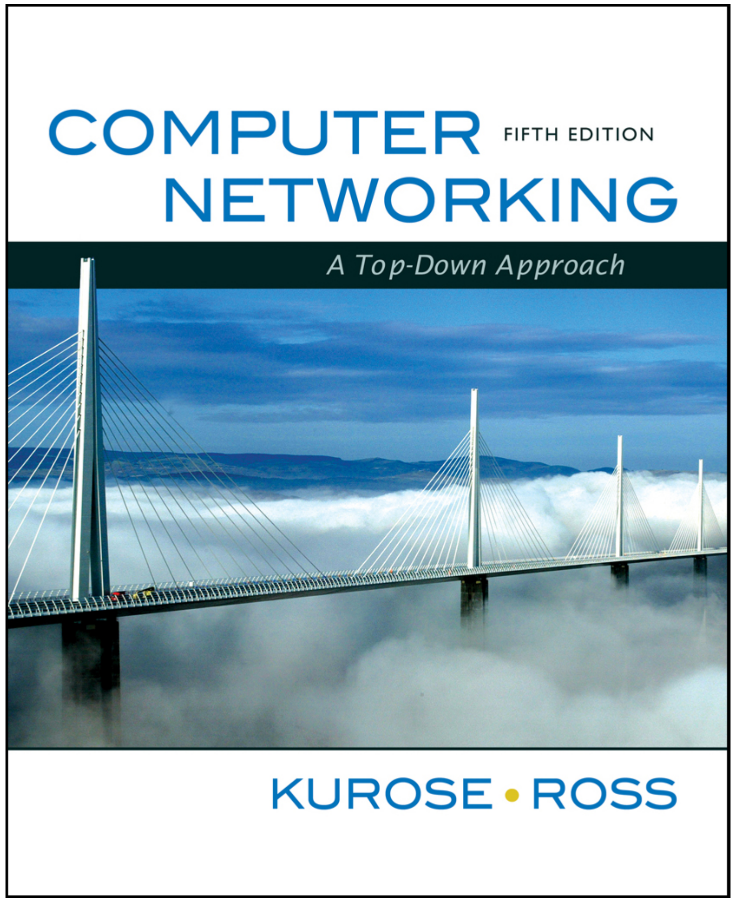
Module 9
Network Layer

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

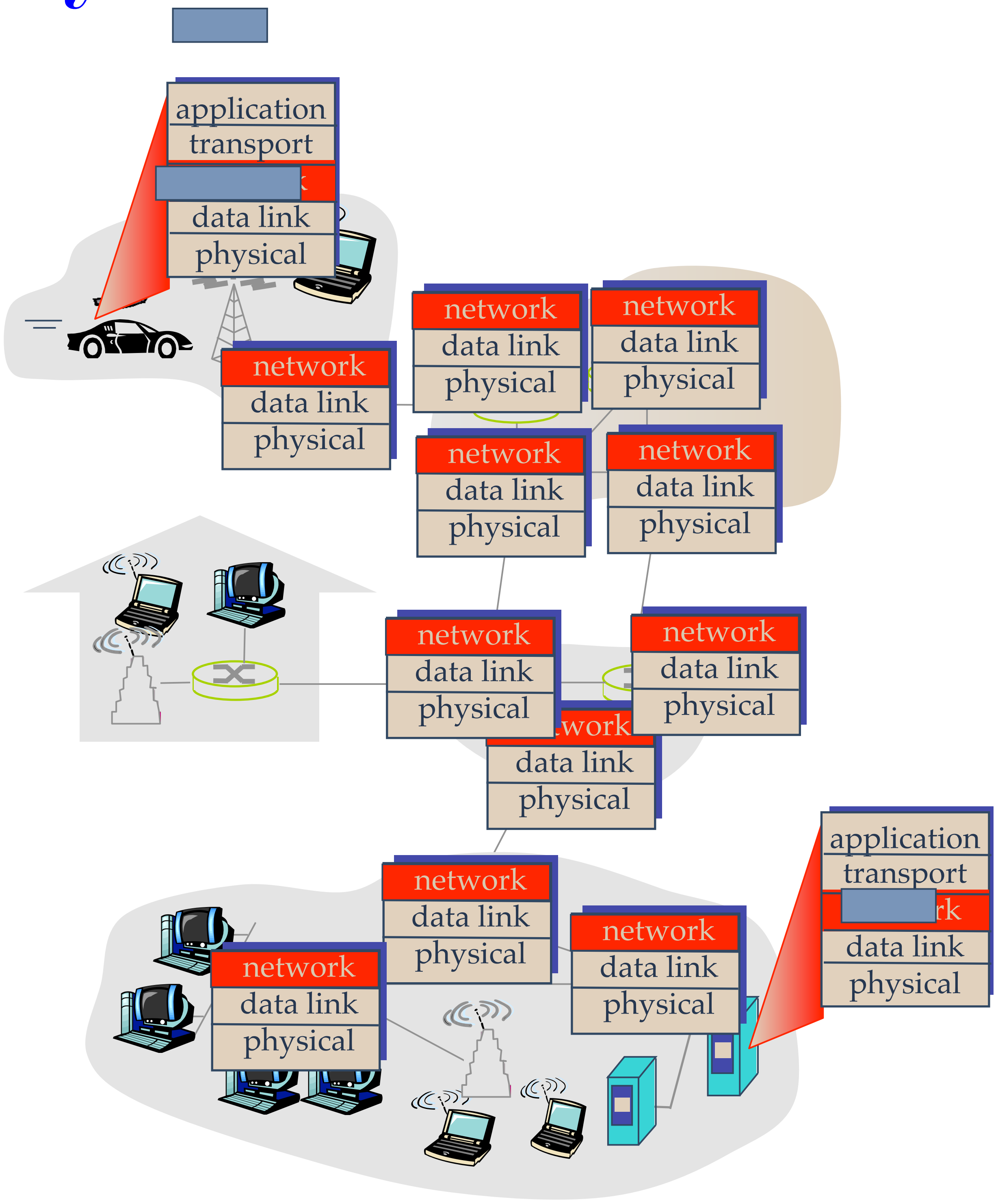


A note on the use of these ppt slides:

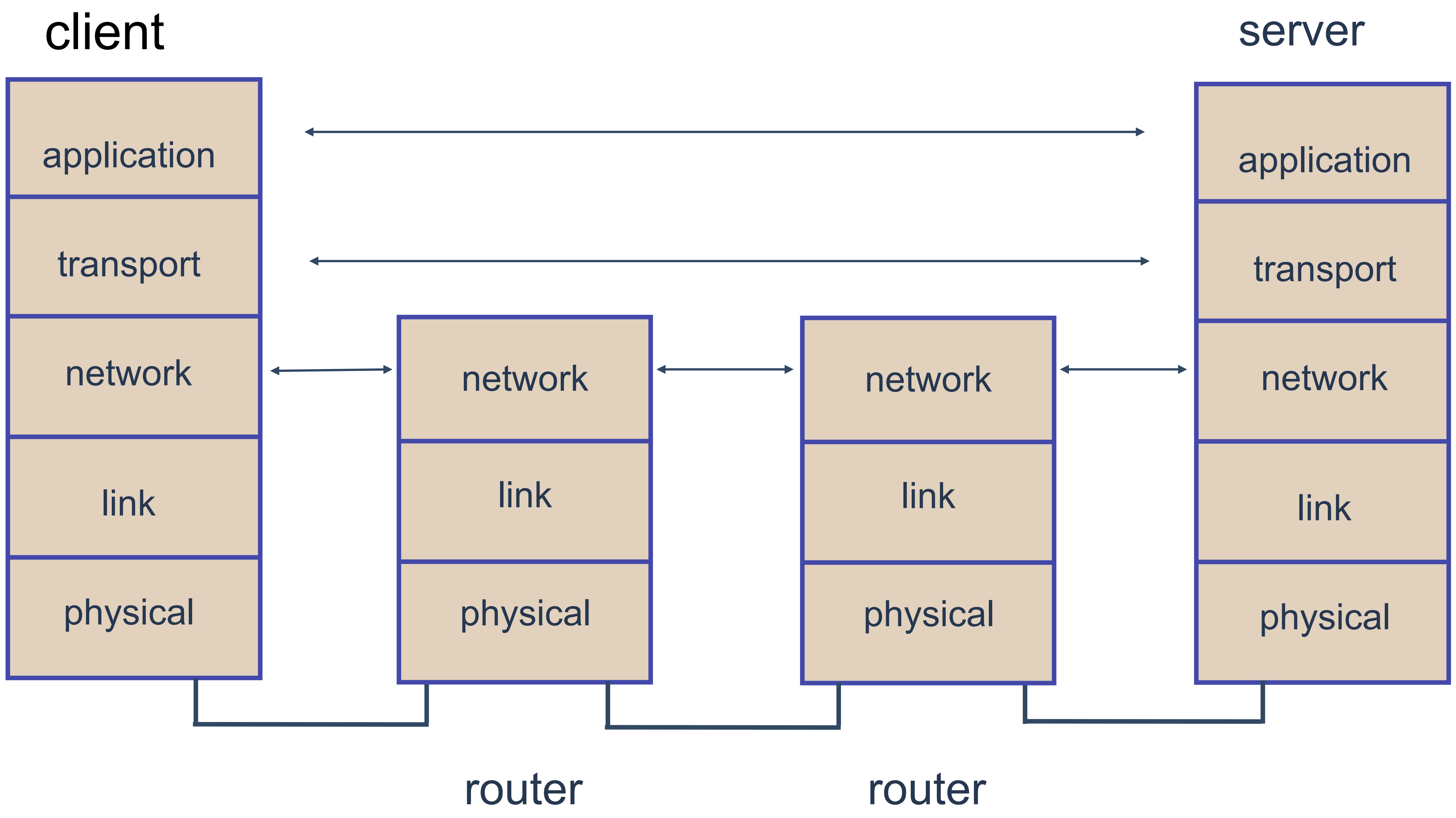
Computer Networking: A Top Down Approach
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

