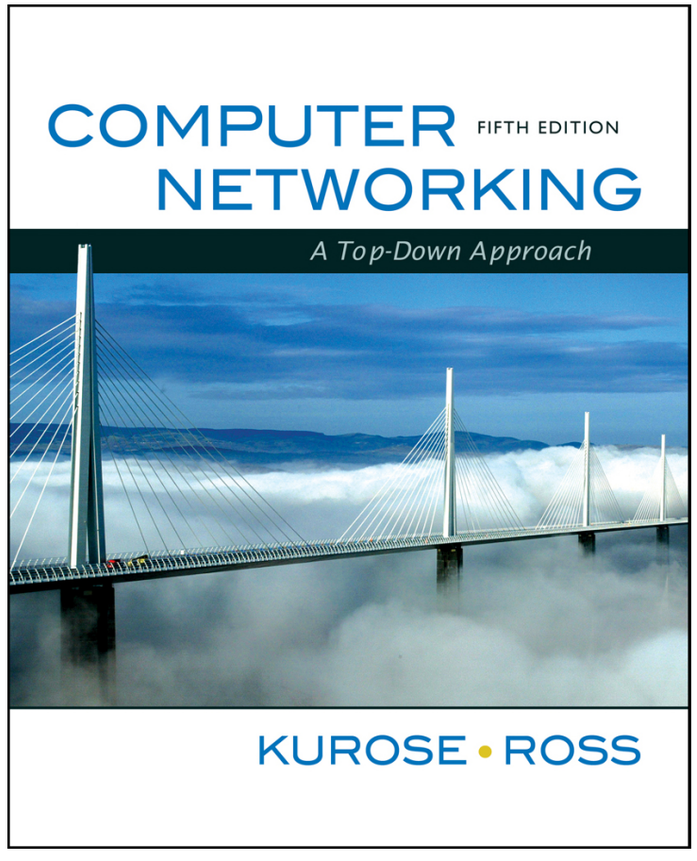
Module 3
Network Layer

Please note: Most of these slides come from this book. Note their copyright notice below…



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

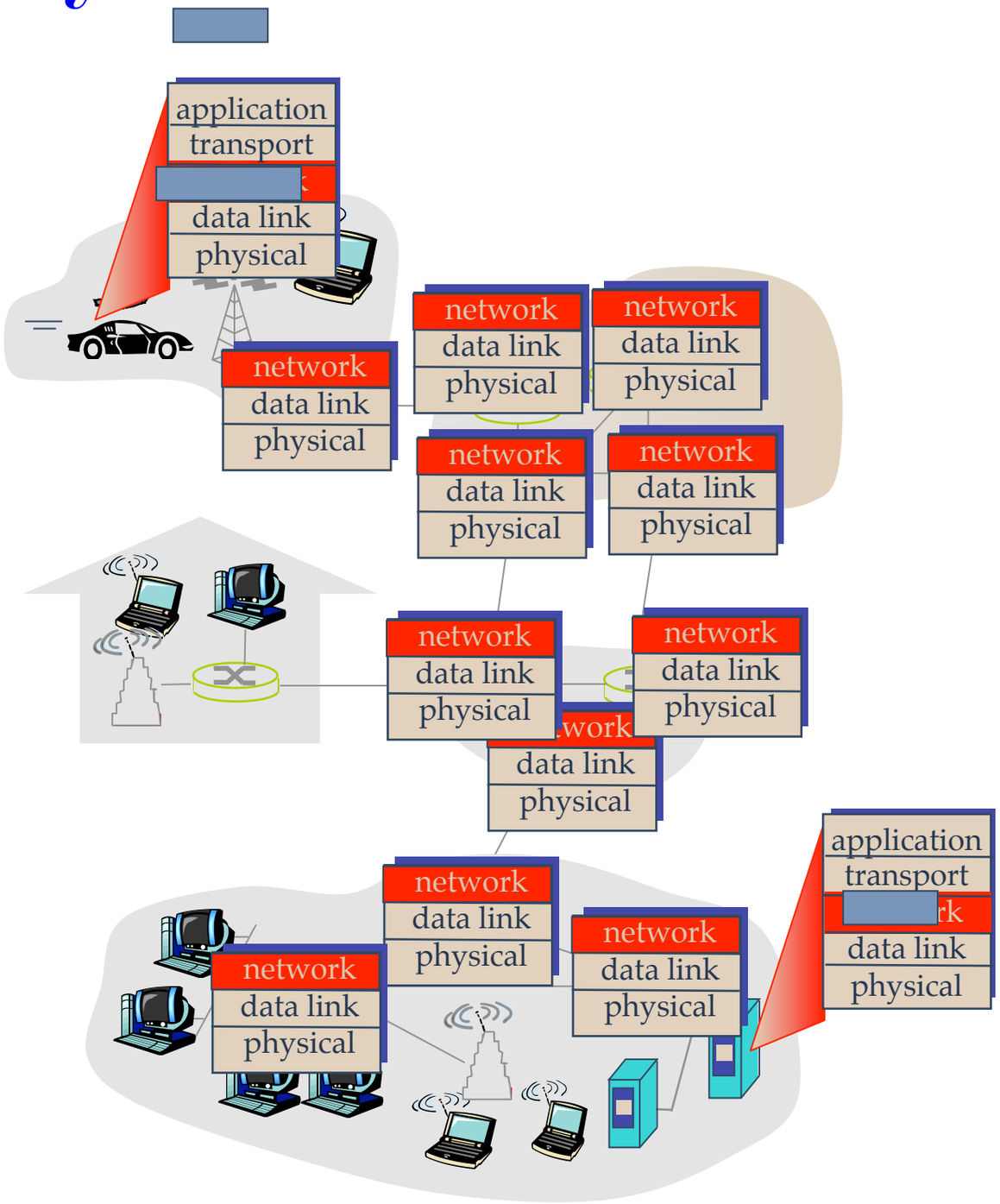
- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

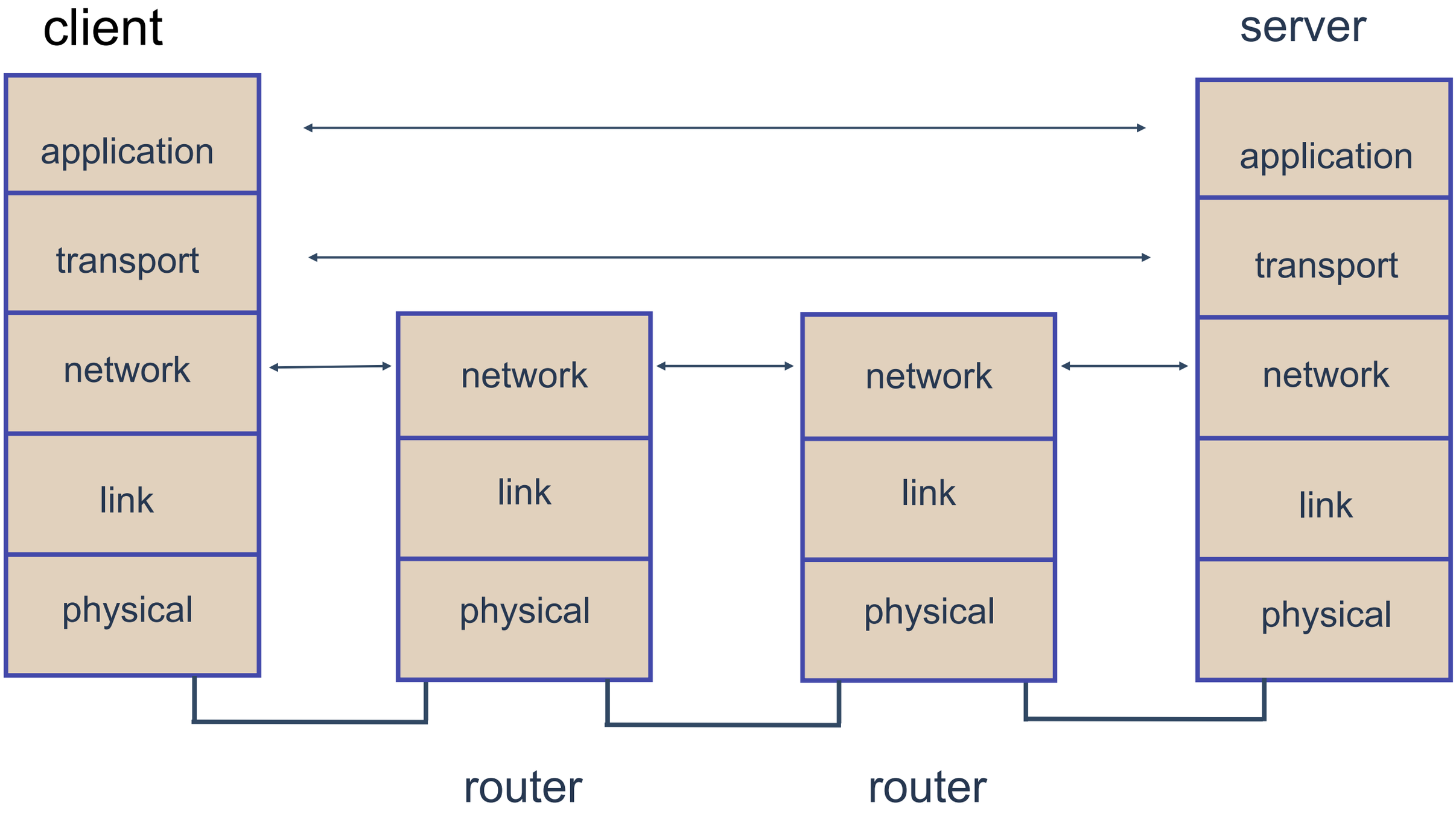
Computer Networking: A Top Down Approach 5th edition. Jim Kurose, Keith Ross Addison-Wesley, April 2009.

Network layer

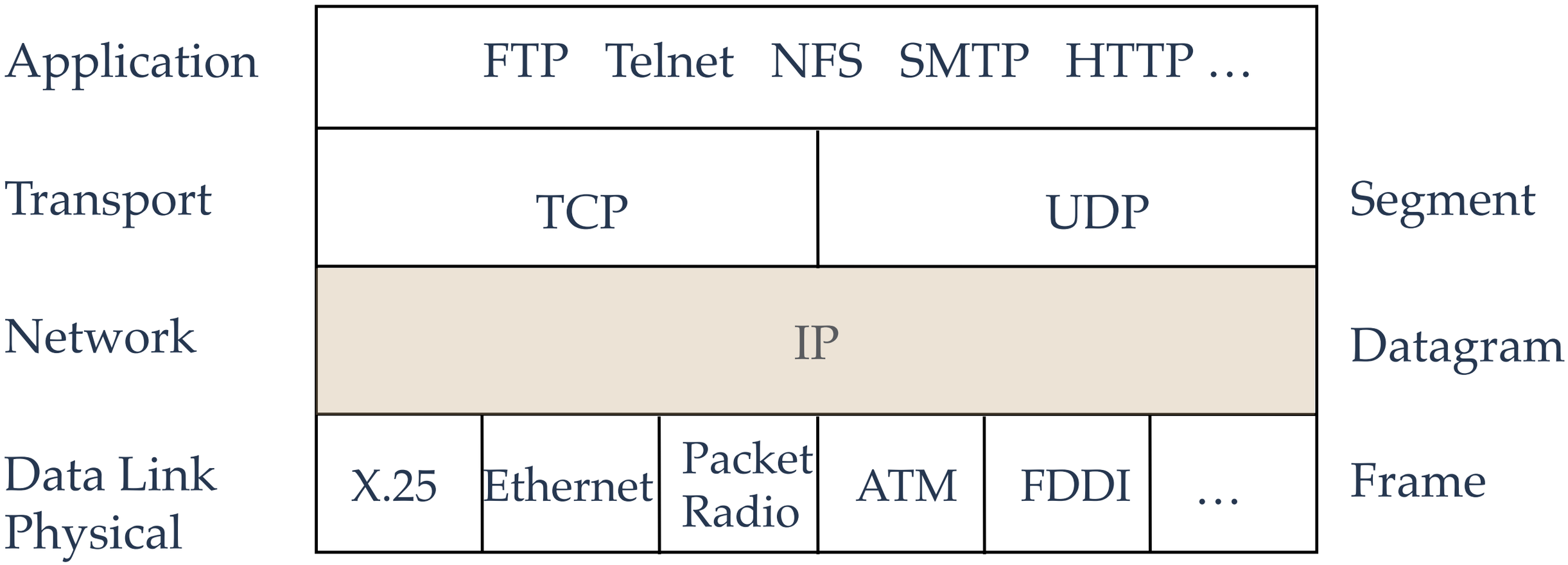
- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



Network Layer is Host-to-Host



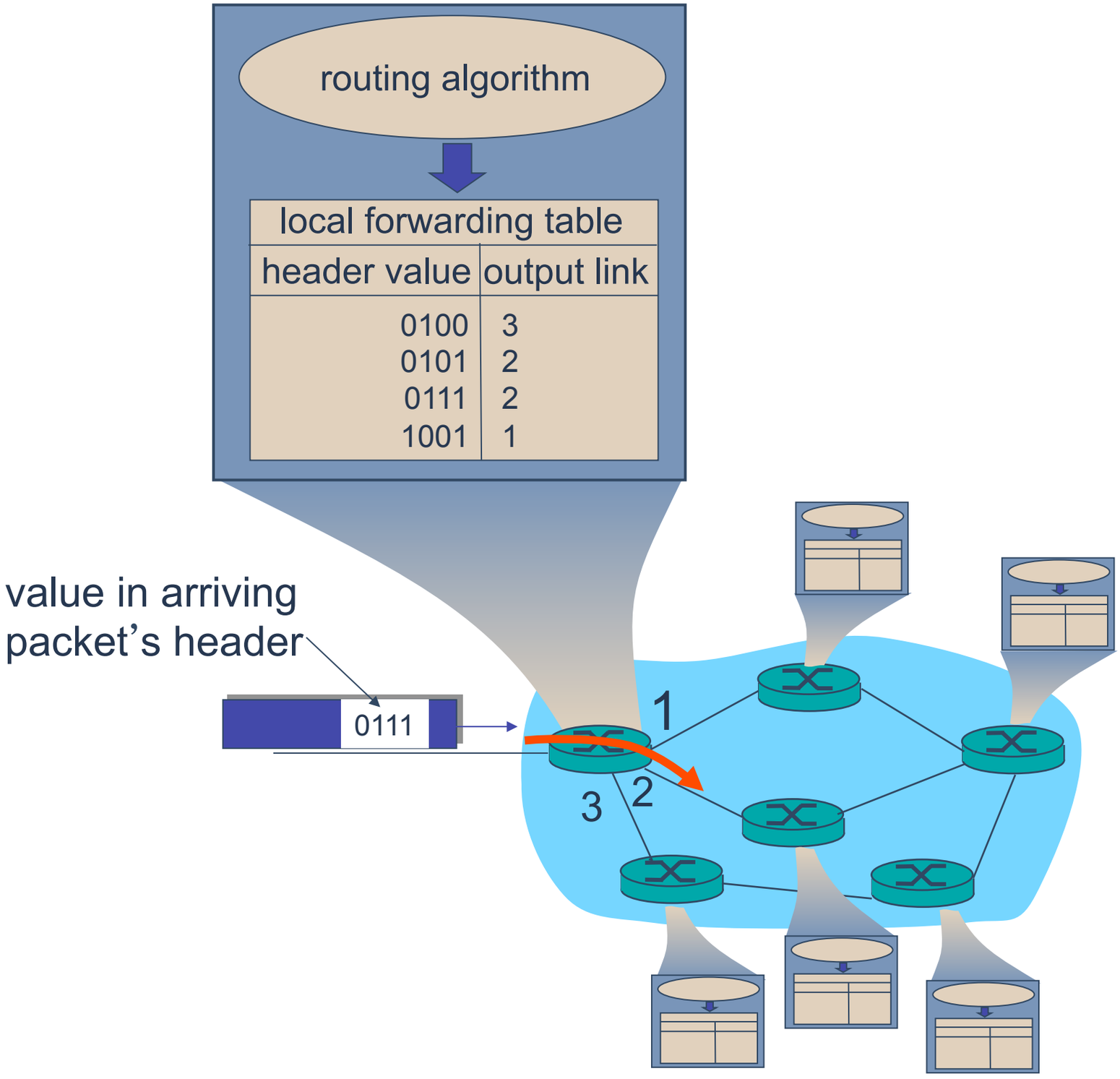
Internet Protocols



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 destination.
  - ➡ *routing algorithms*
- *Connection service:* before datagrams flow, two end hosts *and* intervening routers establish virtual connection (VC)
  - ➡ Needed in *some* network architectures: ATM, frame relay, X.25
  - ➡ Network vs transport layer connection service:
    - ✦ network: between two hosts (may also involve intervening routers in case of VCs)
    - ✦ transport: between two processes

