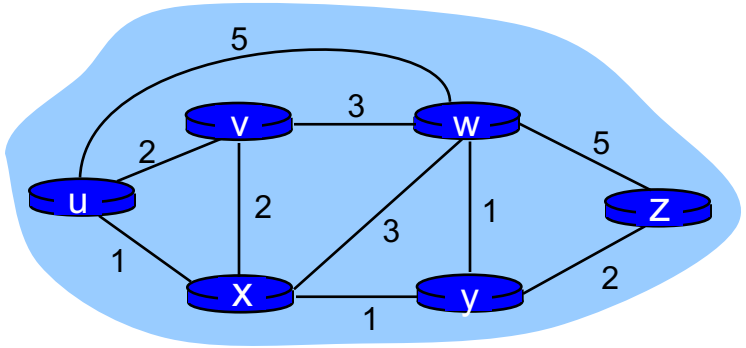
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford Example



clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

node achieving minimum is next hop in shortest path, used in forwarding table

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains
    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance Vector Algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm

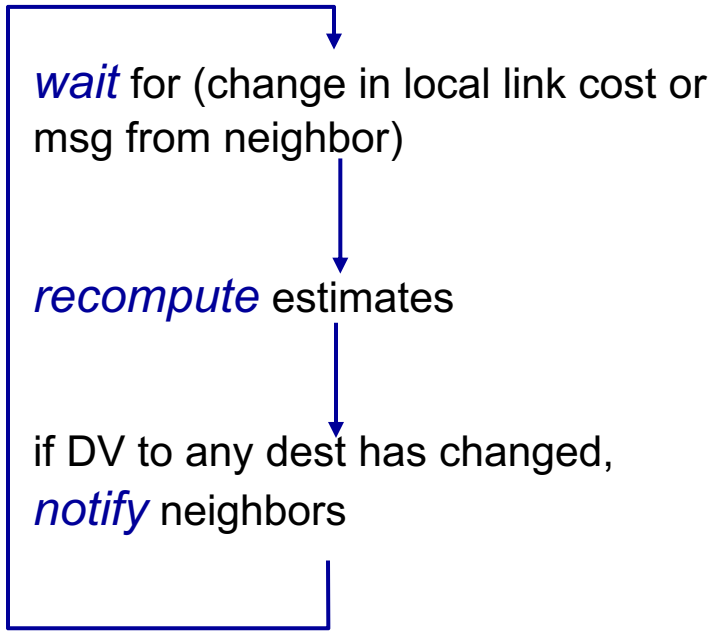
iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



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

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

node x table

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

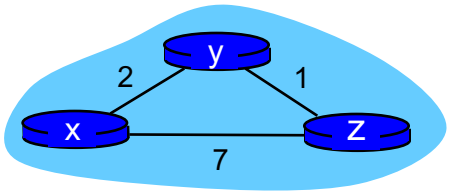
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

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

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

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

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

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

node y table

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

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

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

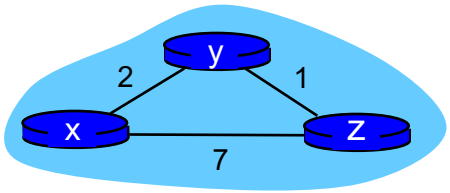
node z table

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

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

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

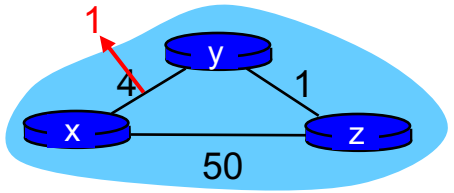
time



Distance Vector: Link Cost Changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

Recap Where We're At

- Routing algorithms
  - Link-state algorithm
  - Distance vector algorithms

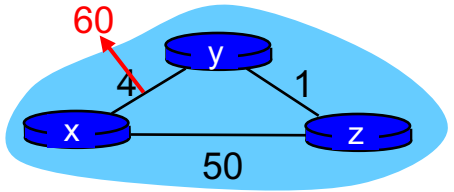
Outline

- Comparison between LS and DV
- Inter-AS and intra-AS routing
- RIP, OSPF
- BGP

Distance Vector: Link Cost Changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 46 iterations before algorithm stabilizes



poisoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)

Comparison of LS and DV Algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Hierarchical Routing

- Our routing study thus far - idealization
  - all routers identical
  - network “flat”
  - … not true in practice

scale: with 400 million destinations:

- can’t store all dest’s in routing tables!
- routing table exchange would swamp links!

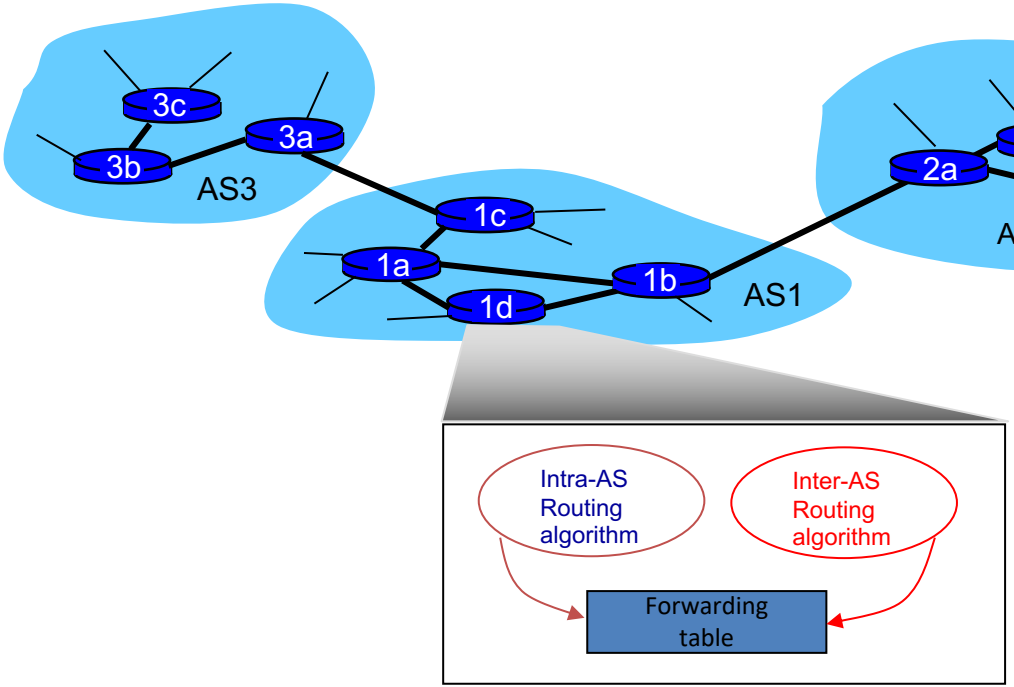
administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

Hierarchical Routing

- Aggregate routers into regions, “autonomous systems” (AS)
- Routers in same AS run same routing protocol
  - “intra-AS” routing protocol
  - routers in different AS can run different intra-AS routing protocol
- Gateway router:
  - at “edge” of its own AS
  - has link to router in another AS