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



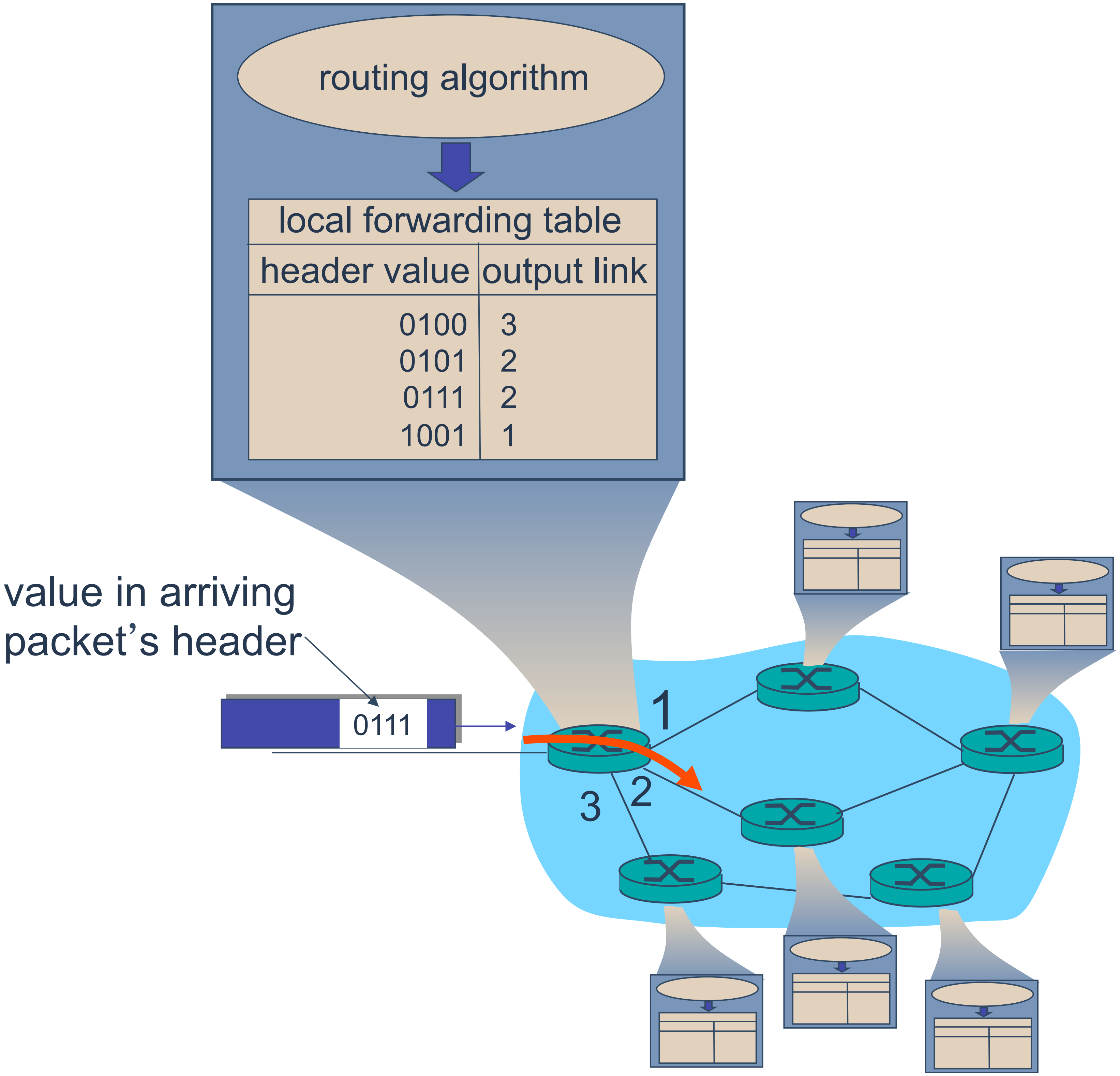
Internet Protocols

Application	FTP Telnet NFS SMTP HTTP …	
Transport	TCP / UDP	Segment
Network	IP	Datagram
Data Link Physical	X.25 / Ethernet / Packet Radio / ATM / FDDI / …	Frame

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 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

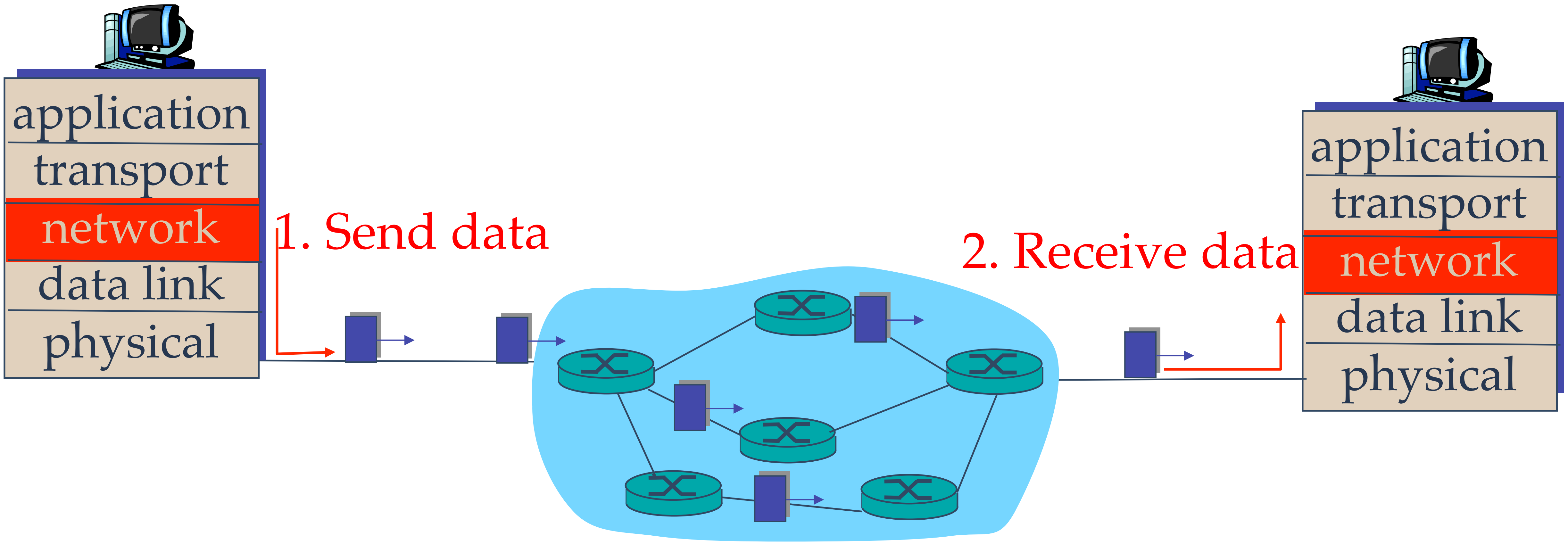
"source-to-destination path behaves much like telephone circuit"

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

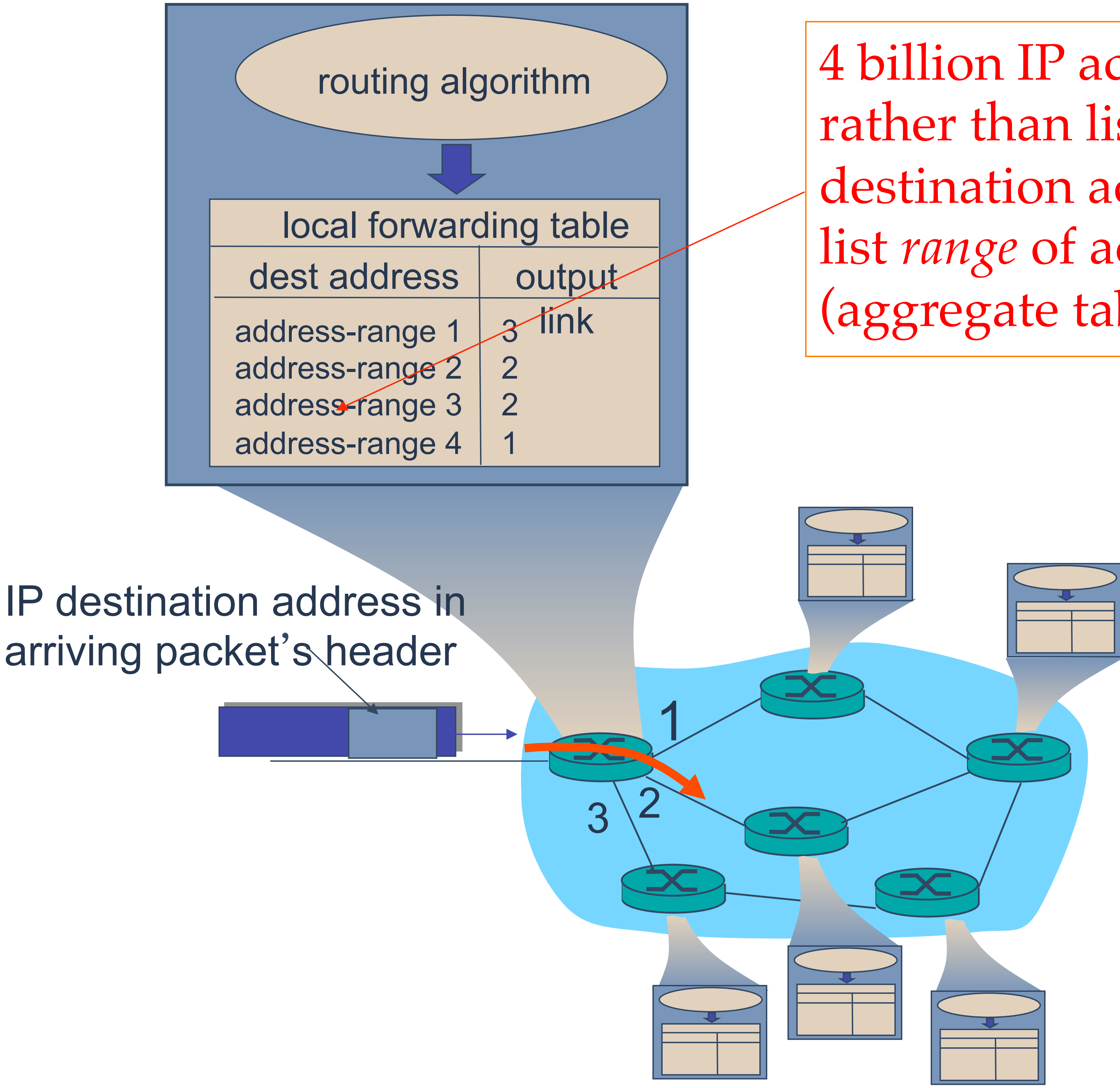
- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-destination path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources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