Interplay Between Routing and Forwarding



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:

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
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

Network layer connection and connection-less service

- Datagram network provides network-layer connectionless service
- Virtual Circuit (VC) network provides network-layer connection service
- Analogous to the transport-layer services, but:
  - ➡ service: host-to-host
  - ➡ no choice: network provides one or the other
  - ➡ implementation: in network core

Virtual Circuits

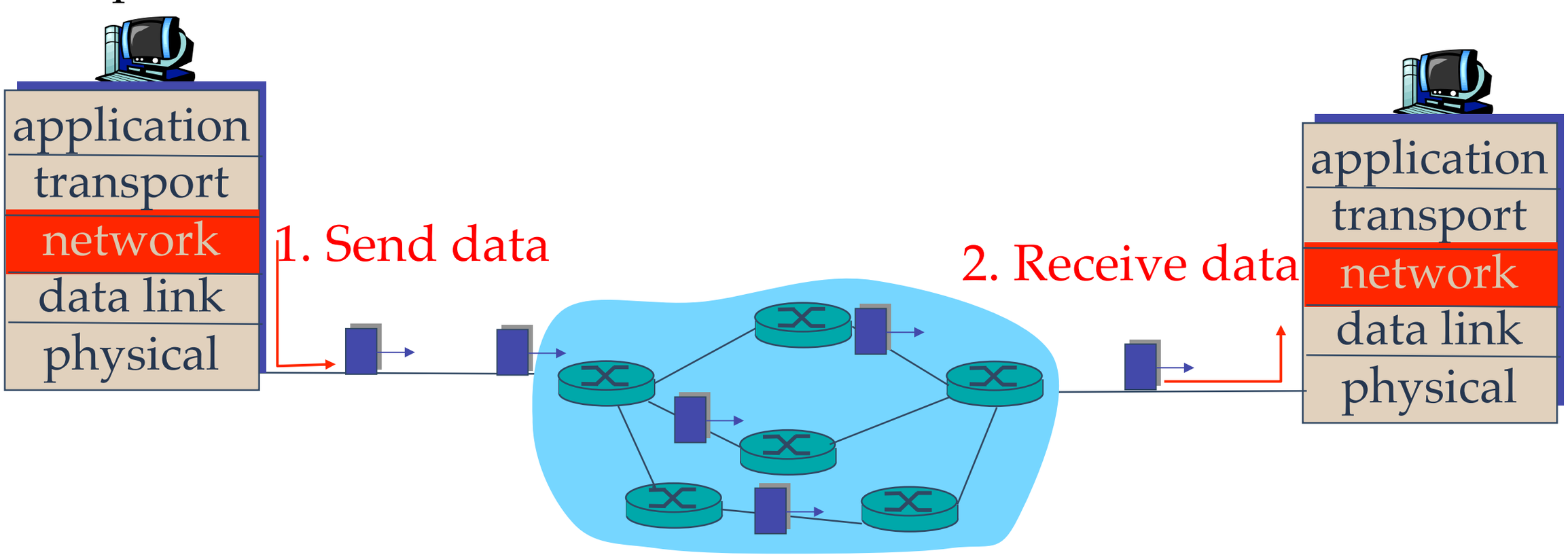
"source-to-destination path behaves much like telephone circuit"

- ➡ performance-wise
- ➡ network actions along source-to-destination path

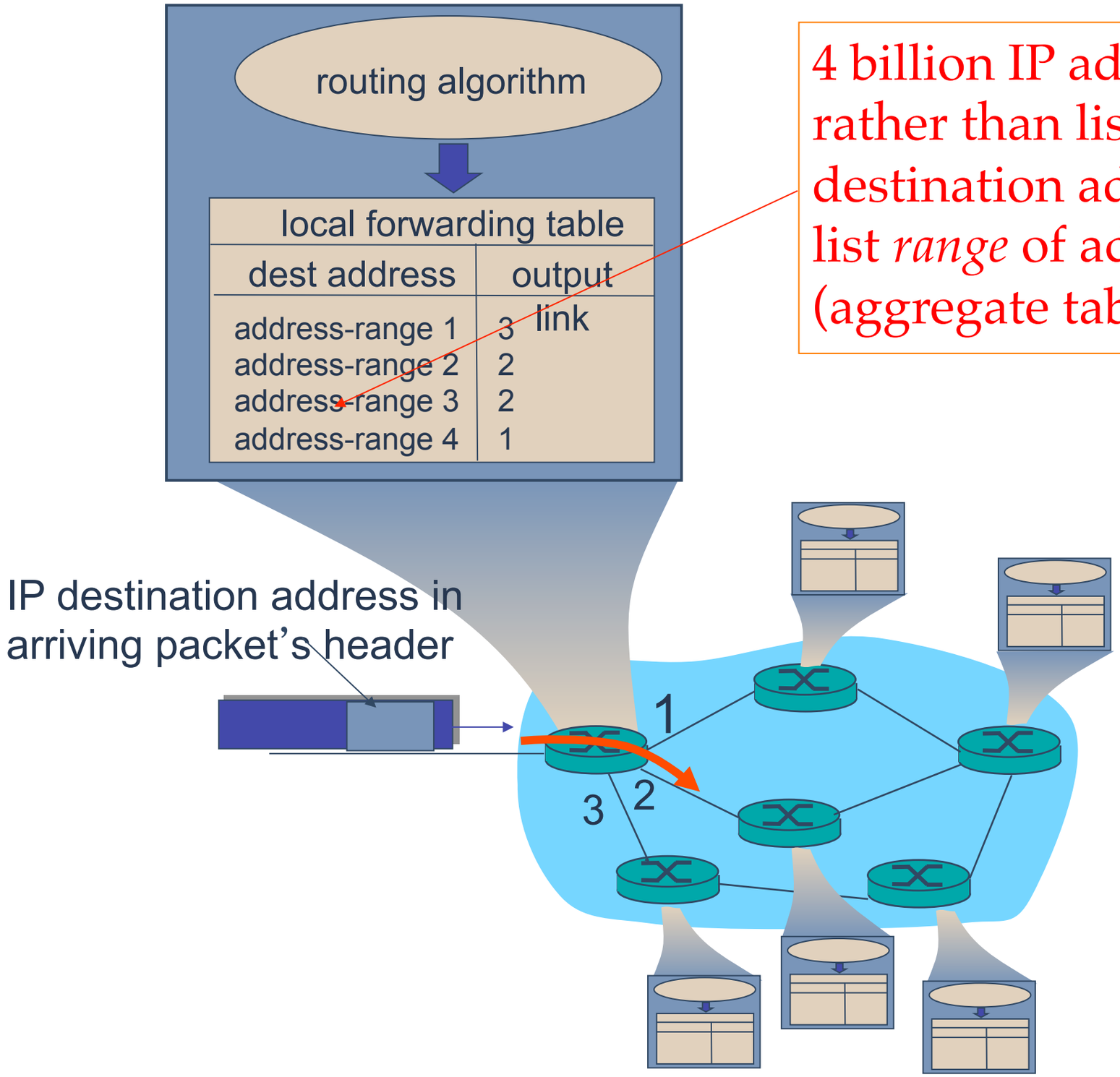
- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-destination path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

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
  - ➡ packets between same source-destination pair may take different paths



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

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

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?

Datagram or VC network: why?

Internet (datagram)

- data exchange among computers
  - ➡ "elastic" service, no strict timing req.
- "smart" end systems (computers)
  - ➡ can adapt, perform control, error recovery
  - ➡ simple inside network, complexity at "edge"
- many link types
  - ➡ different characteristics
  - ➡ uniform service difficult

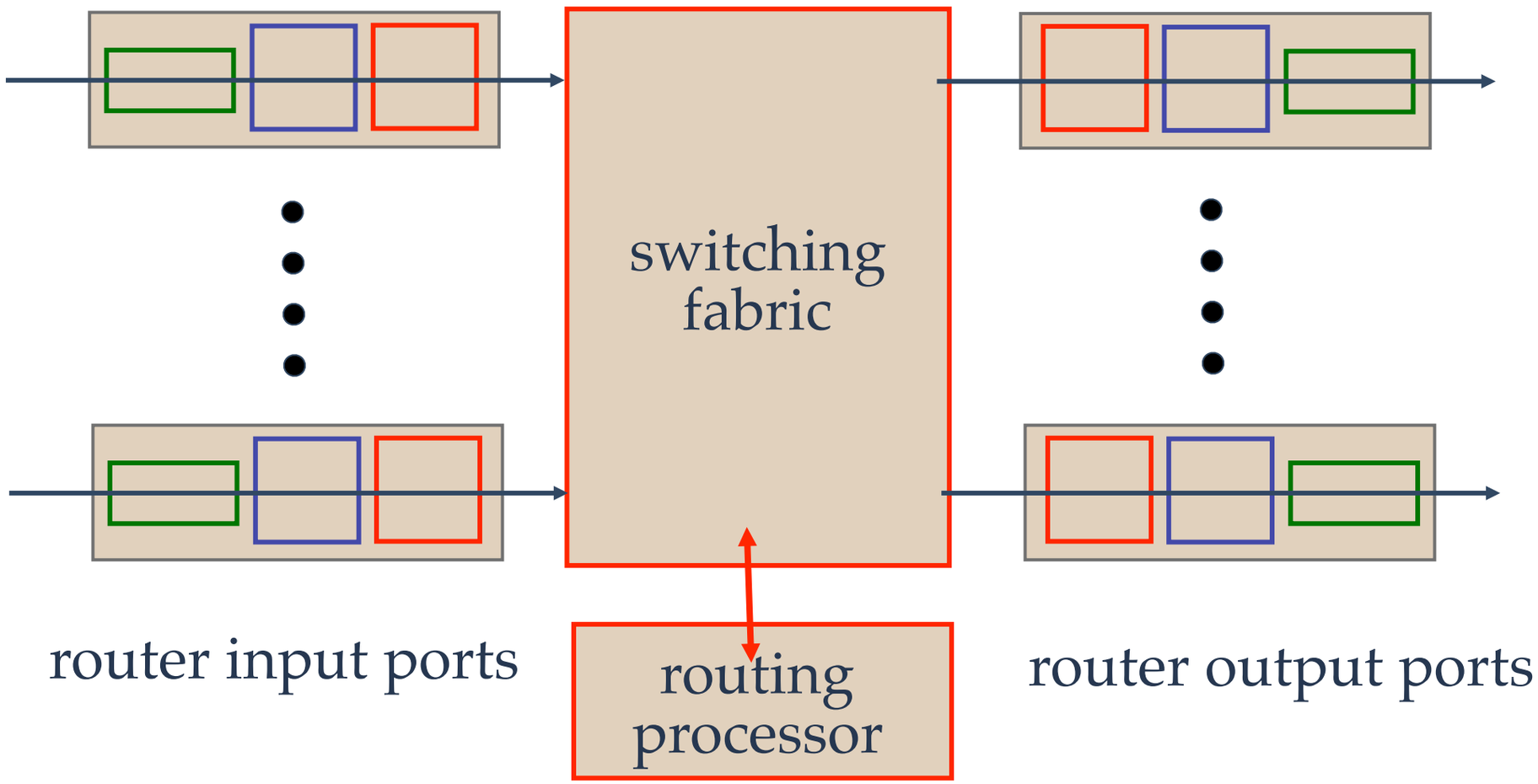
ATM (VC)

- evolved from telephony
- human conversation:
  - ➡ strict timing, reliability requirements
  - ➡ need for guaranteed service
- "dumb" end systems
  - ➡ telephones
  - ➡ complexity inside network

