BLG 337E- Principles of Computer Communications

Assist. Prof. Dr. Berk CANBERK

02/12/ 2014

-Network Layer-

References:

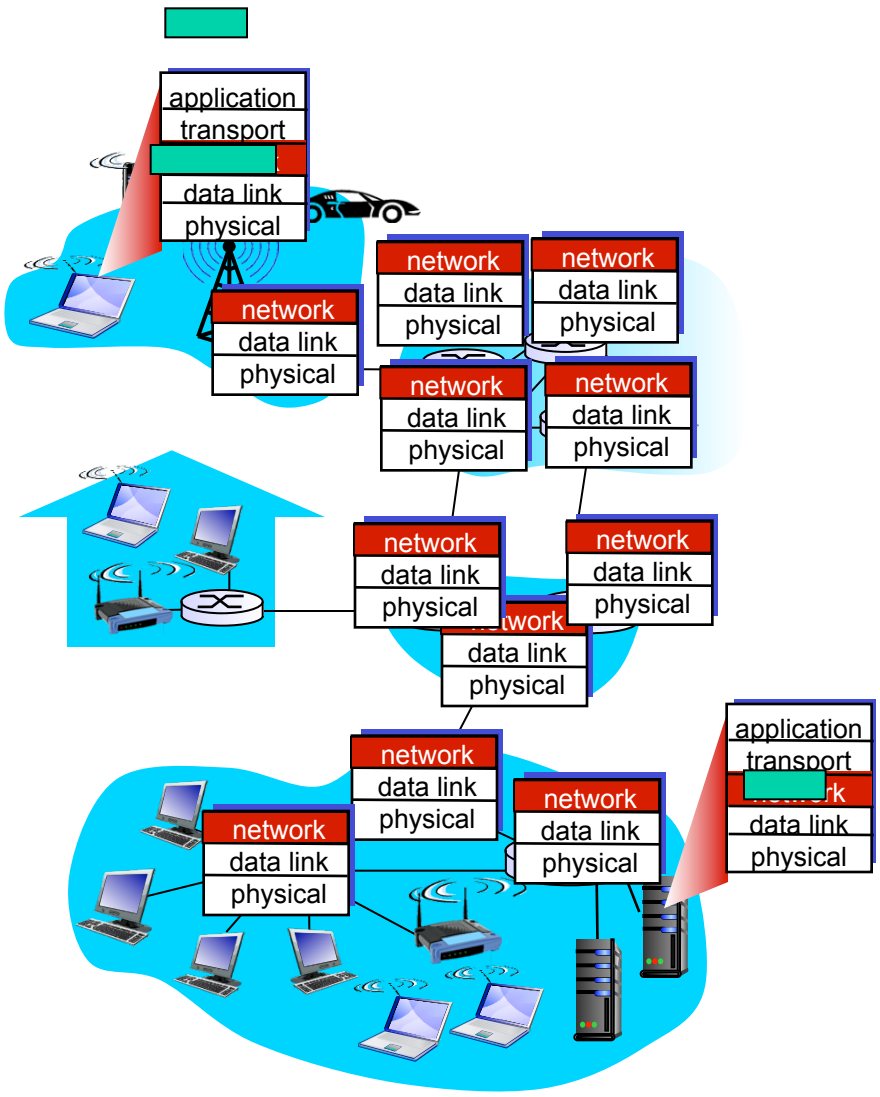
Data and Computer Communications, William Stallings, Pearson-Prentice Hall, 9th Edition, 2010.

-Computer Networking, A Top-Down Approach Featuring the Internet, James F.Kurose, Keith W.Ross, Pearson-Addison Wesley, 6th Edition, 2012.

-Google!

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 connection and connection-less service

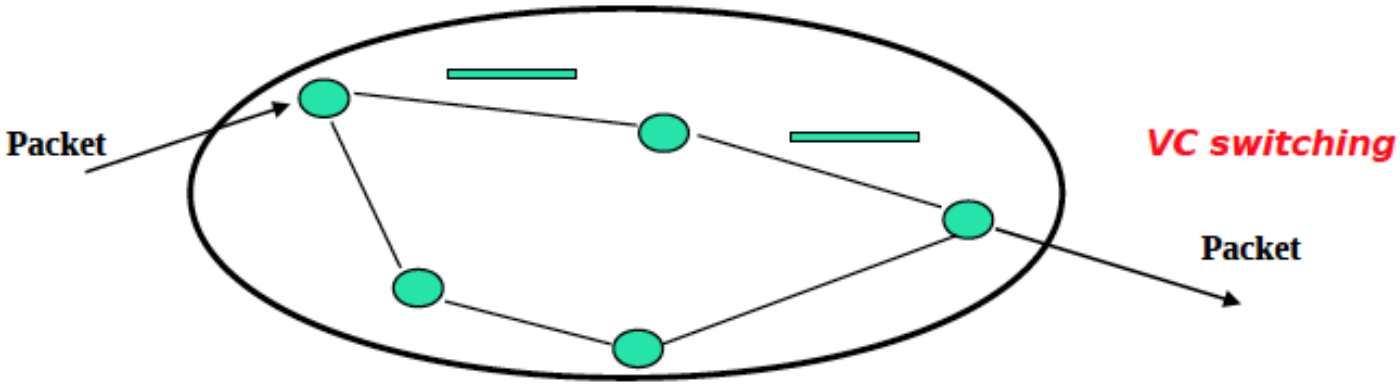
- **Datagram network** provides network-layer connectionless service

- **VC (virtual circuit) network** provides network-layer connection service

Connection Oriented (VC Networks)

"source-to-dest path behaves much like *telephone circuit*"

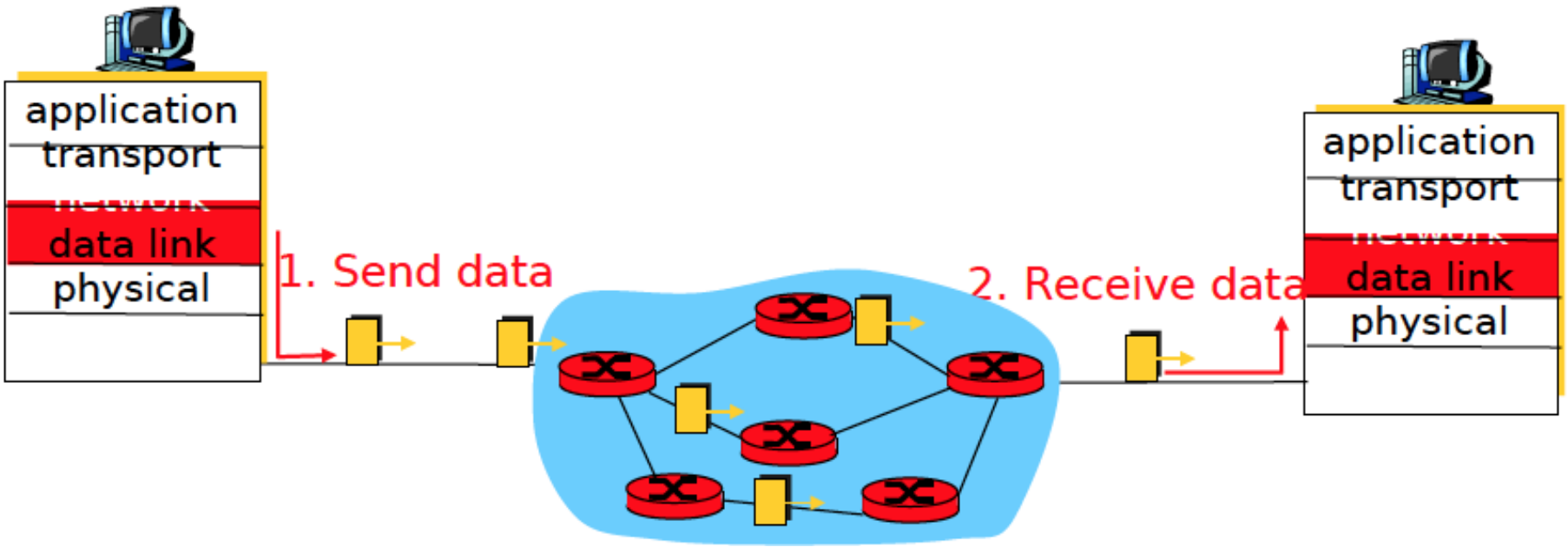
- performance-wise
- network actions along source-to-dest path