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

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

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

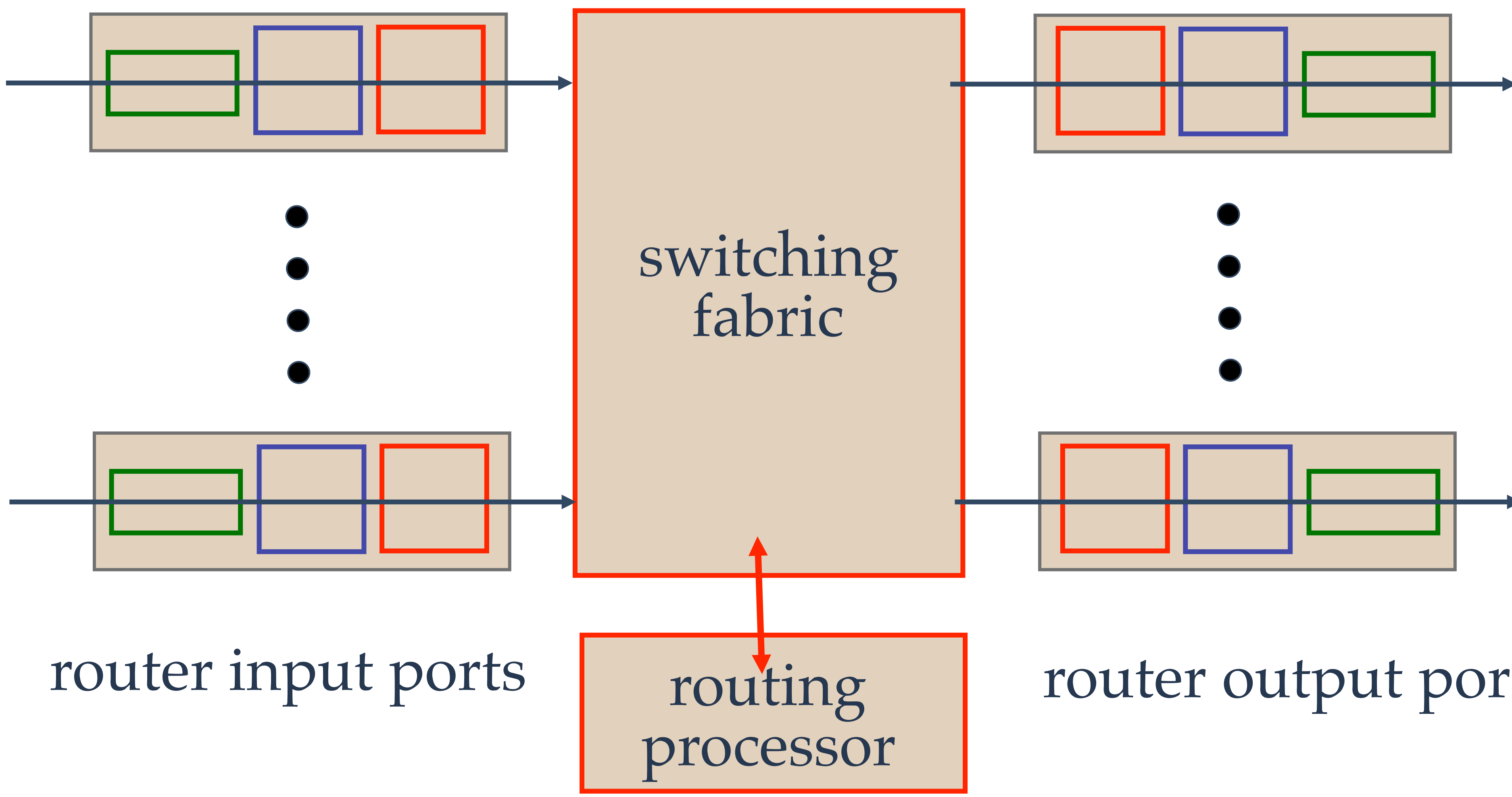
ATM (VC)

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

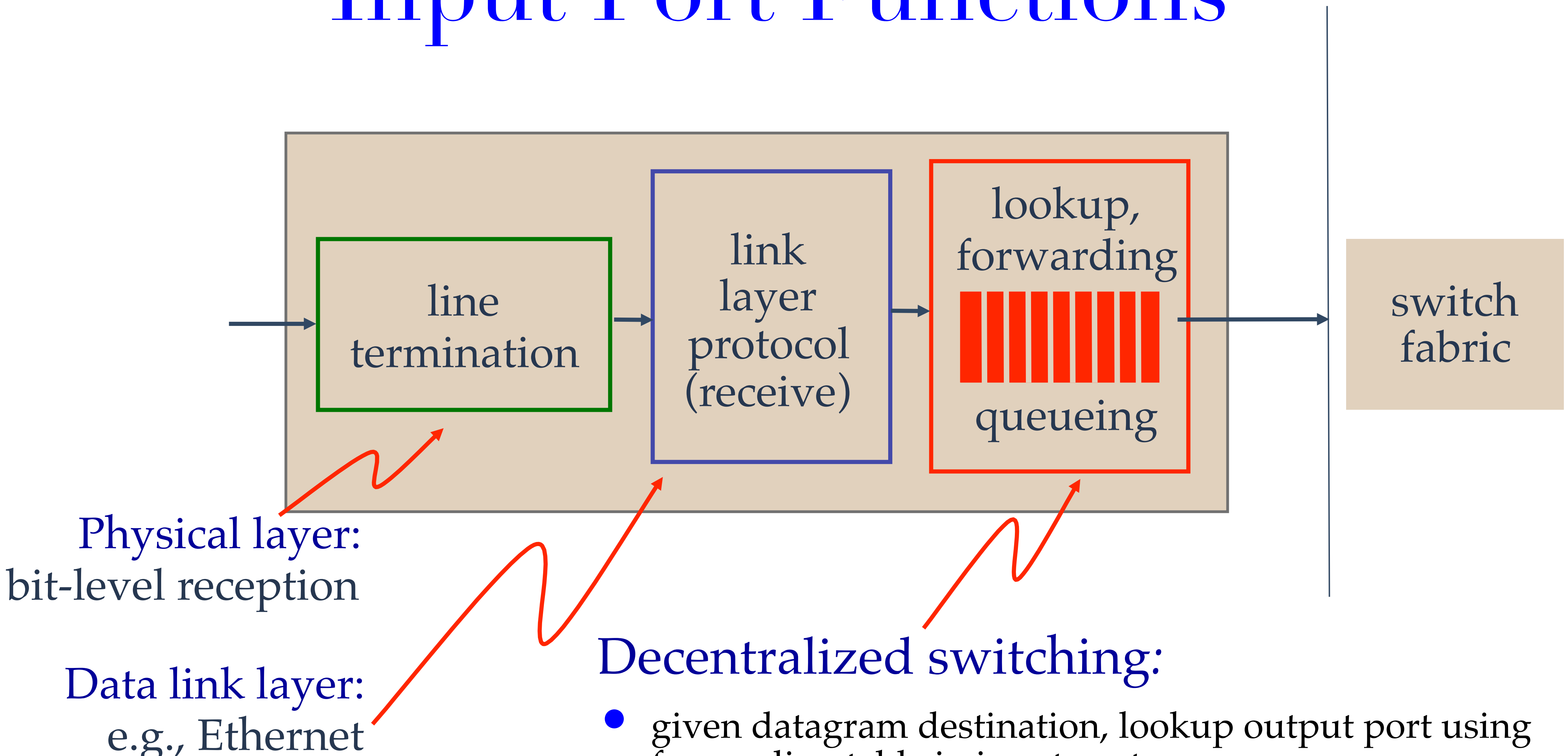
Router Architecture Overview

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



Input Port Functions

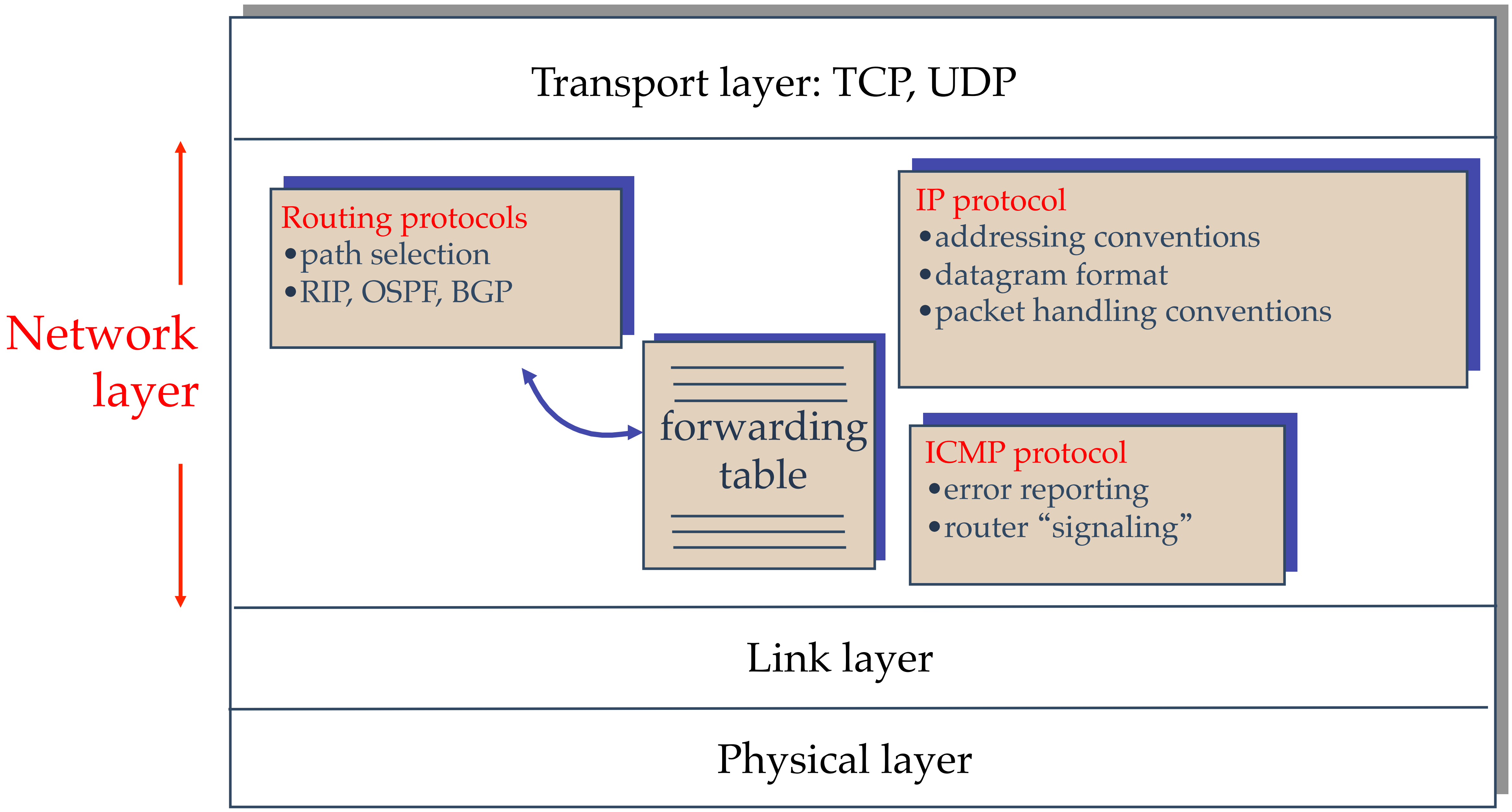


Decentralized switching:

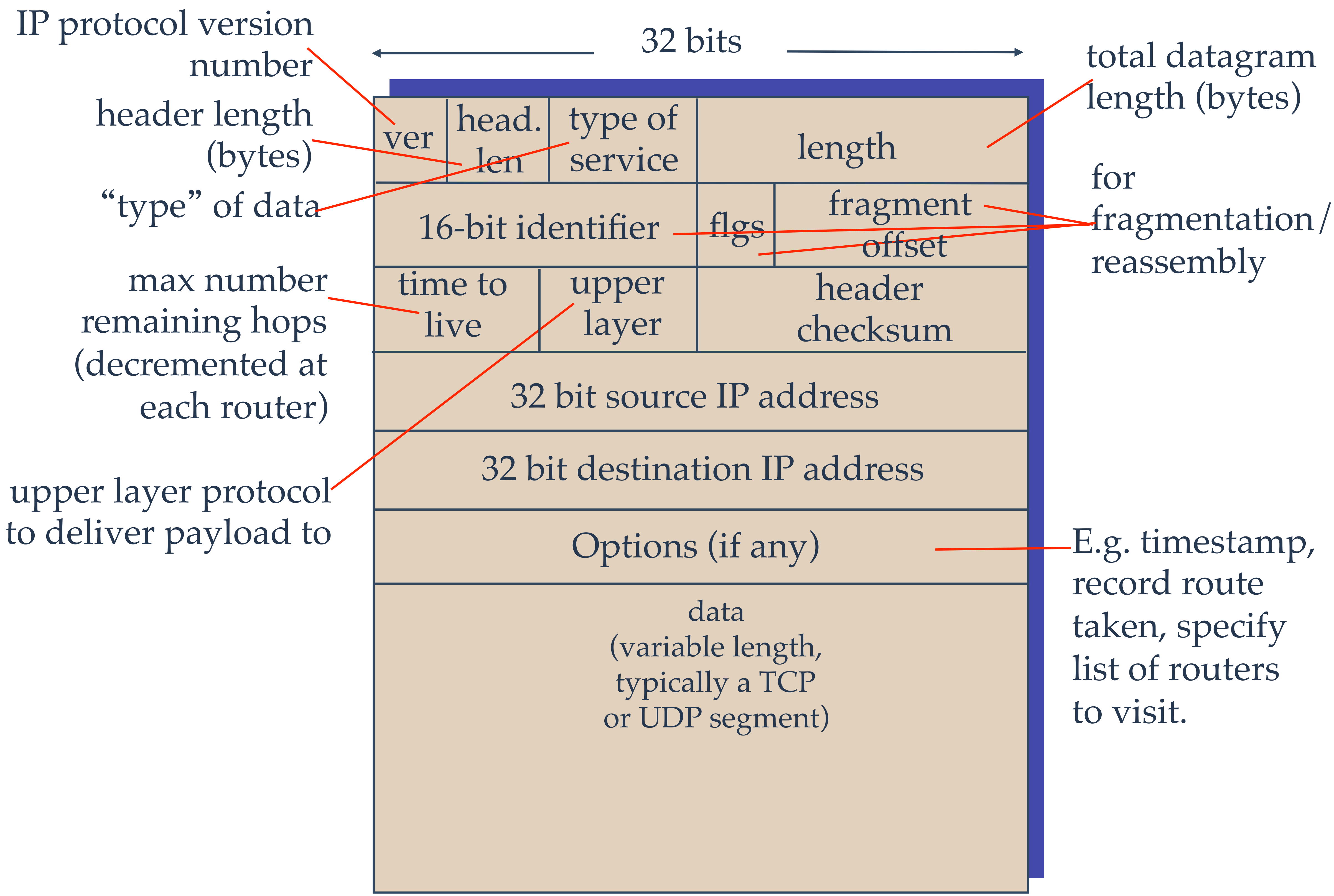
- given datagram destination, lookup output port using forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

The Internet Network layer

Host, router network layer functions:

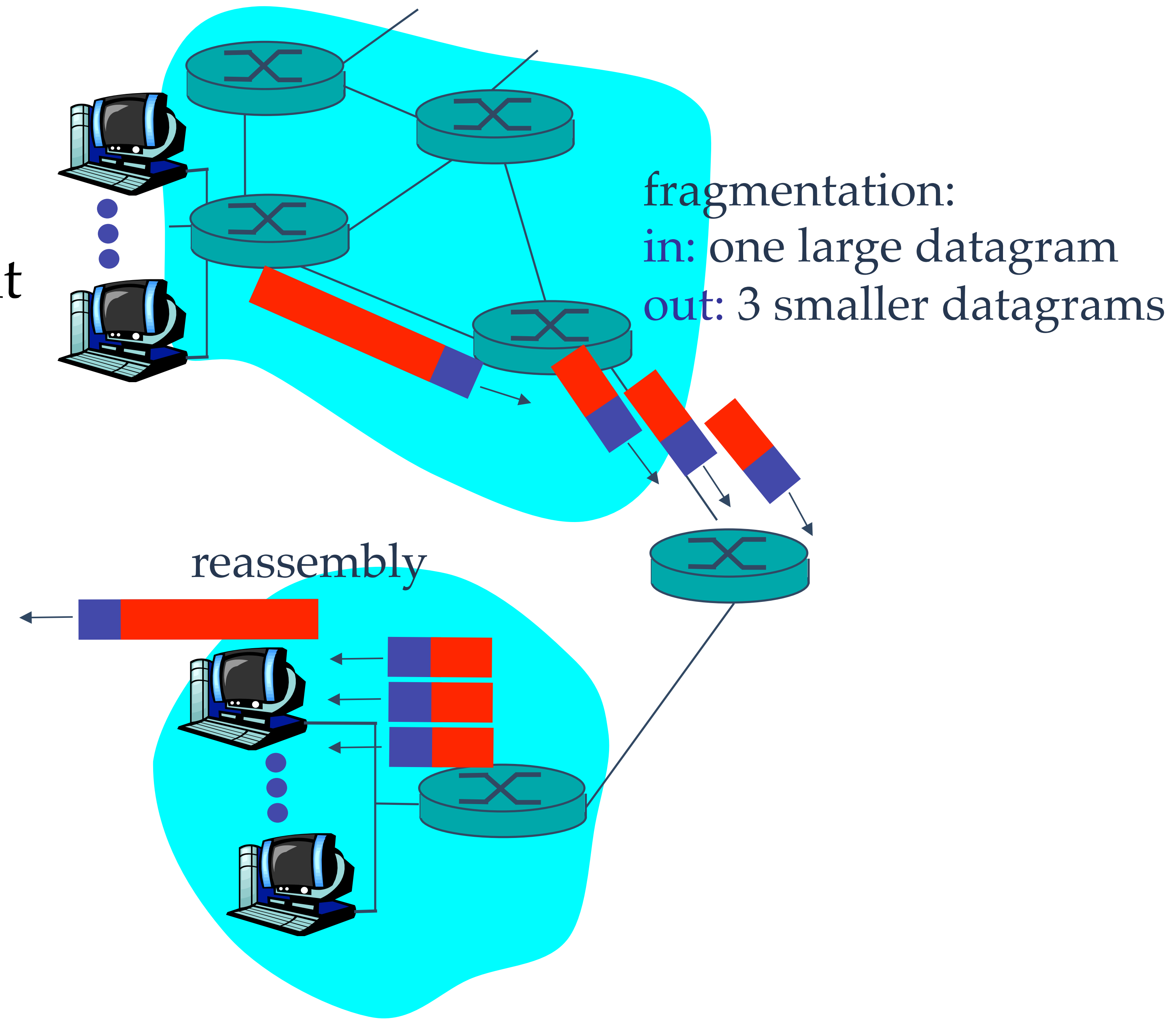


IP datagram format



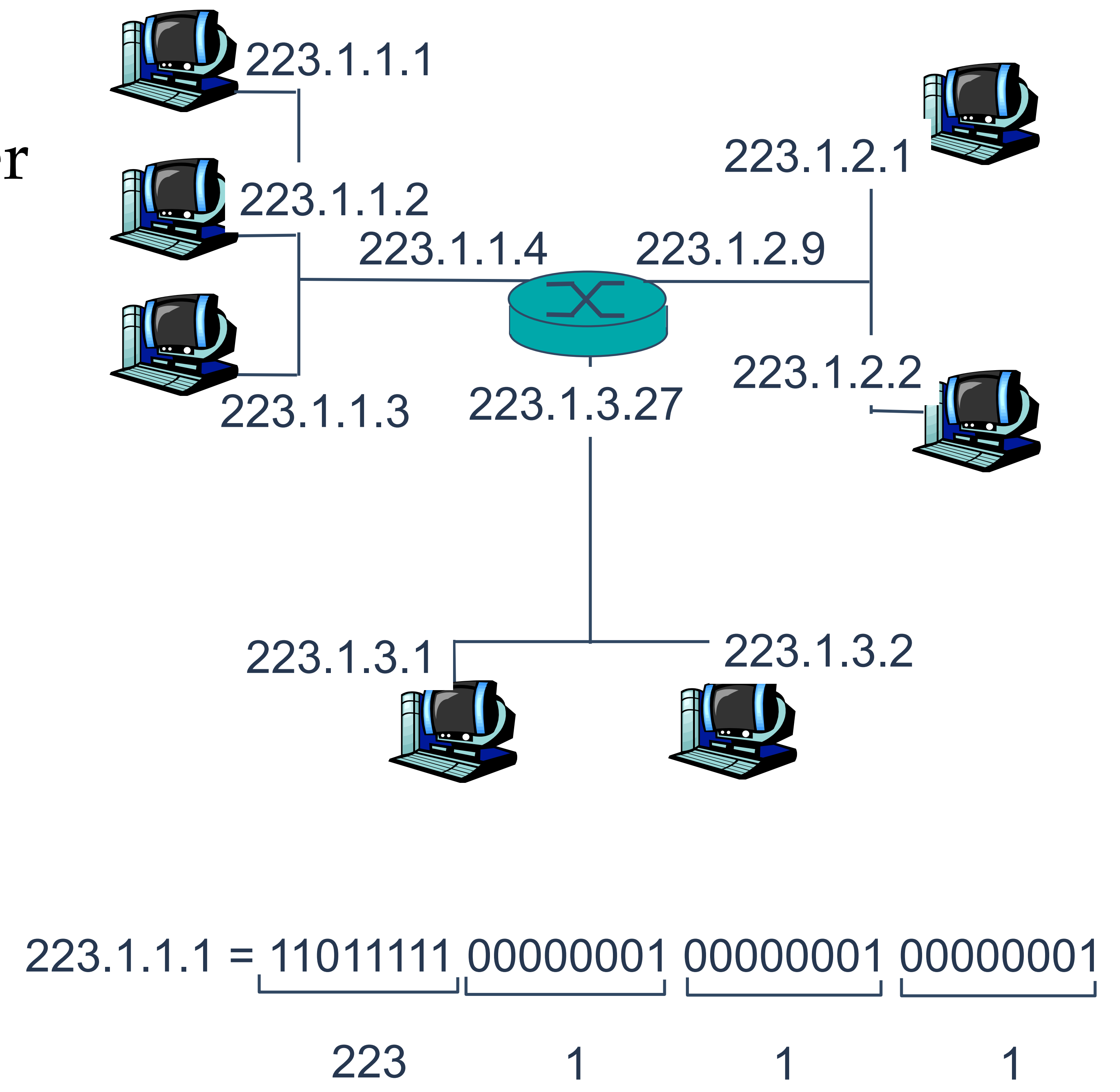
IP Fragmentation & Reassembly

- 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



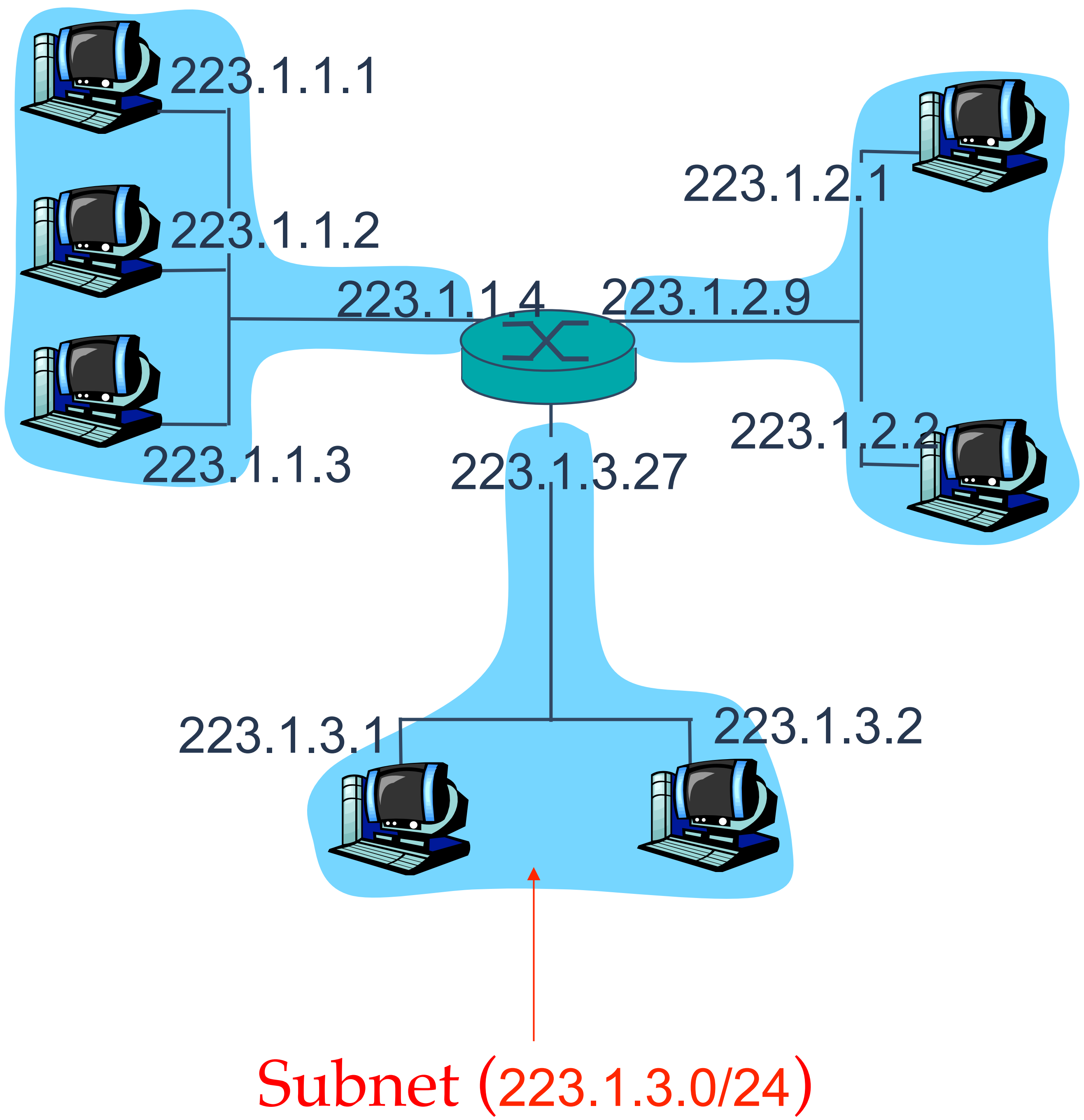
IP Addressing: introduction

- 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



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



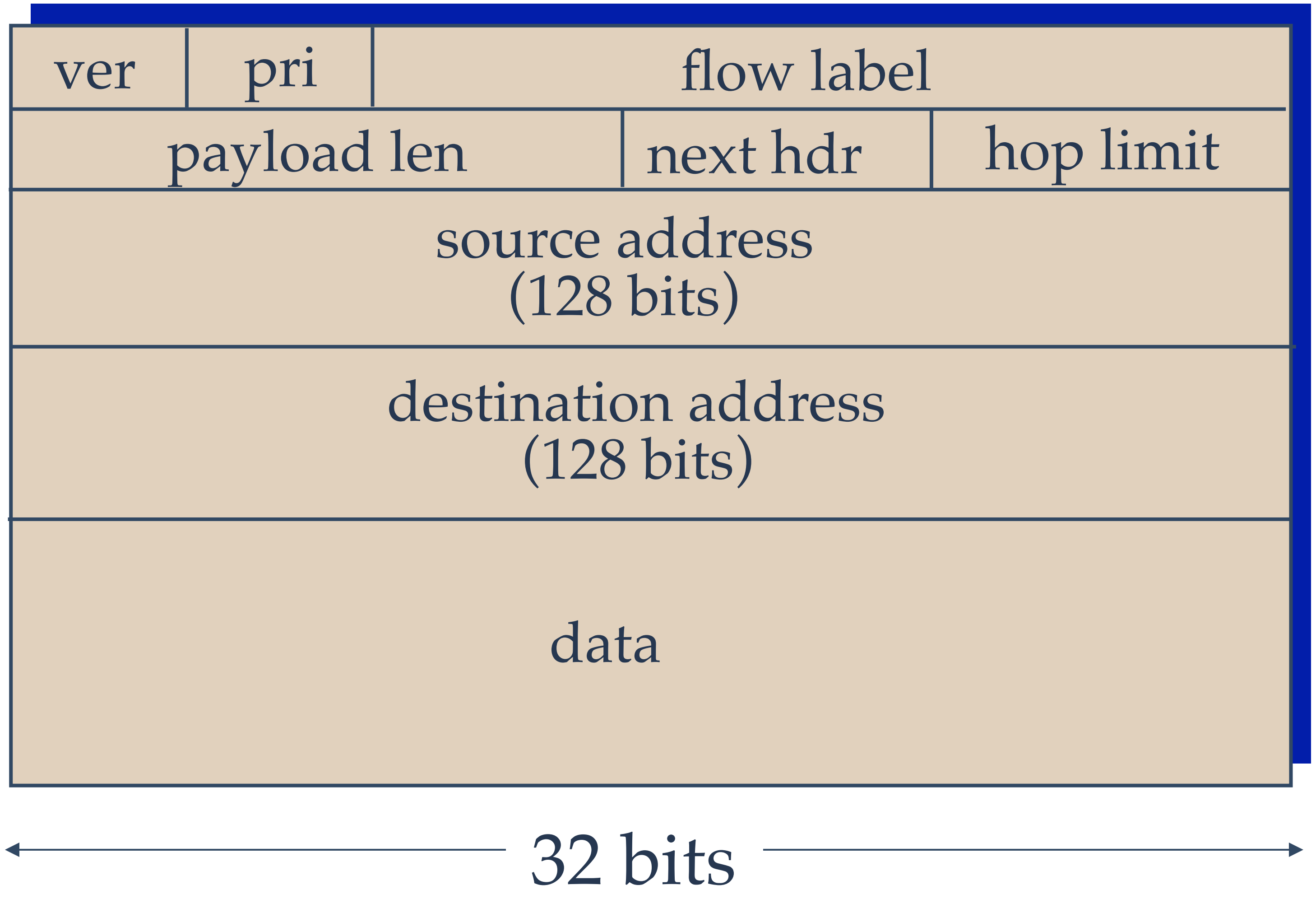
IPv6

- 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)

