ECS524
Network layer

Prof. Steve Uhlig
steve.uhlig@qmul.ac.uk
Office: Eng202

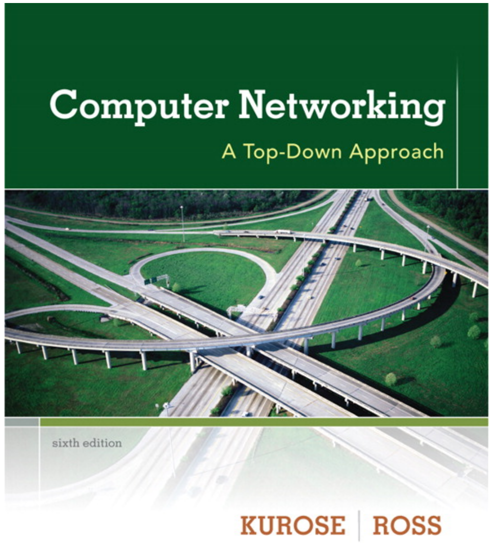
Dr. Felix Cuadrado
felix.cuadrado@qmul.ac.uk
Office: E205

Slides

Disclaimer:

Some of the slides' content is borrowed directly from those provided by the authors of the textbook. They are available from

http://www-net.cs.umass.edu/kurose-ross-ppt-6e



Computer Networking: A Top Down Approach

6th edition

Jim Kurose, Keith Ross

Addison-Wesley

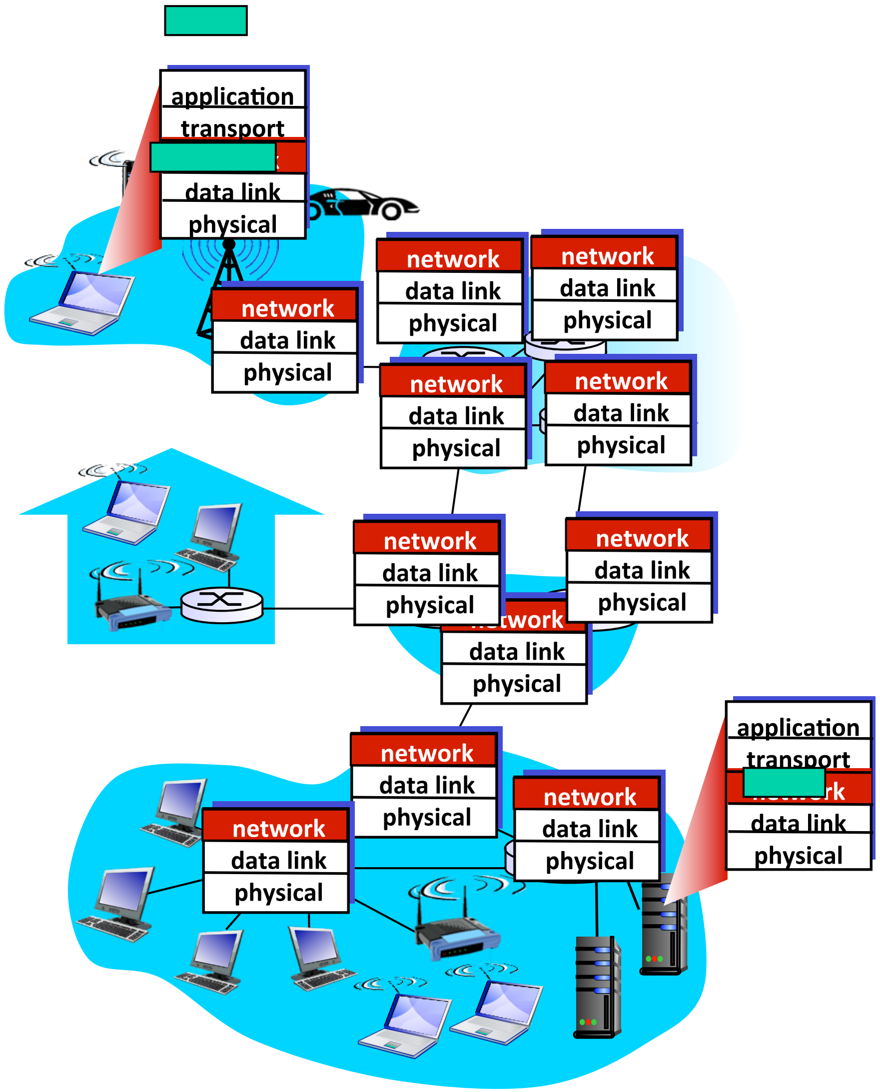
March 2012

The Network layer

- **Introduction**
- IP Addressing
- IP routers
- IP
- Routing: concepts
- Routing: practice

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



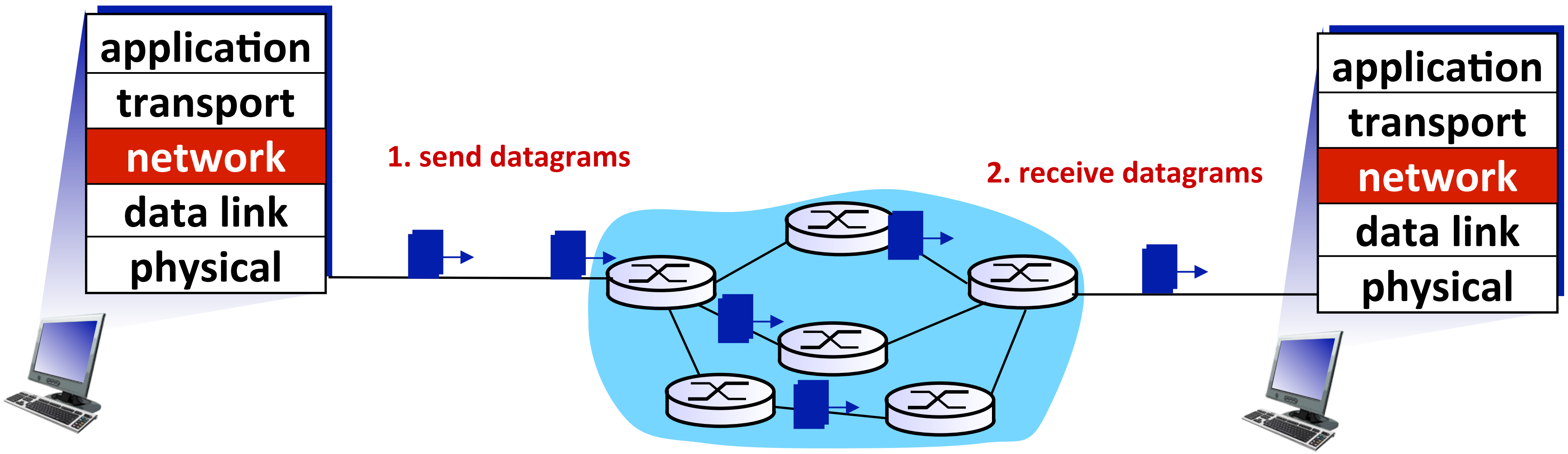
Datagram networks

no call setup at network layer

routers: no state about end-to-end connections

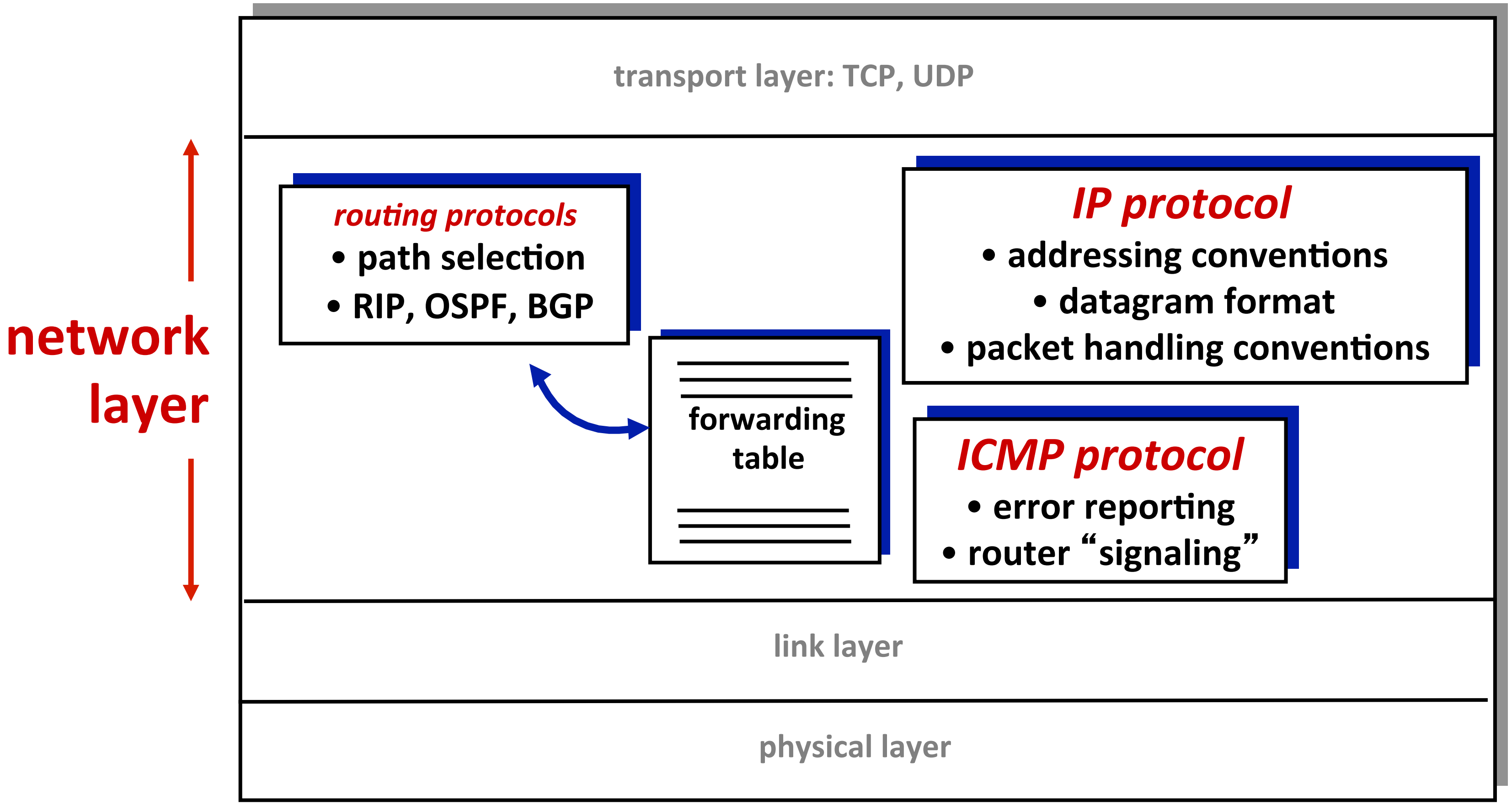
- no network-level concept of "connection"

packets forwarded using destination host address



The Internet network layer

- Host, router network layer functions:

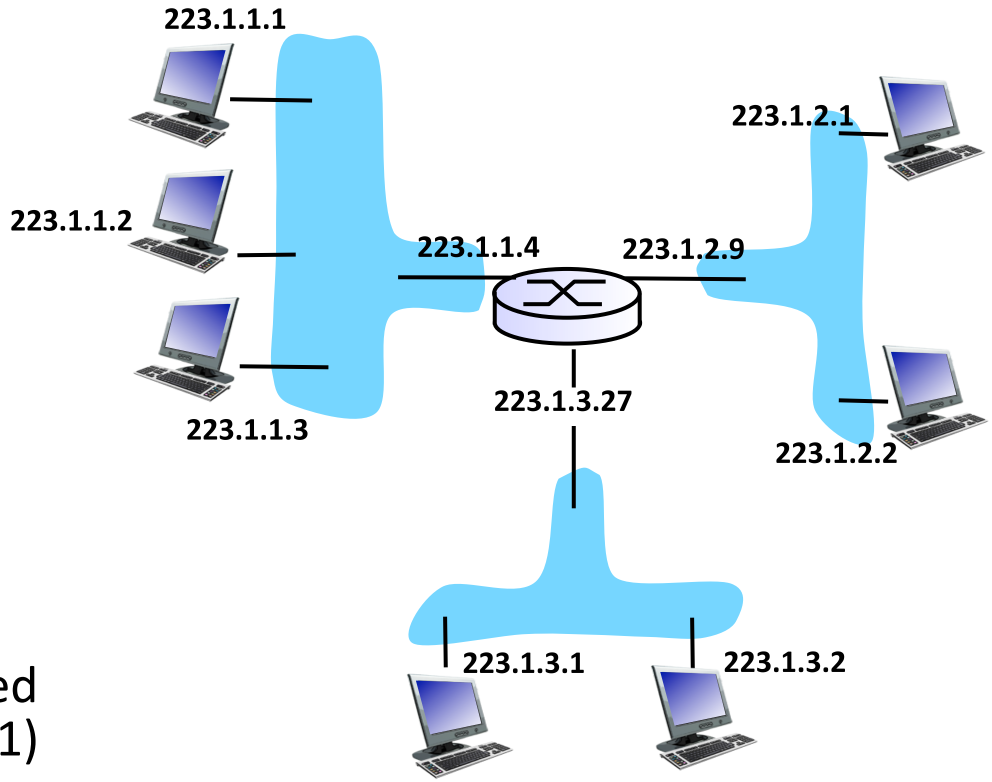


The Network layer

- Introduction
- **IP Addressing**
- IP routers
- IP
- Routing: concepts
- Routing: practice

IP addressing: introduction

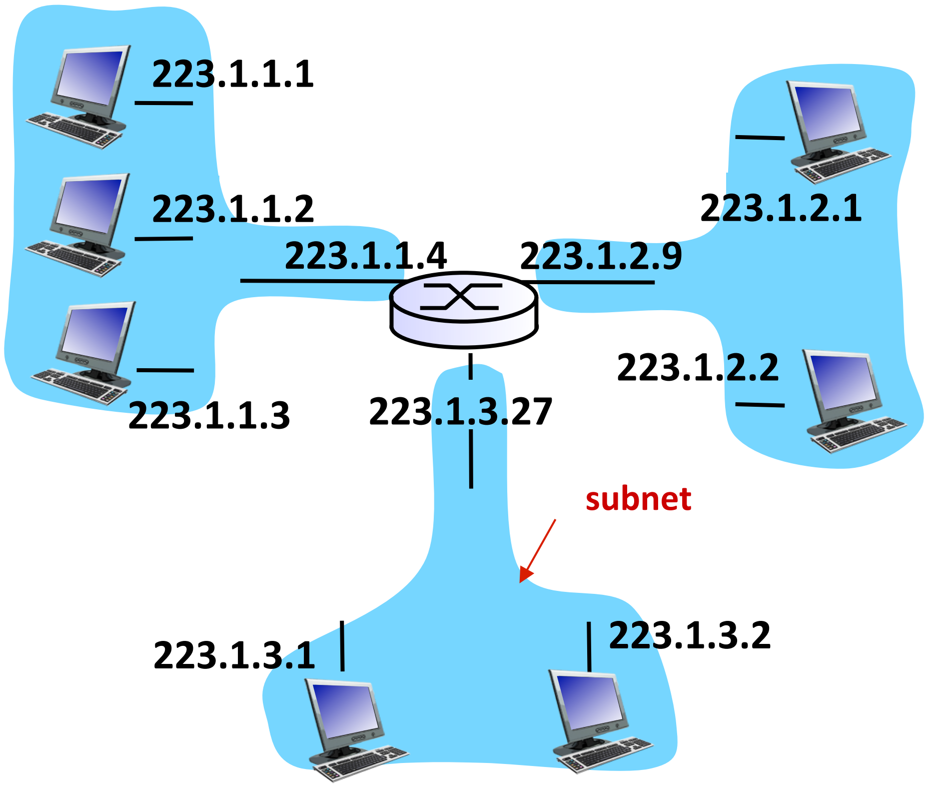
- *IP address:* 32-bit identifier for host, router *interface*
- *Interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