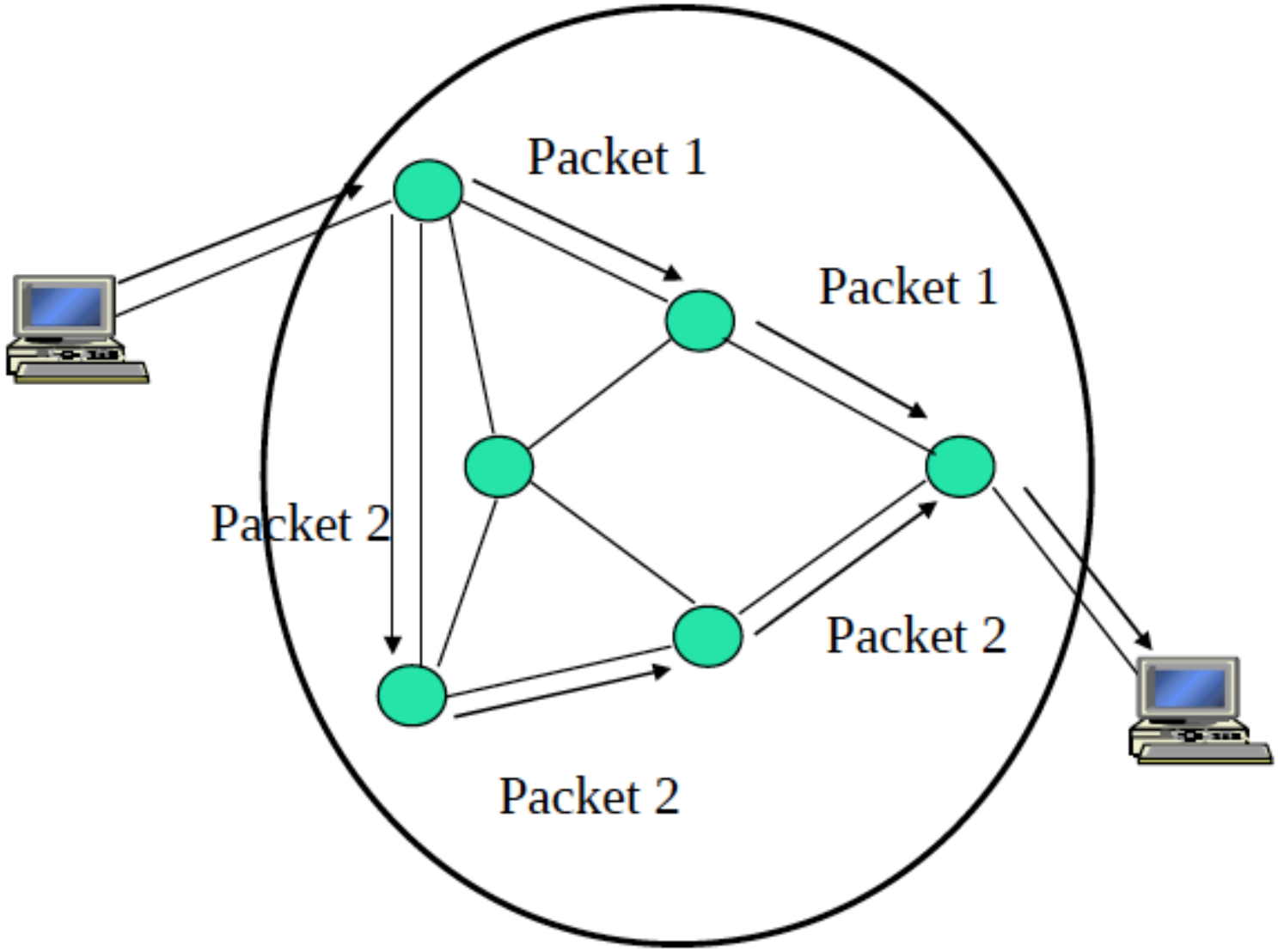


- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- routers on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC

Connectionless (Datagram Networks)

- no call setup at network layer
- routers: do not maintain state for e2e connections
  - no network-level concept of “connection”
- packets forwarded using destination host address
  - packets between the same source-dest pair may take different paths




Packet 1
Packet 1
Packet 2
Packet 2
Packet 2

Routing Table

Destination address	Output port
0785	7
1345	12
1566	6
2458	12

Routing

- ***Routing algorithm*** **:** Part of the Network Layer responsible for deciding on which output line to transmit an incoming packet.
  - Remember: For virtual circuit subnets the routing decision is made ONLY at setup

- **Algorithm properties**:
  - Efficiency, correctness, simplicity, robustness, stability, fairness, optimality, and scalability