Interconnected ASes



- Forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests

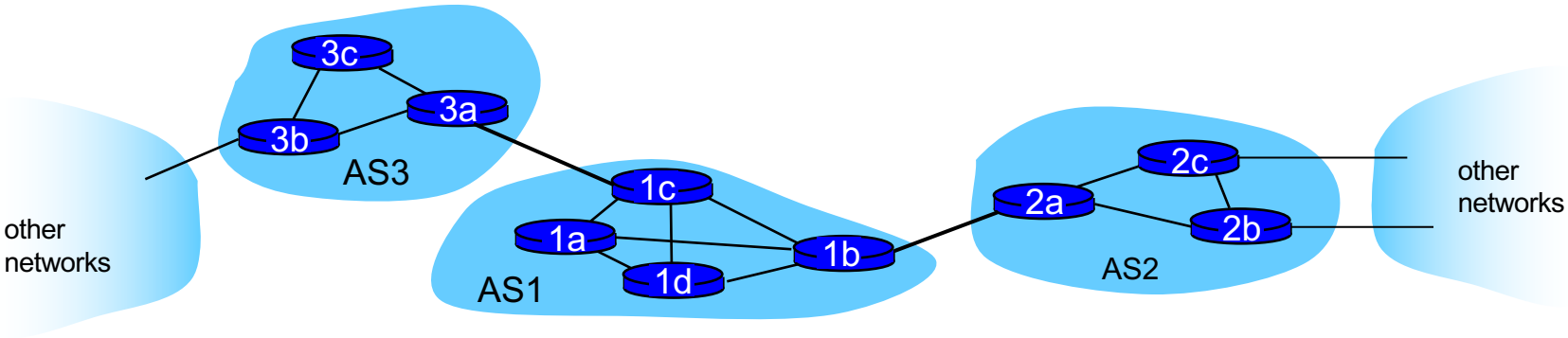
Inter-AS Tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

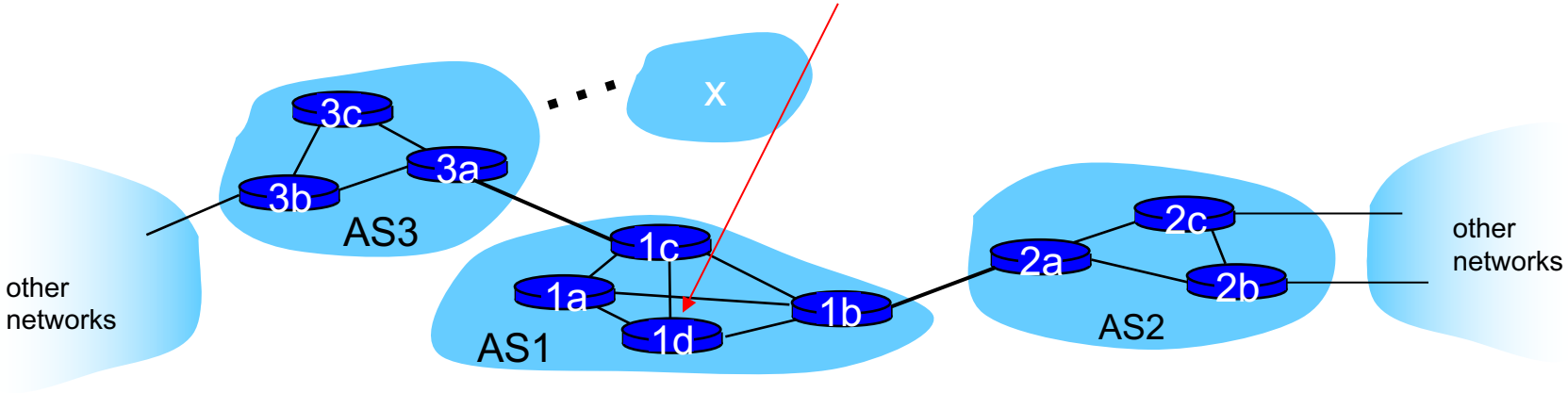
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing



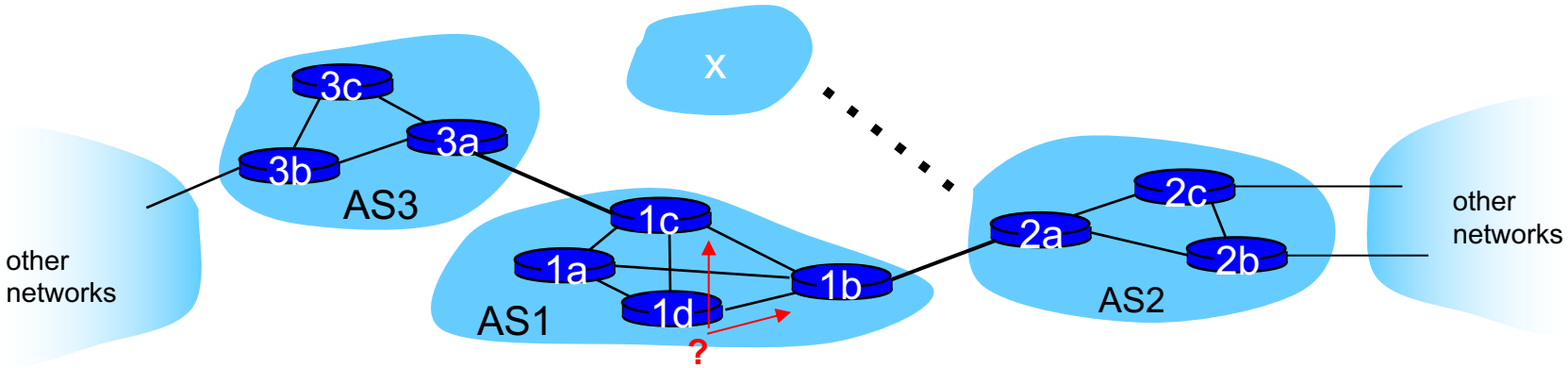
Example: Setting Forwarding Table in Router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet x reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- Router 1d determines from intra-AS routing info that its interface I is on the least cost path to 1c
  - installs forwarding table entry (x,I)



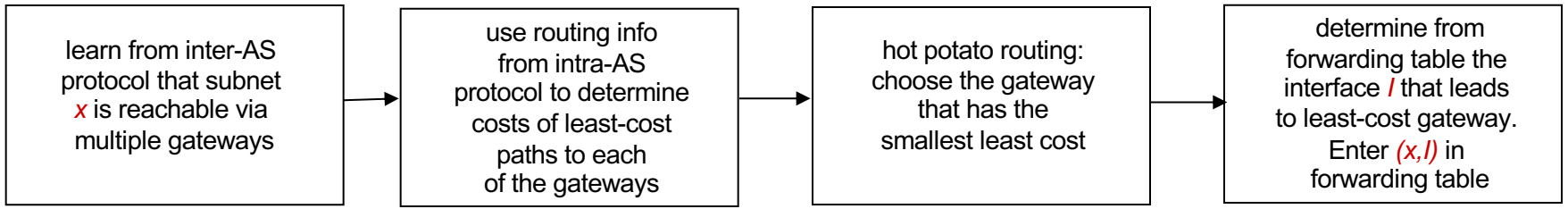
Example: Choosing among Multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet x is reachable from AS3 and from AS2.
- to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - this is also job of inter-AS routing protocol



Example: Choosing among Multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet x is reachable from AS3 and from AS2.
- to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - this is also job of inter-AS routing protocol
- Hot potato routing: send packet towards closest of two routers.