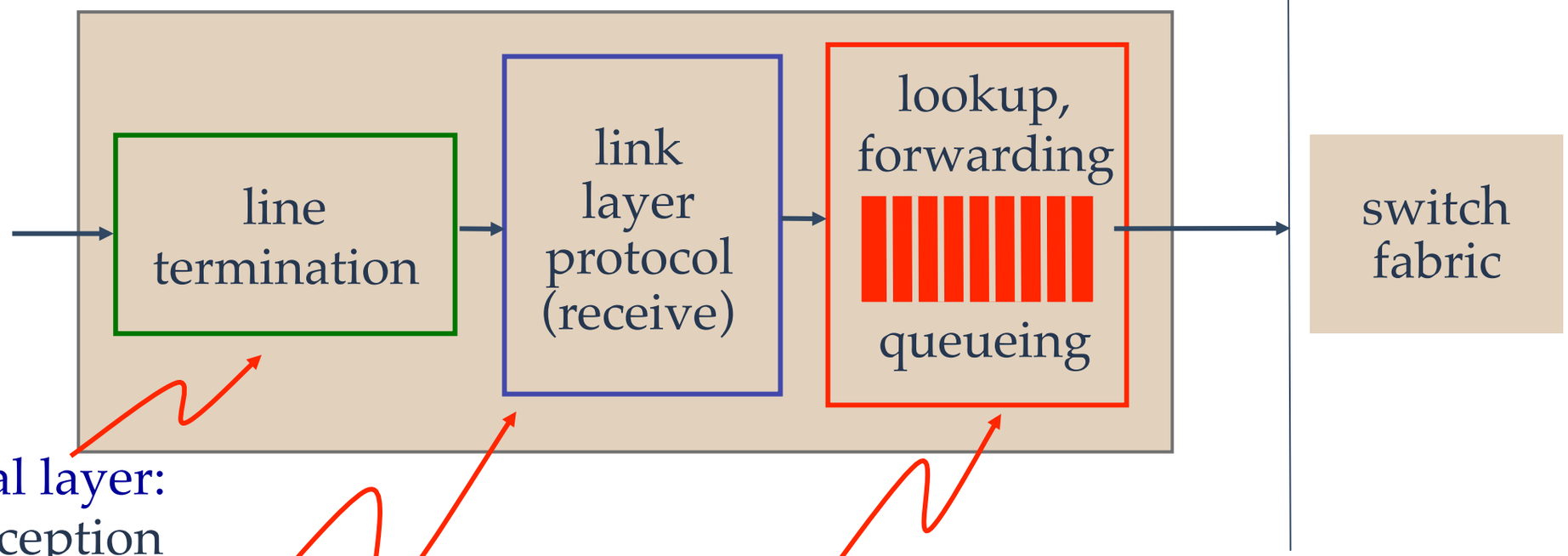
Router Architecture Overview

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

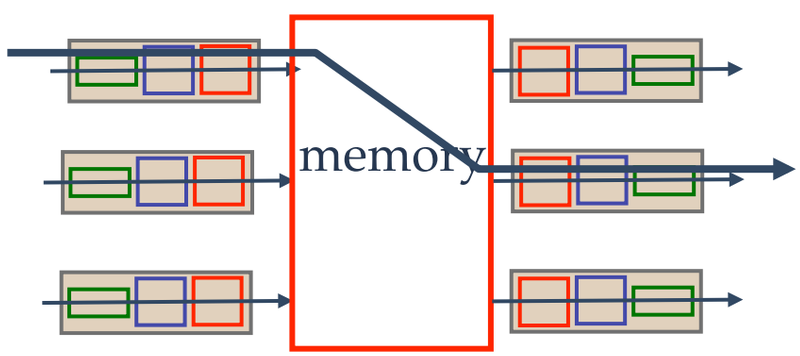
Data link layer:
e.g., Ethernet

Decentralized switching:

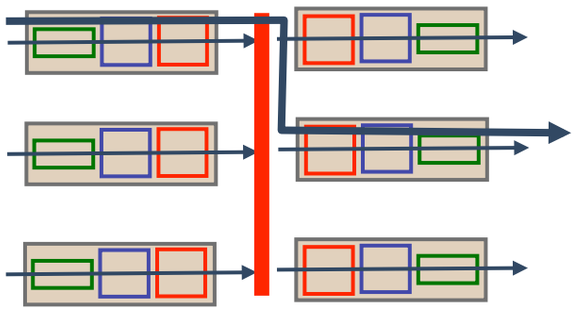
- given datagram destination, lookup output port using forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Switching fabrics

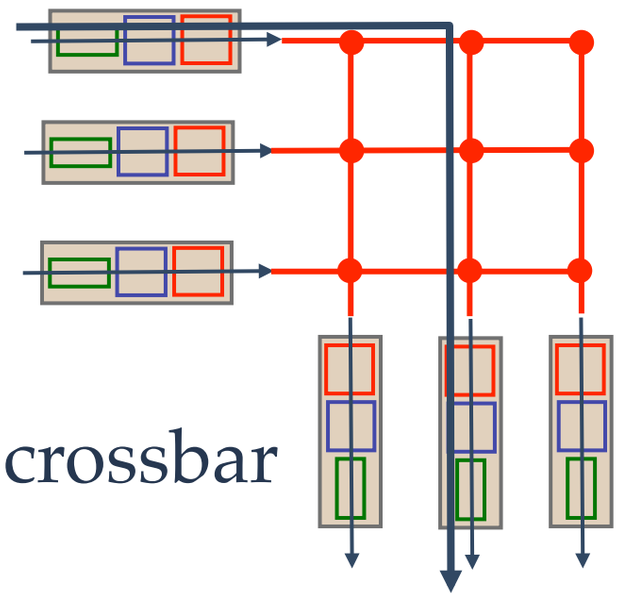
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



memory



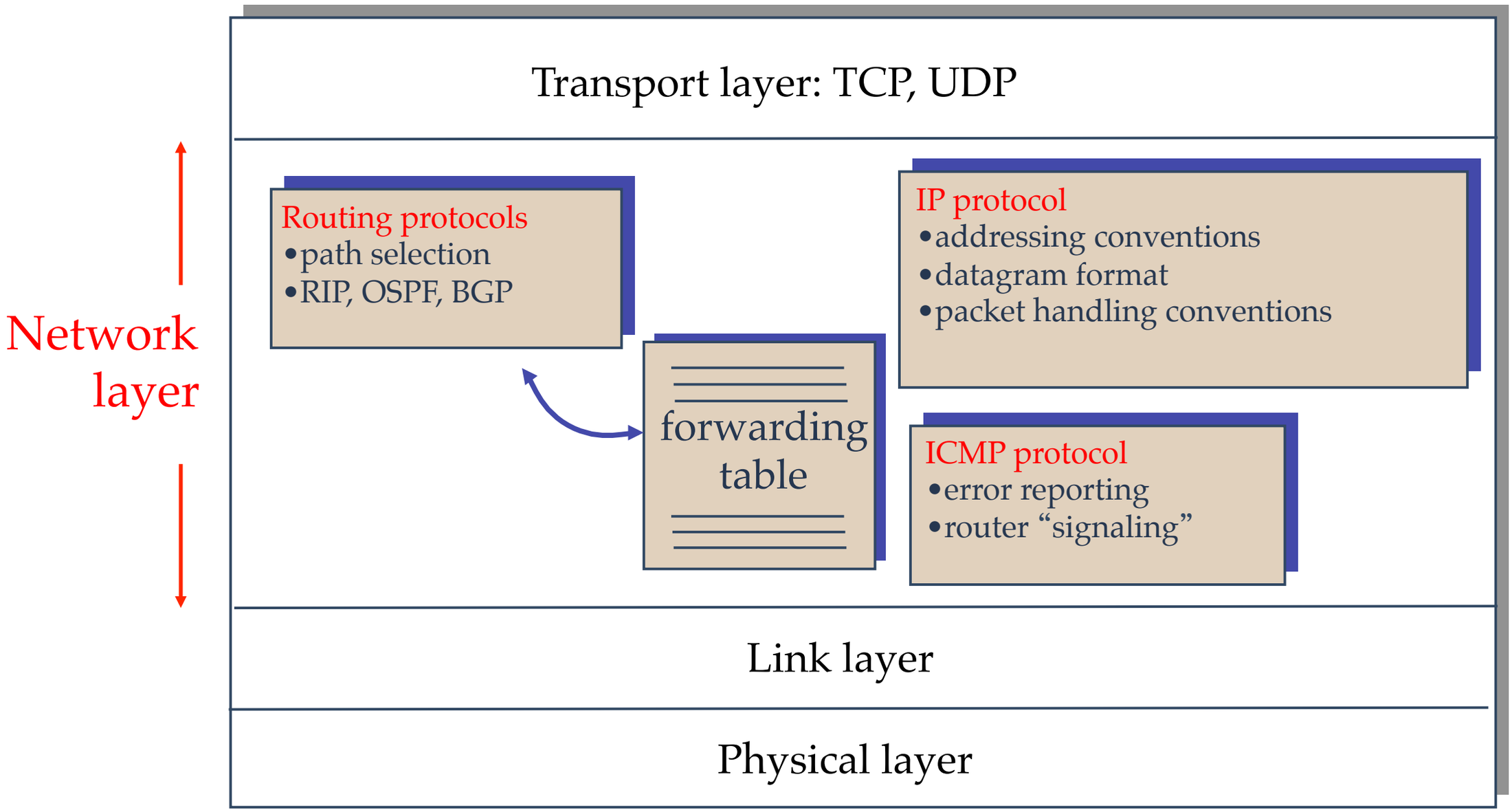
bus



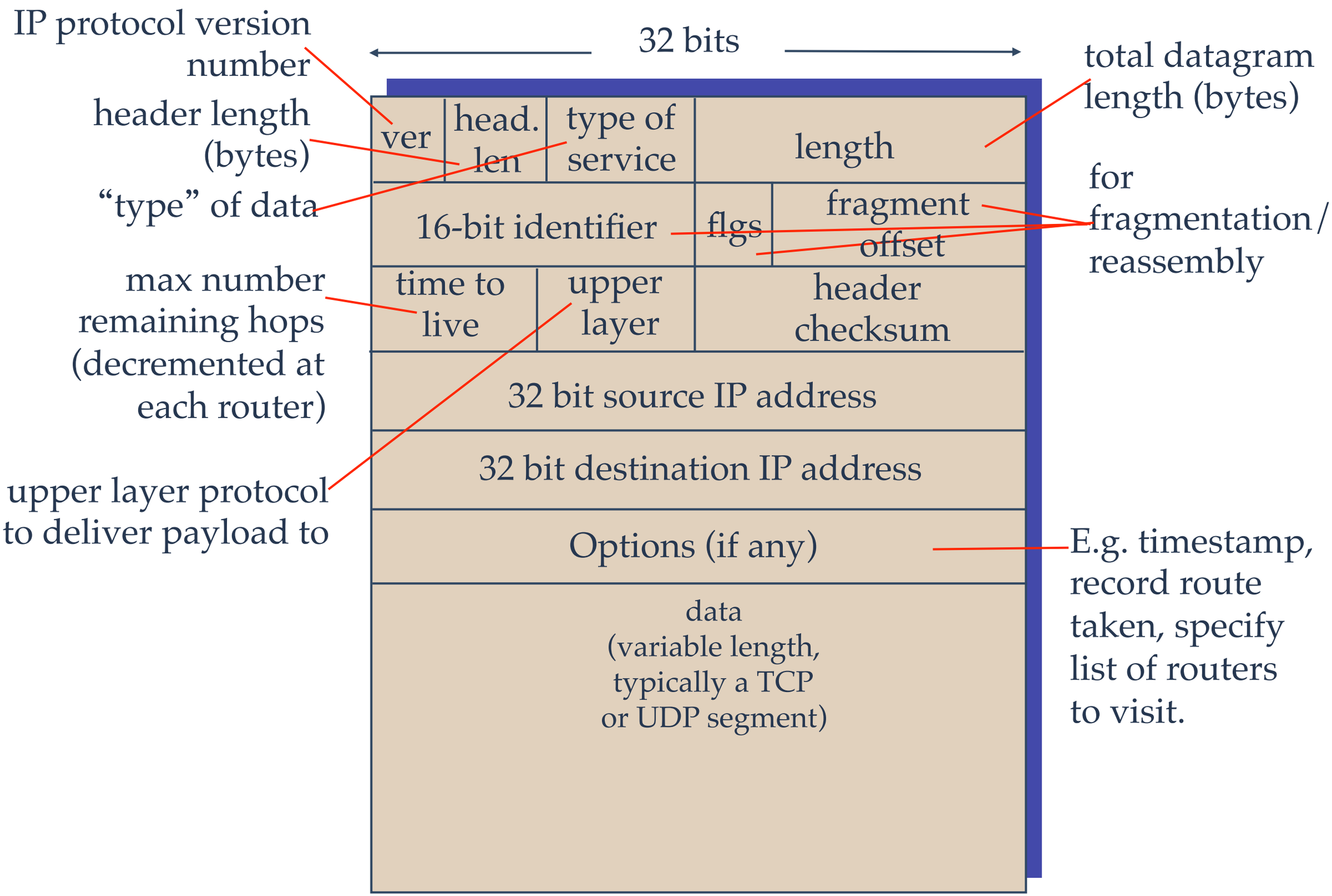
crossbar

The Internet Network layer

Host, router network layer functions:

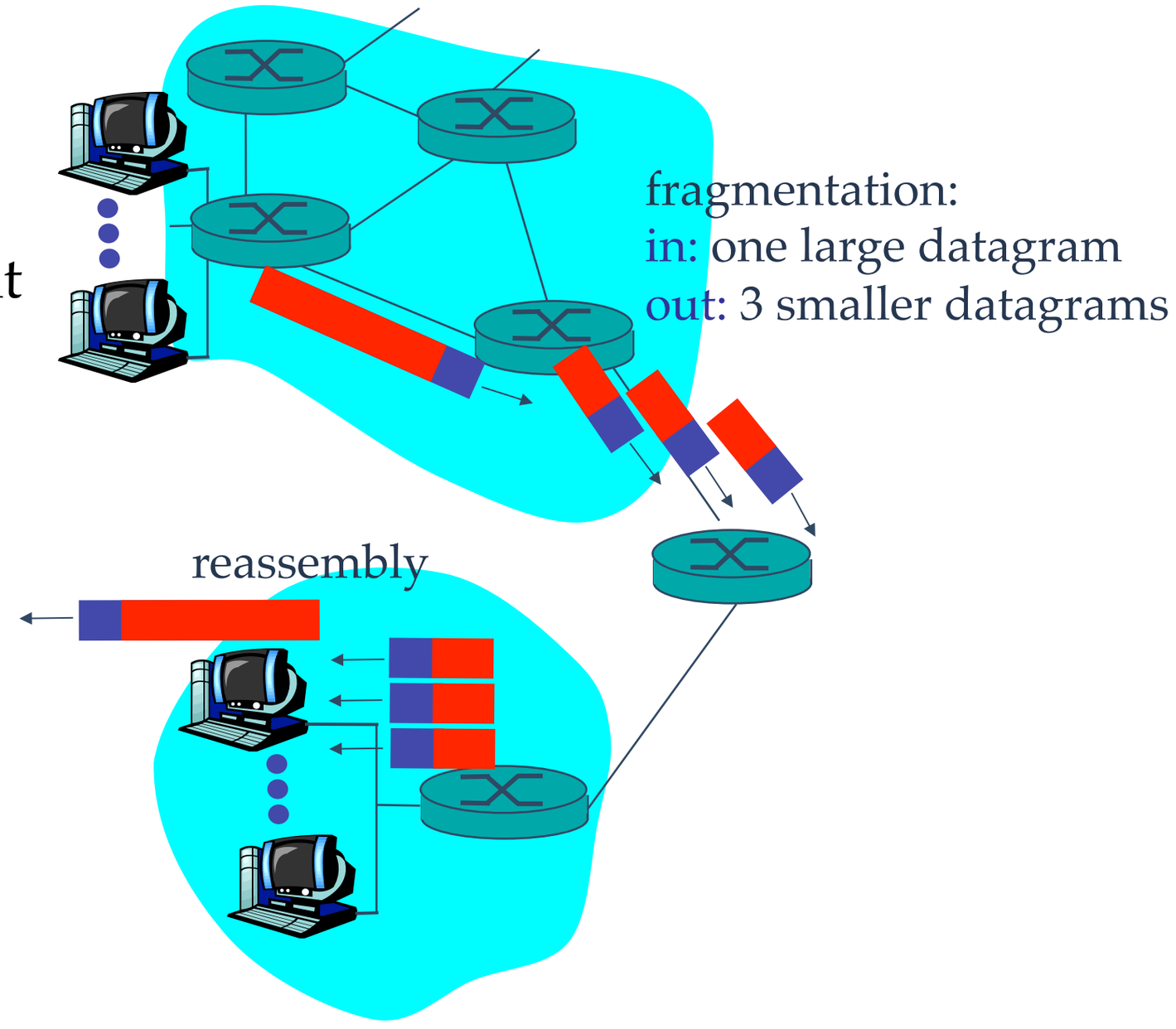


IP datagram format

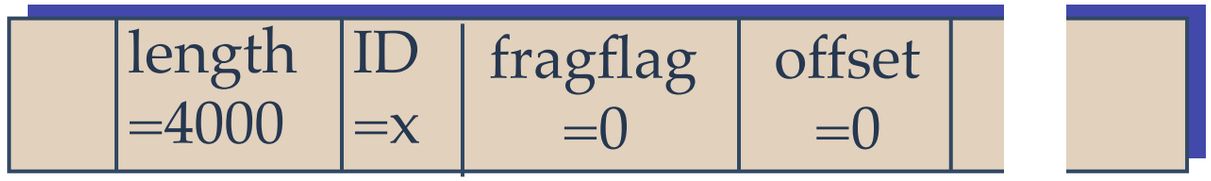


IP Fragmentation & Reassembly

- network links have MTU (maximum transmission unit): 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 and Reassembly