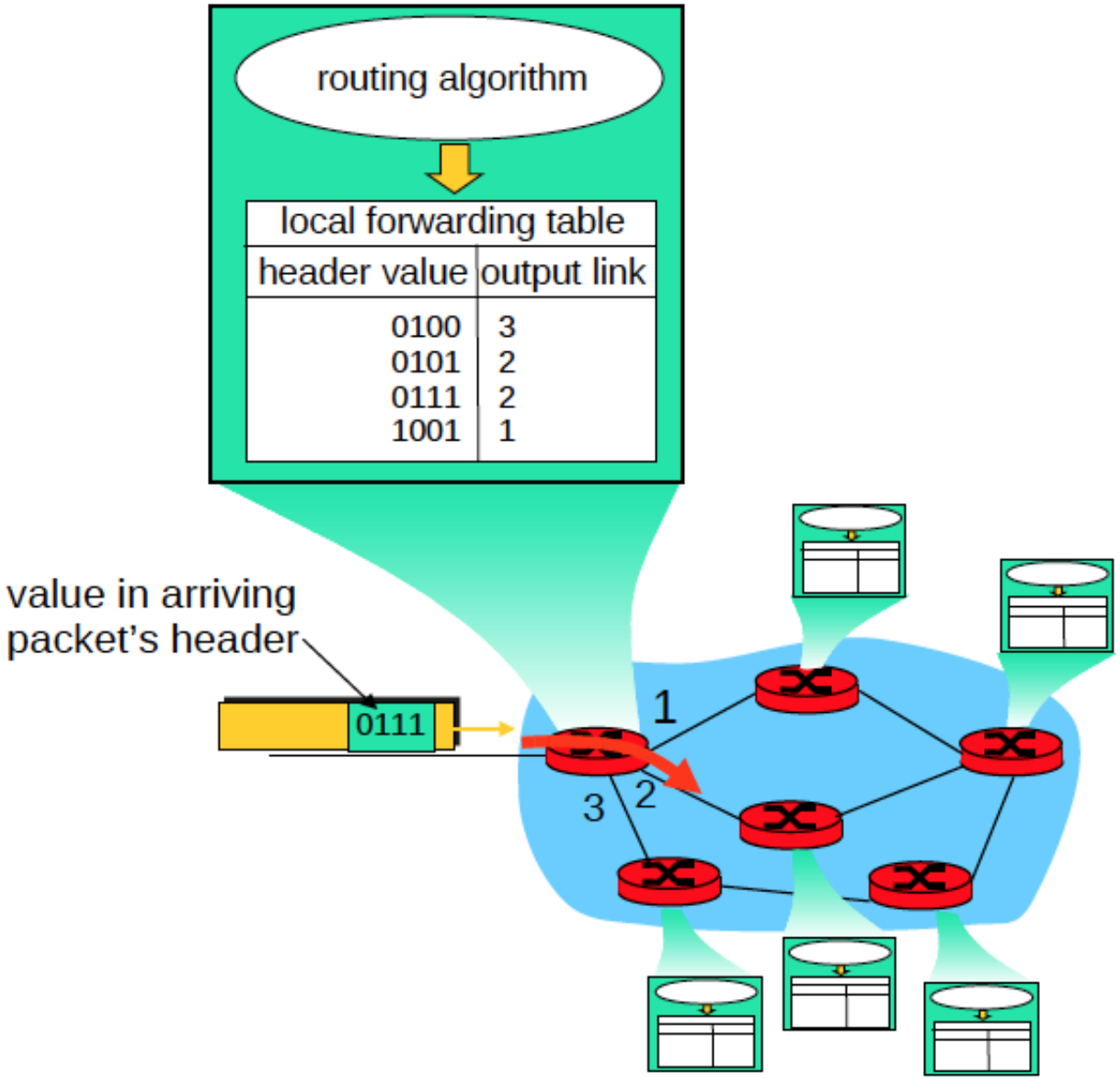
Key Network-Layer Functions

- *routing:* determine route taken by packets from source to dest
- *forwarding:* move packets from router's input to appropriate router output

Analogy:

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

Relation Between Forwarding and Routing



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?

Elements of Routing Techniques

- Performance criteria: Used for selection of routes
  - # of hops, cost, delay, throughput
- Decision Place:
  - Distributed (each node)/Centralized/Source routing
- Decision Time: Packet or VC basis
- Network Information Source:
  - None, local, adjacent node, all nodes
- Network Information Update:
  - Continuous, periodical, on change

Shortest Path Routing

1. Bellman-Ford Algorithm [Distance Vector]
2. Dijkstra's Algorithm [Link State]

What does it mean to be the shortest (or optimal) route?

- Minimize mean packet delay
- Maximize the network throughput
- Minimize the number of hops along the path

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
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

notation:

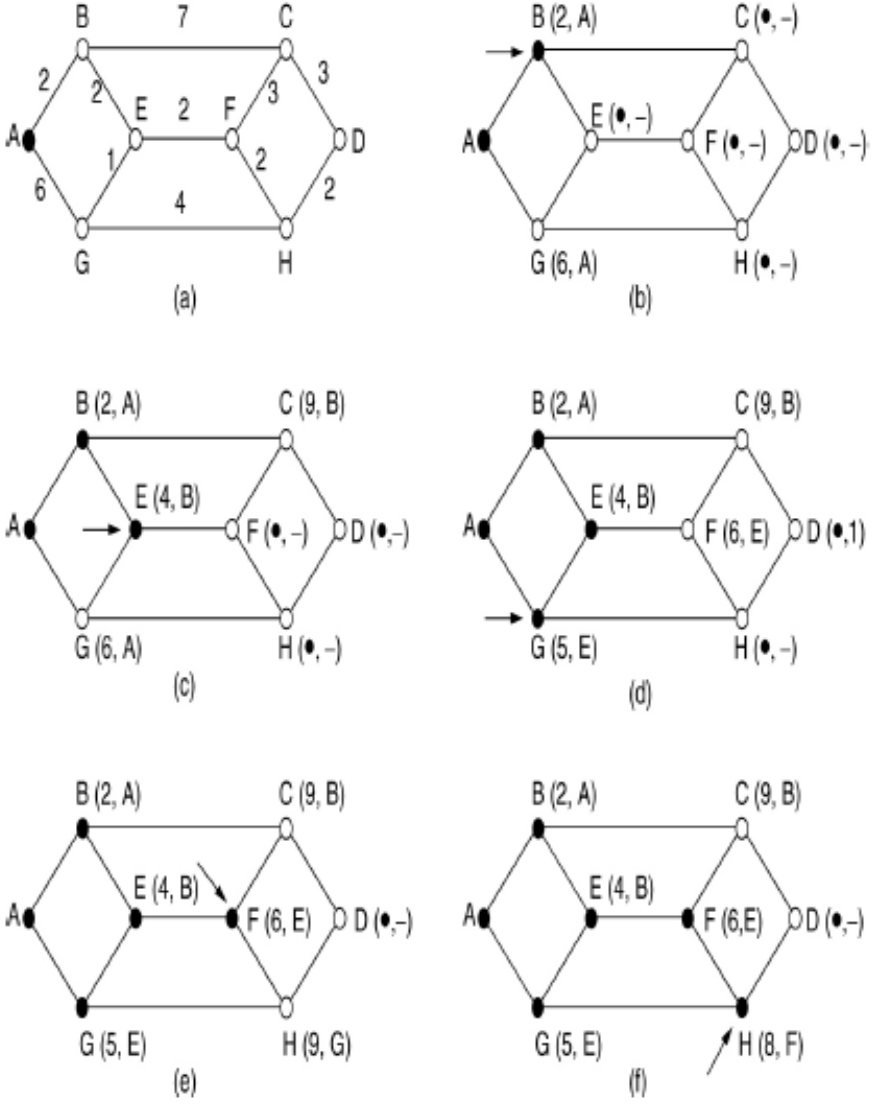
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