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*



network consisting of 3 subnets

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

subnet part | host part

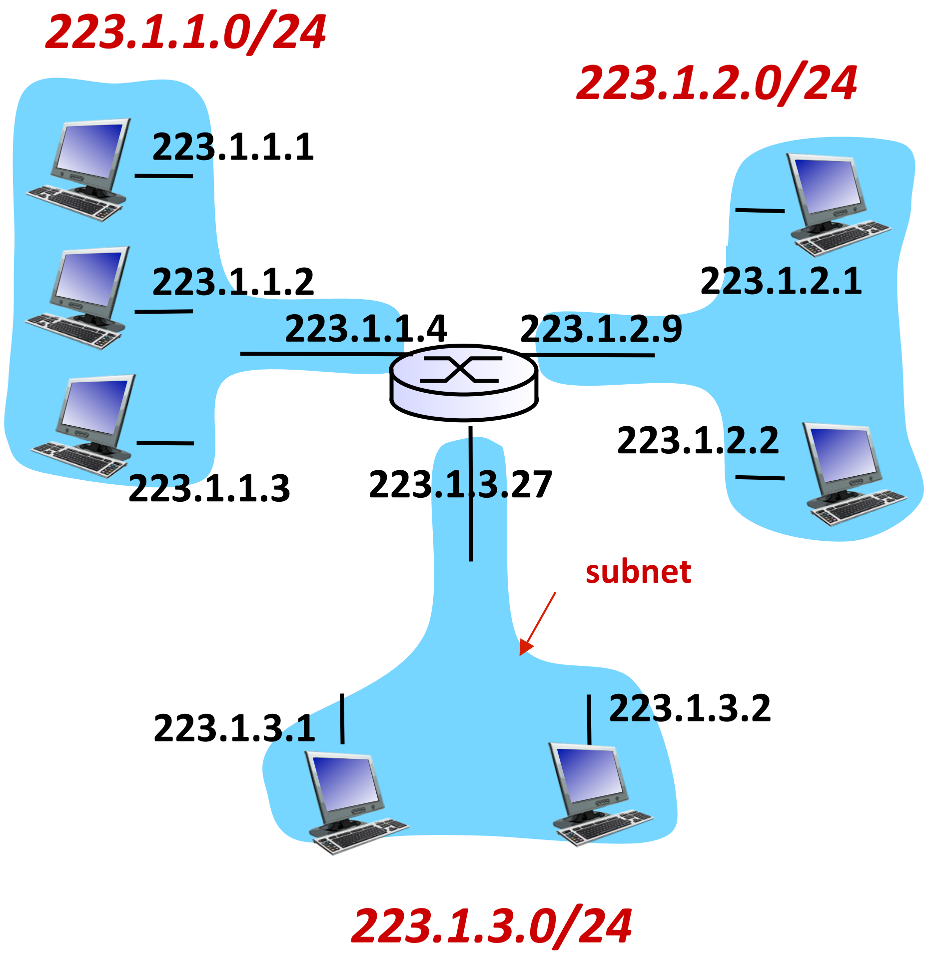
11001000 00010111 00010000 00000000

200.23.16.0/23

Subnets

Recipe

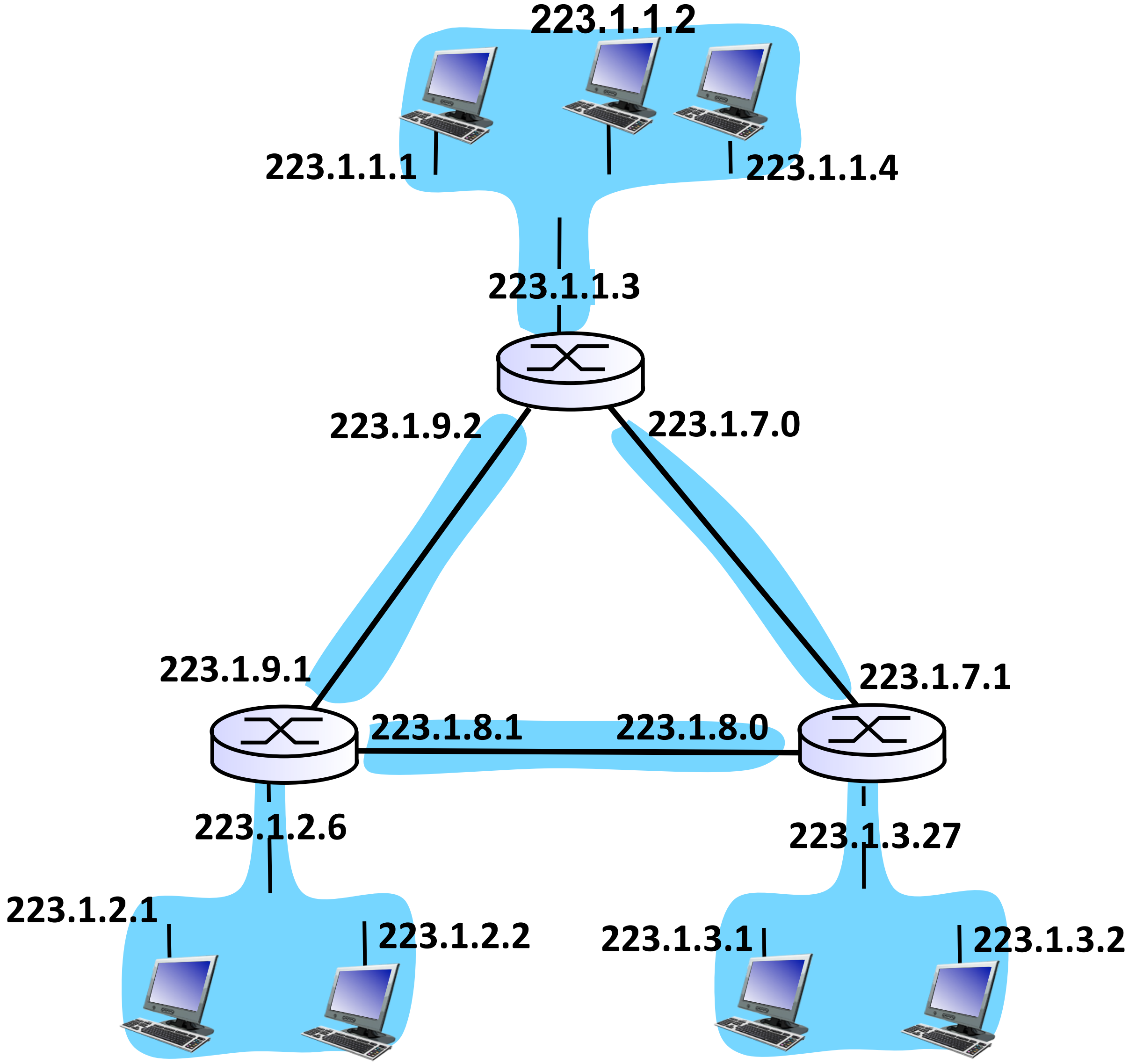
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

How many?

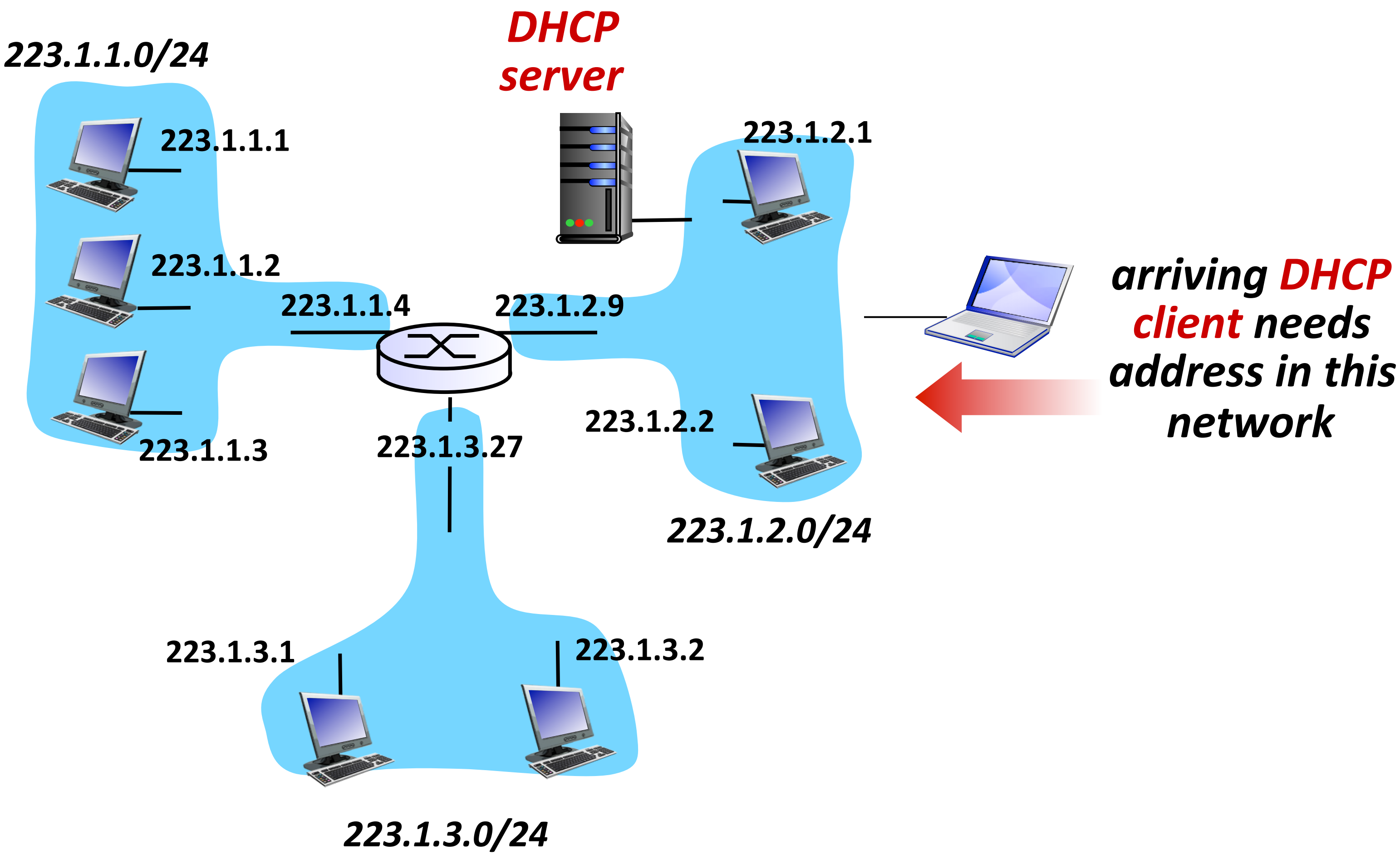


IP address: how to get one?

Q: How does a *host* get an 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"

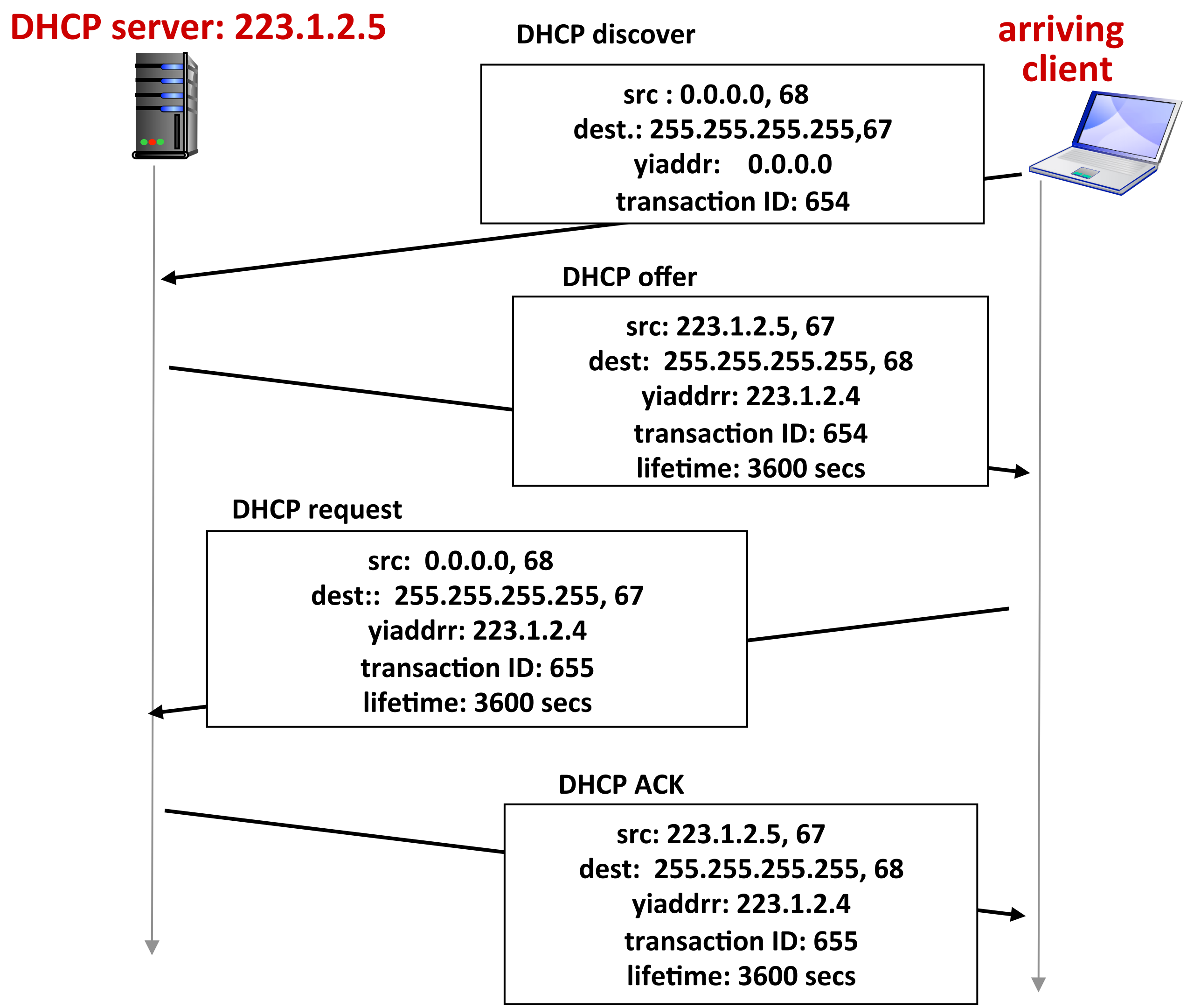
DHCP client-server scenario



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" msg [optional]
  - DHCP server responds with "DHCP offer" msg [optional]
  - host requests IP address: "DHCP request" msg
  - DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario