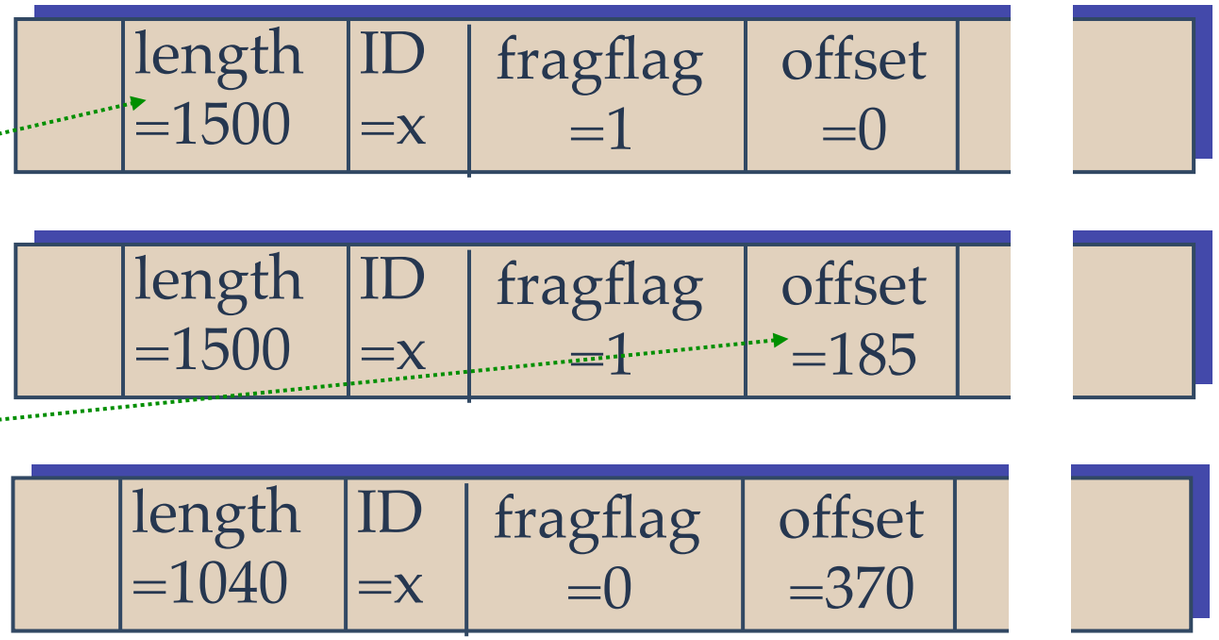
Example

- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams

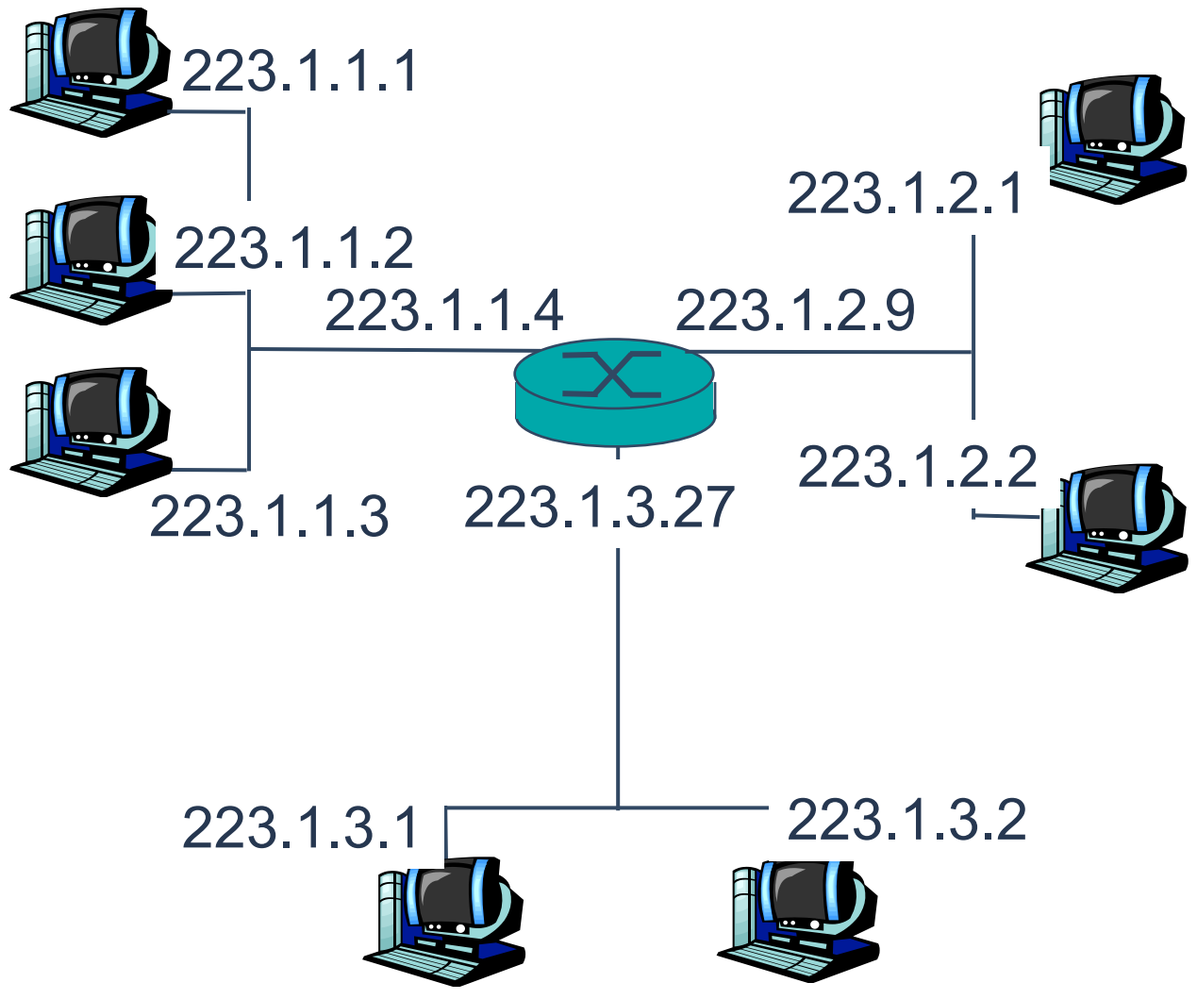
1480 bytes in data field

offset = 1480/8



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 interface
  - ➡ IP addresses associated with each interface