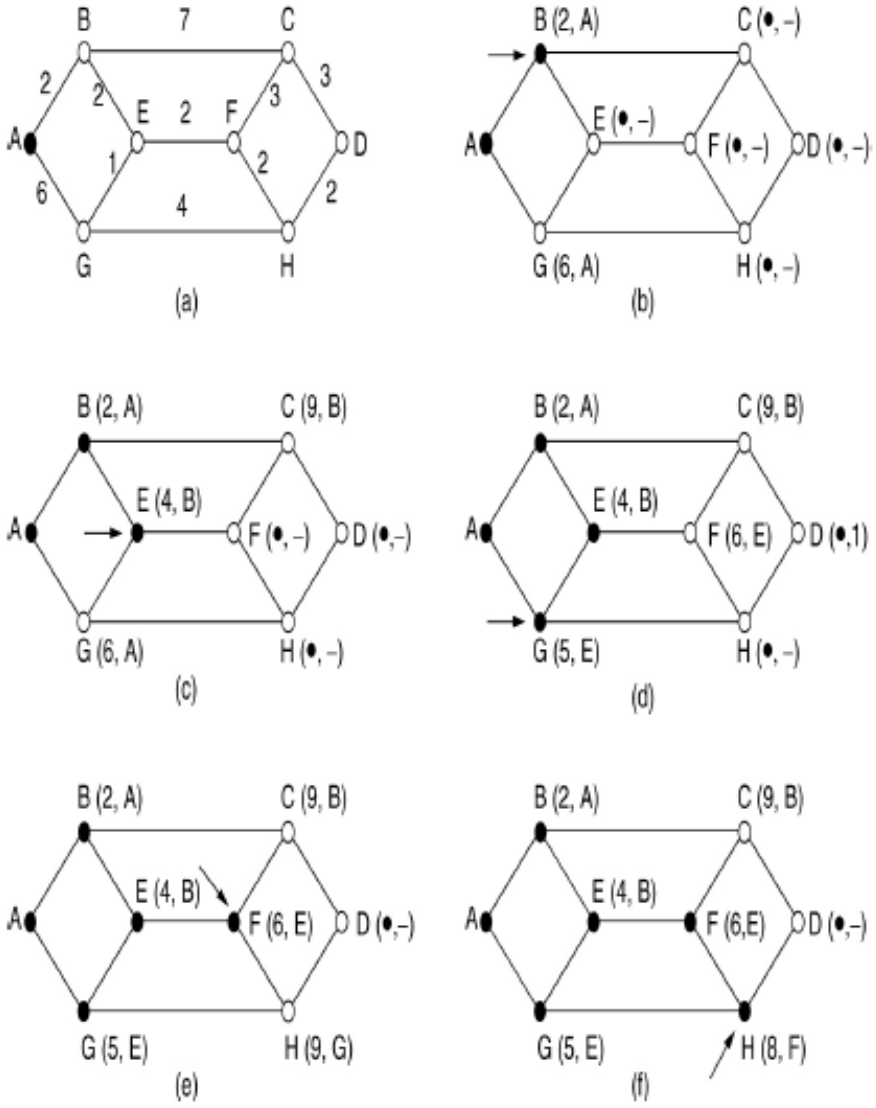
```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9     find w not in N' such that D(w) is a minimum
10     add w to N'
11     update D(v) for all v adjacent to w and not in N' :
12        D(v) = min( D(v), 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'
```

Example: Dijkstra's Shortest Path Algorithm

- Want to find the shortest path from A to D.
- Each node is labeled (in parentheses) with its distance from the source node along the best known path
  - Initially, no paths are known, so all nodes are labeled with infinity
  - A label may be either **tentative** or **permanent**.
  - Initially, all labels are tentative.
- Start out by marking node A as permanent
- Then examine, in turn, each of nodes adjacent to A (the working node), re-labeling each one with the distance to A
- Whenever a node is relabeled, label it with the node from which the probe was made so that the final path can be reconstructed later
- Having examined each of the nodes adjacent to A, examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent
- This one becomes the new working node



- Now start at B and examine all nodes adjacent to it.
- If the sum of the label on B and the distance from B to the node being considered is less than the label on that node → a shorter path, so the node is relabeled
- After all nodes adjacent to working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively-labeled node with the smallest value
- This node is made permanent and becomes the working node for the next round.



Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

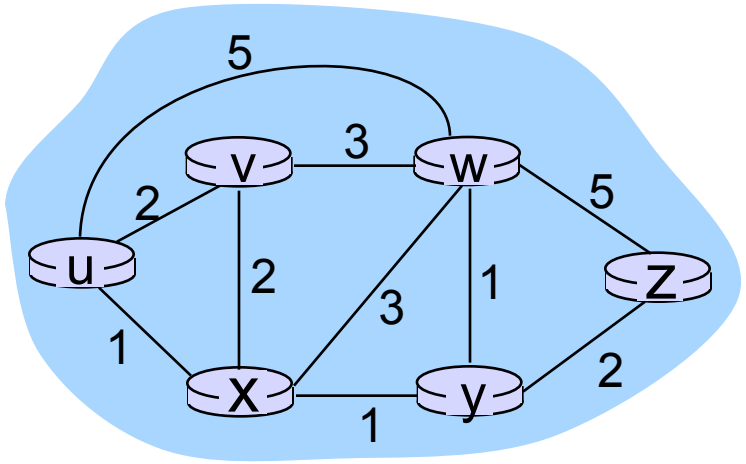
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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:

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

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

Distance vector algorithm

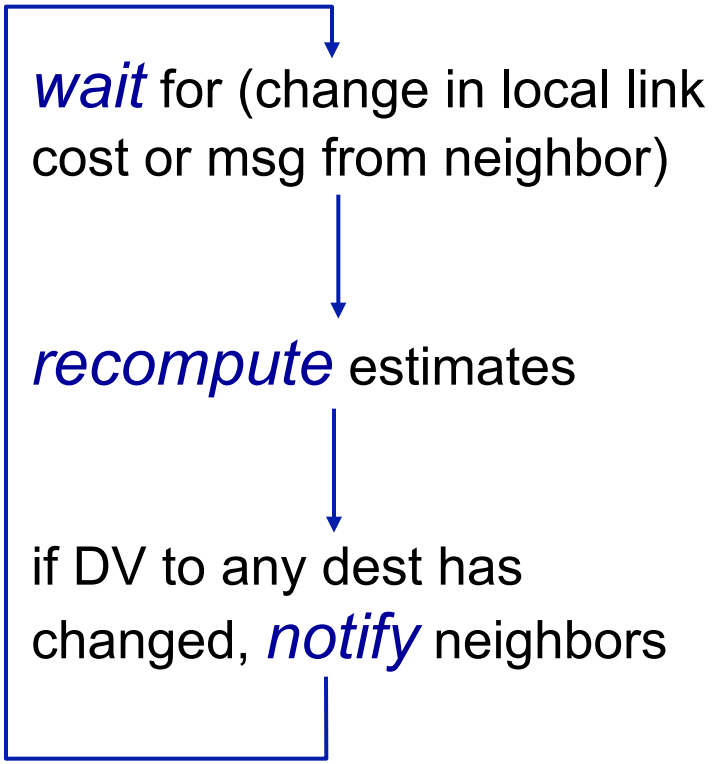
iterative, asynchronous:
each local iteration caused by:

- 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	∞	∞	∞

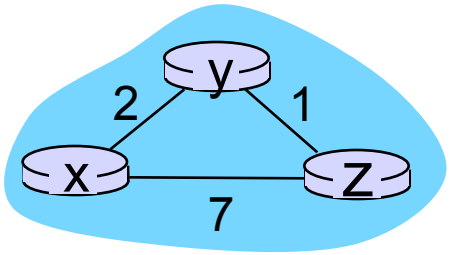
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

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

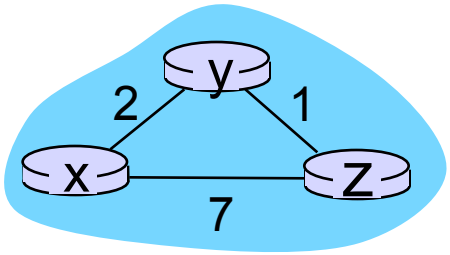
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