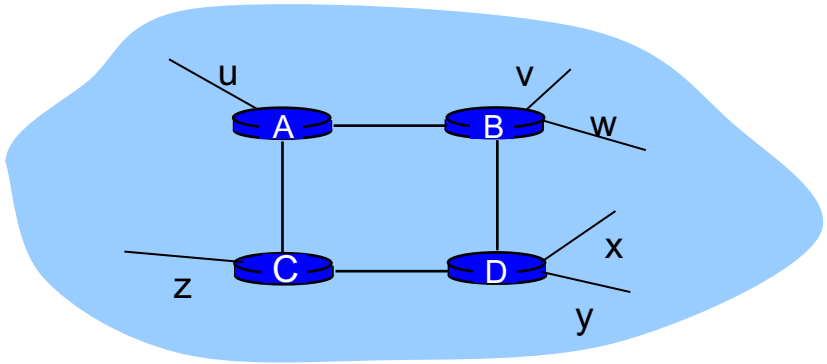


Intra-AS Routing

- Also known as interior gateway protocols (IGP)
- Most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First

RIP (Routing Information Protocol)

- Distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors periodically in response message (aka advertisement)
  - each advertisement: list of multiple destination subnets (in IP addressing sense)

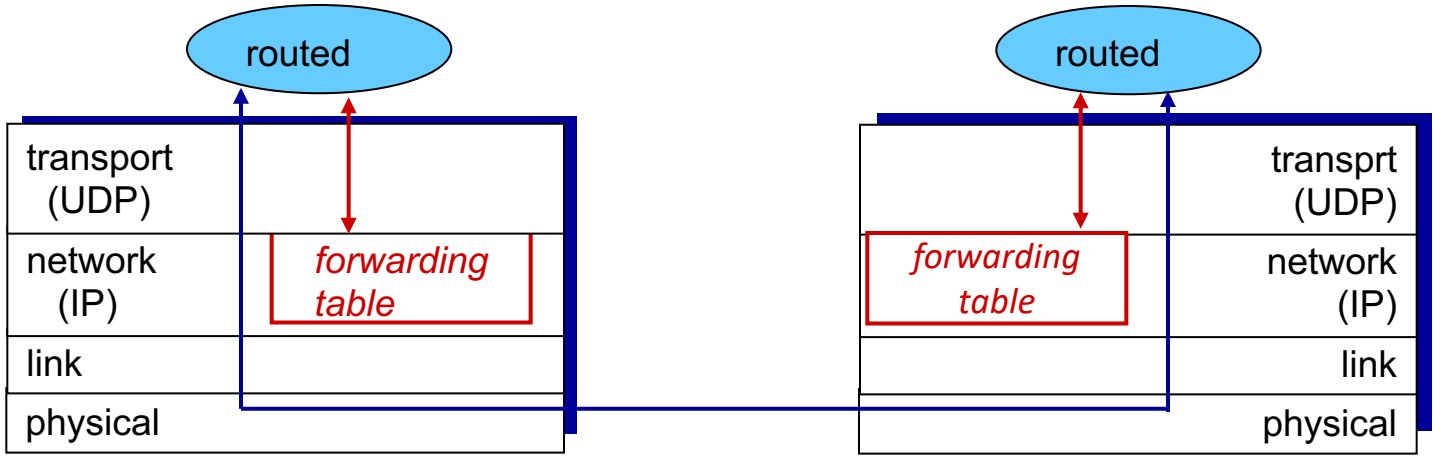


from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP Table Processing

- RIP routing tables managed by application-level process called route-d (daemon)
- Advertisements sent in UDP packets, periodically repeated

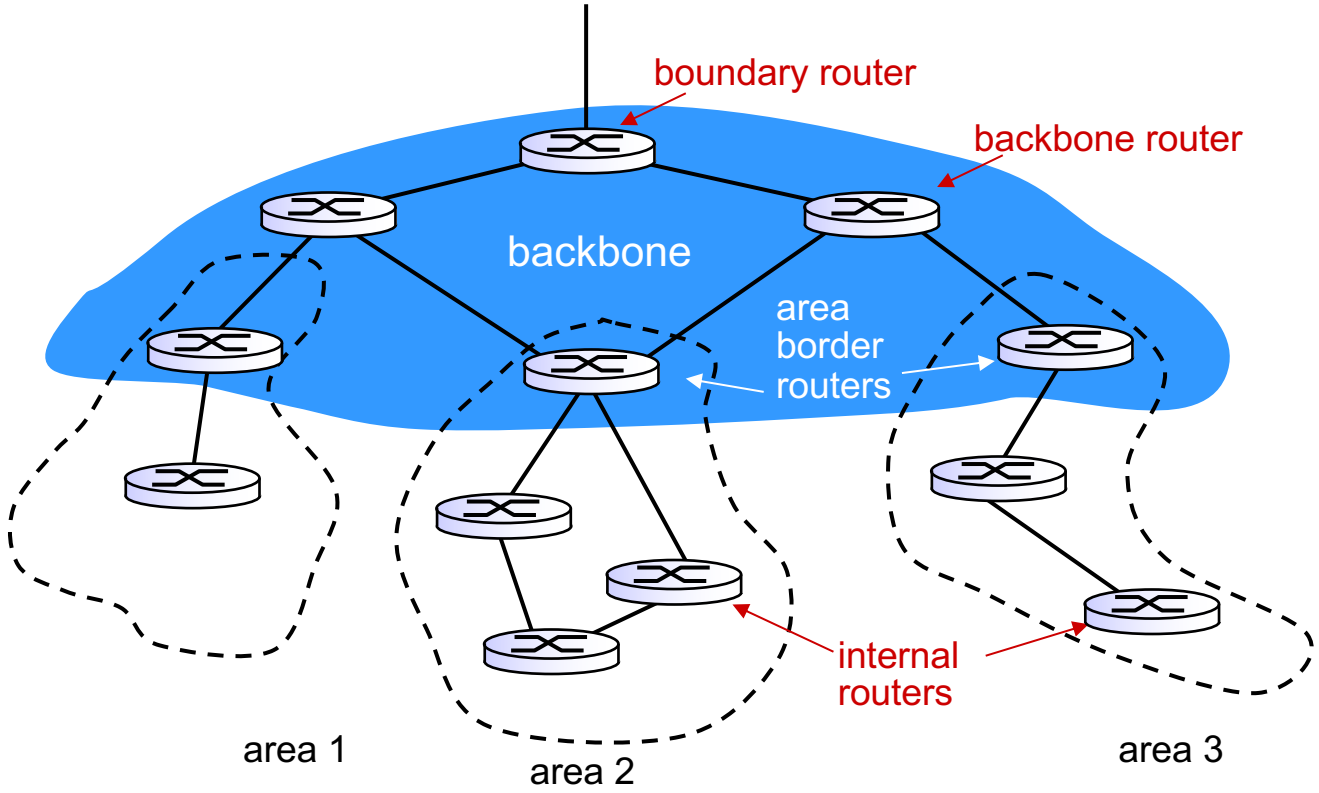


OSPF (Open Shortest Path First)