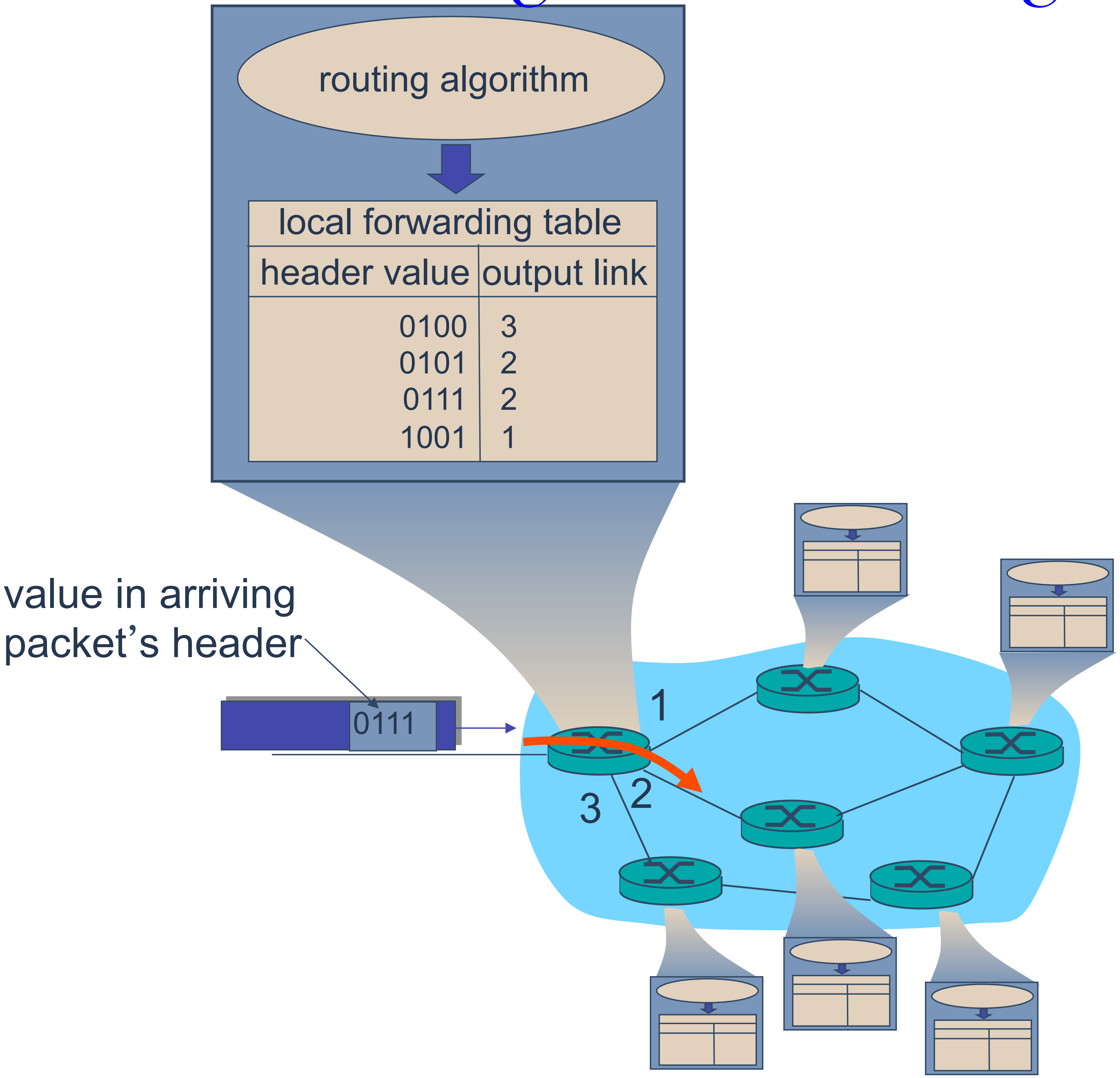
Priority: identify priority among datagrams in flow
Flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).
Next header: identify upper layer protocol for data



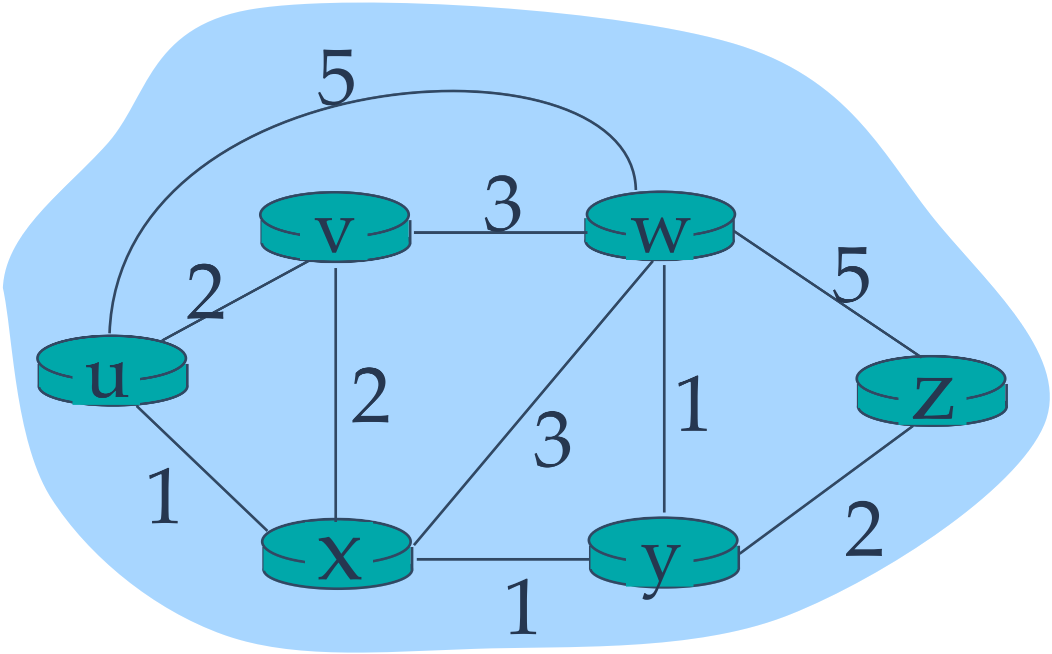
Other Changes from IPv4

- *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

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

set of nodes whose least cost path definitively known

Dijsktra's Algorithm

current value of cost of path from source to destination v

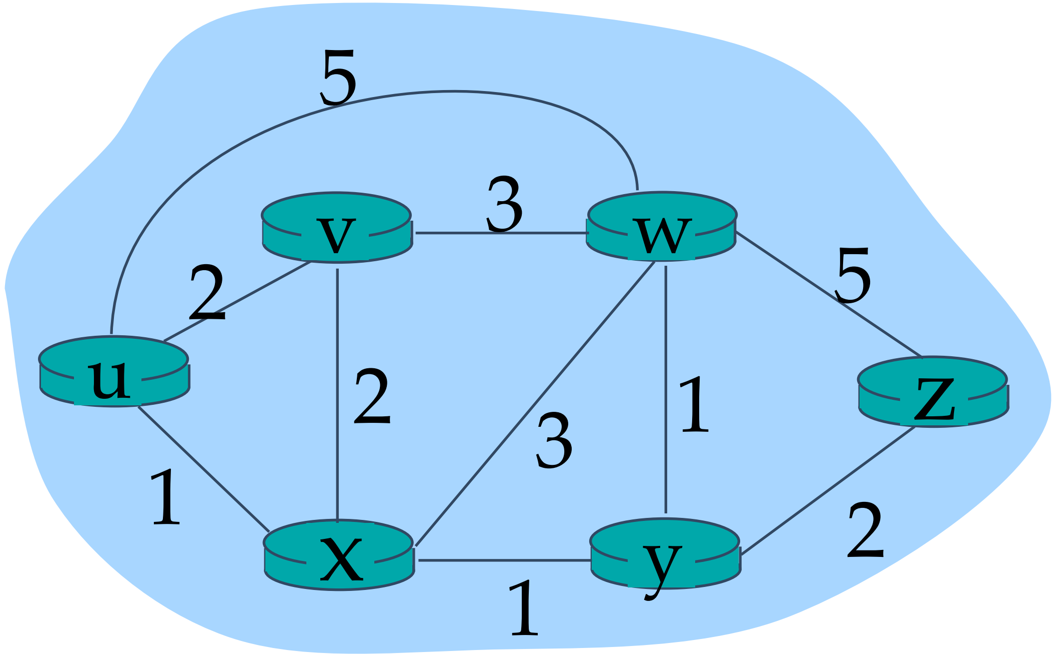
link cost from node x to y; $= \infty$ if not direct neighbors

```
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'
```

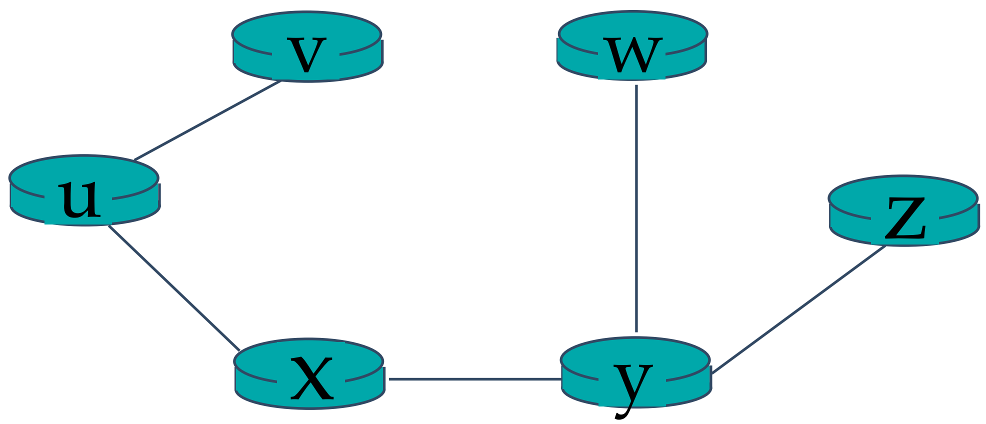
Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	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)