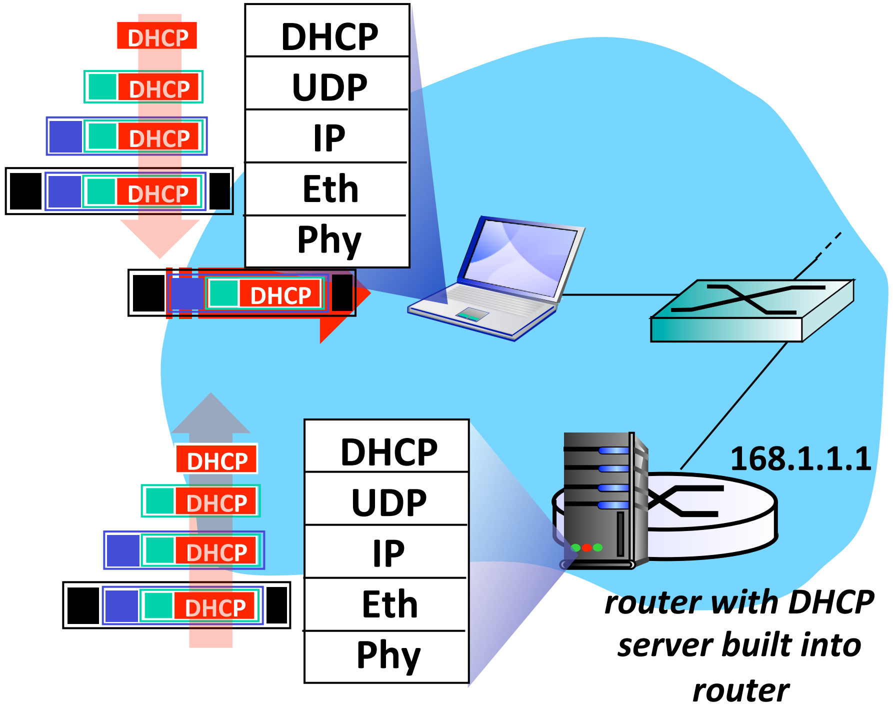


DHCP: more than IP addresses

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)

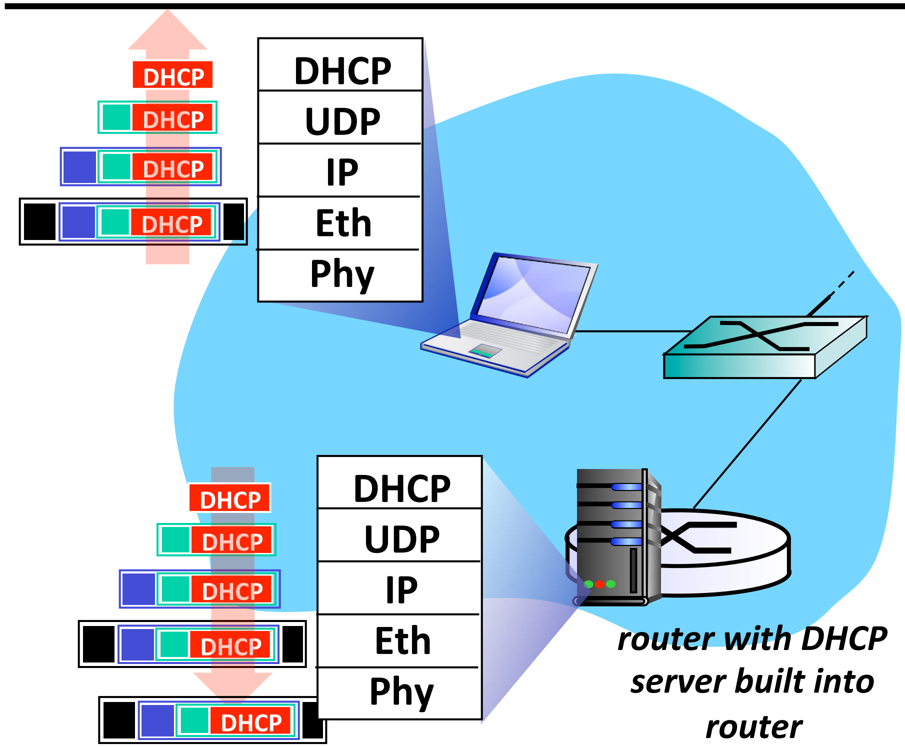
DHCP: example



Connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP

- **DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet**
- **Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server**
- **Ethernet demuxed to IP demuxed, 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, demuxing 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**

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

IP addressing: the last word...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

The Network layer

- Introduction
- Addressing
- **IP routers**
- IP
- Routing: concepts
- Routing: practice

Two key network-layer functions

forwarding: move packets from router's input to appropriate router output

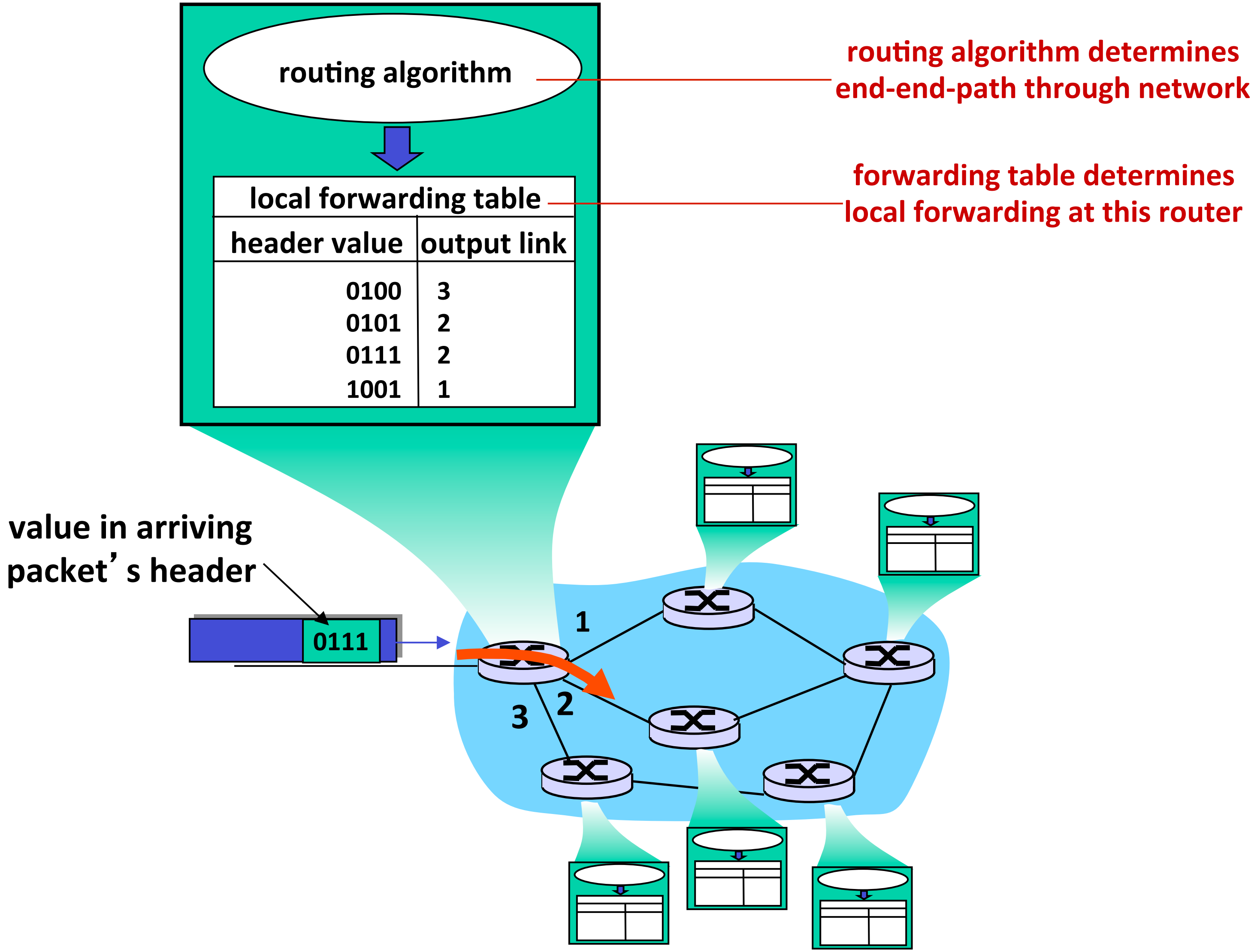
routing: determine route taken by packets from source to dest.

routing algorithms

analogy:

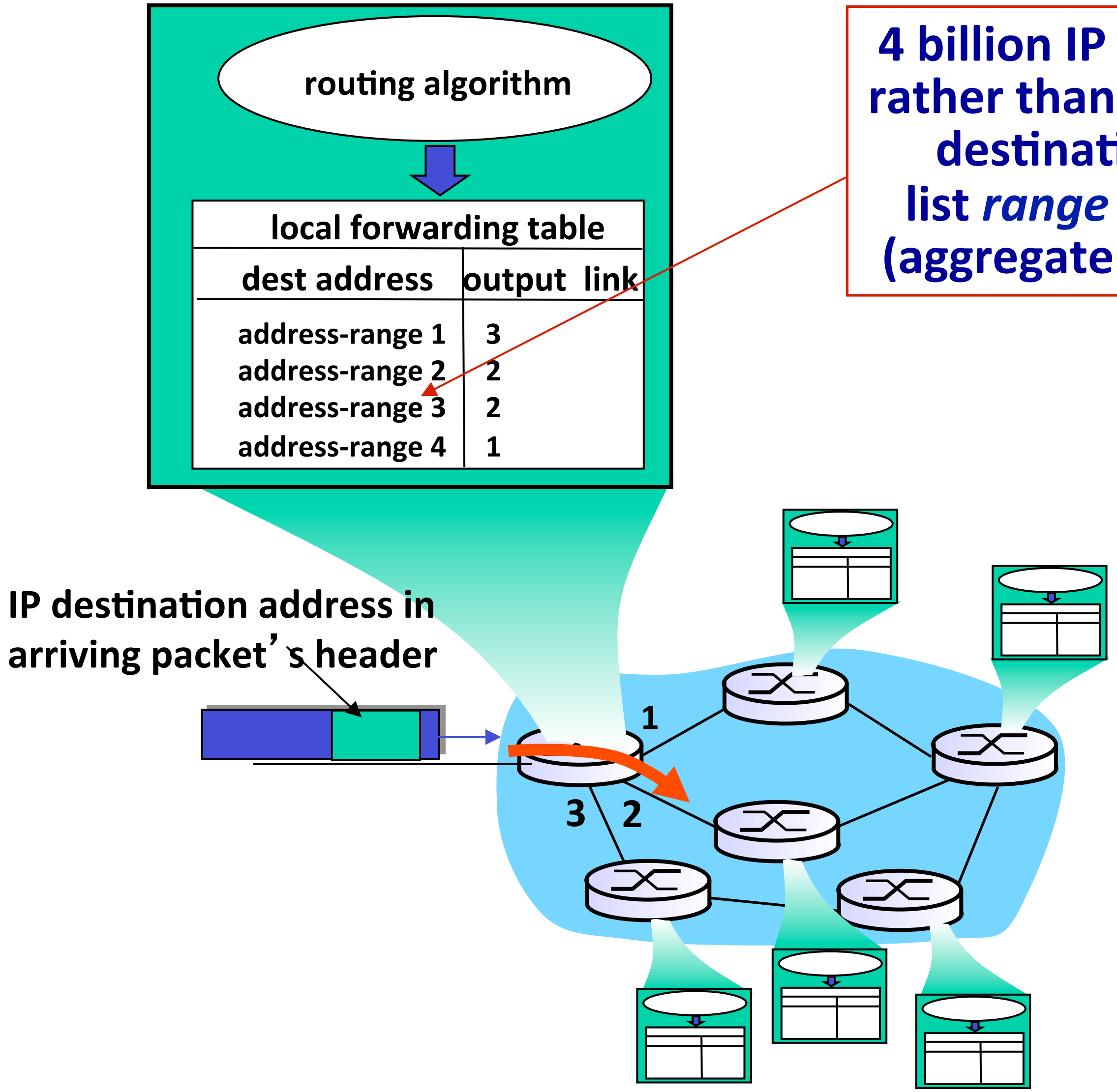
- ***routing:*** **process of planning trip from source to dest**
- ***forwarding:*** **process of getting through single interchange**

Interplay between routing and forwarding



Datagram forwarding table

4 billion IP addresses, so rather than list individual destination address list *range* of addresses (aggregate table entries)



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

Q: **but what happens if ranges don't divide up so nicely?**

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *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:

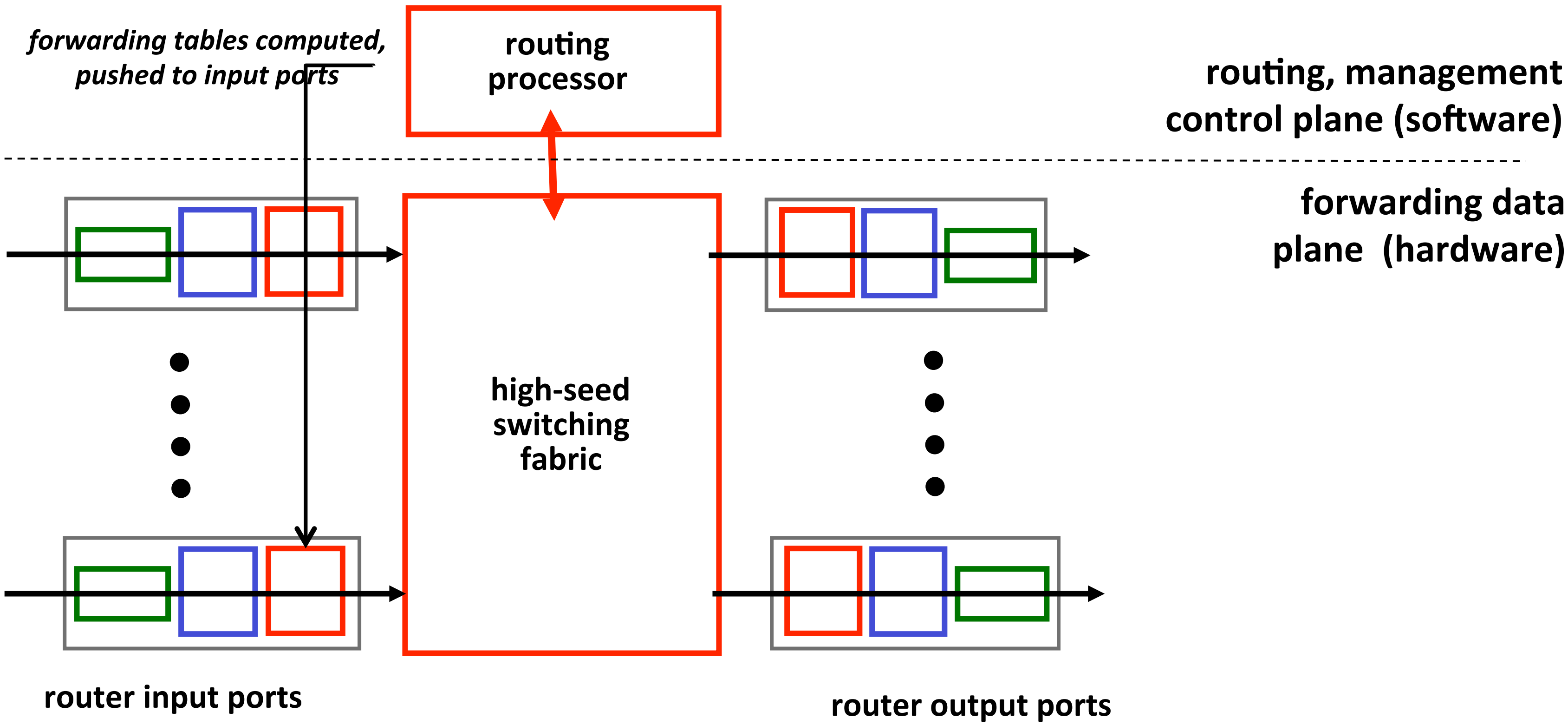
DA: 11001000 00010111 00010110 10100001 — which interface?