- Uses link state algorithm
- OSPF advertisement carries one entry per neighbor
- Advertisements flooded to entire AS
- carried in OSPF messages directly over IP (rather than TCP or UDP)

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.
- area border routers: "summarize" distances to nets in own area, advertise to other Area Border routers..
- boundary routers: connect to other AS's.

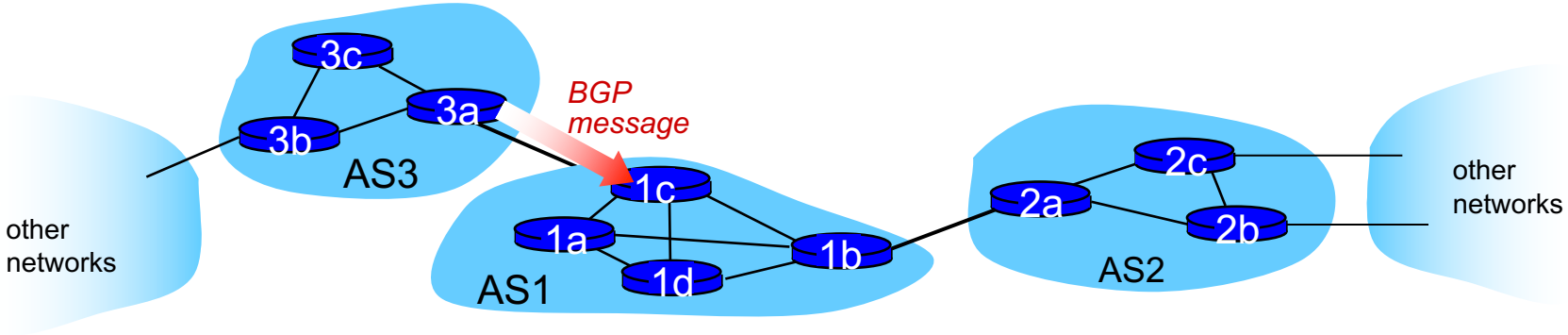


Internet Inter-AS Routing: BGP

- 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 ASs.
  - 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"

BGP Basics

- BGP session: two BGP routers (“peers”) exchange BGP messages:
  - advertising paths to different destination network prefixes (“path vector” protocol)
  - exchanged over semi-permanent TCP connections
- When AS3 advertises a prefix to AS1:
  - AS3 promises it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



Recap Where We're At

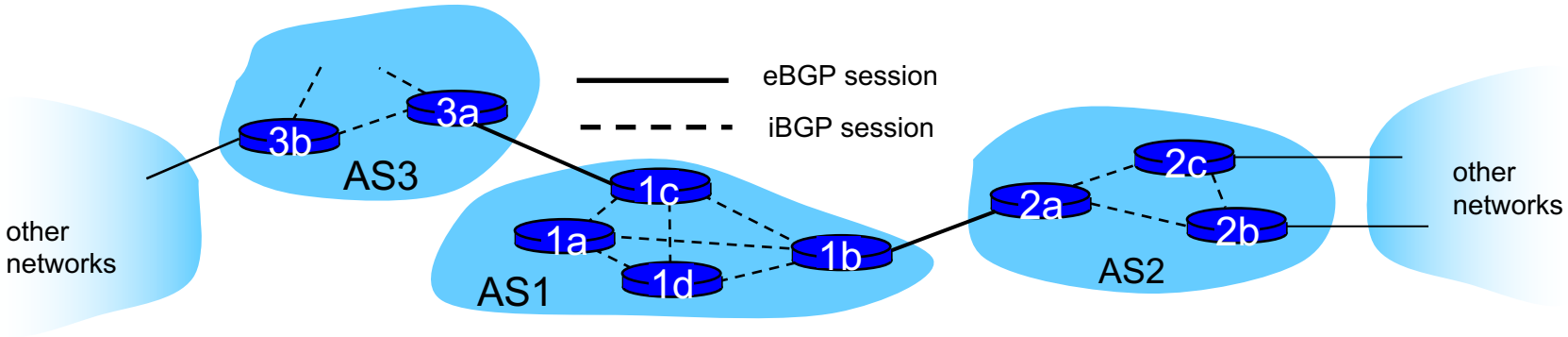
- Comparison between LS and DV
- Inter-AS and intra-AS routing
- RIP, OSPF
- BGP

Outline

- BGP (continue)
- Broadcast

BGP Basics: Distributing Path Information

- Using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



Path Attributes and BGP Routes

- Advertised prefix includes BGP attributes
  - prefix + attributes = “route”
- Two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- Gateway router receiving route advertisement uses import policy to accept/decline
  - e.g., never route through AS x
  - policy-based routing

BGP Route Selection

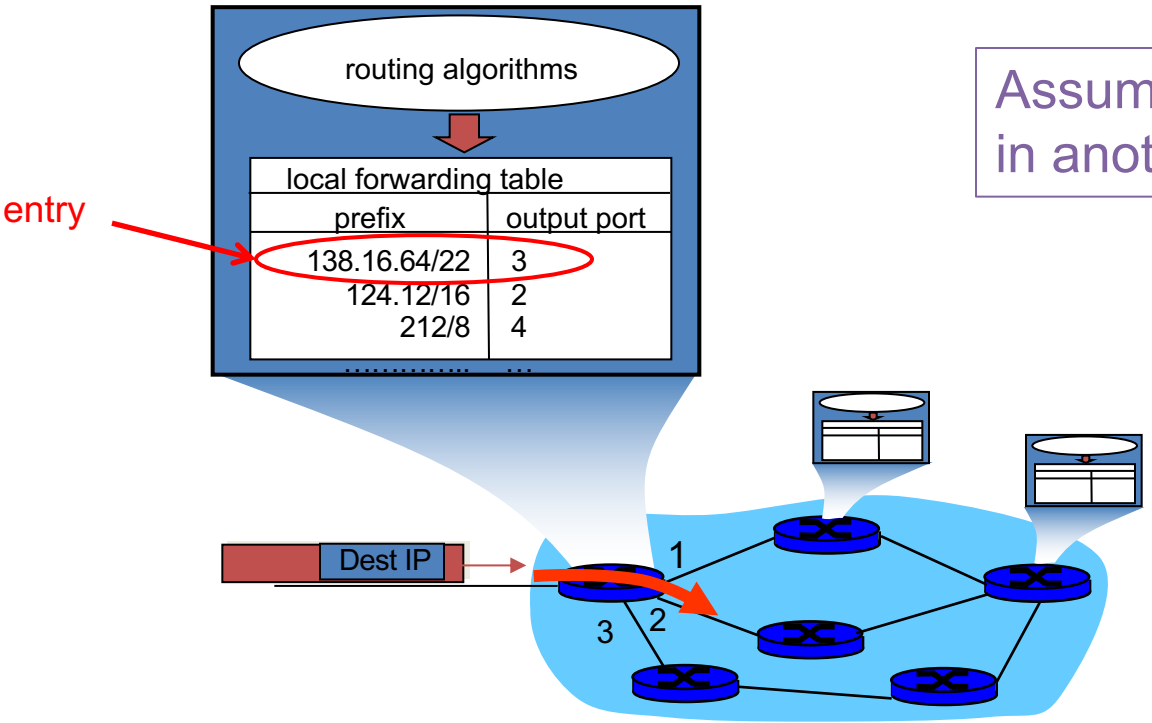
- Router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