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

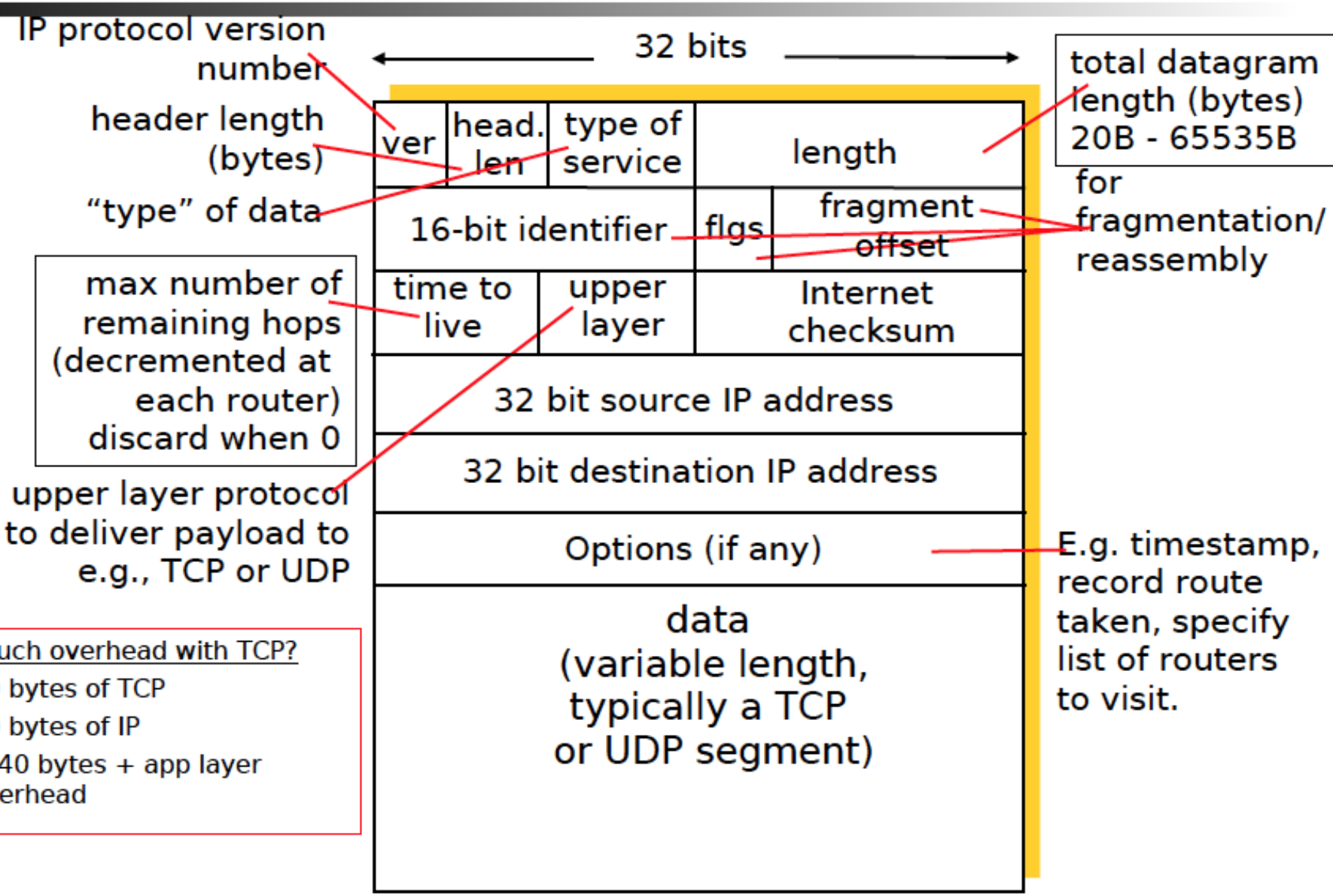
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

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



time

IP Datagram

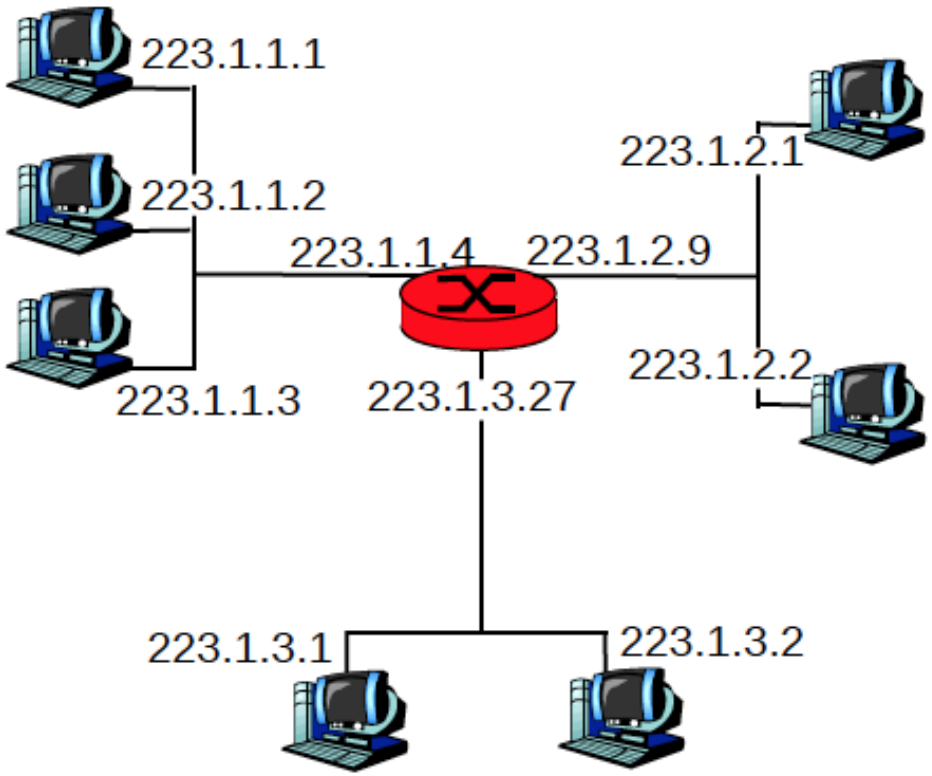


how much overhead with TCP?

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

IP Addressing

- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - routers typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with each interface