DA: 11001000 00010111 00011000 10101010 — which interface?

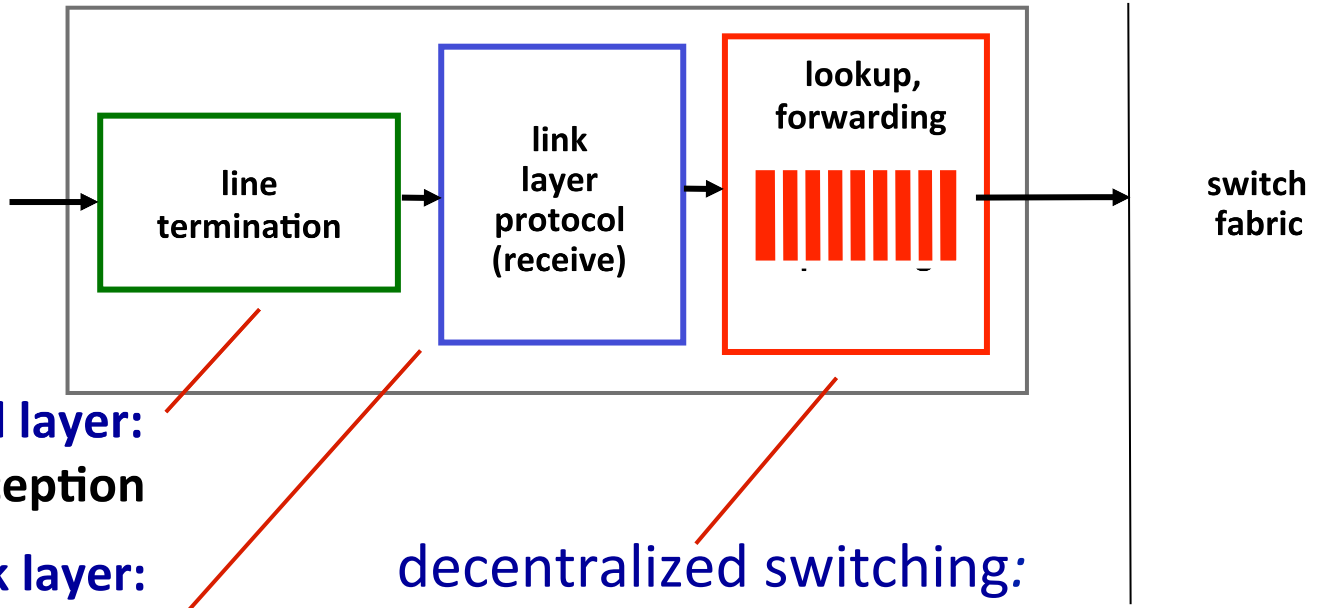
Router architecture overview

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
see chapter 5

decentralized switching:

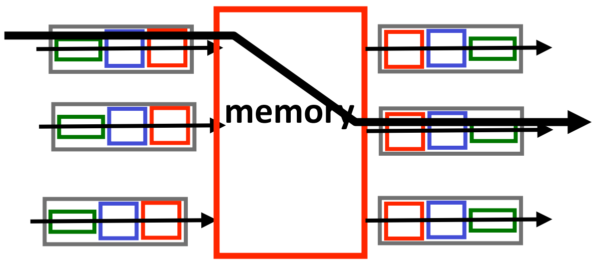
given datagram dest., lookup output port using forwarding table in input port memory *("match plus action")*

goal: complete input port processing at 'line speed'

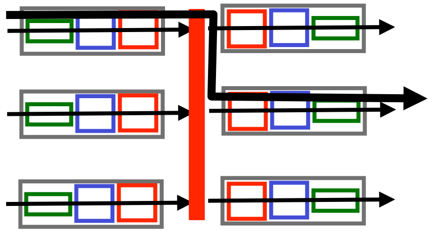
queuing: if datagrams arrive faster than forwarding rate into switch fabric

Switching fabrics

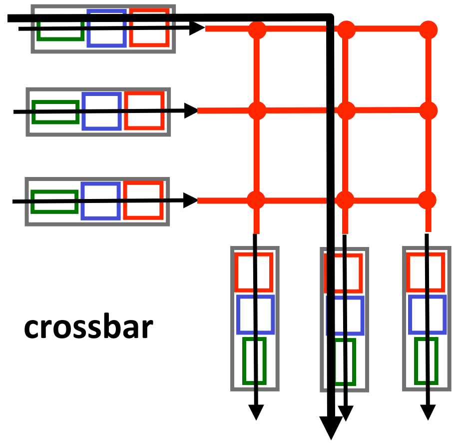
- Transfer packet from input buffer to appropriate output buffer
- Switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- Three types of switching fabrics



memory



bus

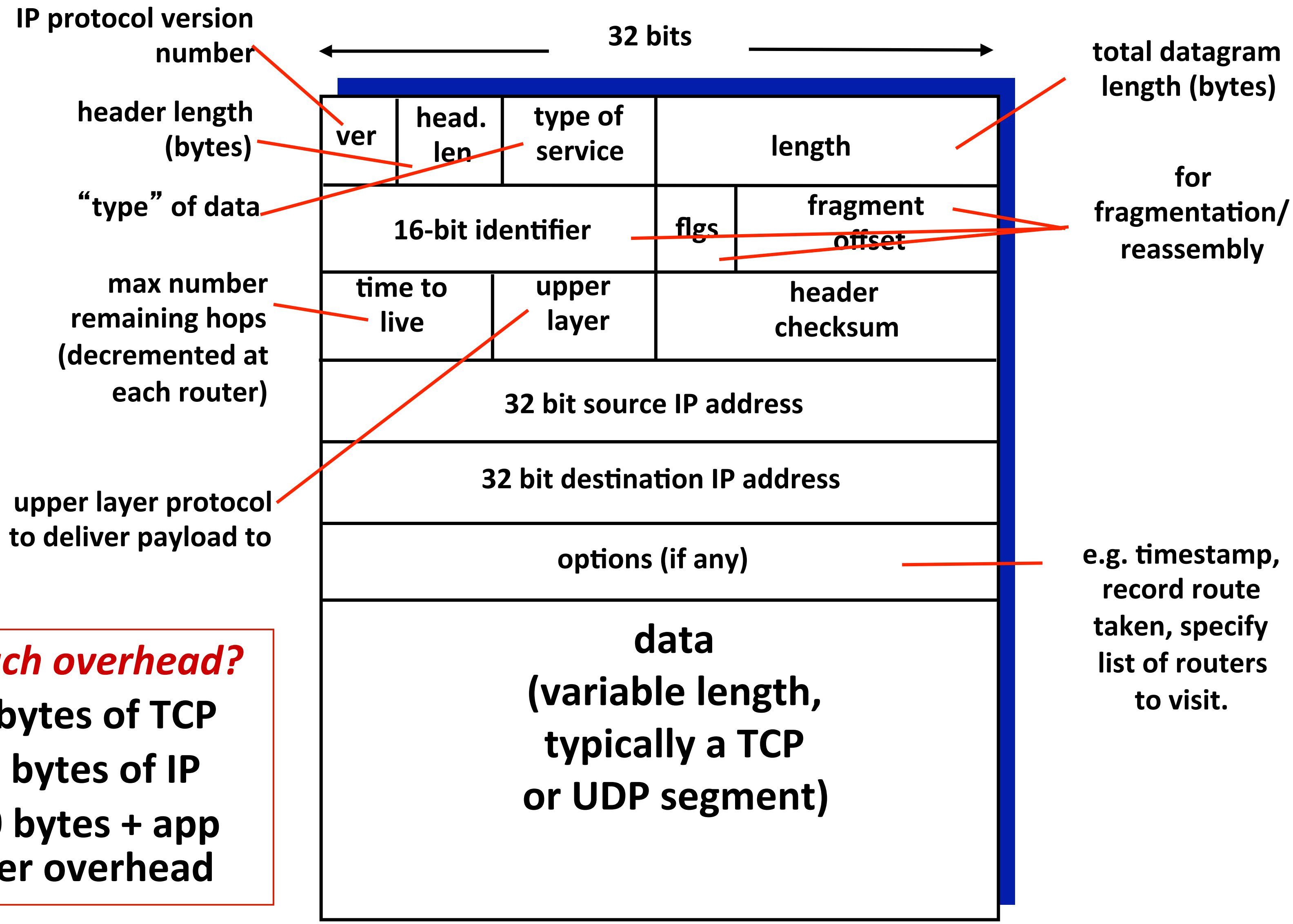


crossbar

The Network layer

- Introduction
- IP Addressing
- IP router
- **IP**
- Routing: concepts
- Routing: practice