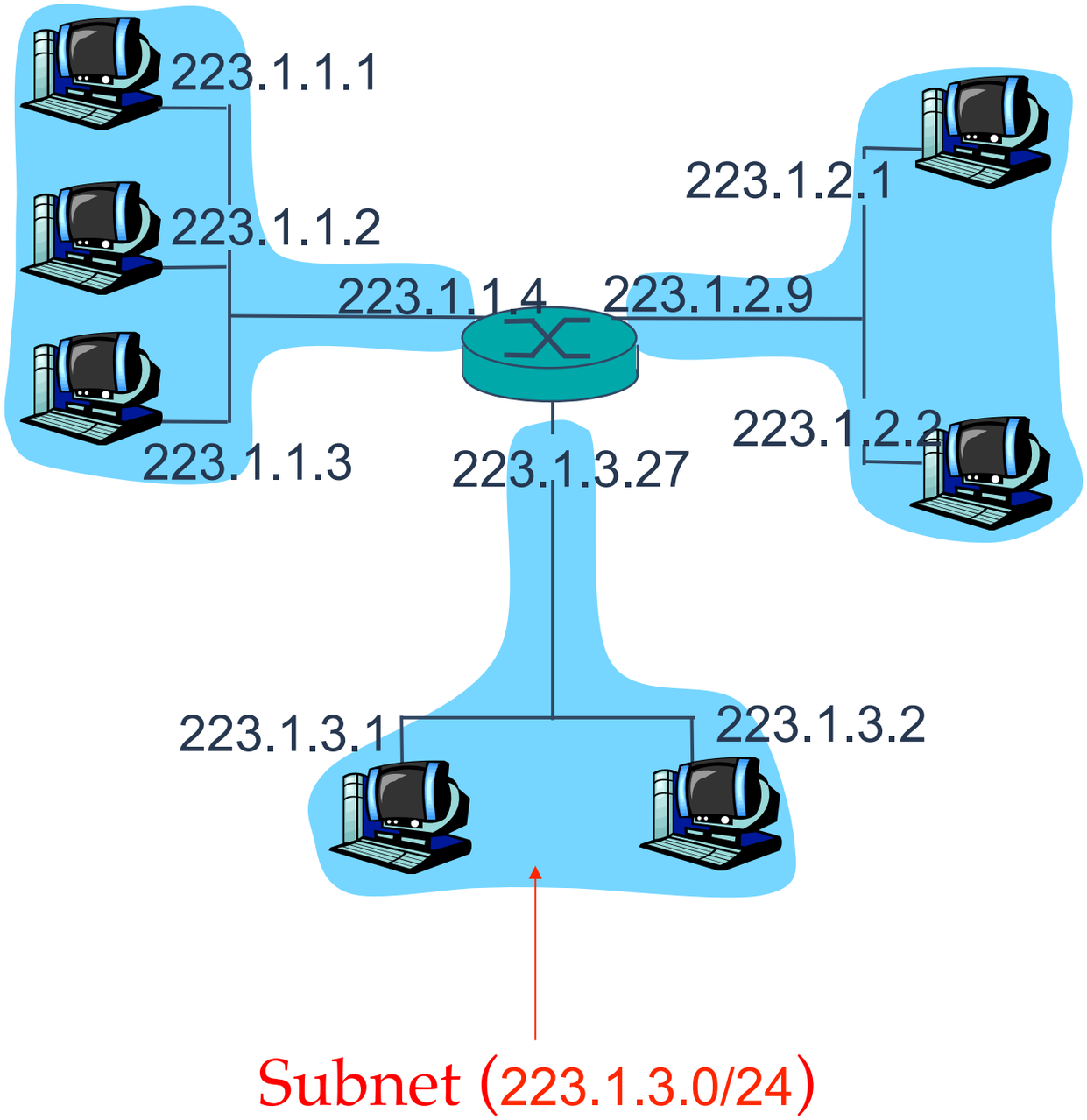


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

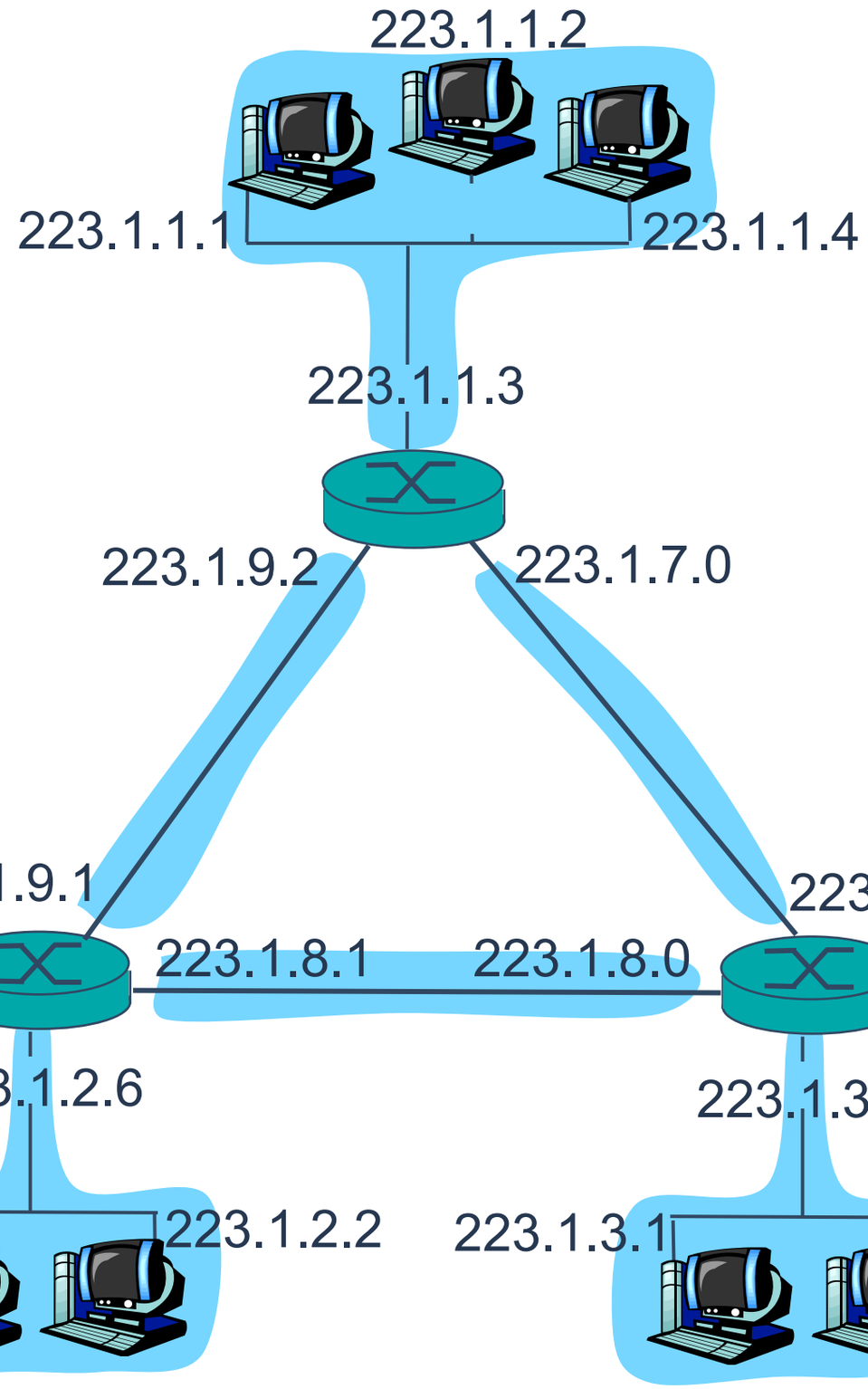
Subnets

- 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



Subnets

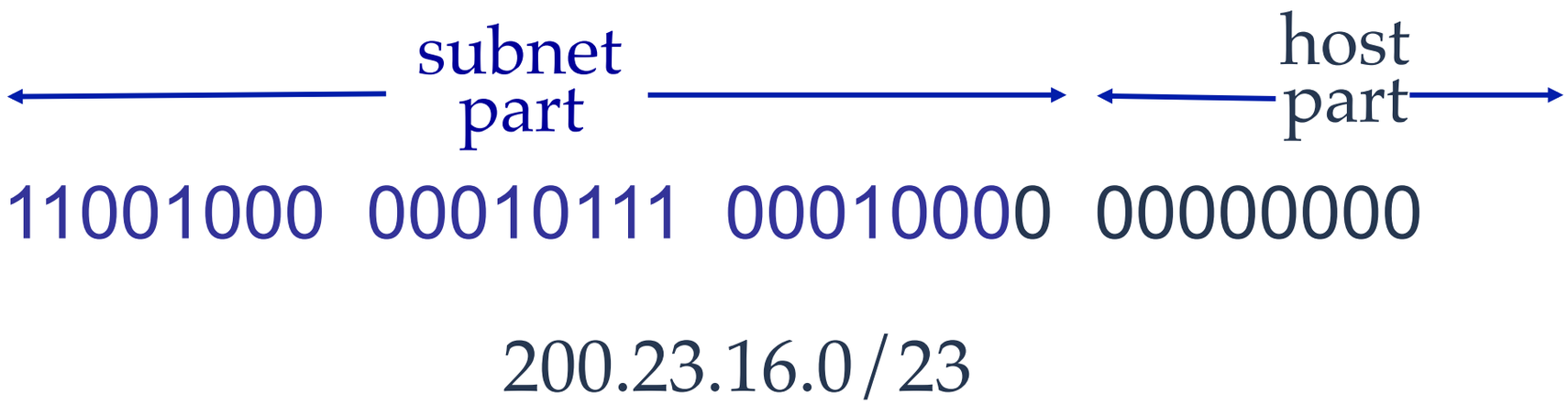
How many?



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



IP addresses: how to get one?

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

- Static allocation: hard-coded by system admin in a file
- 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)

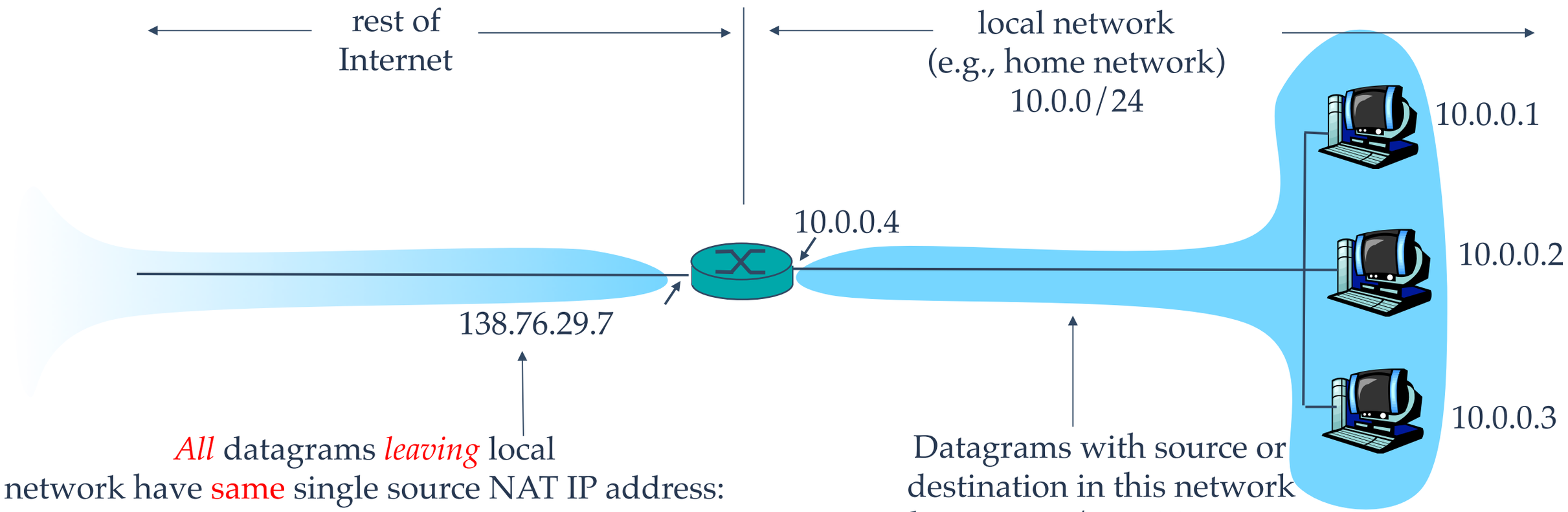
Support for mobile users who want to join network

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

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).



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 messages 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 destination
  - first has TTL =1
  - second has TTL=2, etc.
  - unlikely port number
- When nth datagram arrives to nth router:
  - router discards datagram
  - and sends to source an ICMP message (type 11, code 0)
  - ICMP message includes name of router & IP address
- when ICMP message arrives, source calculates RTT
- traceroute does this 3 times

Stopping criterion

- UDP segment eventually arrives at destination host
- destination returns ICMP “port unreachable” packet (type 3, code 3)
- when source gets this ICMP, stops.

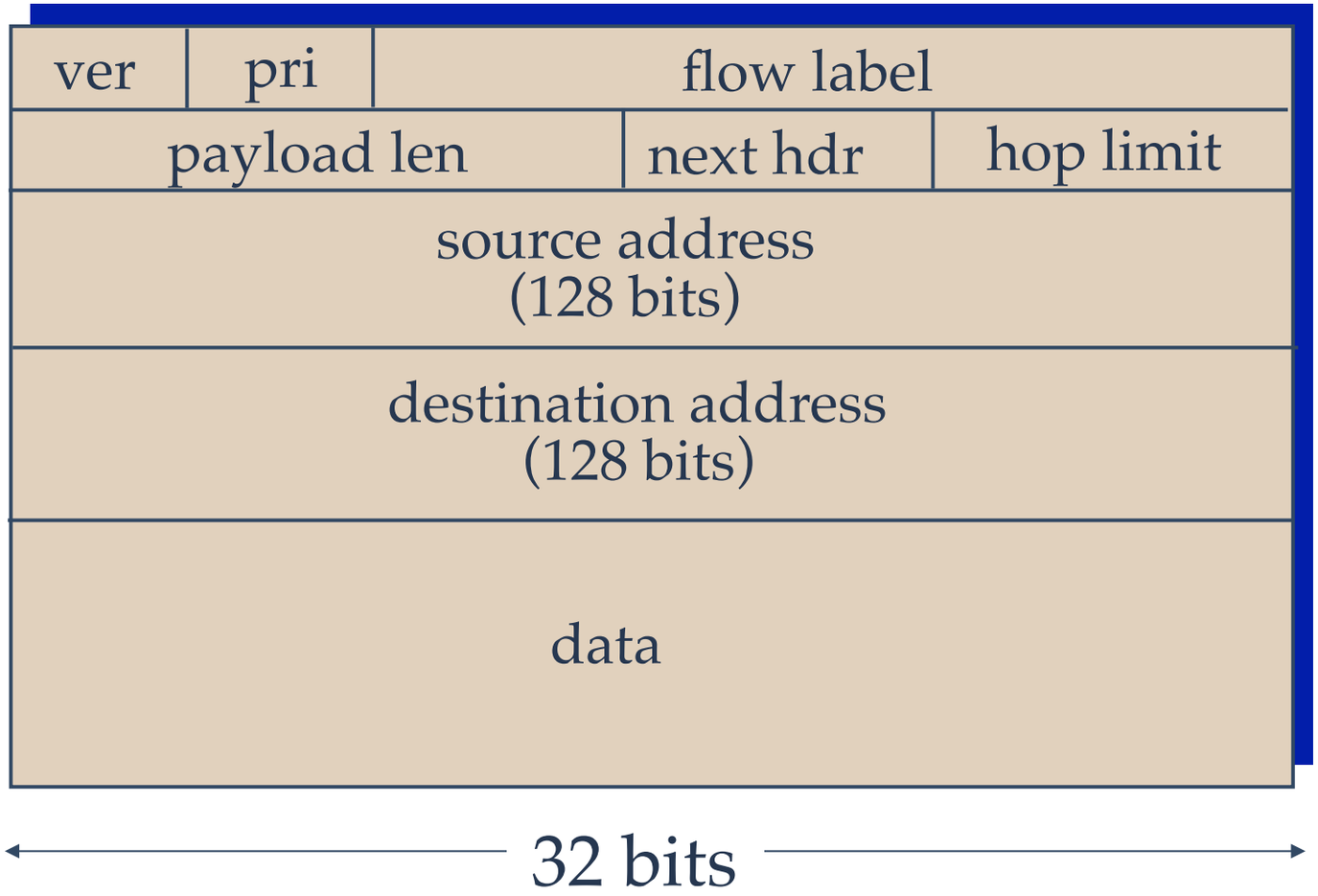
IPv6

- 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 Header (Cont)

Priority: identify priority among datagrams in flow
Flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).
Next header: identify upper layer protocol for data