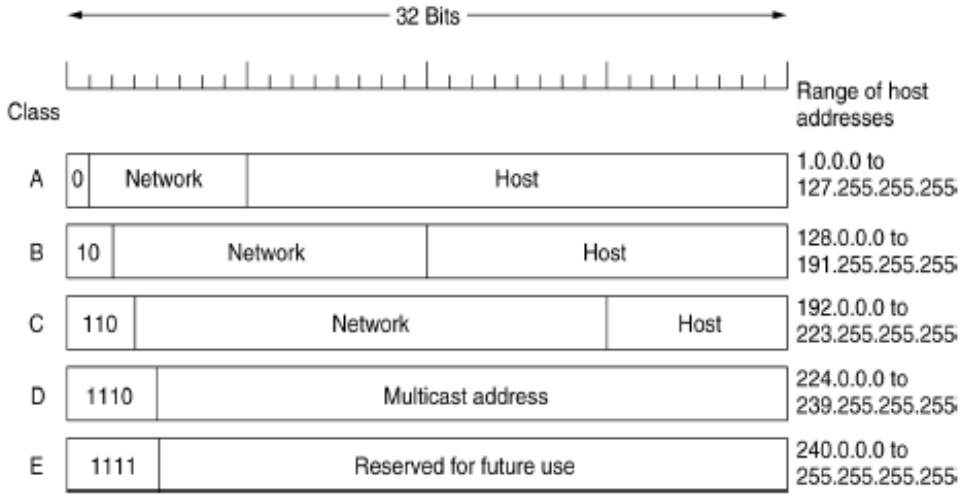


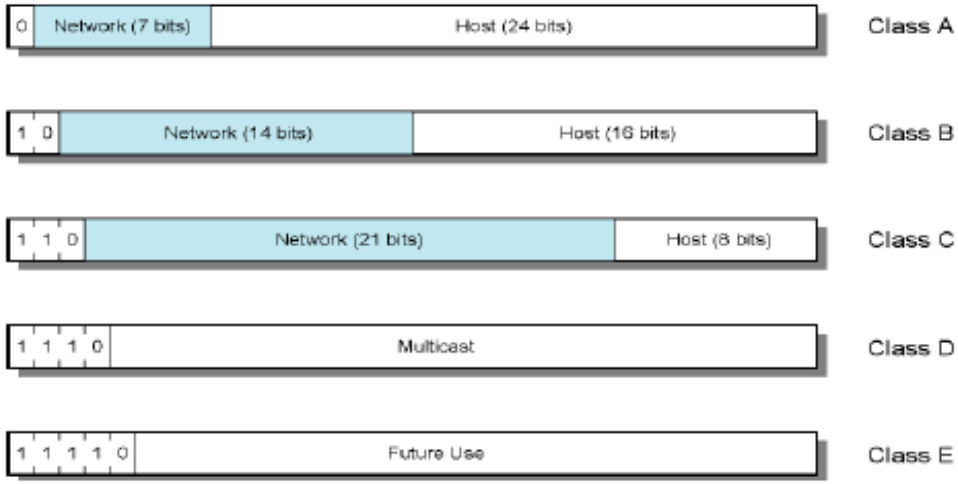
223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP Addresses

- 32 bit global internet address
- Network part and host part
- **Class A**
  - Start with binary 0
  - All 0 reserved
  - 01111111 (127) reserved for loopback
  - Range 1.x.x.x to 126.x.x.x
  - All allocated
- **Class B**
  - Start 10
  - Range 128.x.x.x to 191.x.x.x
  - Second Octet also included in network address
  - 2^{14} = 16,384 class B addresses
  - All allocated
- **Class C**
  - Start 110
  - Range 192.x.x.x to 223.x.x.x
  - Second and third octet also part of network address
  - 2^{21} = 2,097,152 addresses
  - Nearly all allocated



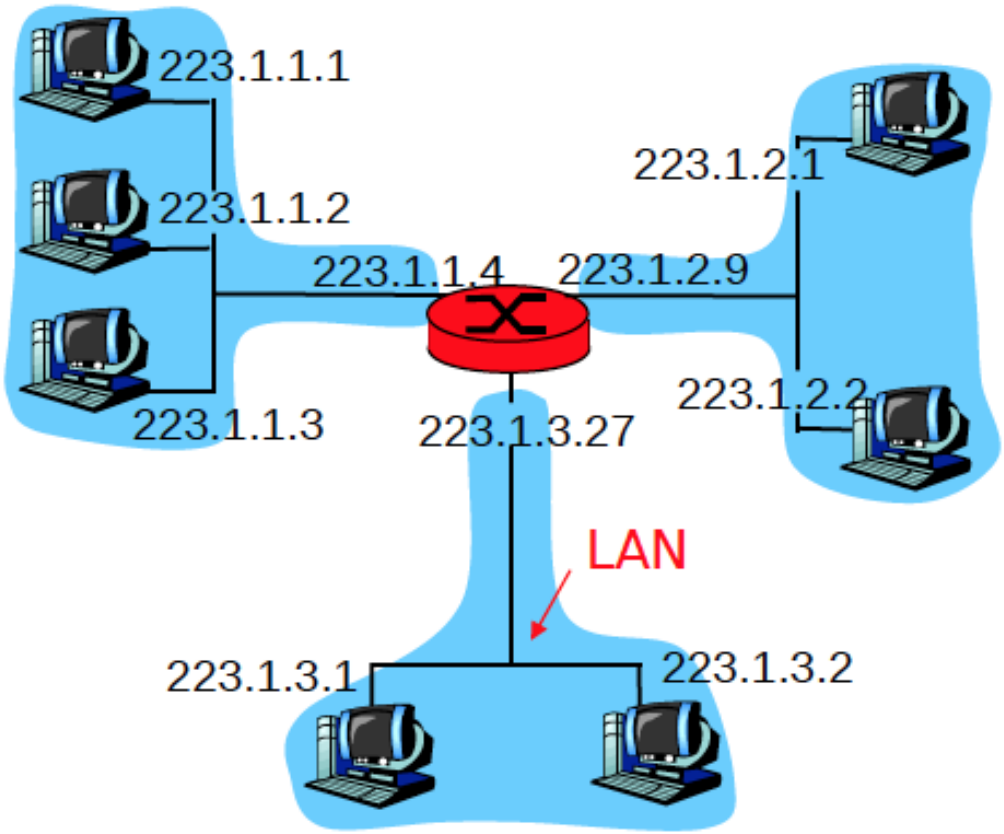


Subnets and Subnet Masks

- Classes are too coarse
  - Additional partitioning may be needed
- Each LAN assigned subnet number
  - Host portion of address partitioned into subnet number and host number
  - ***Subnet mask*** *indicates which bits are subnet number and which are host number*
- Consider a class B address 144.122.x.x has 16 bits network part 16 bits hosts part
  - 2^{16} hosts → may be unnecessary
  - Difficult to keep routing for 2^{16} entries
  - Additional partitions may be required (e.g., departments etc.)
  - e.g. use 6 bits for subnet, and 10 bits for hosts
    - Network+subnet=16+6=22 bits=subnetmask
    - 64 subnets with 1022 hosts (0, and 255 are **not** available as host octet)
  - IP address **AND** Subnetmask = Network Address

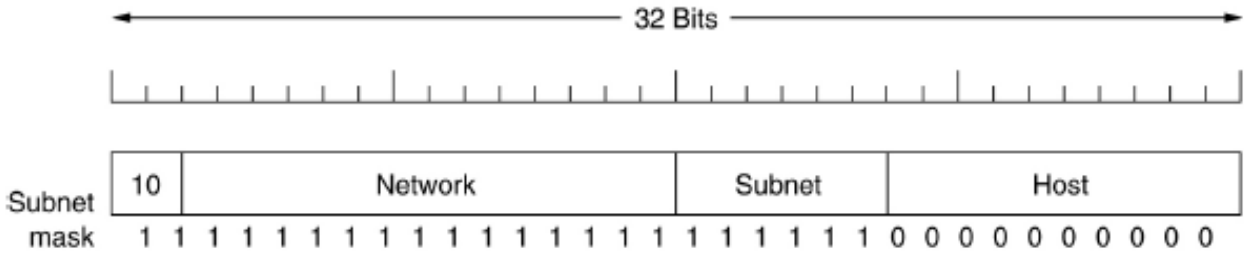
Subnets

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



network consisting of 3 subnets

Subnets



(**A class B network (130.50.x.x) subnetted into 64 subnets**)