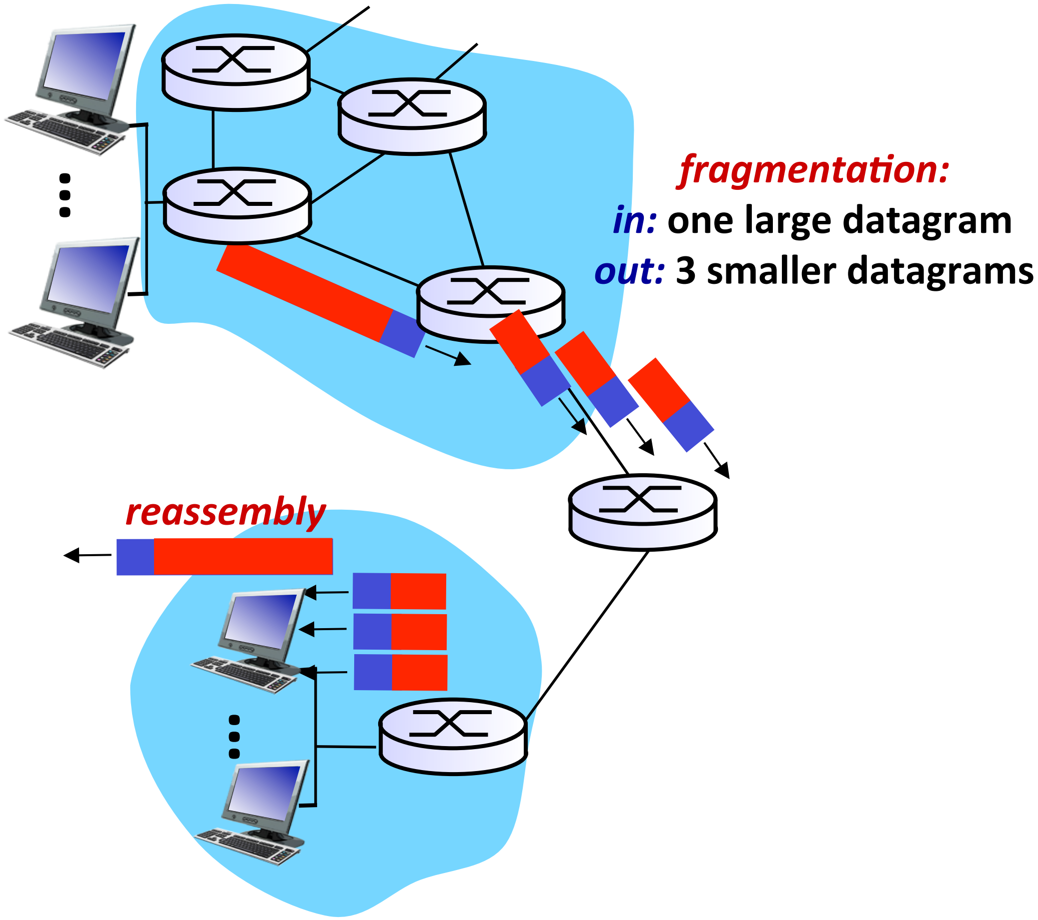
IP datagram format



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

IP fragmentation, reassembly

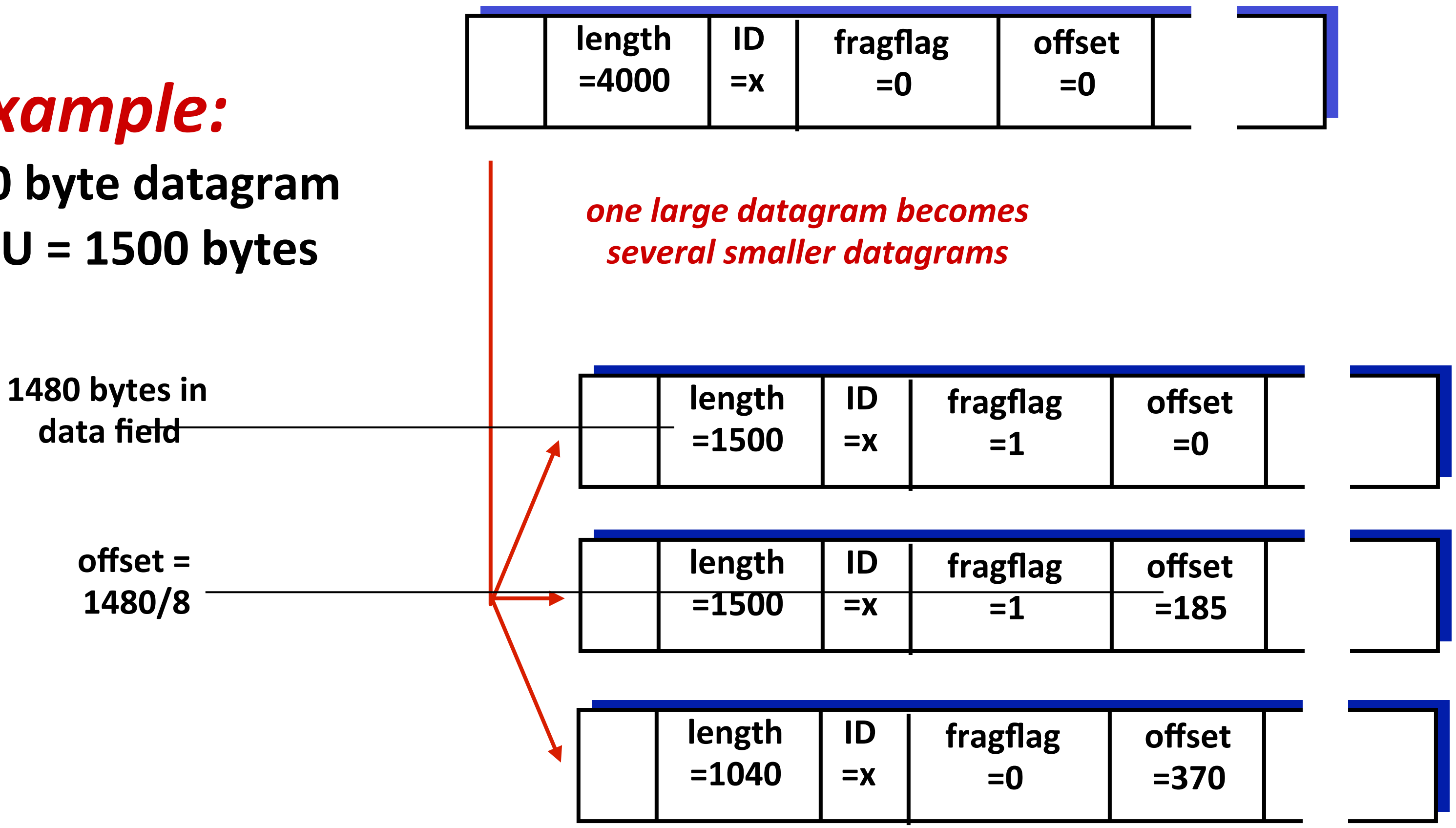
- Network links have MTU (max.transfer size) - largest possible link-level frame
  different link types, different MTUs
- Large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - reassembled" only at final destination
  - IP header bits used to identify, order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



IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



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

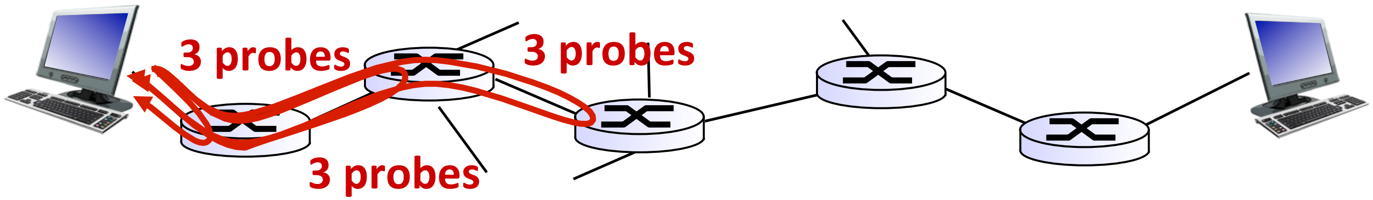
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
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.
  - unlikely port number
- When *n*th 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

stopping criteria:

- **UDP segment eventually arrives at destination host**
- **Destination returns ICMP "port unreachable" message (type 3, code 3)**
  - **source stops**

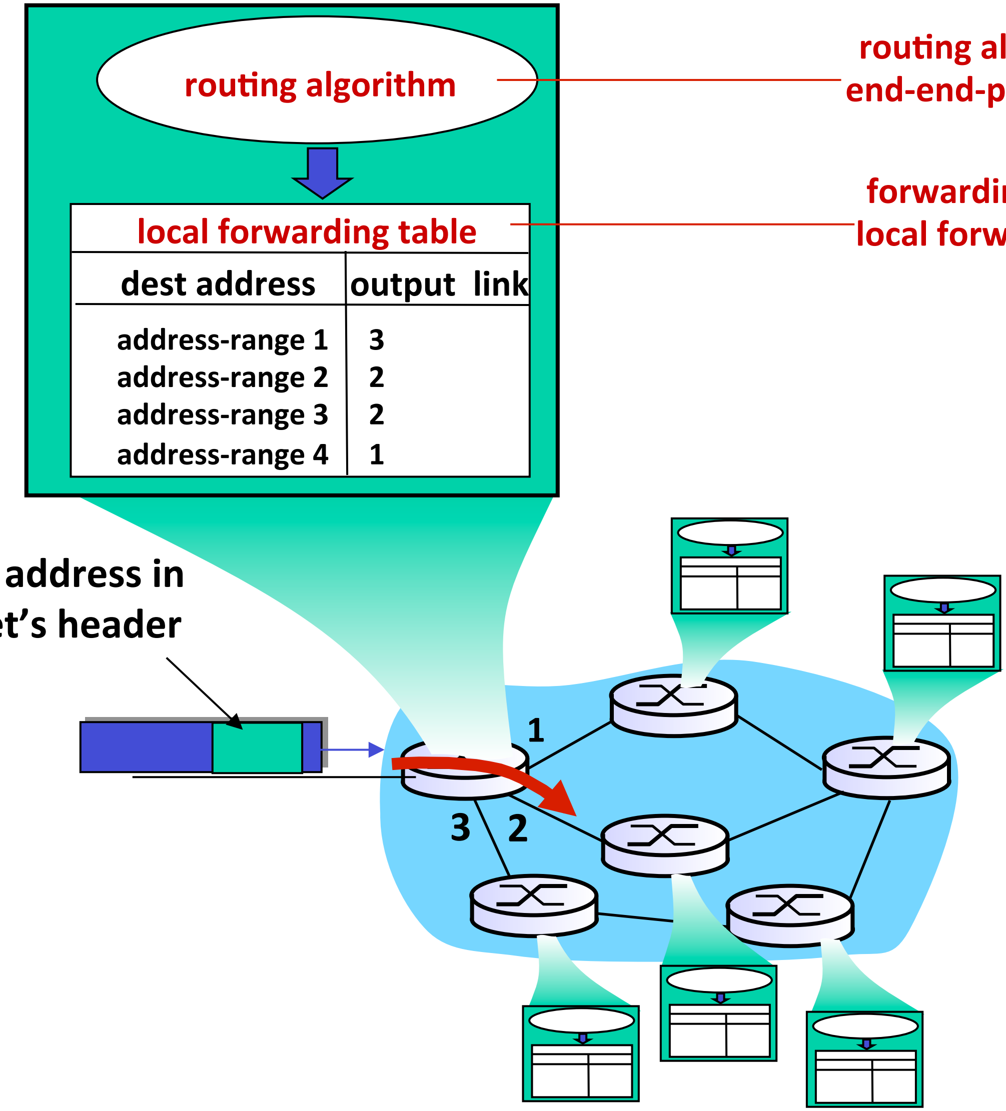


The Network layer

- Introduction
- IP router
- IP
- **Routing: concepts**
- Routing: practice

Interplay between routing, forwarding

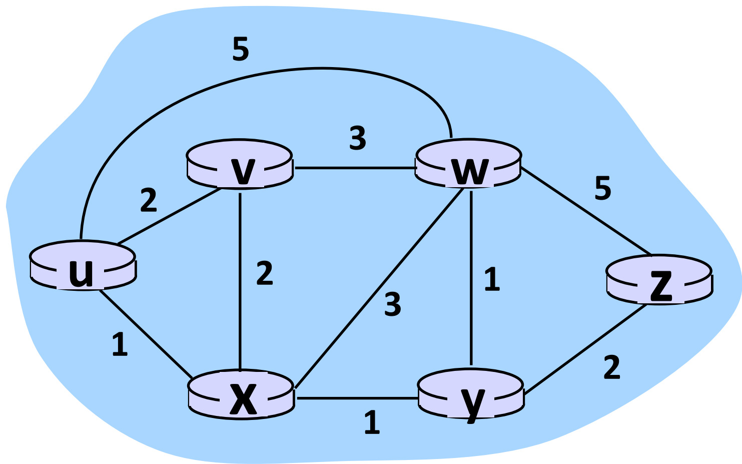
IP destination address in arriving packet's header



routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

Graph abstraction



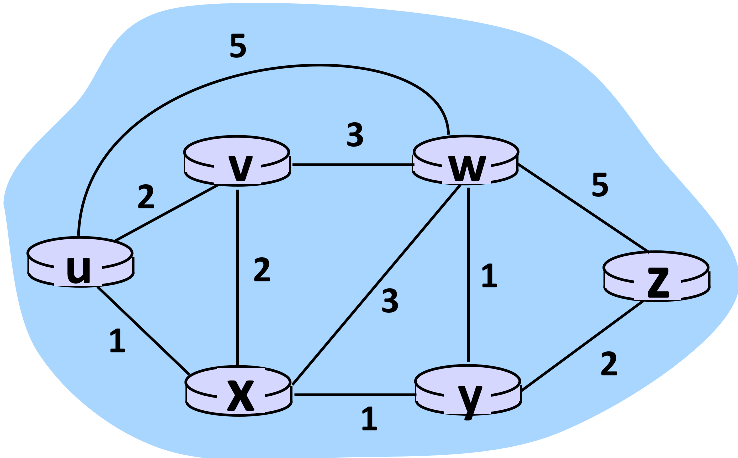
graph: G = (N,E)

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) }

***aside:* 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



$c(x,x')$ = 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

A Link-State Routing Algorithm

Queen Mary University of London

Dijkstra's algorithm

- Network 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
  - gives *forwarding table* for that node
- Iterative: after k iterations, know least cost path to k destinations

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

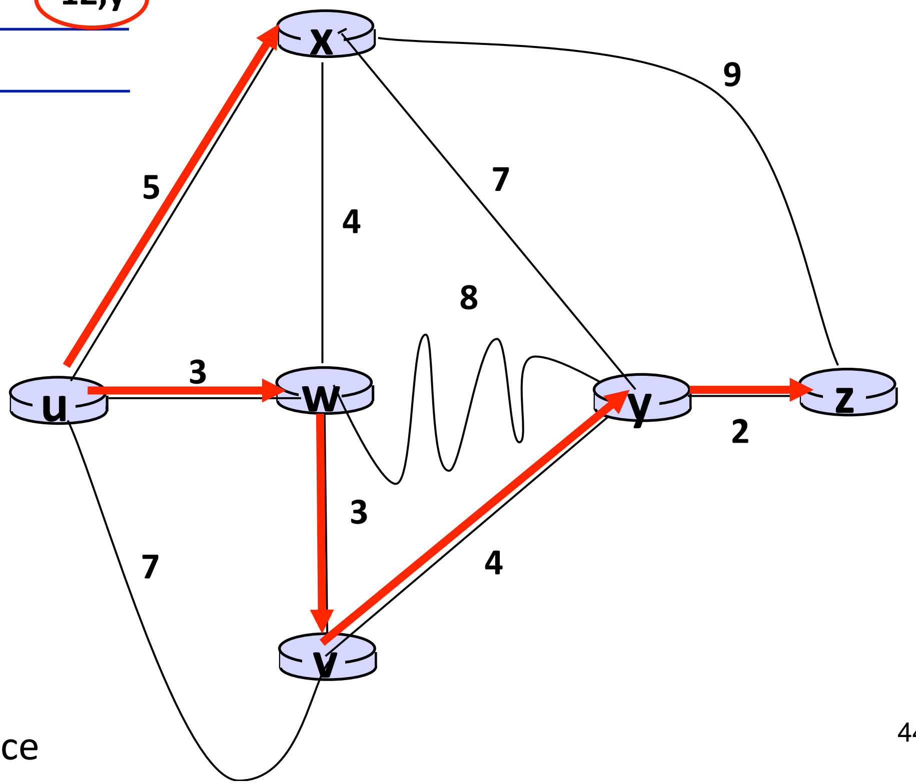
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
     D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
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					

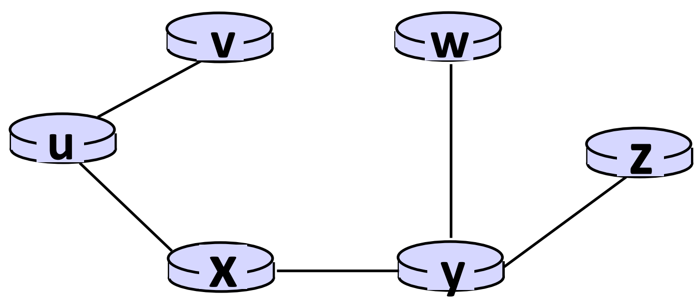
Notes:

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



Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:

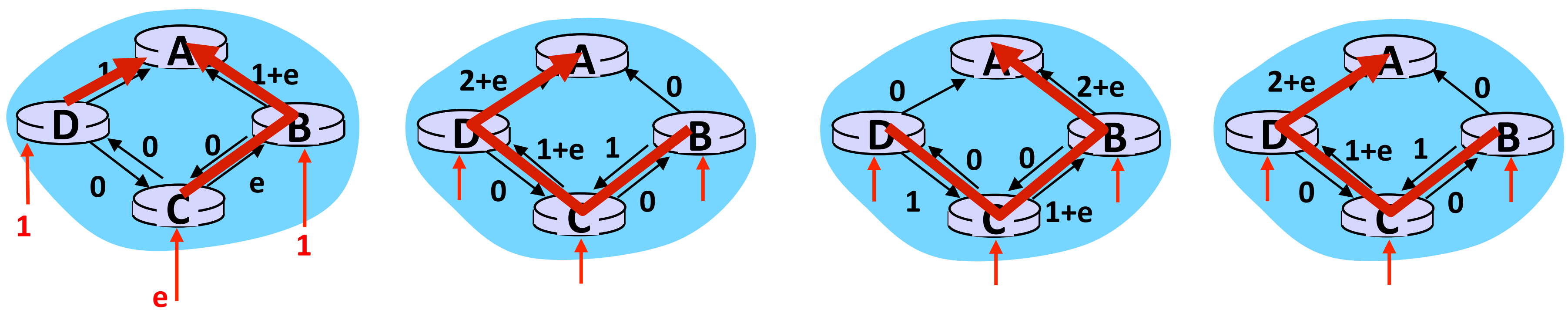


resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Dijkstra's algorithm, discussion

- *Algorithm complexity:* n nodes
  - each iteration: need to check all nodes, w, not in N
  - n(n-1)/2 comparisons: $O(n^2)$
  - more efficient implementations possible: $O(n*\log(n))$