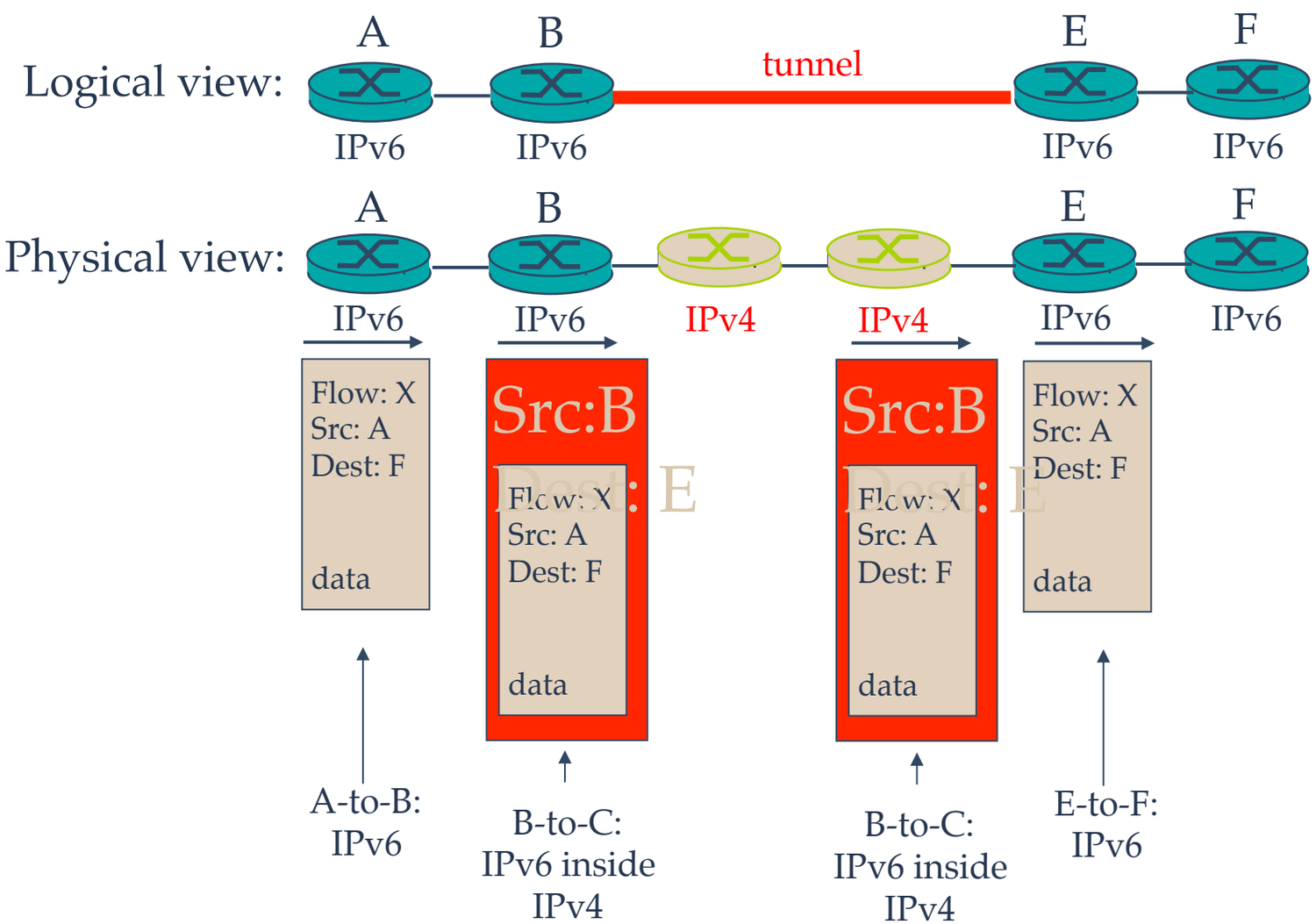


Other Changes from IPv4

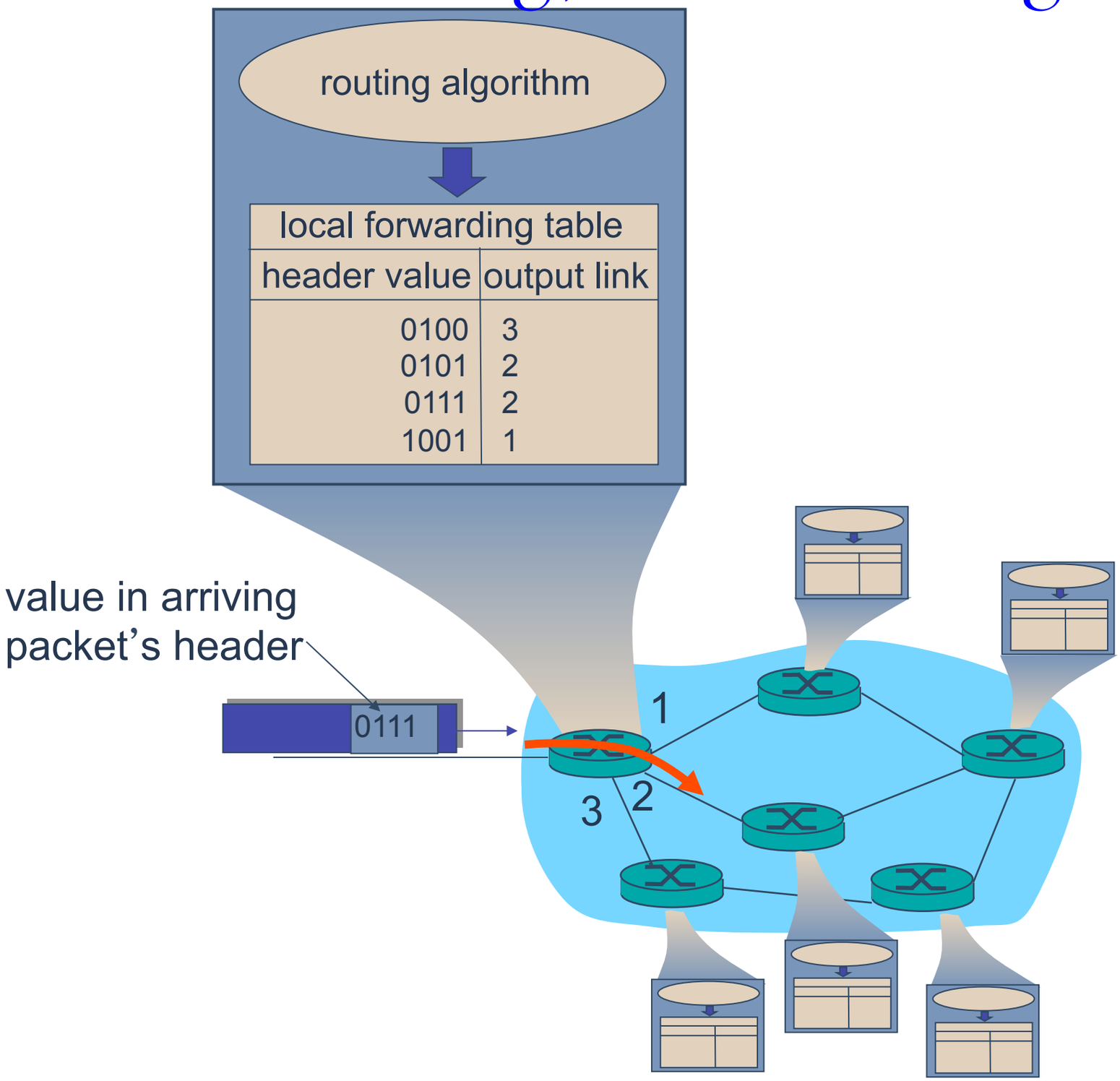
- *Checksum*: removed entirely to reduce processing time at each hop
- *Options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - ➡ additional message types, e.g. "Packet Too Big"
  - ➡ multicast group management functions

Transition From IPv4 To IPv6

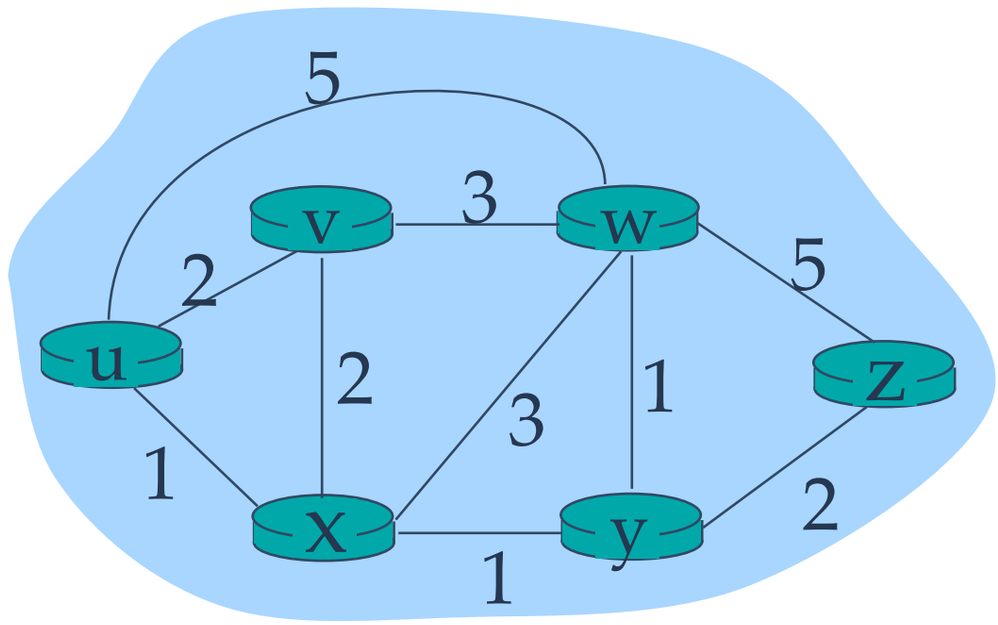
- Not all routers can be upgraded simultaneous
  - ➡ no “flag days”
  - ➡ How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers



Routing Algorithms – 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)}
- c(x,y) = cost of link (x,y)
  - e.g., c(w,z) = 5
  - cost could always be 1, or inversely related to bandwidth, or inversely related to congestion
- Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

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

Decentralized:

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

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

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - ➡ accomplished via "link state broadcast"
  - ➡ all nodes have same info
- Dijkstra's algorithm: 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

Dijsktra's Algorithm

set of nodes whose least cost path definitively known

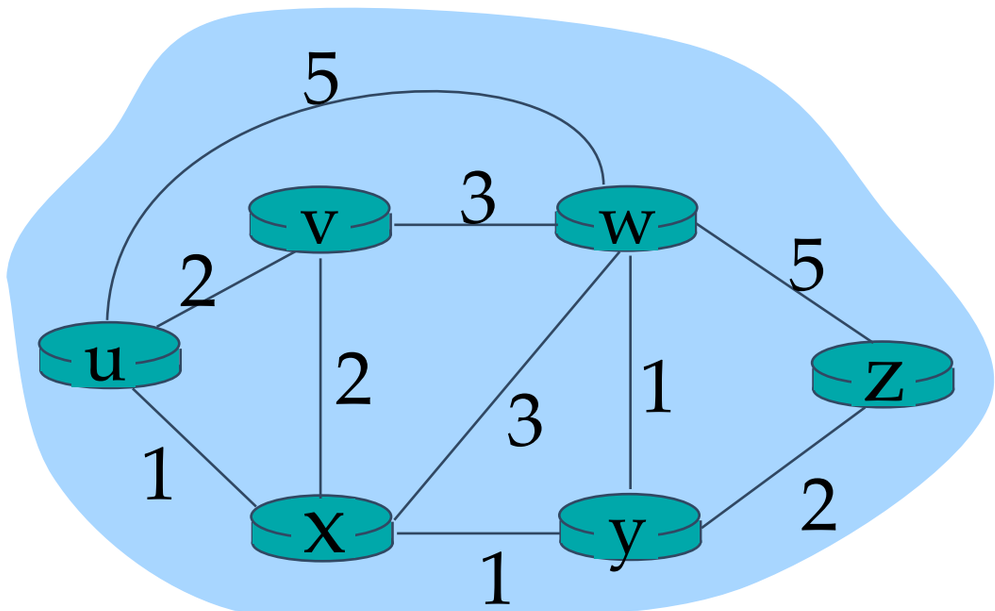
current value of cost of path from source to destination v

link cost from node x to y; = ∞ if not direct neighbors

```
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'
```

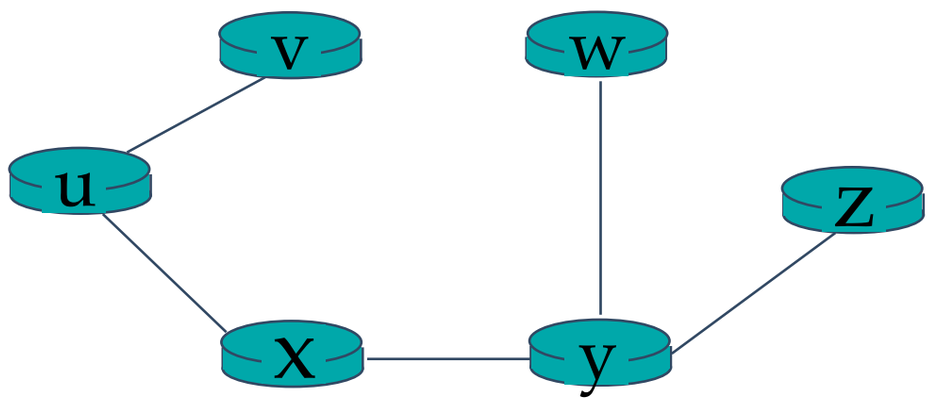
Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



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)

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

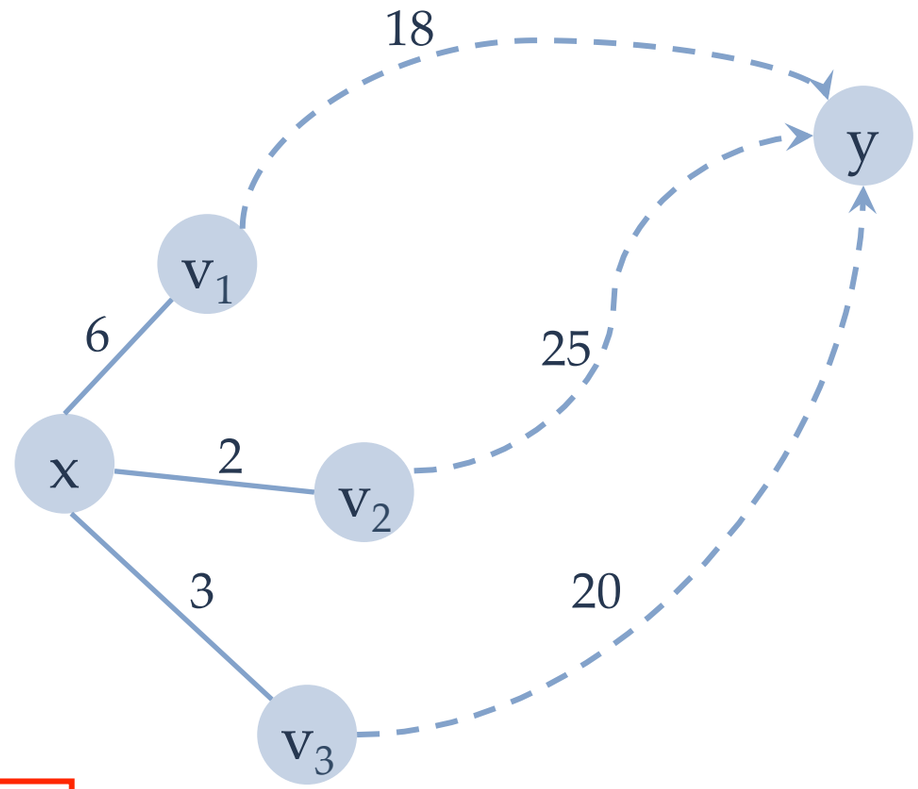
Define

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

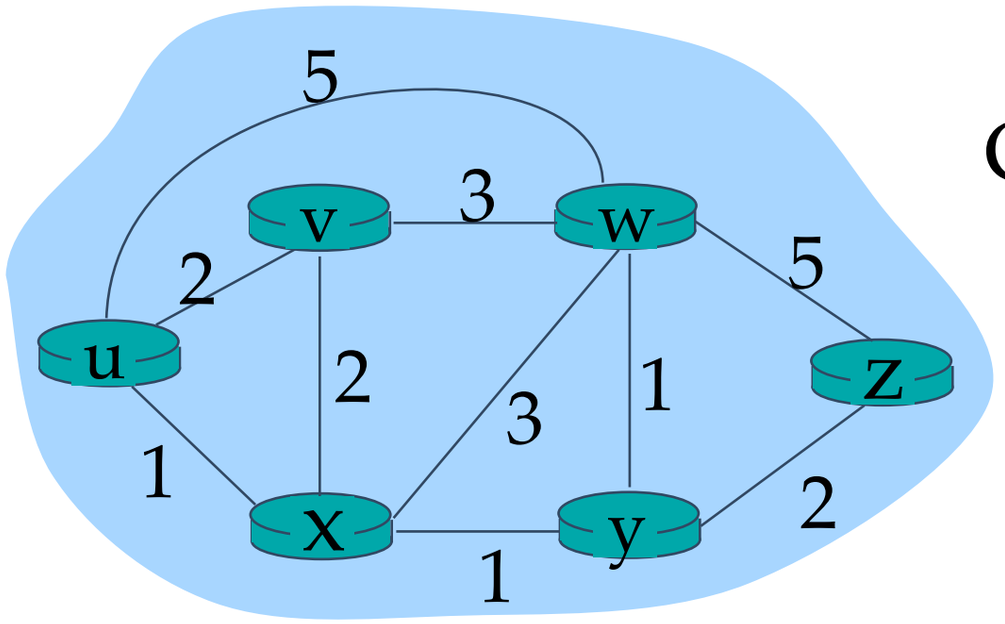
Then

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

where min is 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 that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

- Each node x maintains the following
  - Its own distance vector $\mathbf{D}_x = [D_x(y): y \in N]$ (N is the set of nodes)
    - $D_x(y)$ = estimate of least cost from x to y
  - cost to each neighbor v: c(x,v)
  - its neighbors' distance vectors. For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$
- 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:

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

- Under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm

Iterative, asynchronous:

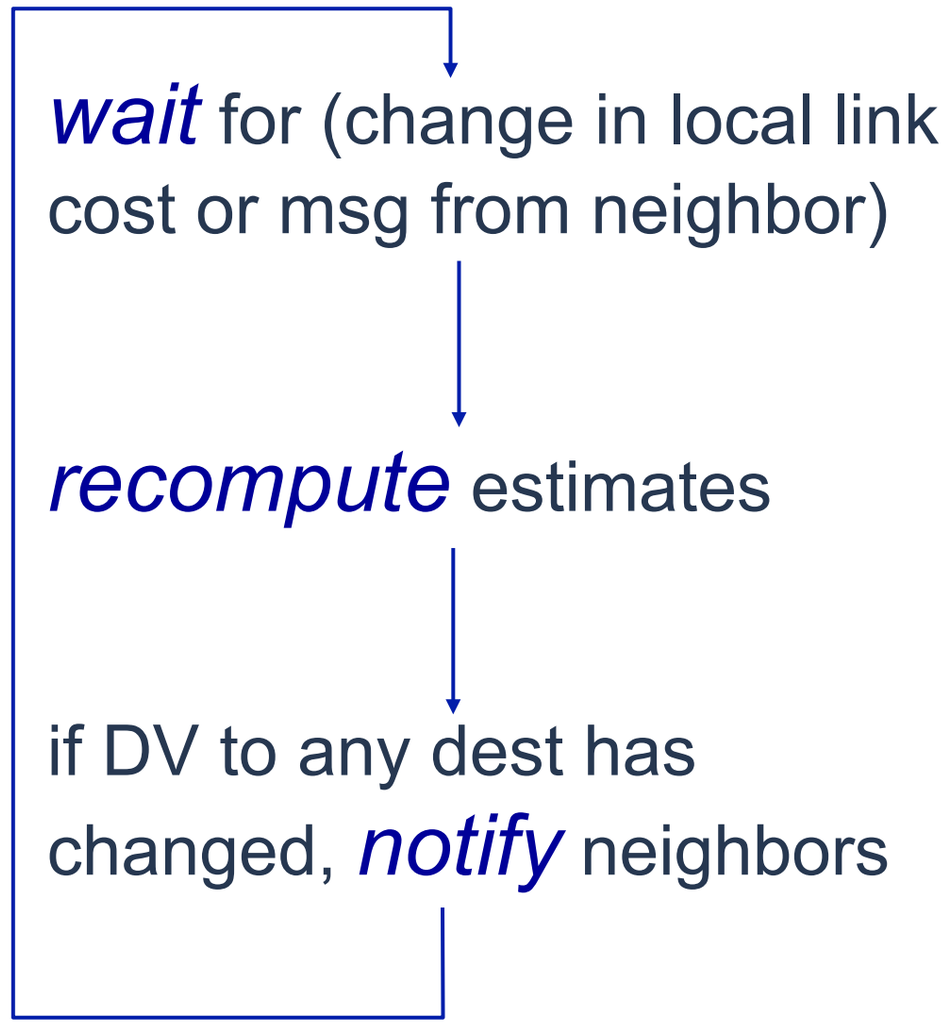
each local iteration caused by:

- 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:



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

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

node x table

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

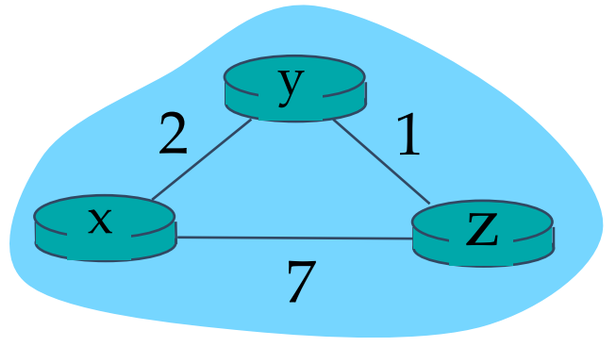
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

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

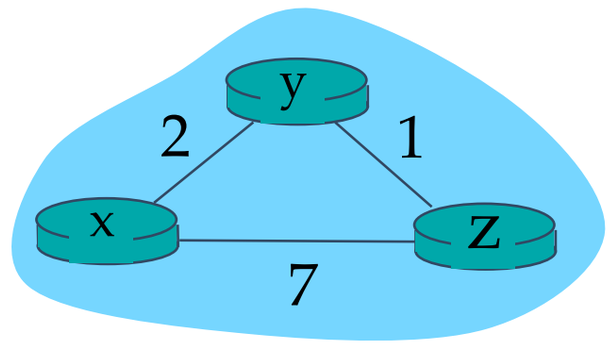
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

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

Comparison of LS and DV algorithms

Message complexity

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

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) messages
  - ➡ 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

So far we assumed

- All routers are identical
- Network is "flat"
- These are not true in practice

scale: with 200 million destinations:

- can't 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

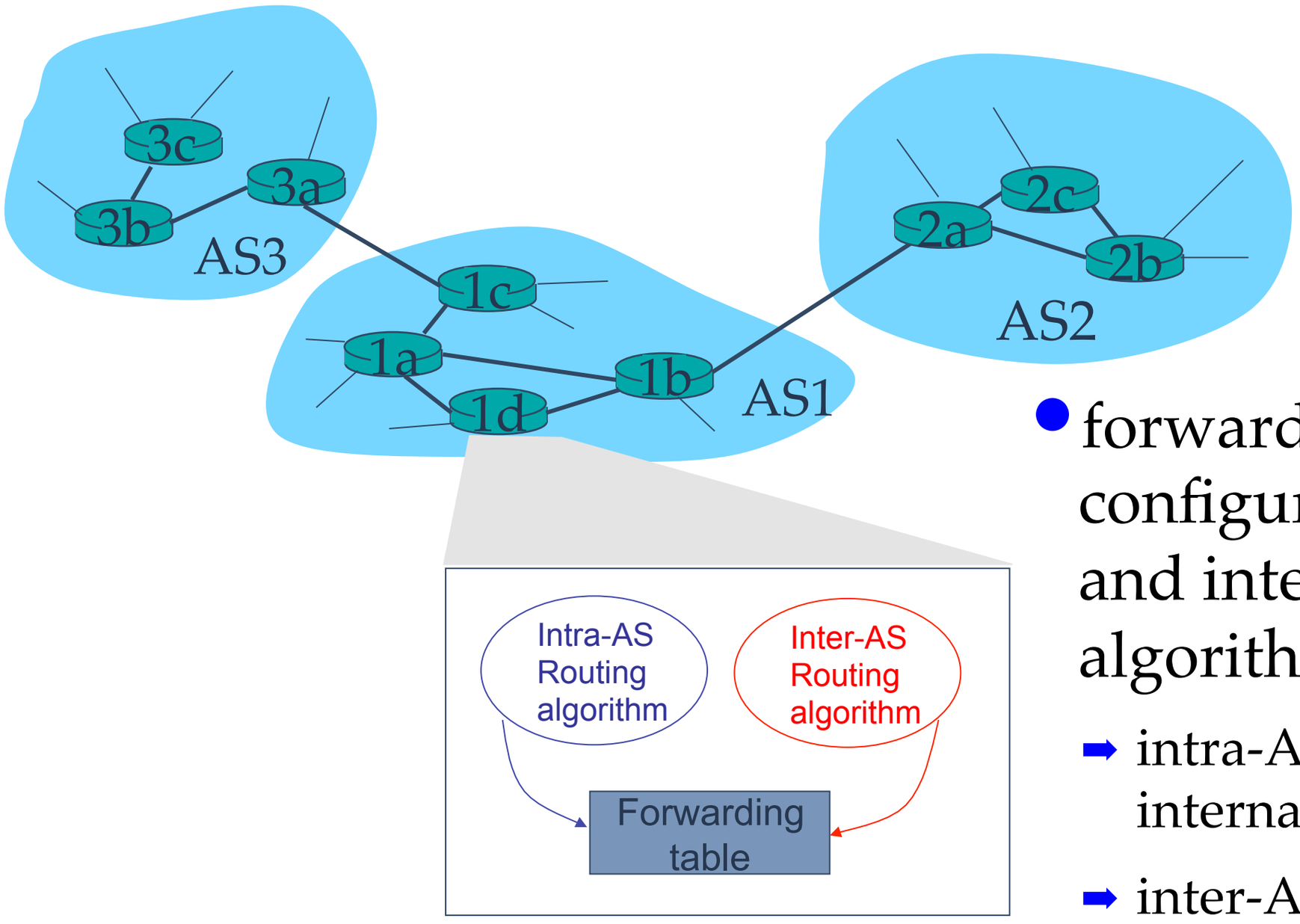
Hierarchical Routing

- 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 destinations
  - ➡ inter-AS & intra-As sets entries for external destinations

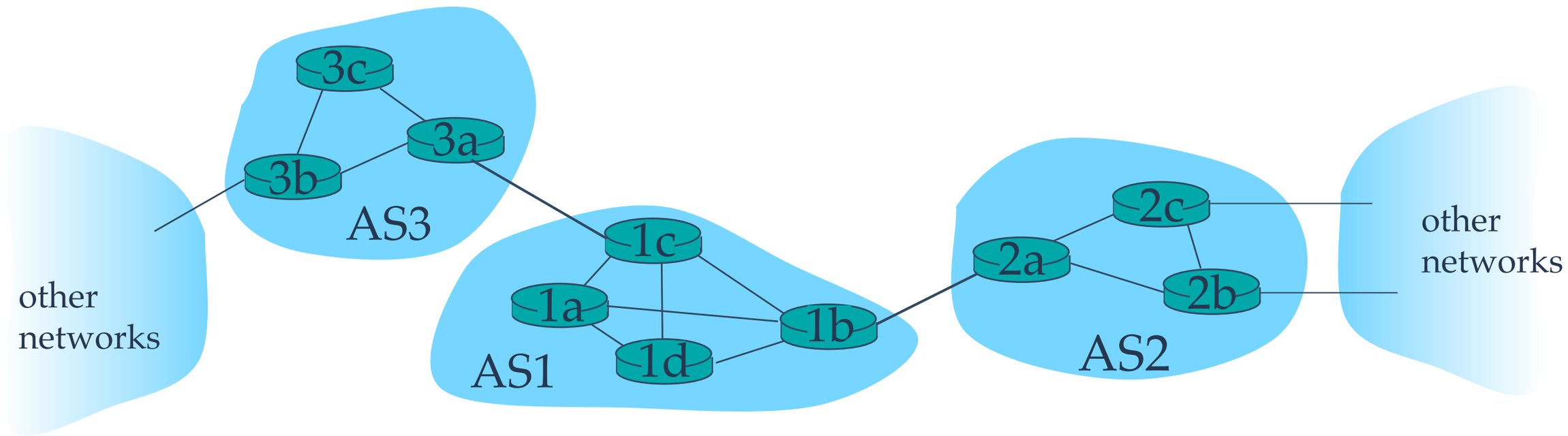
Inter-AS tasks

- 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, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!

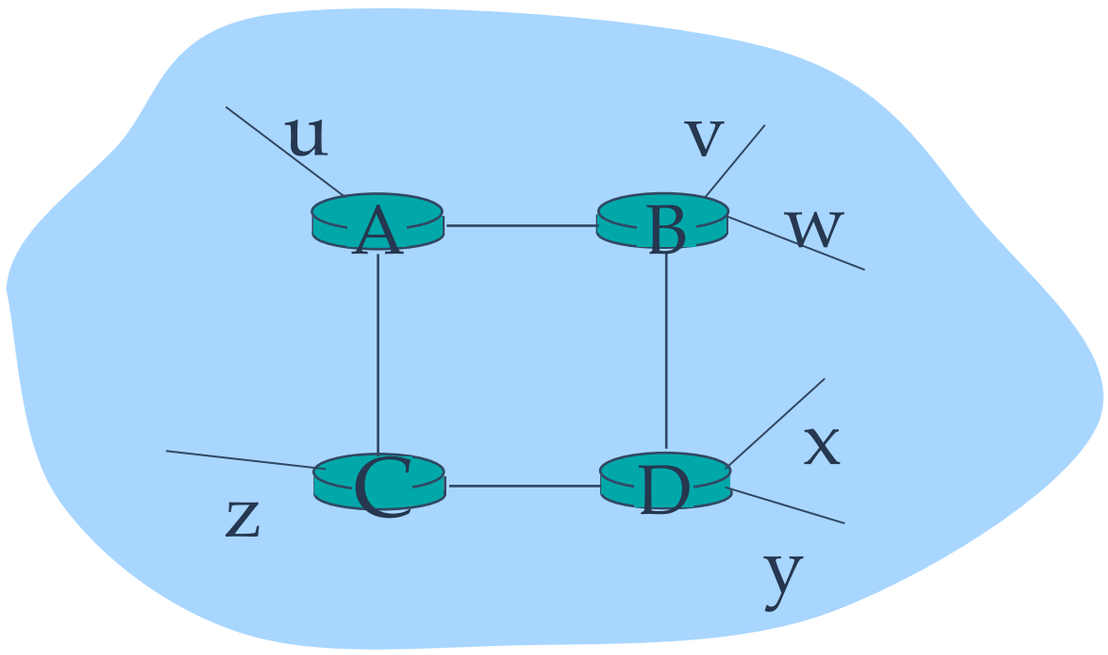


Intra-AS Routing

- also known as Interior Gateway Protocols (IGP)
- most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol (open - Internet)
  - OSPF: Open Shortest Path First (open – Internet)
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

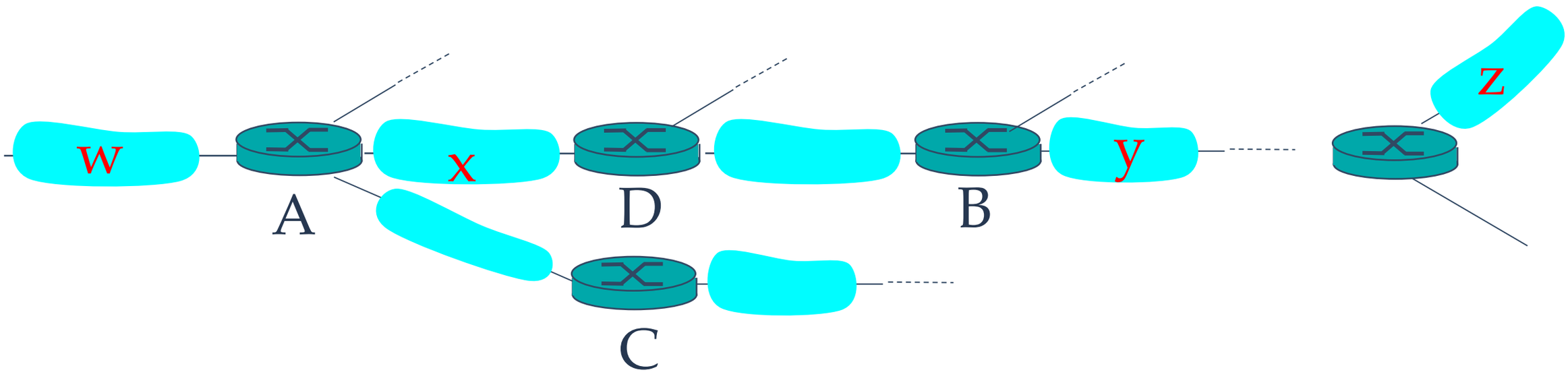
- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets (in IP addressing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



from router A to destination subnets:

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

RIP: Example



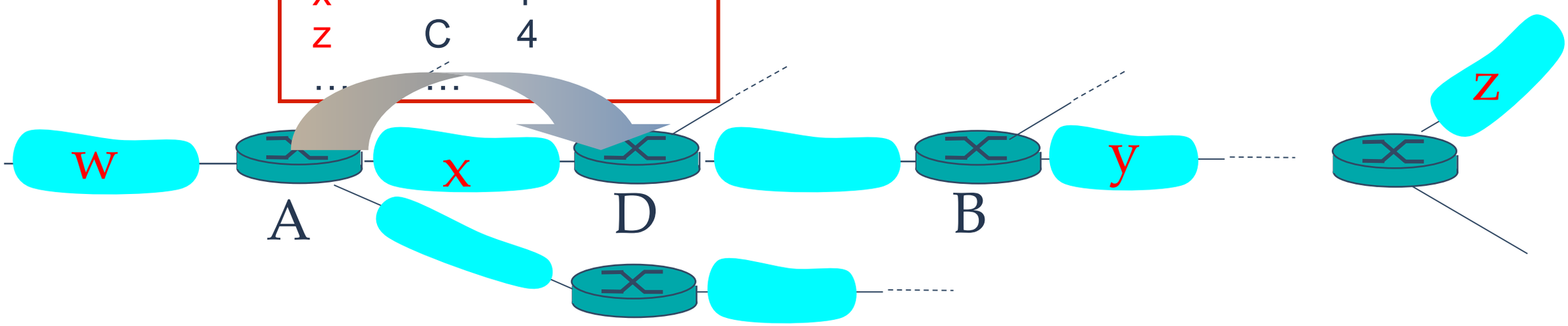
routing table in router D

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
…	…	



C

routing table in router D

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

A

5

OSPF (Open Shortest Path First)

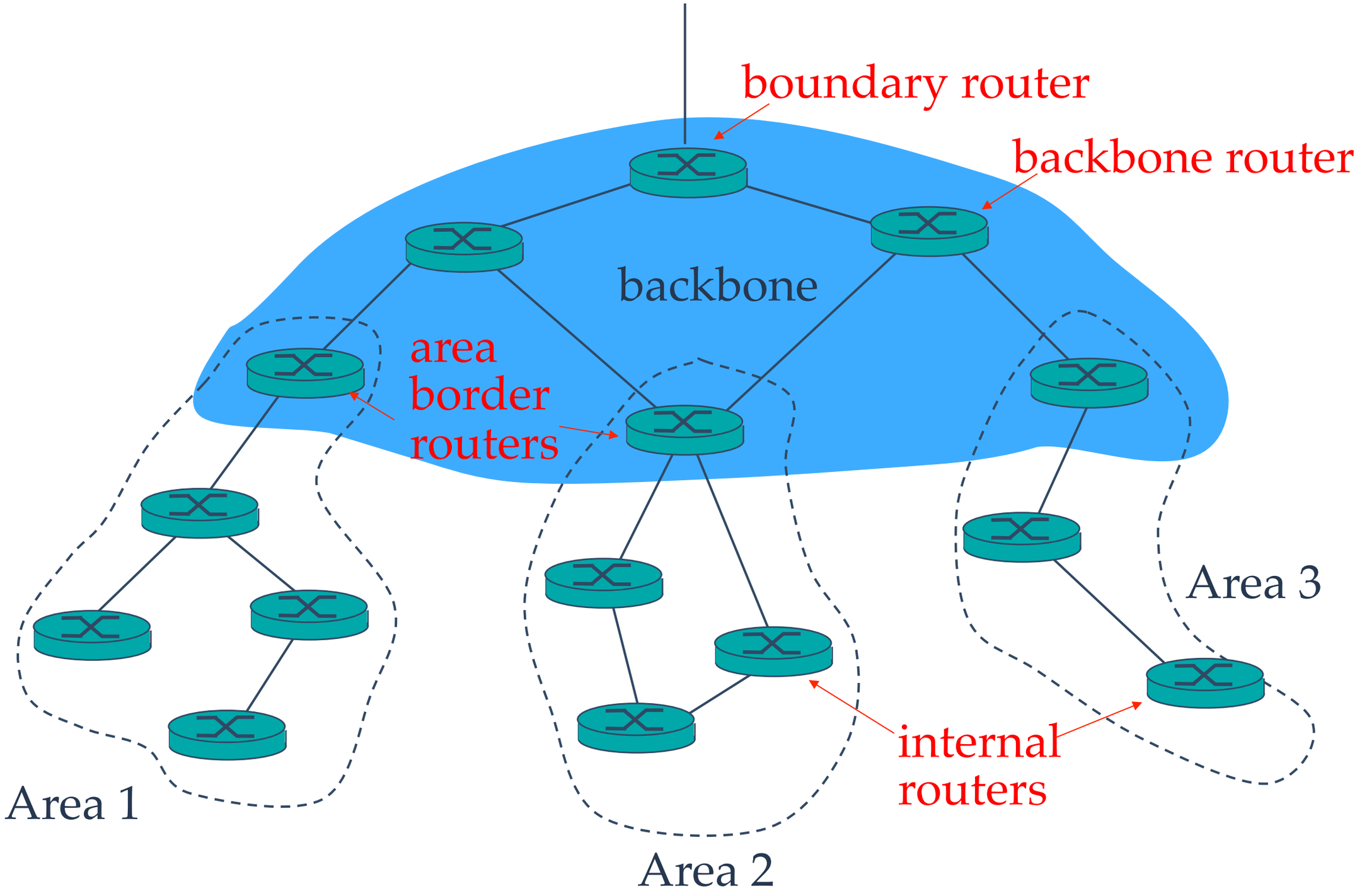
- "open": publicly available
- uses Link State algorithm
  - ➡ LS packet dissemination
  - ➡ topology map at each node
  - ➡ route computation using Dijkstra's algorithm

- OSPF advertisement carries one entry per neighbor router
- advertisements disseminated to entire AS (via flooding)
  - ➡ carried in OSPF messages directly over IP (rather than TCP or UDP)

OSPF "advanced" features (not in RIP)

- 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 (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 OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

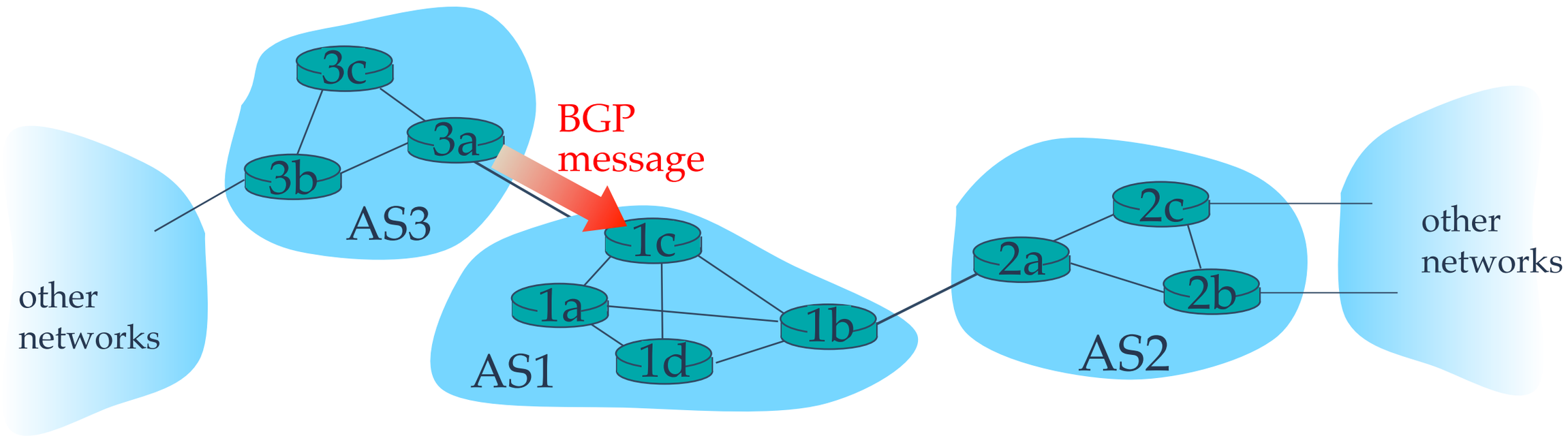
- two-level hierarchy: local area, backbone.
  - ➡ link-state advertisements only in area
  - ➡ each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *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"*

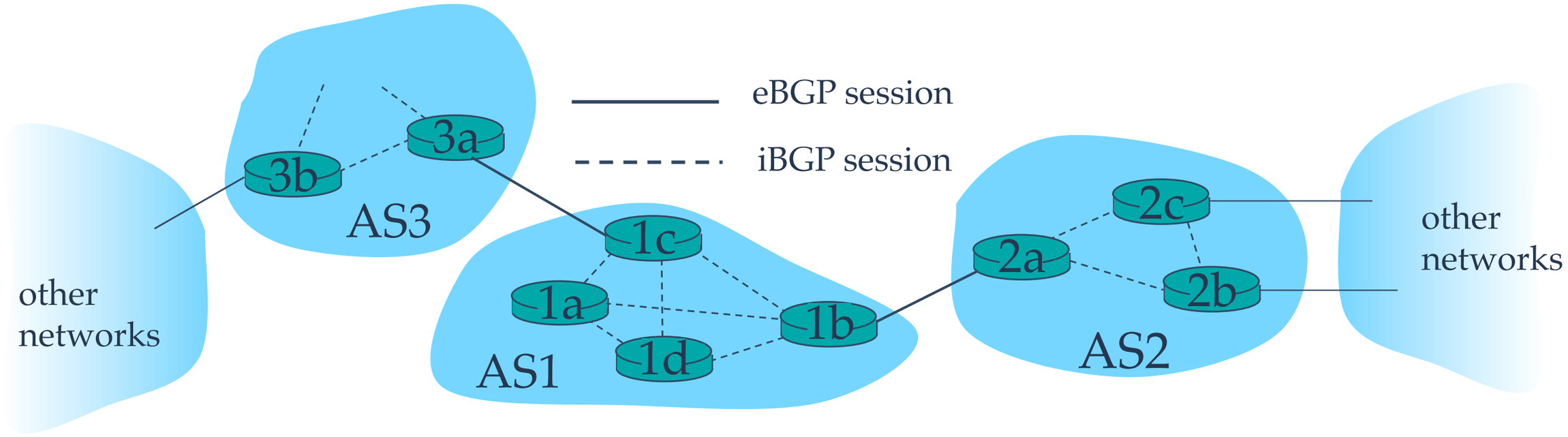
BGP basics

- 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

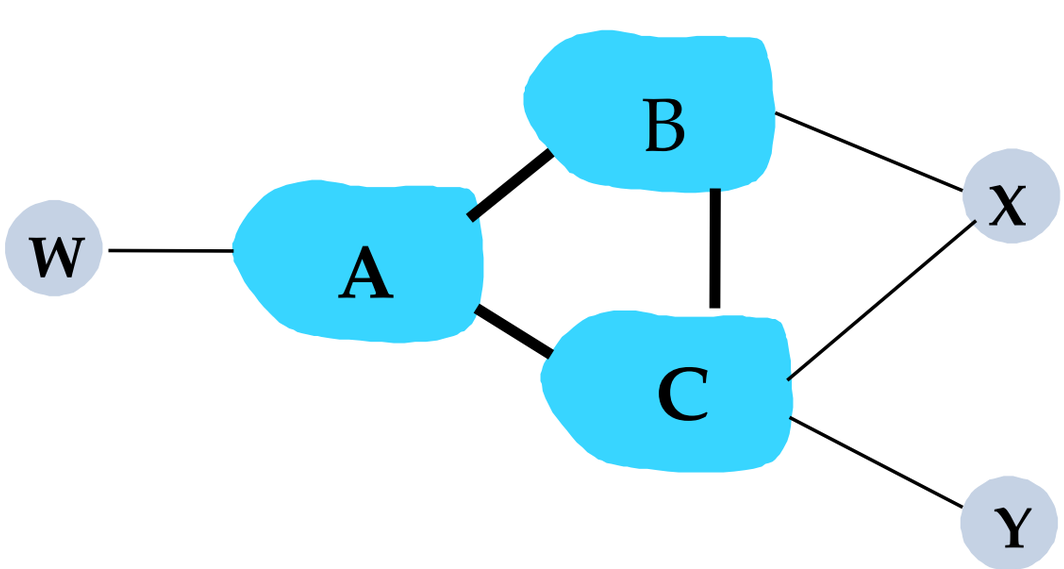


BGP basics: distributing path information

- 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.



BGP routing policy



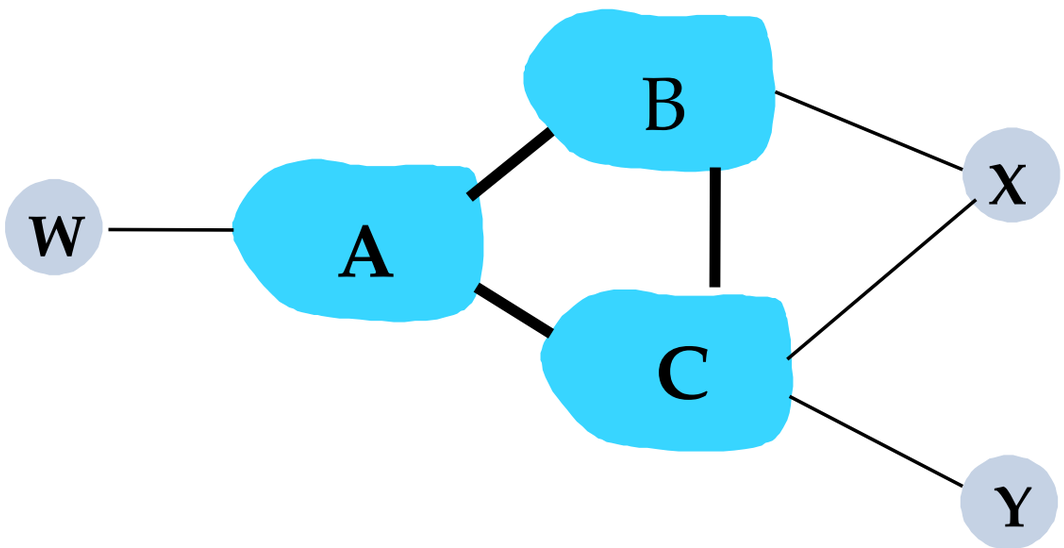
legend:

provider network

customer network:

- 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 (2)



legend:

provider network

customer network:

- 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- and 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