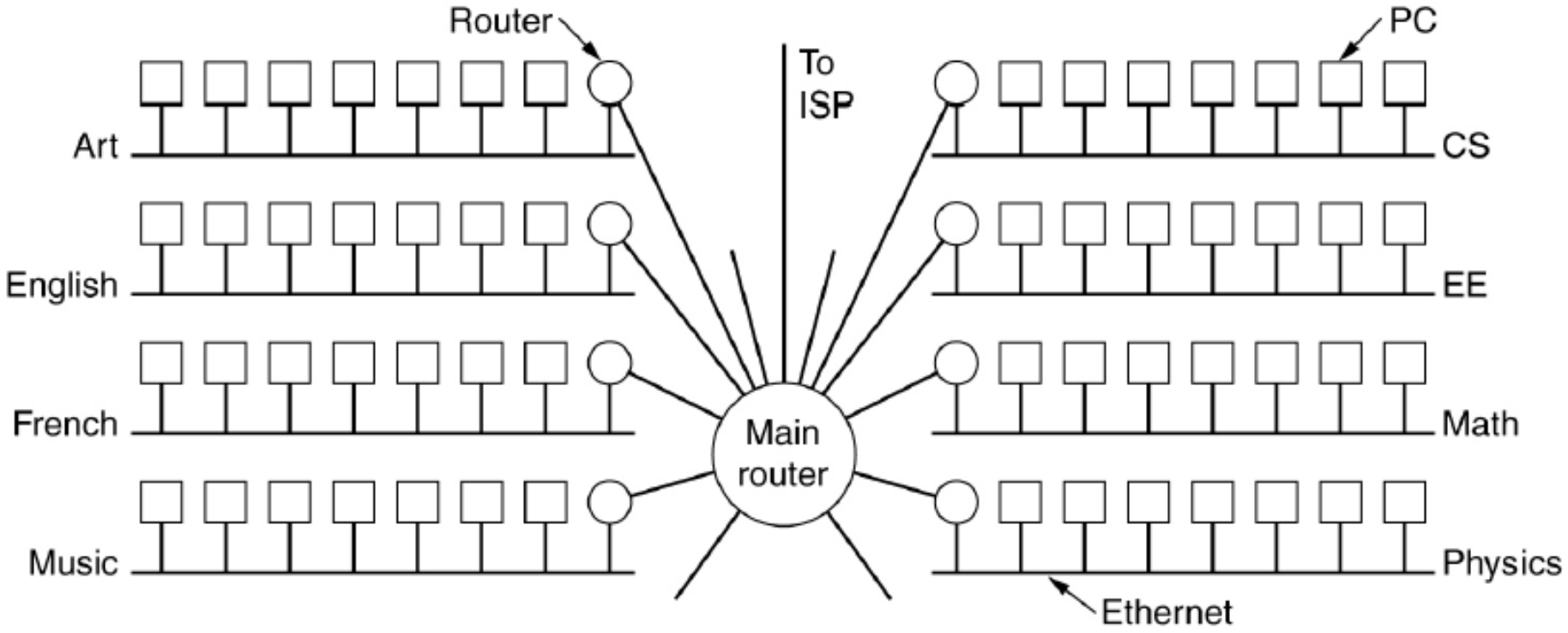
- To implement subnetting, the main router needs a subnet mask that indicates the split between network + subnet number and host.
- 255.255.252.0 or /22 to indicate that the subnet mask is 22 bits long.
- Subnet 0 might use IP addresses starting at 130.50.0.1; Subnet 1 might start at 130.50.4.1; Subnet 3 might start at 130.50.8.1; and so on

Subnet 0: 10000010 00110010 000000|00 00000001
Subnet 1: 10000010 00110010 000001|00 00000001
Subnet 2: 10000010 00110010 000010|00 00000001
...
Subnet 63: 10000010 00110010 111111|00 00000001

- When an IP packet arrives, router does a Boolean AND with network's subnet mask to get rid of host number and look up the resulting address in its tables
  - e.g. packet addressed to 130.50.15.6 and arriving at the main router is ANDed with the subnet mask 255.255.252.0/22 to give the address 130.50.12.0.
  - This address is looked up to find out which output line to use to get to Subnet 3

Subnets

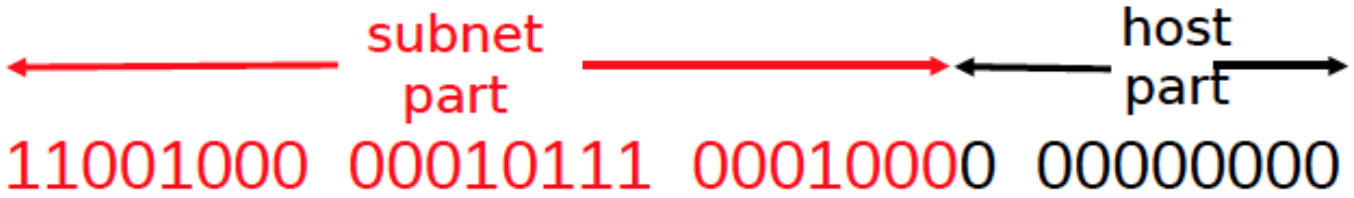
A campus network consisting of LANs for various departments.



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



200.23.16.0/23

CIDR – Classless InterDomain Routing

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Address	Mask
C: 11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E: 11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000
O: 11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

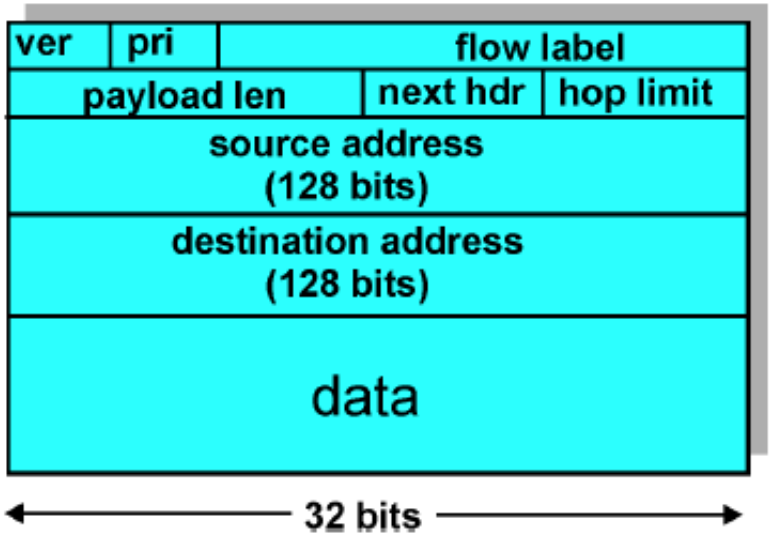
- Consider when a packet comes in addressed to 194.24.17.4, which in binary is represented as the following 32-bit string 11000010 00011000 00010001 00000100
- First it is Boolean ANDed with the Cambridge mask to get
  11000010 00011000 00010000 00000000
- This value does not match the Cambridge base address, so the original address is next ANDed with the Edinburgh mask to get
  11000010 00011000 00010000 00000000
- This value does not match the Edinburgh base address, so Oxford is tried next, yielding
  11000010 00011000 00010000 00000000
- This value does match the Oxford base. If no longer matches are found farther down the table, the Oxford entry is used and the packet is sent along the line named in it.
  - what if 192.168.20.19 is the destination; 192.168.20.16/28 and 192.168.0.0/16 are two entries in the routing table?
  - (longest prefix matching)