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

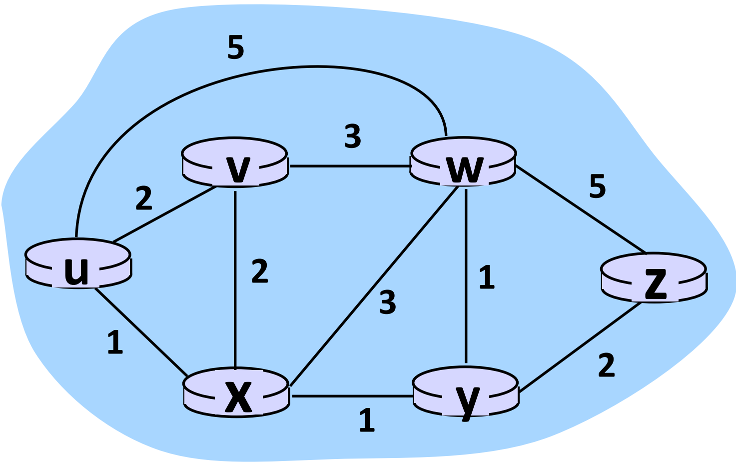
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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:

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

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:

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

- **under minor, natural conditions, 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:

wait **for (change in local link cost or msg from neighbor)**

recompute **estimates**

if DV to any dest has changed, *notify* neighbors

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

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

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

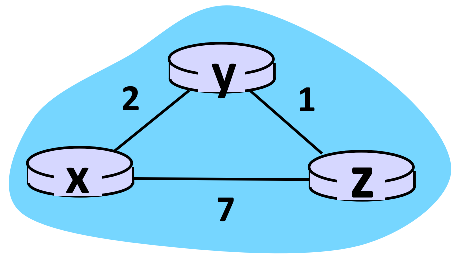
node y table

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

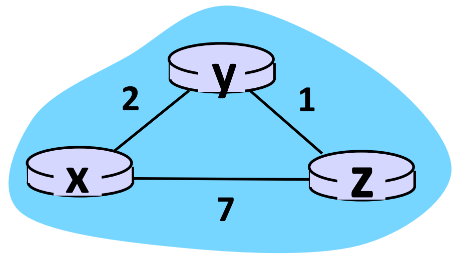
node z table

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

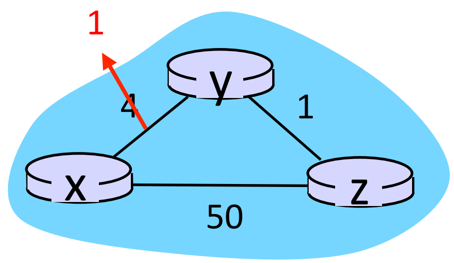
time →



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.

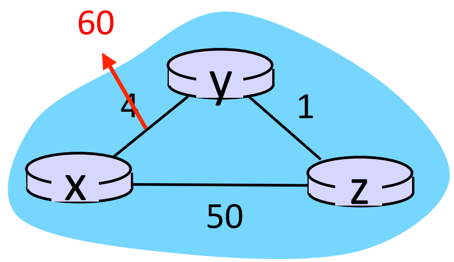
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

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*.

Distance vector: link cost changes

link cost changes:

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



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)
- will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

Message complexity

- ***LS:*** with n nodes, E links, O(nE) messages sent
- ***DV:*** exchange between neighbors only, number of exchanged messages varies

Speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) messages
  - Micro-loops
- ***DV:*** convergence time varies
  - Potential transient routing loops
  - count-to-infinity problem

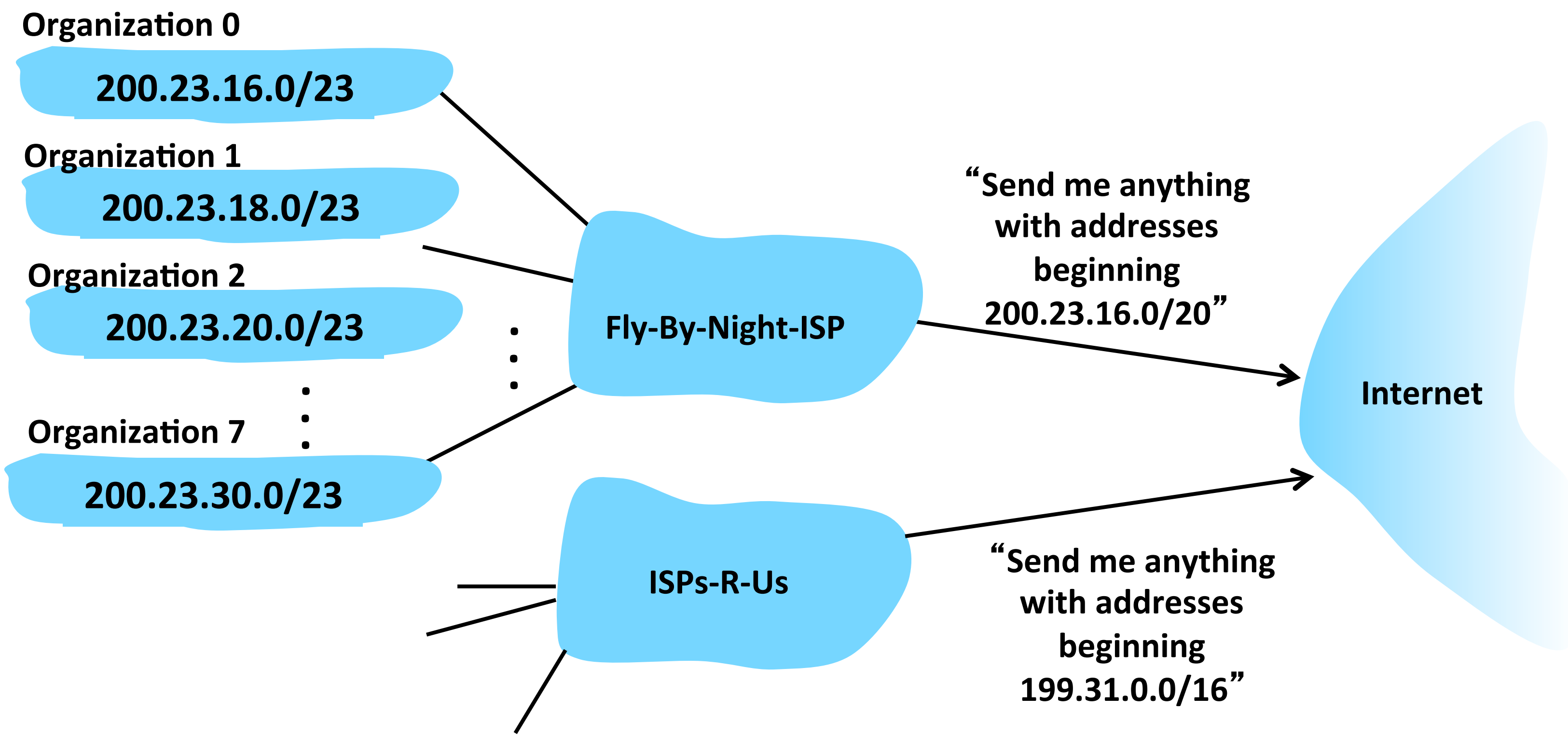
Robustness: what happens if router malfunctions?

- *LS:*
  - node can advertise incorrect *link* cost
  - each node computes only its *own* table => local
- *DV:*
  - DV node can advertise incorrect *path* cost
  - each node's table used by others

Errors propagate through the whole network!

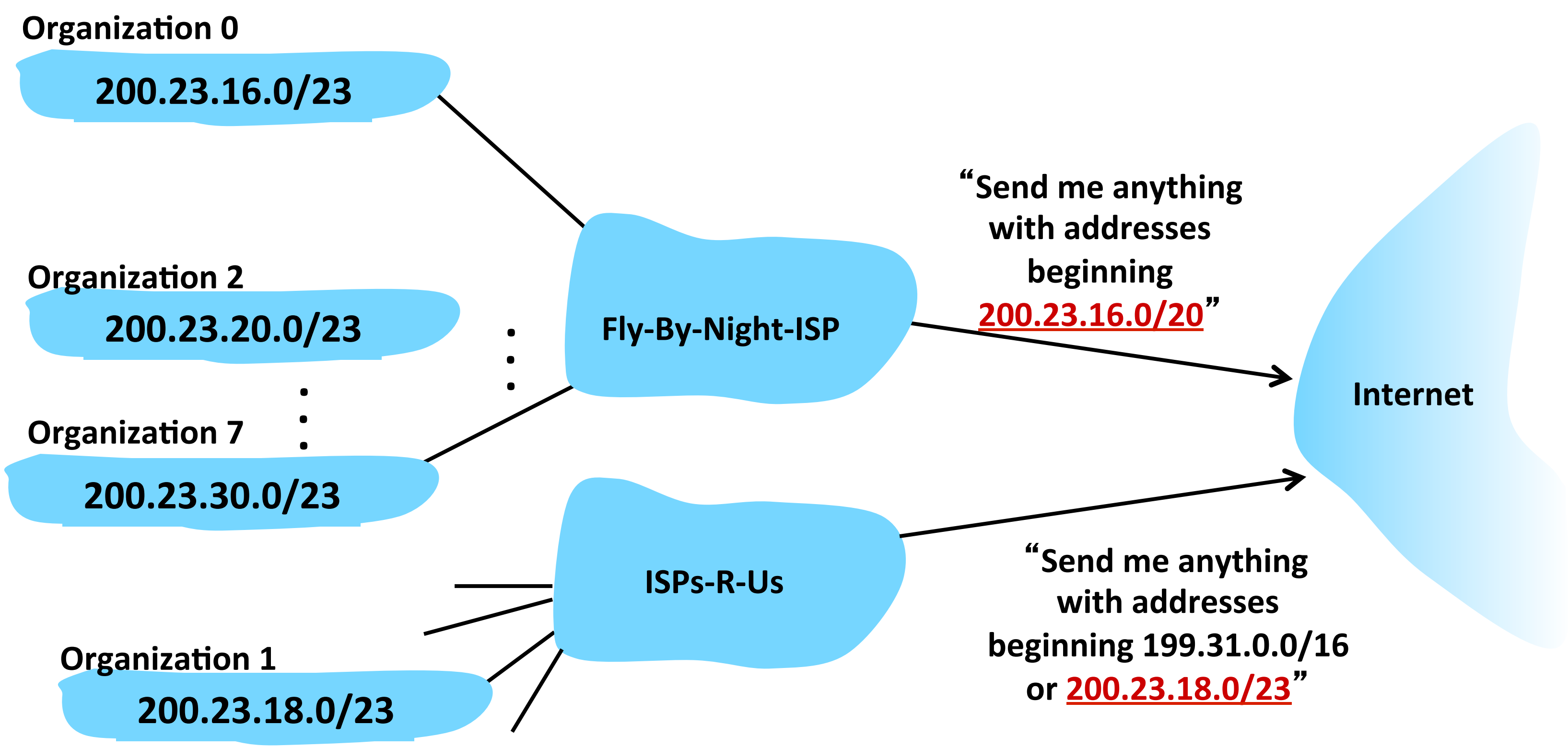
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



Hierarchical routing

Our routing study thus far is an idealization

- **all routers identical**
- **network "flat"**

… *not* true in practice

Scale: with 600 million destinations:

- Can't store all destinations in routing tables!
- Routing table exchange would take too much time!

Administrative autonomy

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

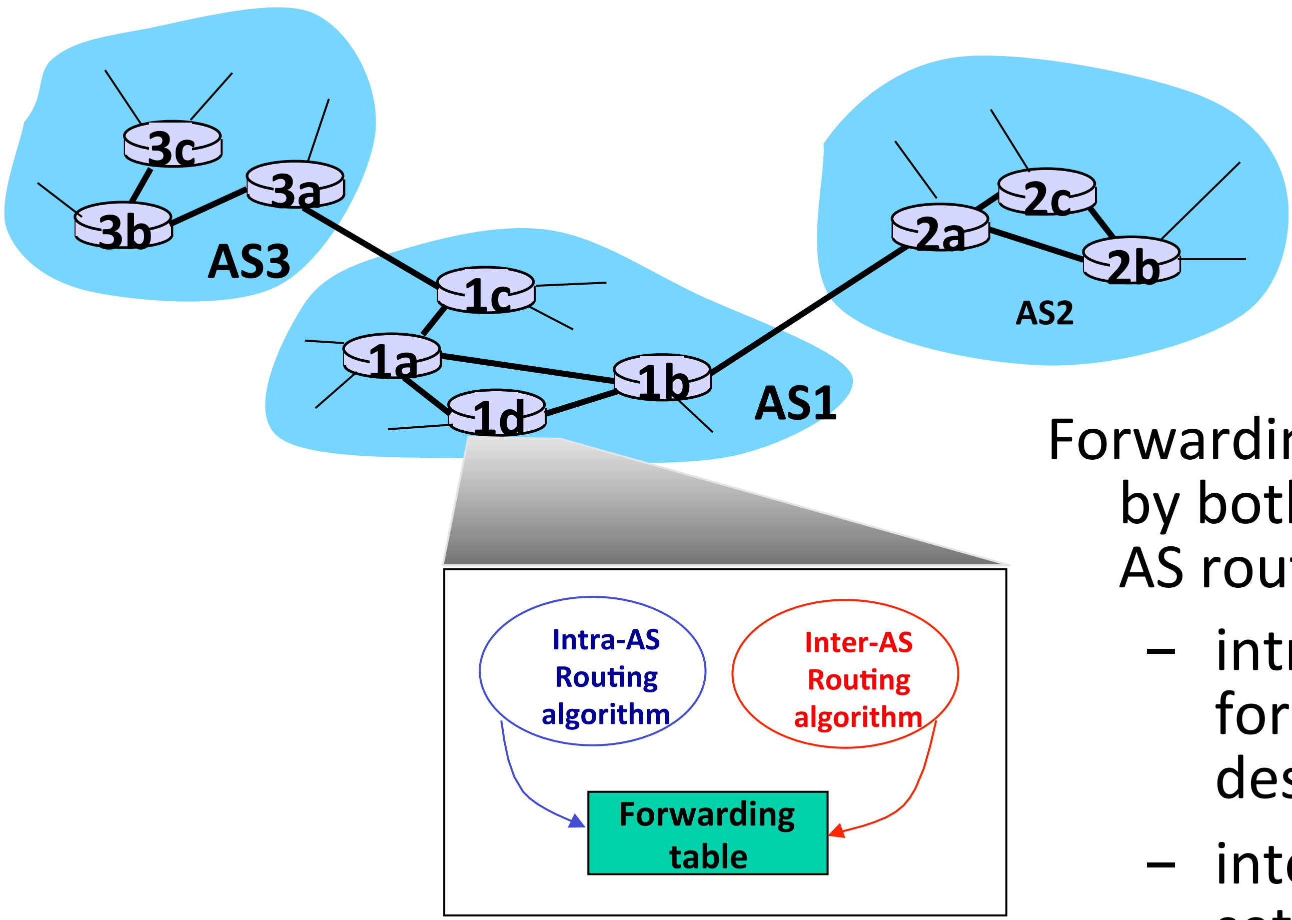
Hierarchical routing

- 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(s) to router in another AS

Interconnected ASes



Forwarding table configured by both intra- **and** inter-AS routing algorithm

- intra-AS sets entries for internal destinations
- inter-AS & intra-AS sets entries for external destinations