BGP Messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

How Does Entry Get in Forwarding Table?

- Ties together hierarchical routing with BGP and OSPF

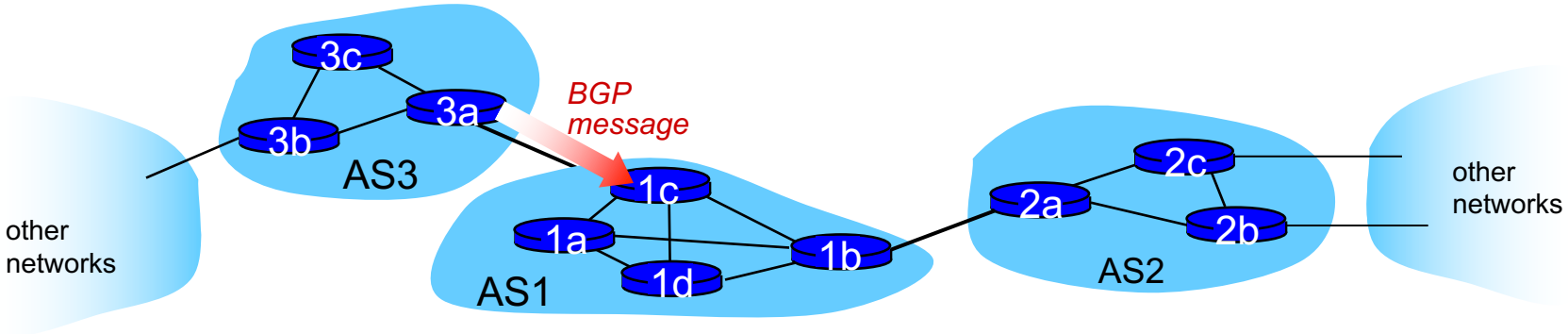


Assume prefix is in another AS.

How Does Entry Get in Forwarding Table?

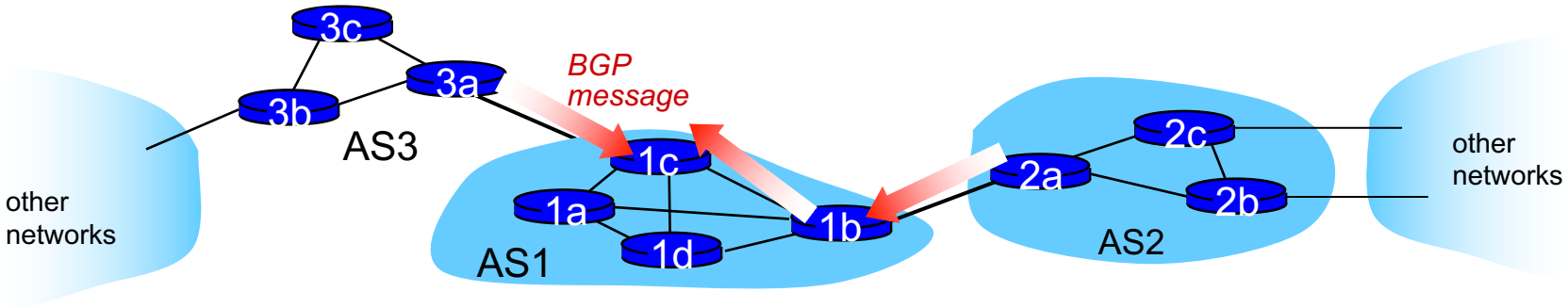
- High-level overview
  1. Router becomes aware of prefix
  2. Router determines output port for prefix
  3. Router enters prefix-port in forwarding table

Router Becomes Aware of Prefix



- BGP message contains “routes”
- “route” is a prefix and attributes: AS-PATH, NEXT-HOP,…
- Example: route:
  - Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

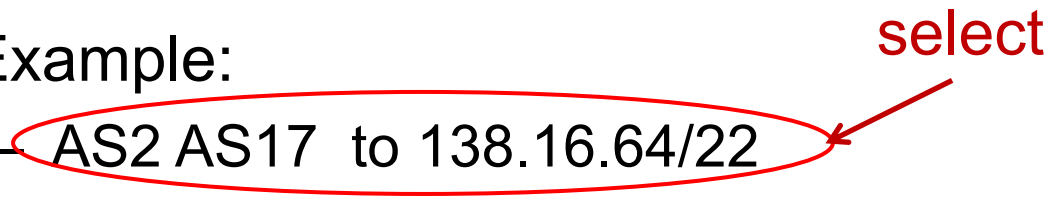
Router May Receive Multiple Routes



- Router may receive multiple routes for same prefix
- Has to select one route

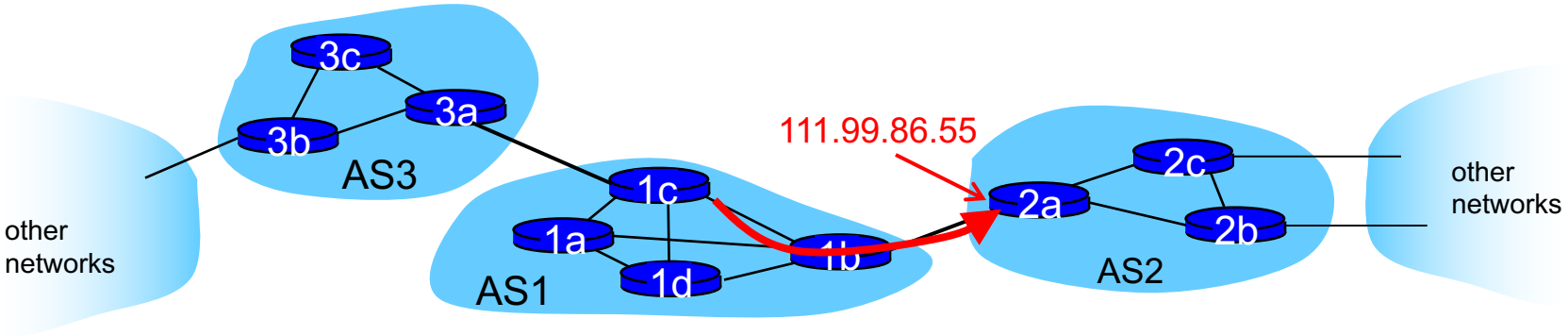
Select Best BGP Route to Prefix

- Router selects route based on shortest AS-PATH
- Example:
  - AS2 AS17 to 138.16.64/22
  - AS3 AS131 AS201 to 138.16.64/22