check http://www.subnet-calculator.com/

IPv6

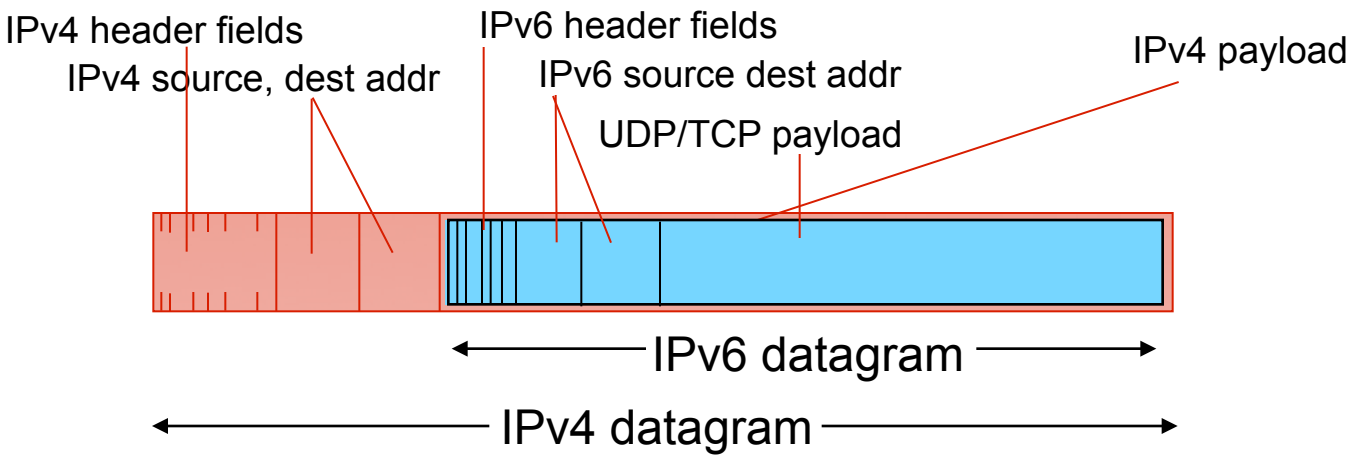
Priority: identify priority among datagrams in flow
Flow Label: identify datagrams in the same "flow."
(concept of"flow" not well defined).
Next header: identify upper layer protocol for data

- 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

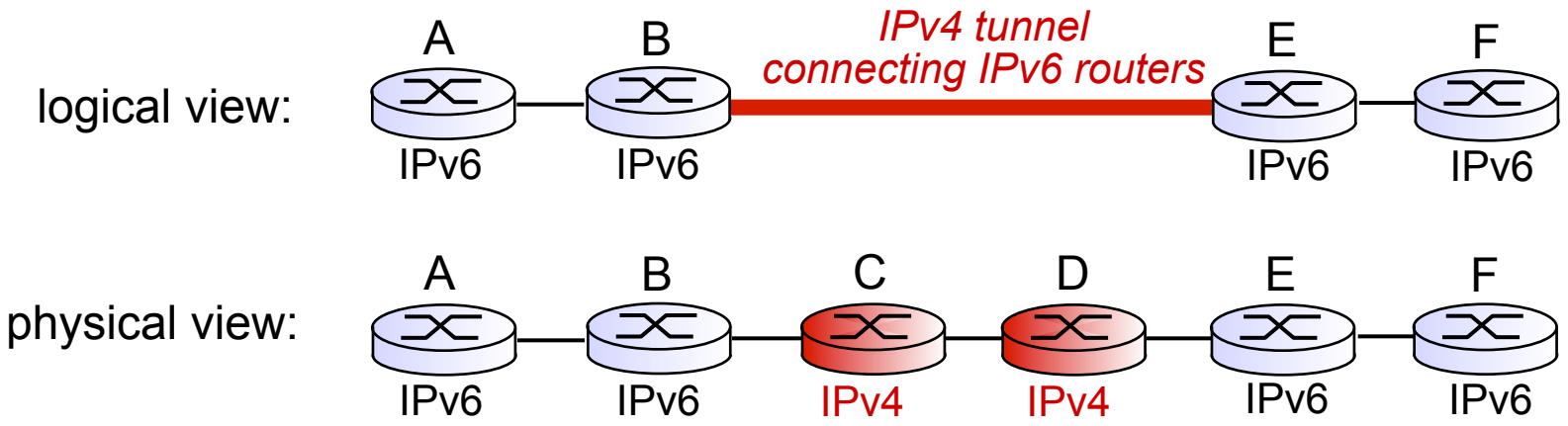


Transition from IPv4 to IPv6

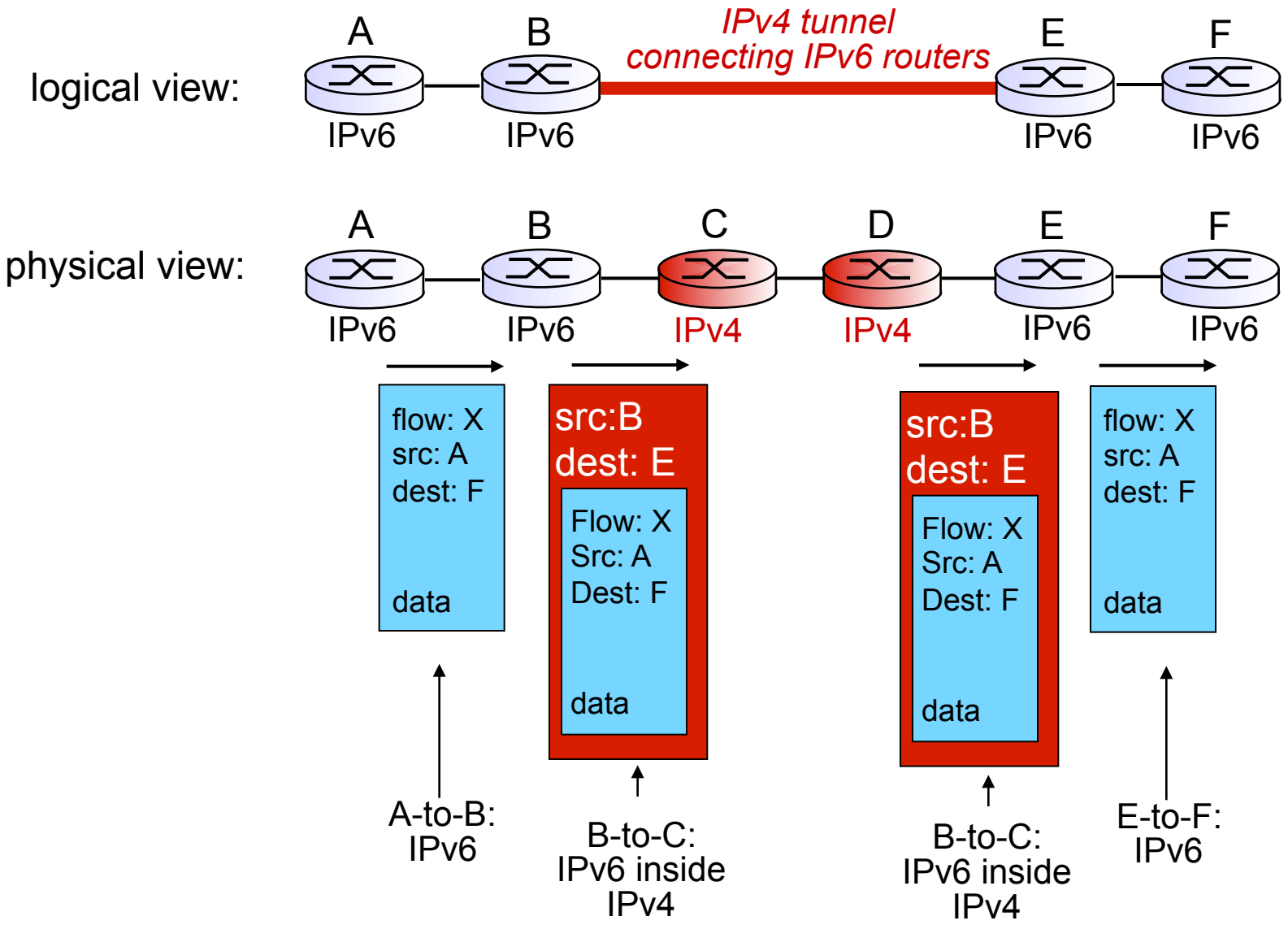
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


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