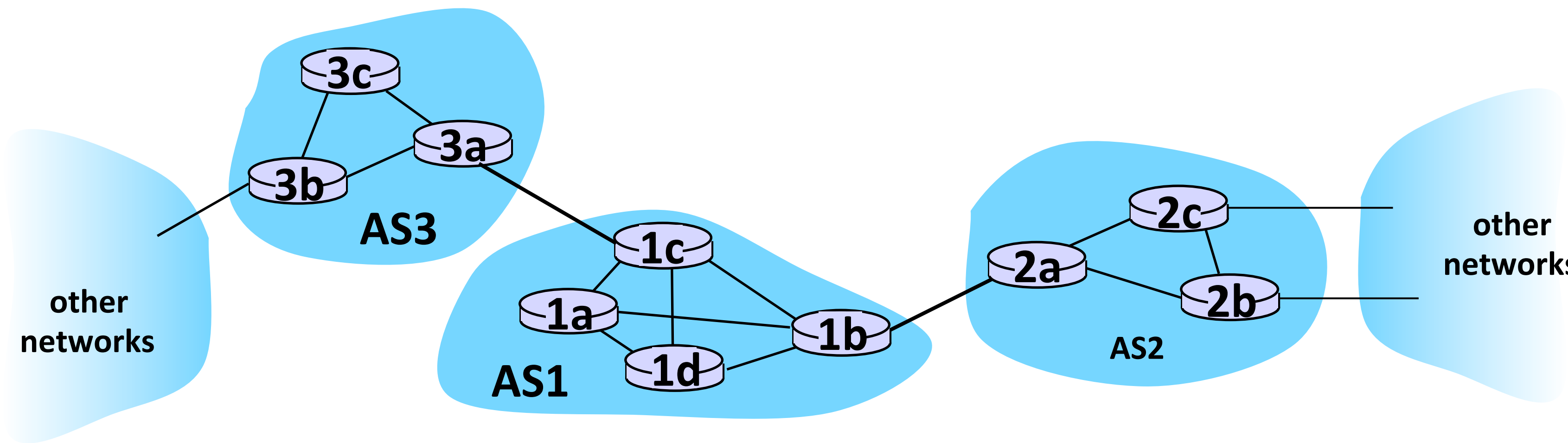
Inter-AS tasks

- 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 destinations are reachable through AS2 and AS3
2. Propagate this reachability information to all routers in AS1

= job of inter-AS routing!



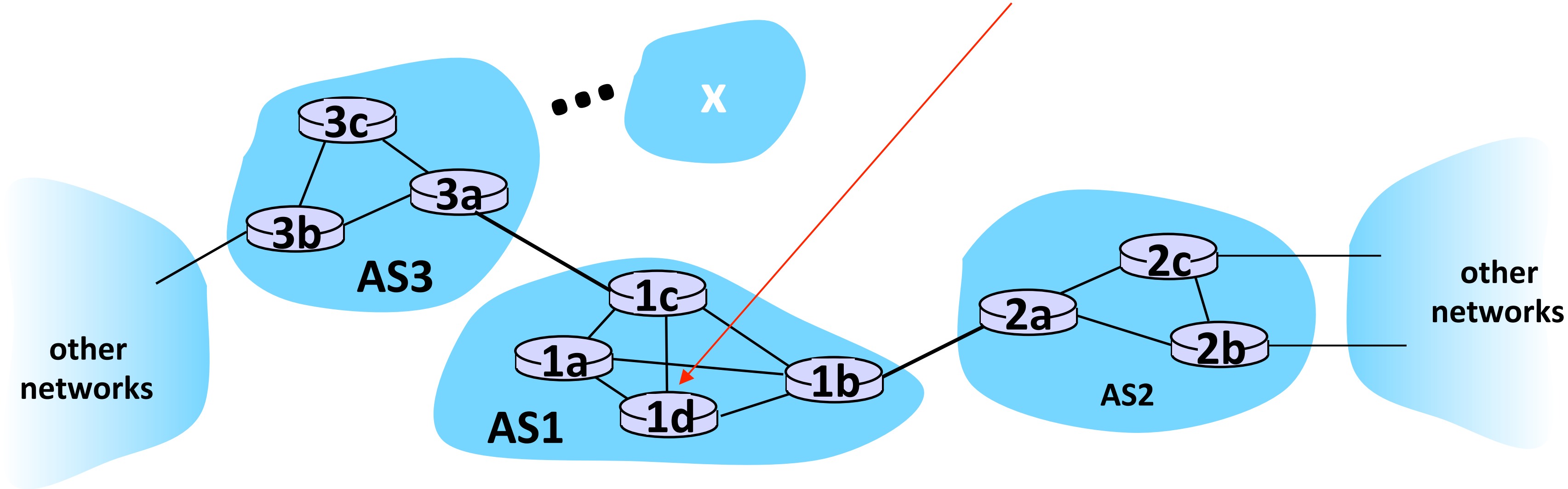
Example: setting forwarding table in router 1d

- 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:

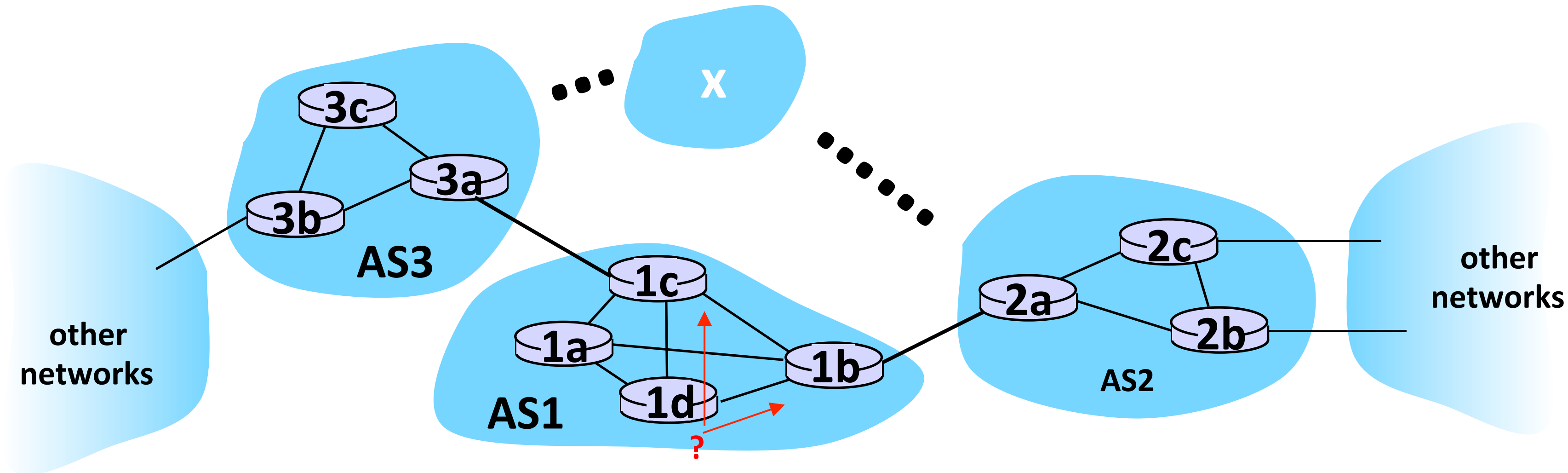
  Router 1d installs forwarding table entry *(x,I)*



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 destination x

This is also the job of the 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 towards which gateway it should forward packets for destination x

  This is also the job of the inter-AS routing protocol!
- *Hot potato routing: send* packet towards closest of two routers.

learn from inter-AS protocol that subnet *x* is reachable via multiple gateways → **use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways** → **hot potato routing: choose the gateway that has the smallest least cost** → **determine from forwarding table the interface *I* that leads to least-cost gateway. Enter *(x,I)* in forwarding table**

The Network layer

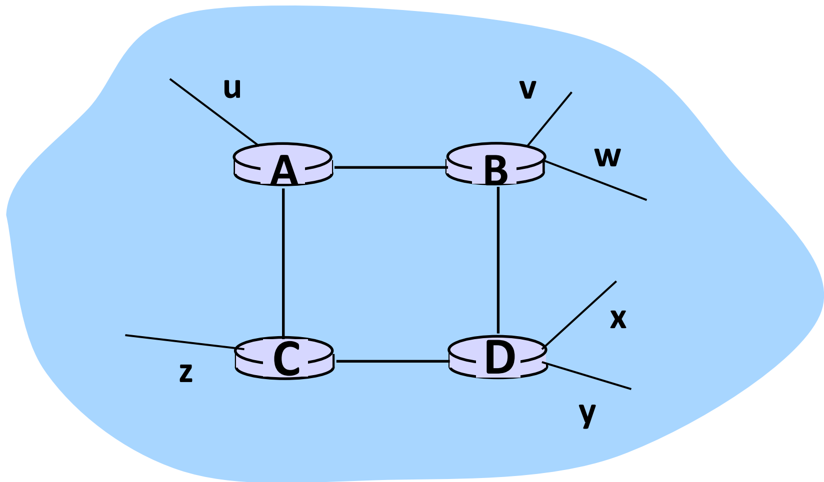
- Introduction
- IP router
- IP
- Routing: concepts
- **Routing: practice**

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
  - ISIS: Intermediate System-Intermediate System

RIP (Routing Information Protocol)

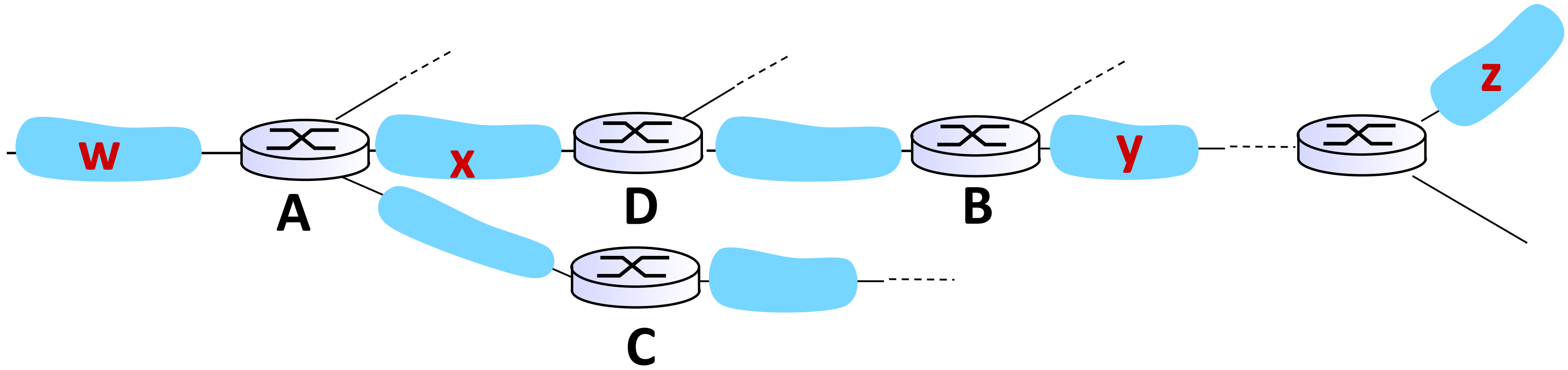
- Included in BSD-UNIX distribution in 1982
- Distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets* *(in IP addressing sense)*



from router A to destination *subnets:*

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

RIP: example



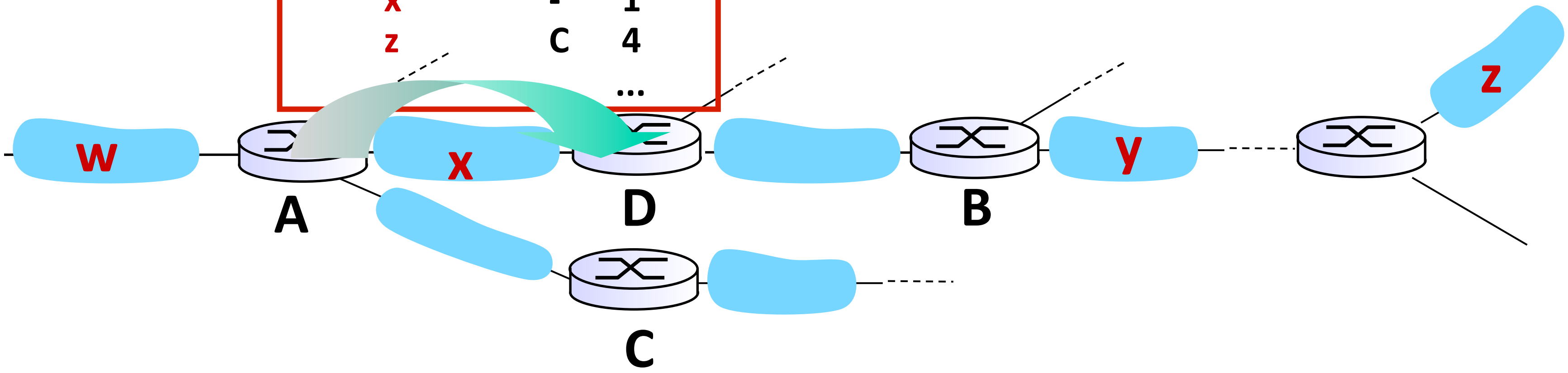
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

RIP: example

A-to-D advertisement

dest	next	hops
w	-	1
x	-	1
z	C	4
		...



routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
....	....	....