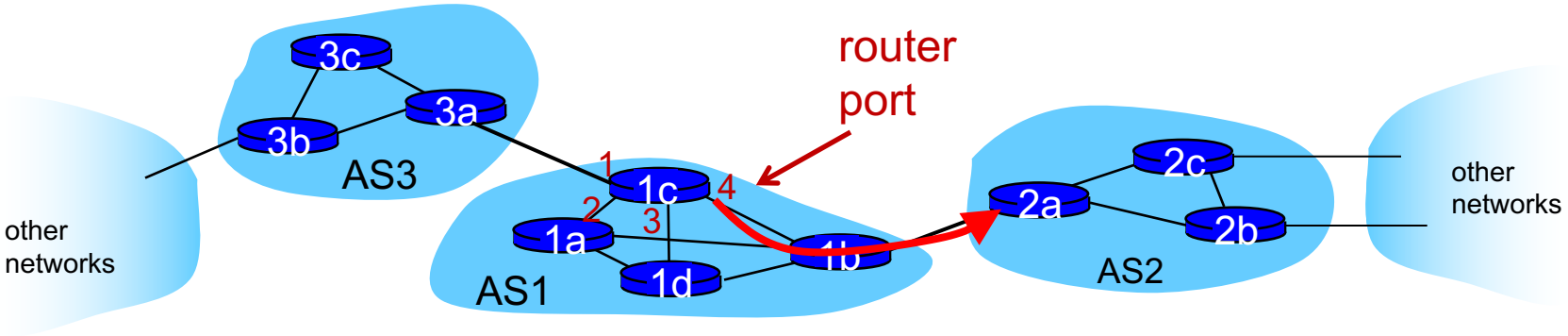
Find Best Intra-route to BGP route

- Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- Example:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- Router uses OSPF to find shortest path from 1c to 111.99.86.55



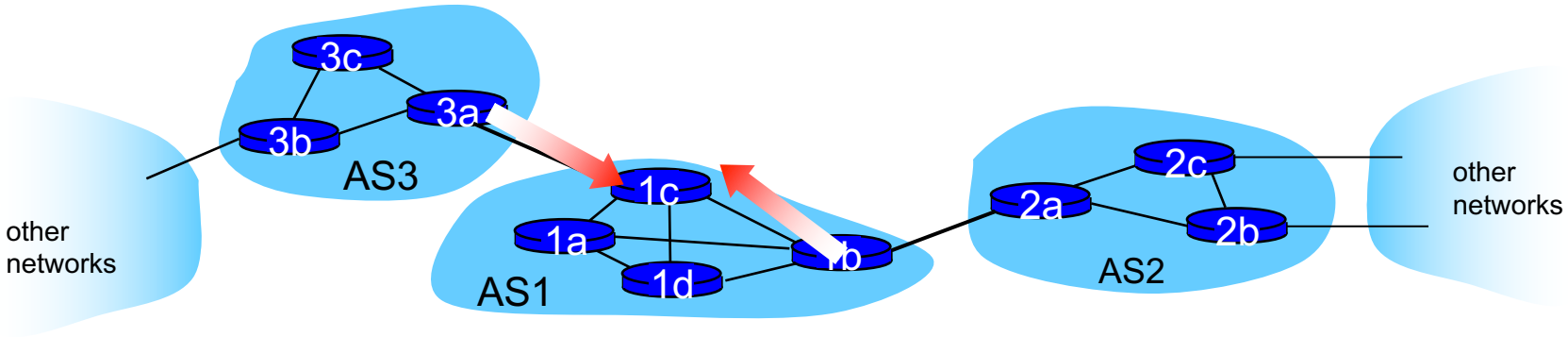
Router Identifies Port for Route

- Identifies port along the OSPF shortest path
- Adds prefix-port entry to its forwarding table:
  - (138.16.64/22 , port 4)



Hot Potato Routing

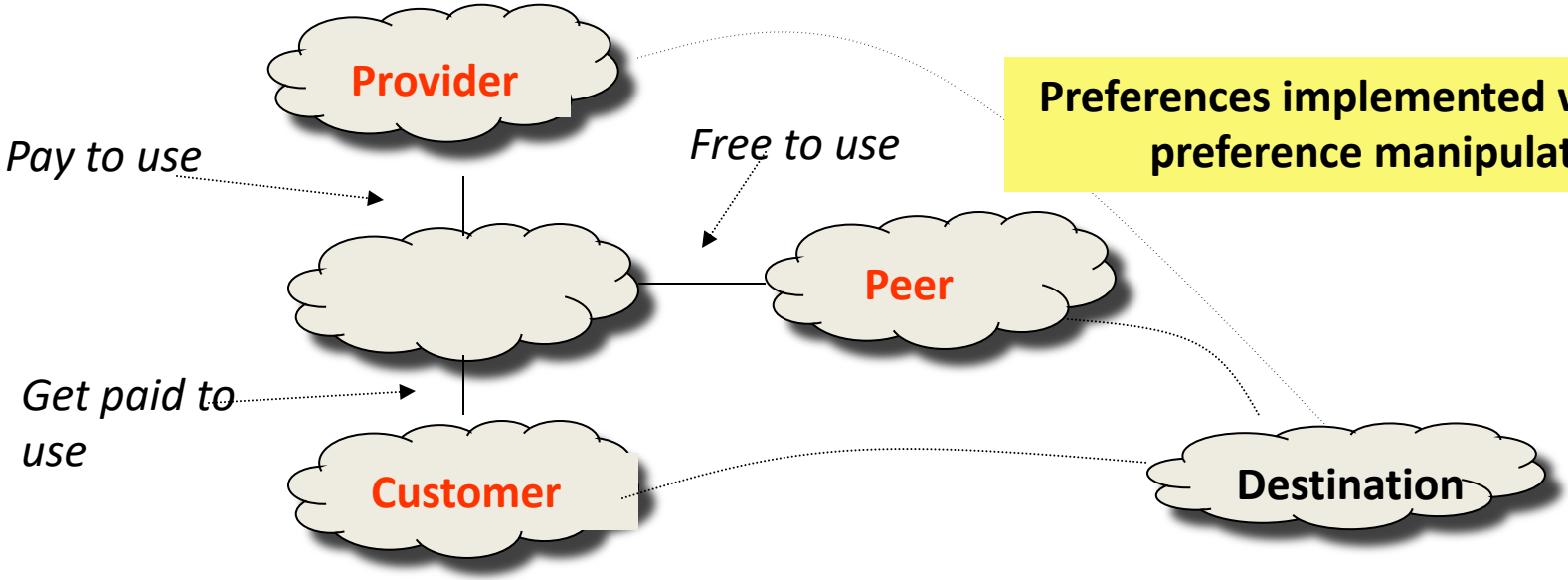
- Suppose there are two or more best inter-routes.
- Then choose route with closest NEXT-HOP
  - Use OSPF to determine which gateway is closest
    - Q: From 1c, chose AS3 AS131 or AS2 AS17?
    - A: route AS3 AS131 since it is closer



How Does Entry Get in Forwarding Table?