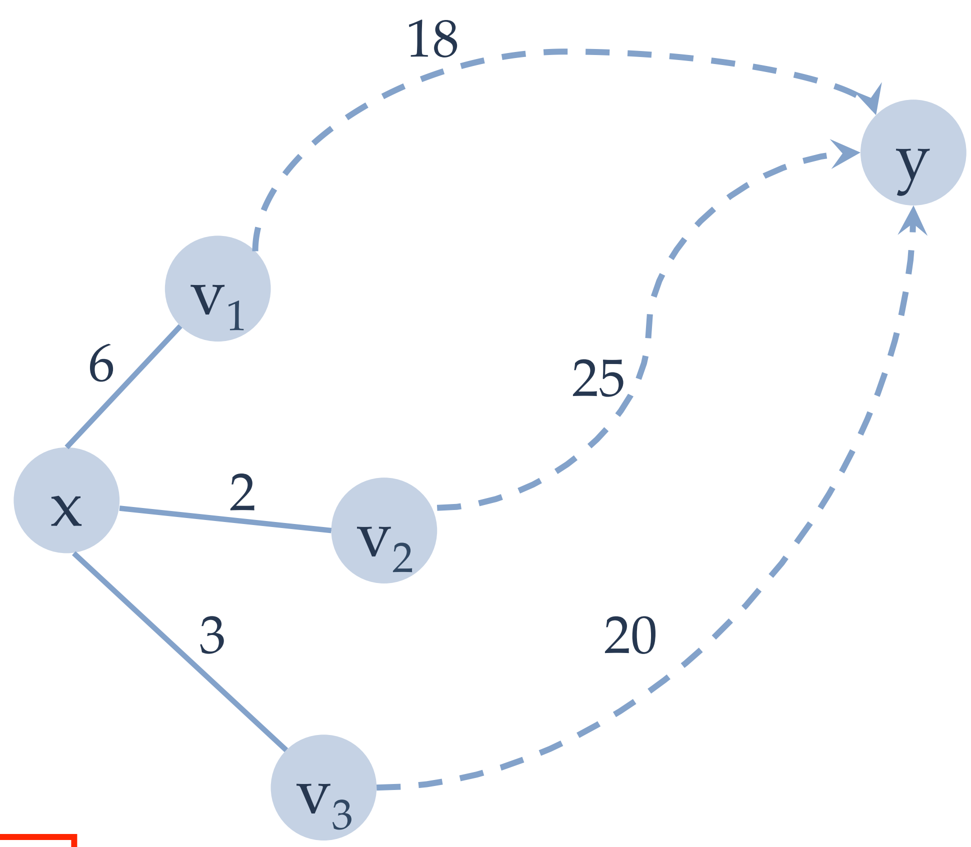
Define

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

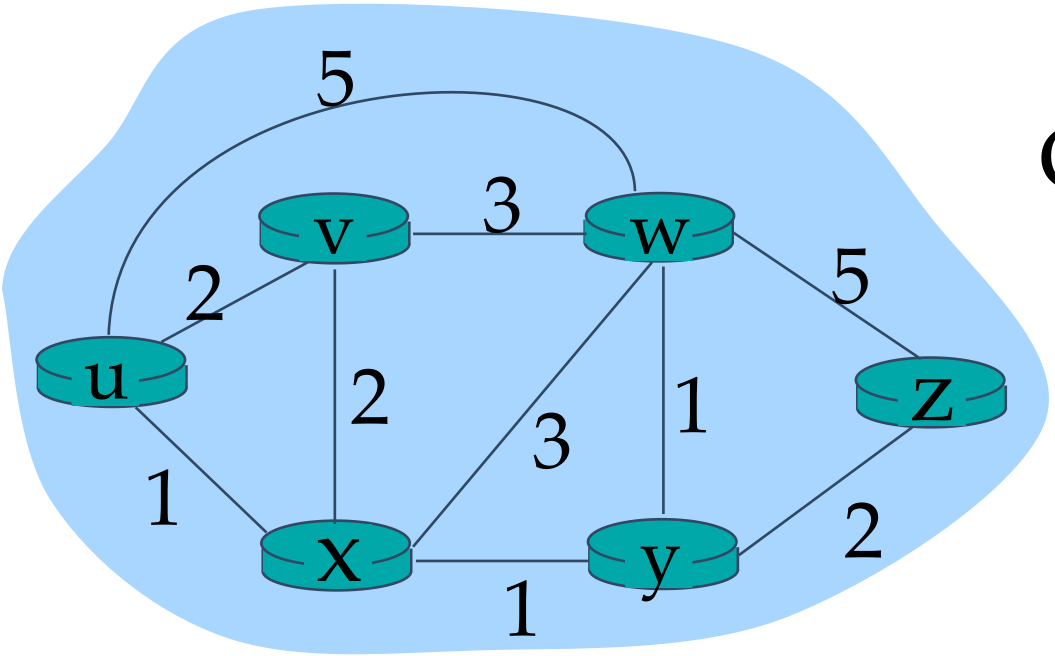
Then

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

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:

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

Node 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:

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

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

Distance Vector Algorithm

Iterative, asynchronous:

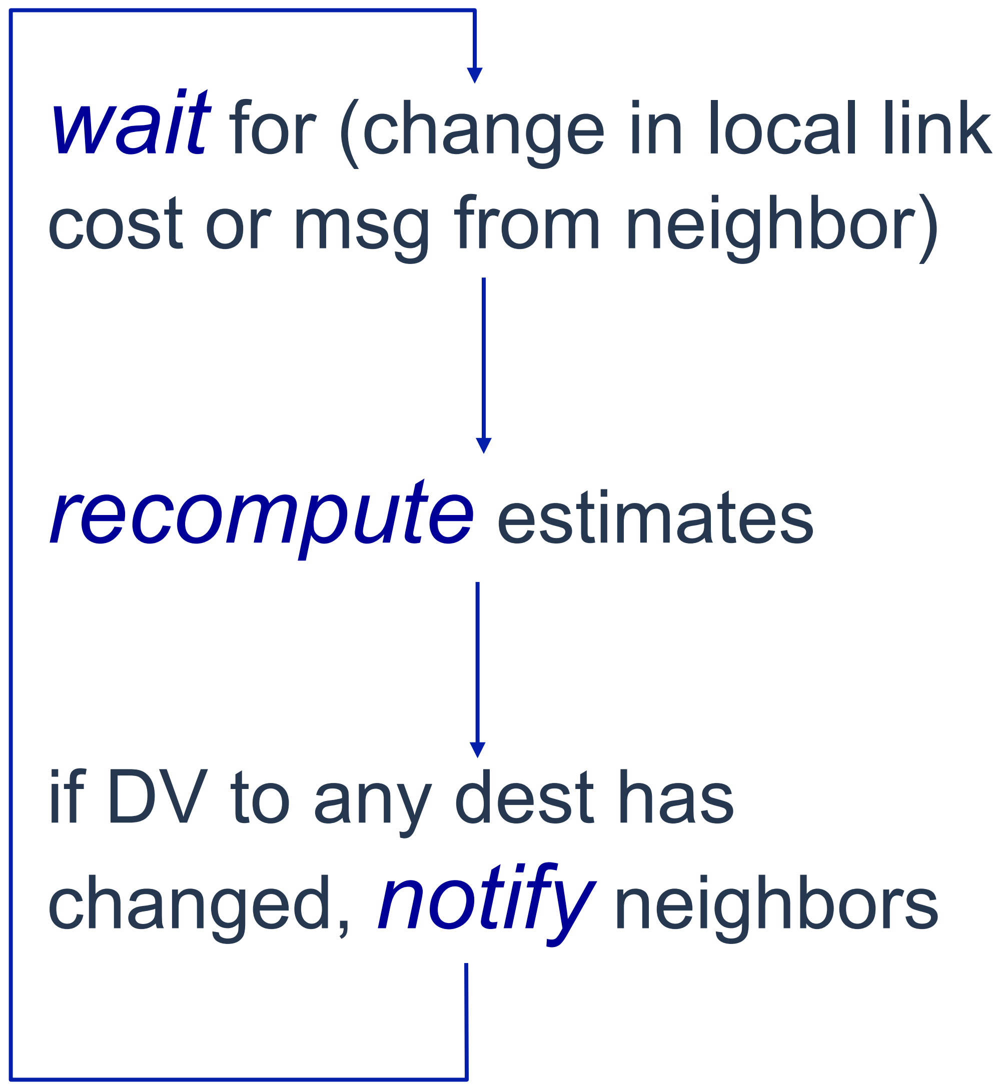
each local iteration caused by:

- 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:



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

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

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

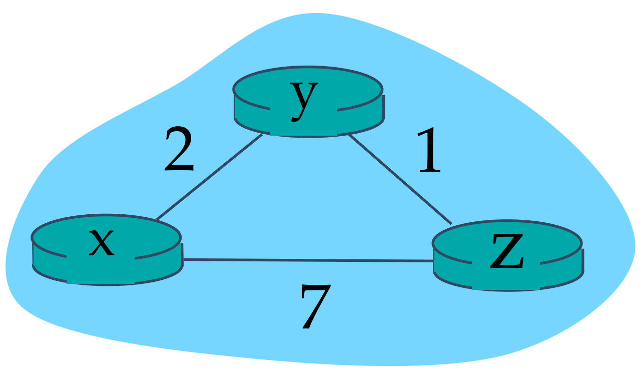
node y table

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

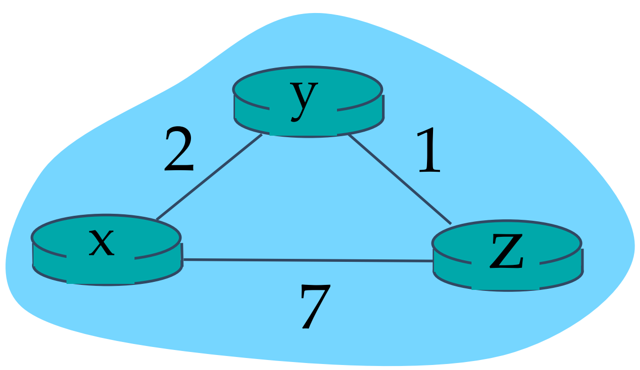
node z table

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

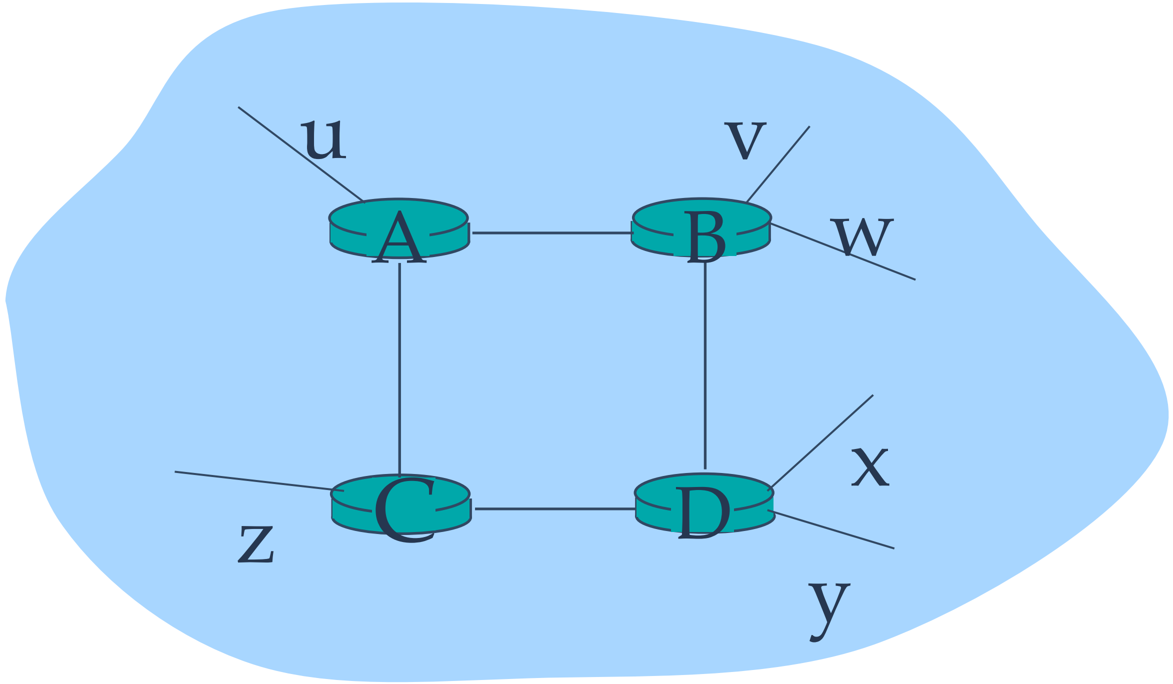
- aggregate routers into regions, autonomous systems (AS)
- routers in same AS run same routing protocol
  - ➡ intra-AS routing protocol
  - ➡ routers in different AS can run different intra-AS routing protocol

gateway router

- ➡ at "edge" of its own AS
- ➡ has link to router in another AS

RIP (Routing Information Protocol)

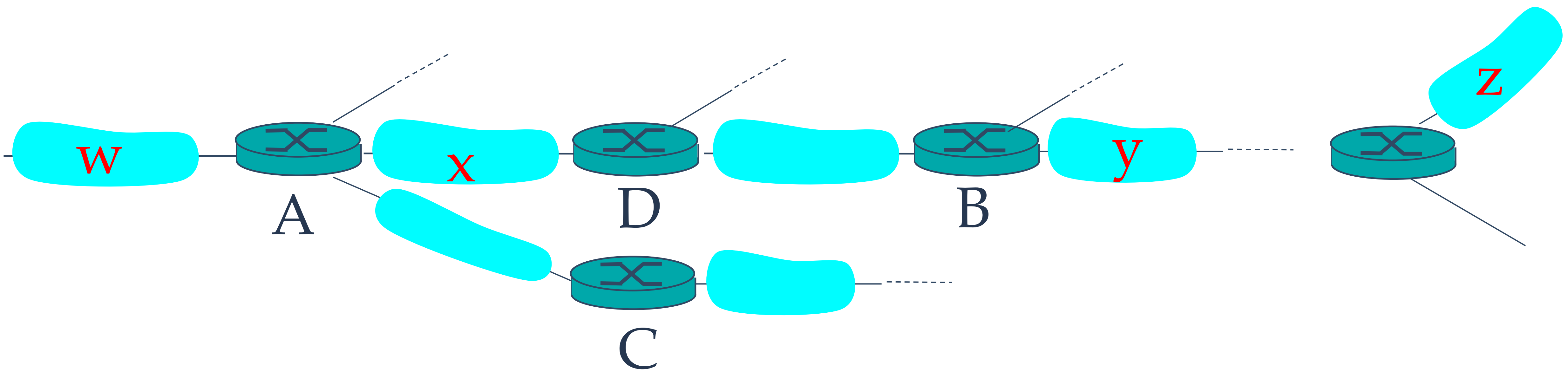
- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets* *(in IP addressing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



from router A to destination subnets:

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

RIP: Example



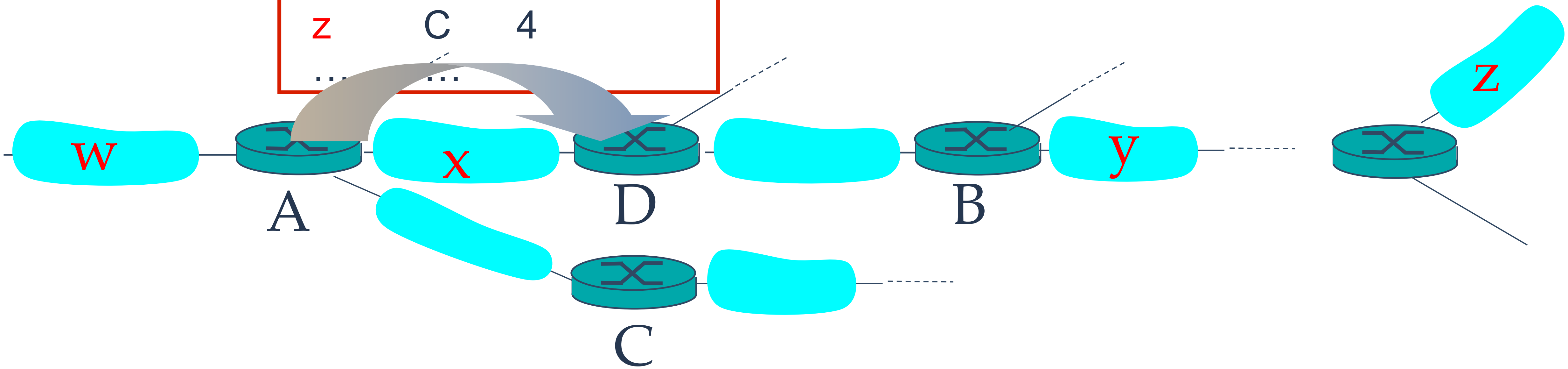
routing table in router D

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
….	….	….

OSPF (Open Shortest Path First)

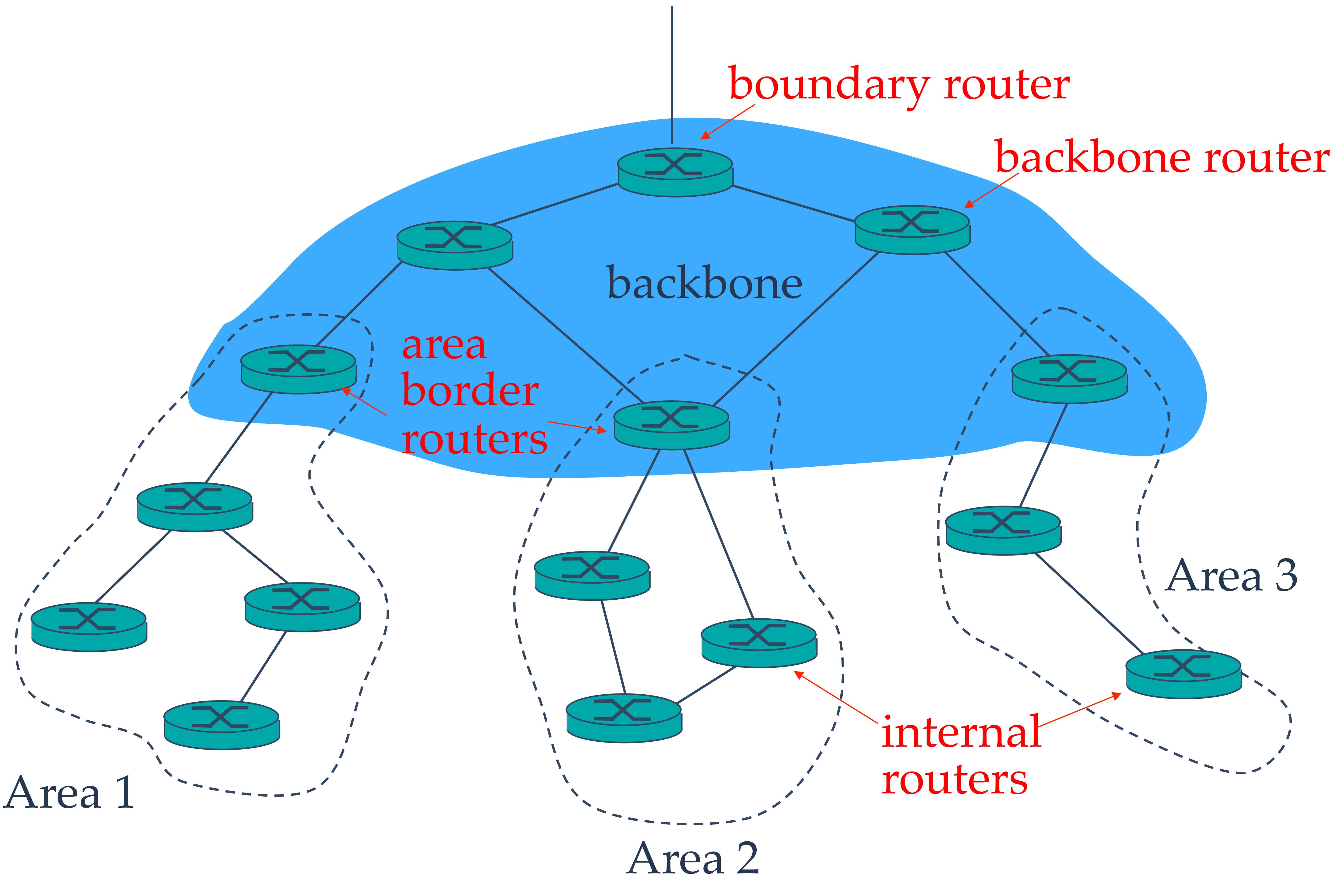
- "open": publicly available
- uses Link State algorithm
  - ➡ LS packet dissemination
  - ➡ topology map at each node
  - ➡ route computation using Dijkstra's algorithm

- 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.