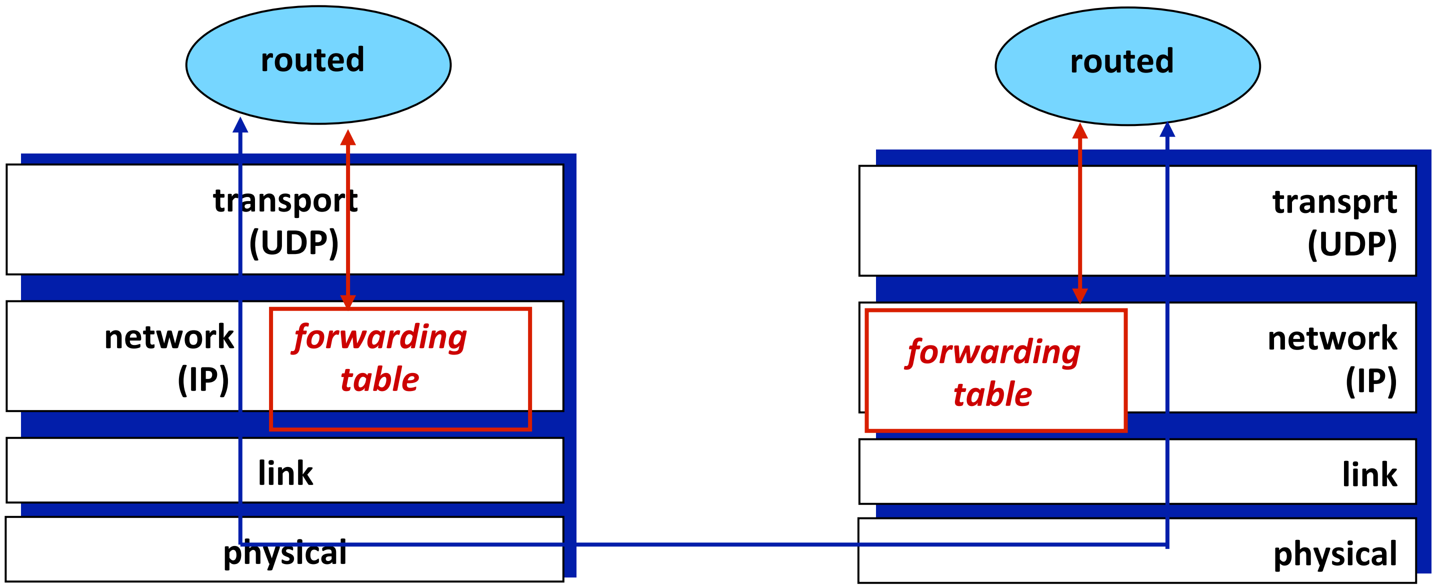
RIP: link failure, recovery

If no advertisement heard after 180 sec
➔neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

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)

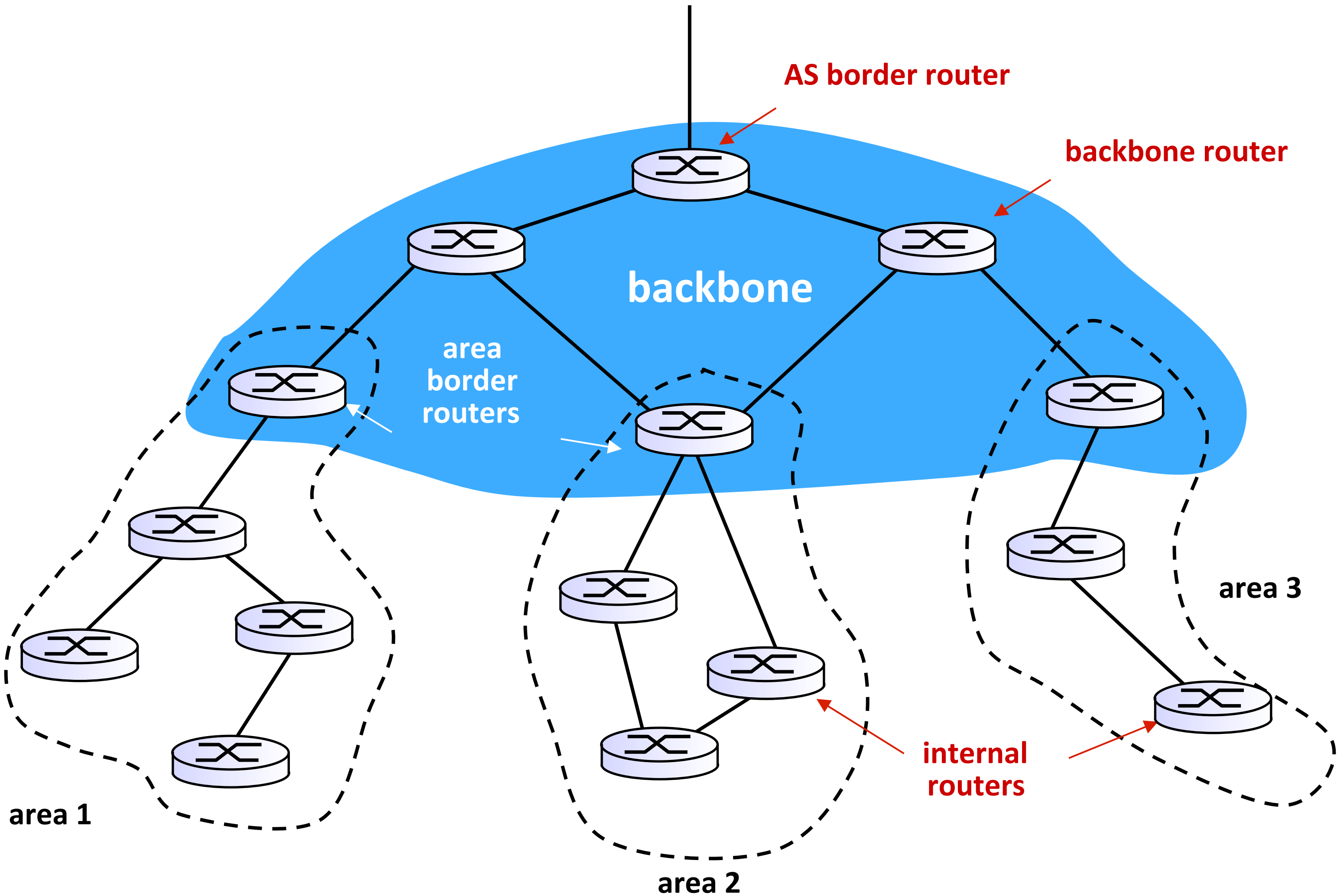
- "Open": publicly available
- Uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor
- Advertisements flooded to *entire* AS

  carried in OSPF messages directly over IP (rather than TCP or UDP
- *IS-IS routing* protocol: very similar to OSPF

OSPF "advanced" features

- *Security:* all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (ECMP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort ToS; high for real time ToS)
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical multi-area OSPF in large domains

Multi-area OSPF



Multi-area OSPF

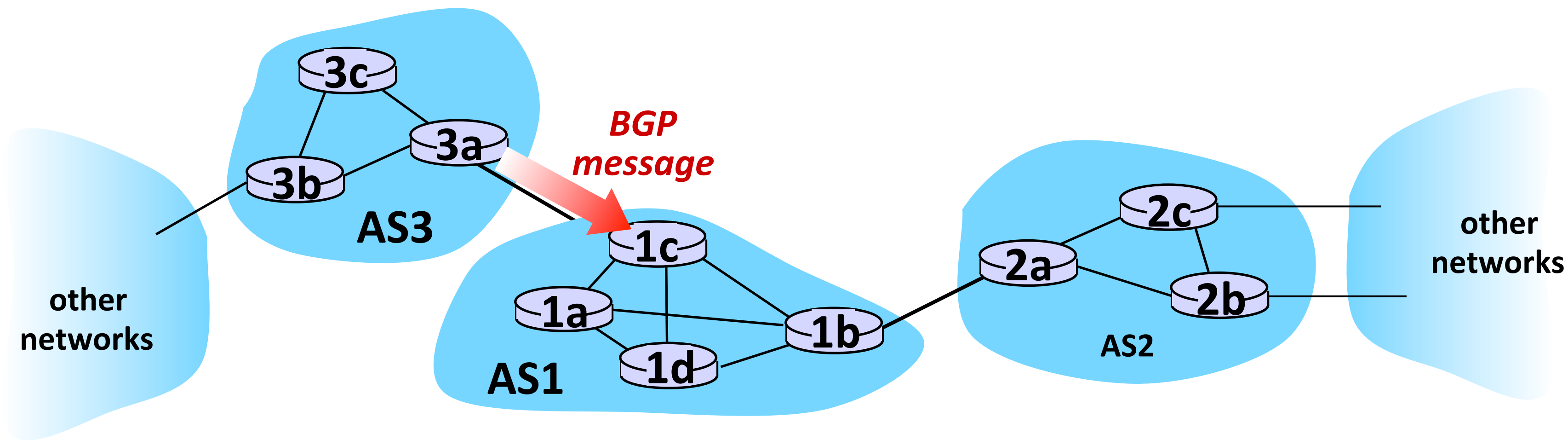
- *Two-level hierarchy:* local area, backbone.
  - Link-state advertisements only in area
  - Each node has detailed area topology; only know direction (shortest path) to nets in other areas
- *Area border routers:* "summarize" distances to prefixes in own area, advertise to other Area Border routers
- *Backbone routers:* run OSPF routing limited to backbone
- *AS Border routers:* connect to other AS's

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 prefix to advertise its existence to rest of Internet: *"I am here"*

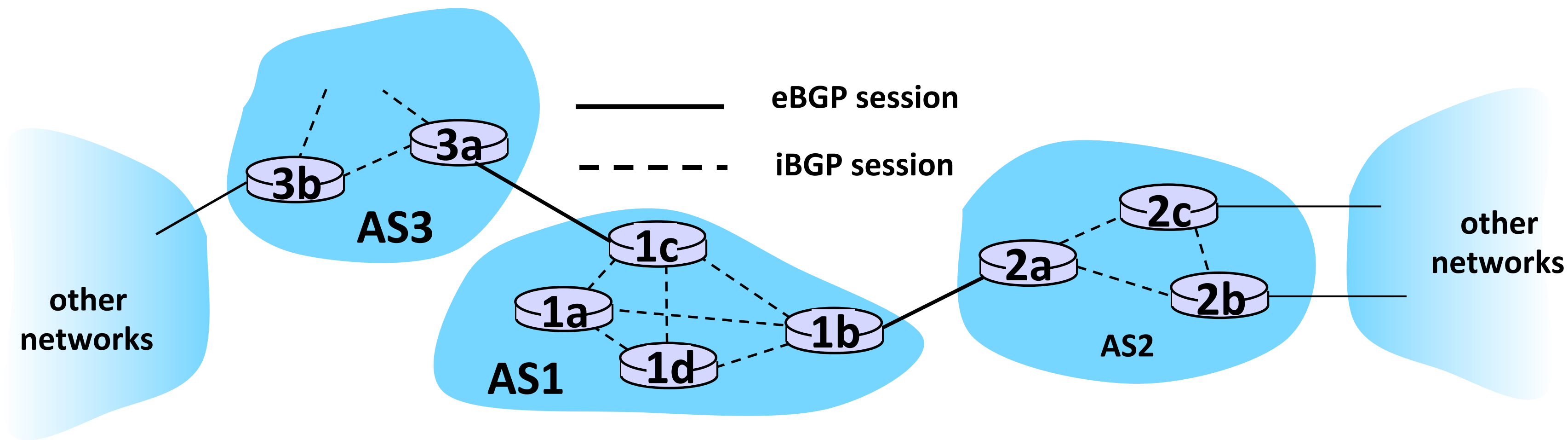
BGP basics

- 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



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 internal-AS exit router to next-hop AS
- Border router receiving route advertisement uses import policy to accept/decline

  e.g., never route through AS X *policy-based* routing

BGP route selection

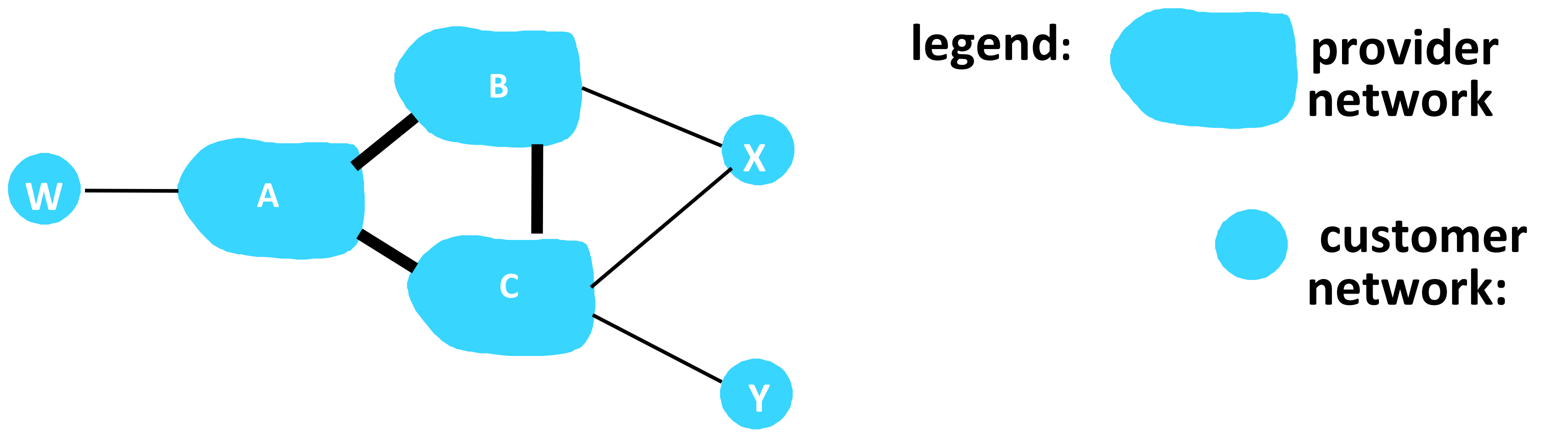
Router may learn about more than 1 route to destination AS, selects route based on:

1. Local preference value attribute: policy decision
2. Shortest AS-PATH
3. Closest NEXT-HOP router: hot potato routing
4. Additional criteria (tie-break)

BGP messages

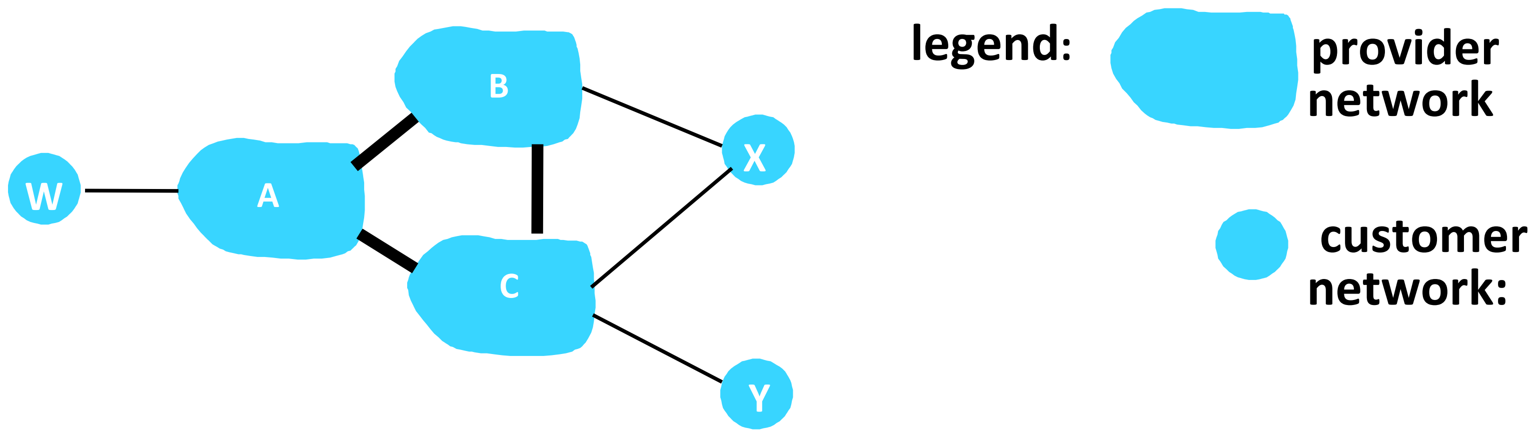
- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or implicitly withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: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

BGP routing policy



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

BGP routing policy (2)



- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no "revenue" for routing C-B-A-W 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 network
- 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