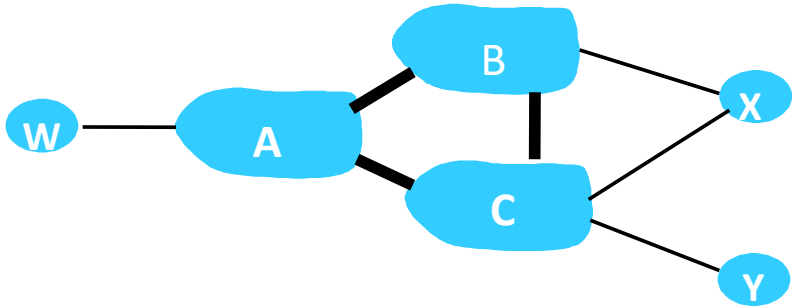
- Summary
  1. Router becomes aware of prefix
     - via BGP route advertisements from other routers
  2. Determine router output port for prefix
     - Use BGP route selection to find best inter-AS route
     - Use OSPF to find best intra-AS route leading to best inter-AS route
     - Router identifies router port for that best route
  3. Enter prefix-port entry in forwarding table

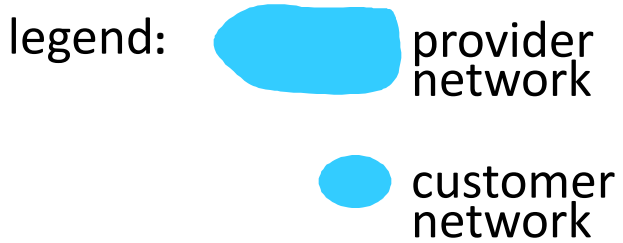
Internet Business Model (Simplified)



- **Customer/Provider:** One AS pays another for reachability to some set of destinations
- **"Settlement-free" Peering:** Bartering. Two ASes exchange routes with one another.

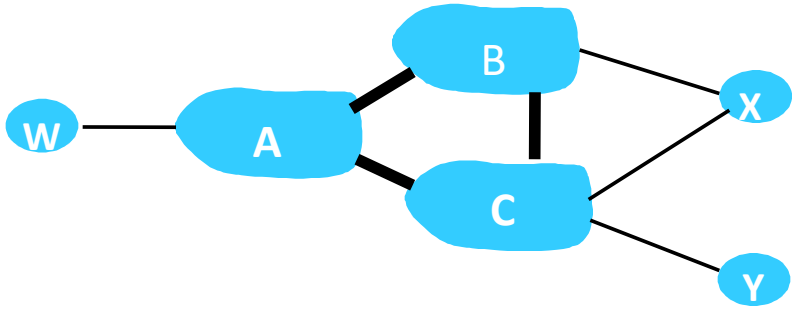
BGP Routing Policy

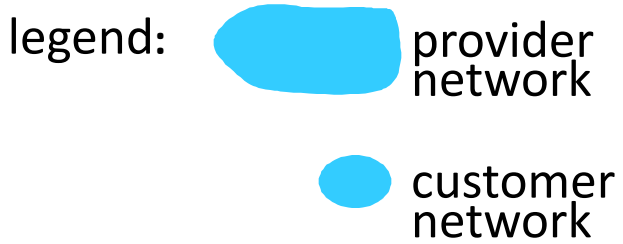




- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP Routing Policy





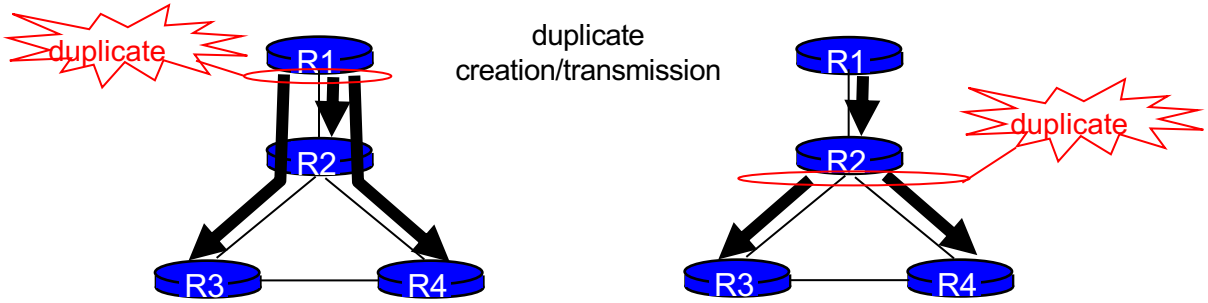
- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No … B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route only to/from its customers

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

Broadcast Routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient:



source duplication

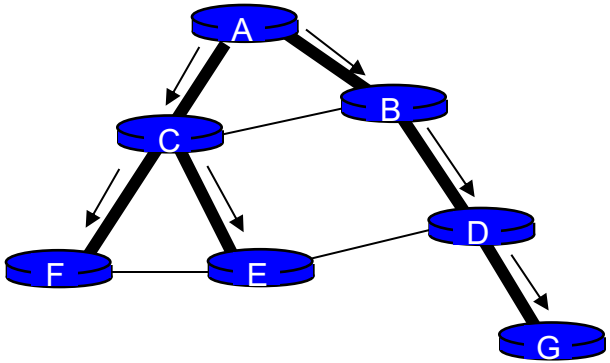
in-network duplication

In-network Duplication

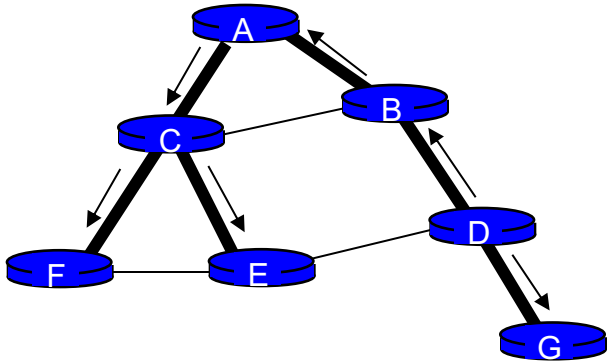
- Flooding: when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- Controlled flooding: node only broadcasts packet if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadacsted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- Spanning tree:
  - no redundant packets received by any node

Spanning Tree

- First construct a spanning tree
- Nodes then forward/make copies only along spanning tree



(a) broadcast initiated at A



(b) broadcast initiated at D

Recap Where We’re At

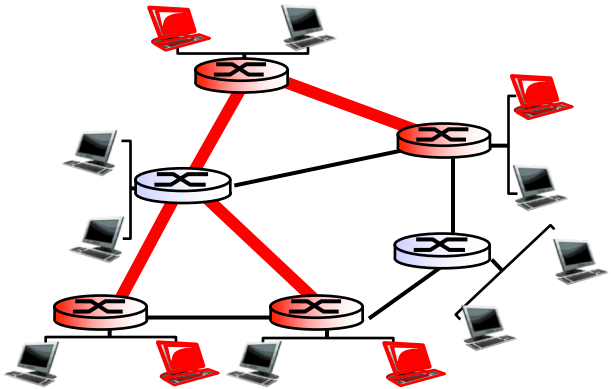
- BGP
- Broadcast

Outline

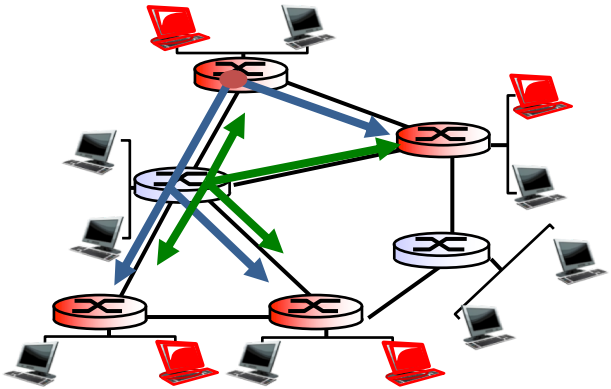
- Multicast

Multicast Routing: Problem Statement

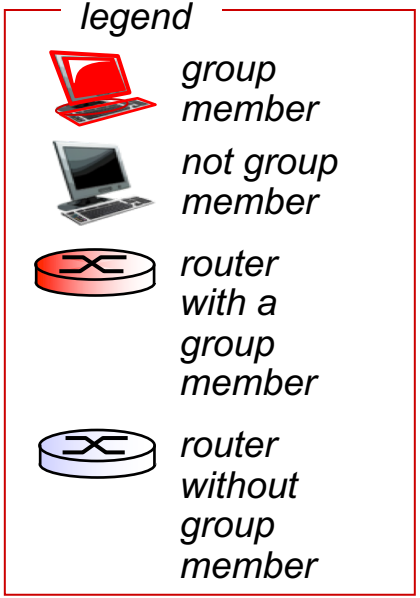
- Goal: find a tree (or trees) connecting routers having local multicast group members
  - shared-tree: same tree used by all group members
  - source-based: different tree from each sender to receivers



shared tree



source-based trees

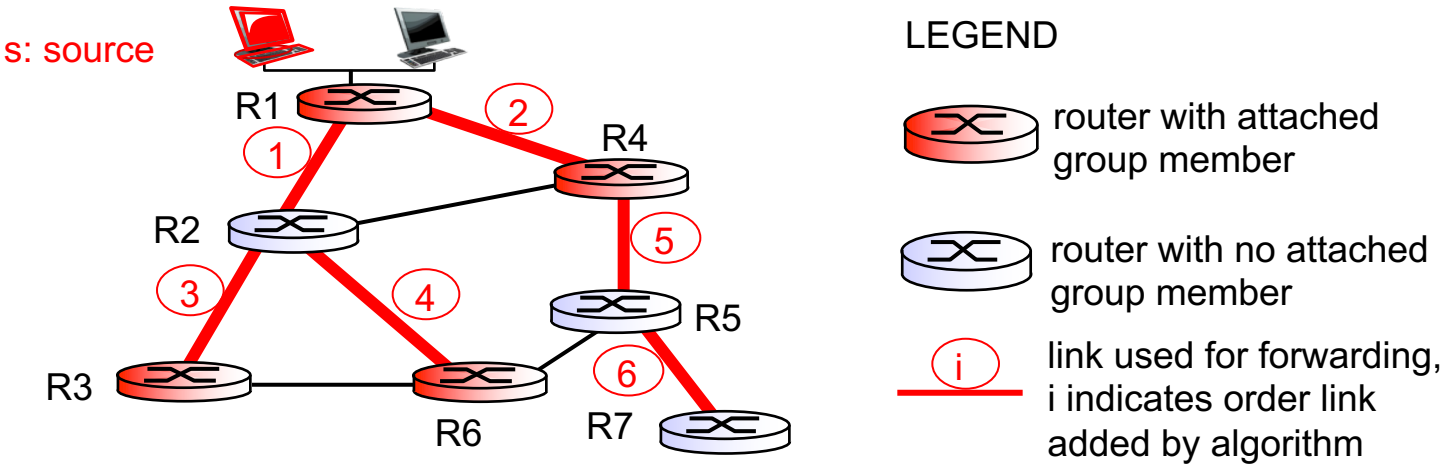


Approaches for Building Multicast Trees

- Approaches:
  - source-based tree: one tree per source
    - shortest path trees
    - reverse path forwarding
  - group-shared tree: group uses one tree

Shortest Path Tree

- Multicast forwarding tree: tree of shortest path routes from source to all receivers



Reverse Path Forwarding

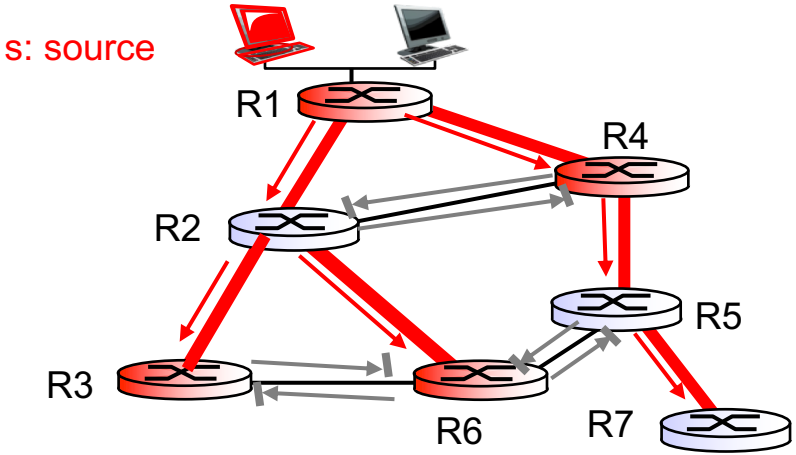
- Rely on router's knowledge of unicast shortest path from it to sender
- Each router has simple forwarding behavior:

if (mcast datagram received on incoming link on shortest path back to center)

then flood datagram onto all outgoing links

else ignore datagram

Reverse Path Forwarding: Example



LEGEND

router with attached group member

router with no attached group member

datagram will be forwarded

datagram will not be forwarded

- Result is a source-specific reverse spanning tree