

CMSC417

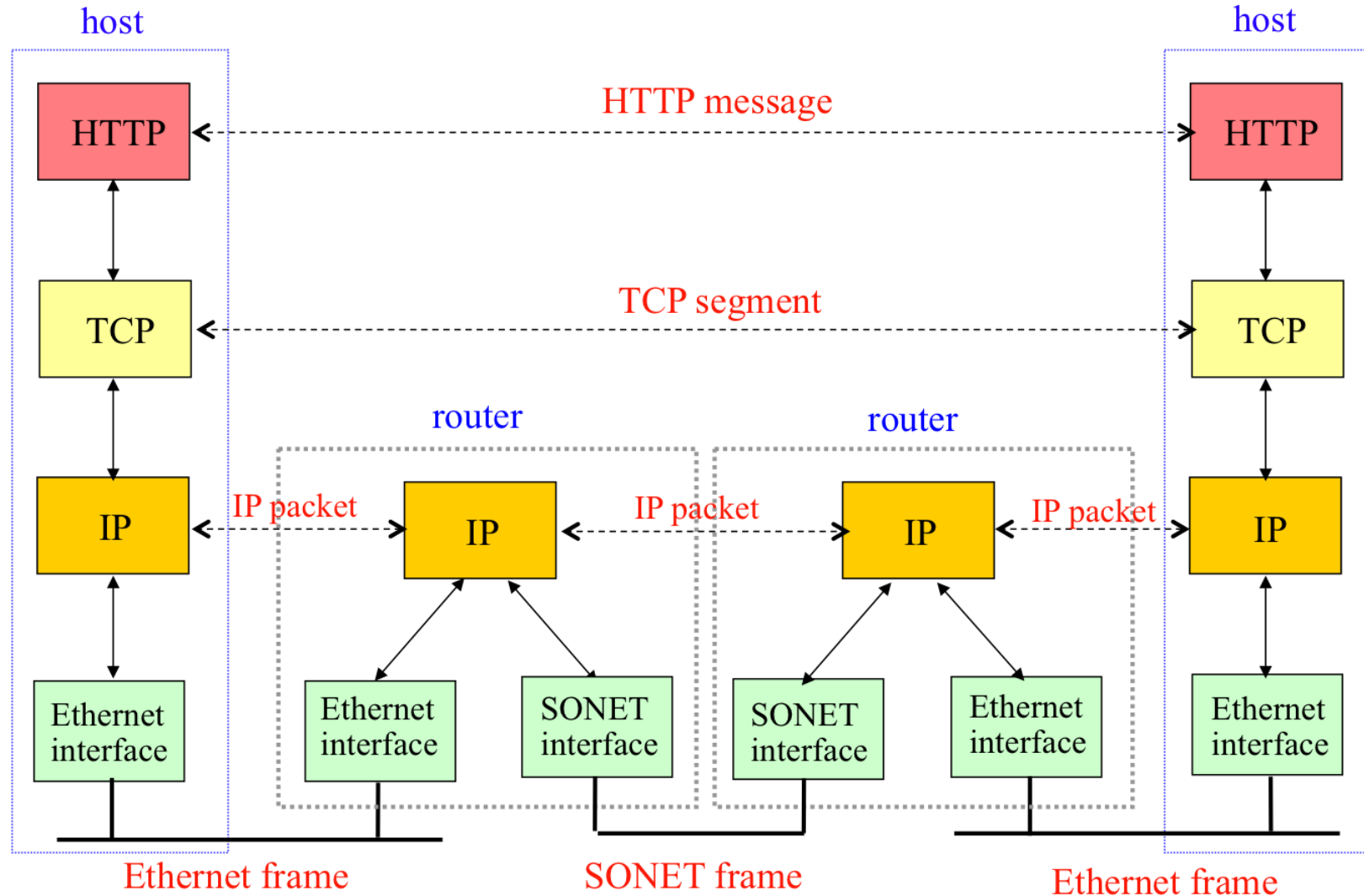
Computer Networks

Prof. Ashok K. Agrawala

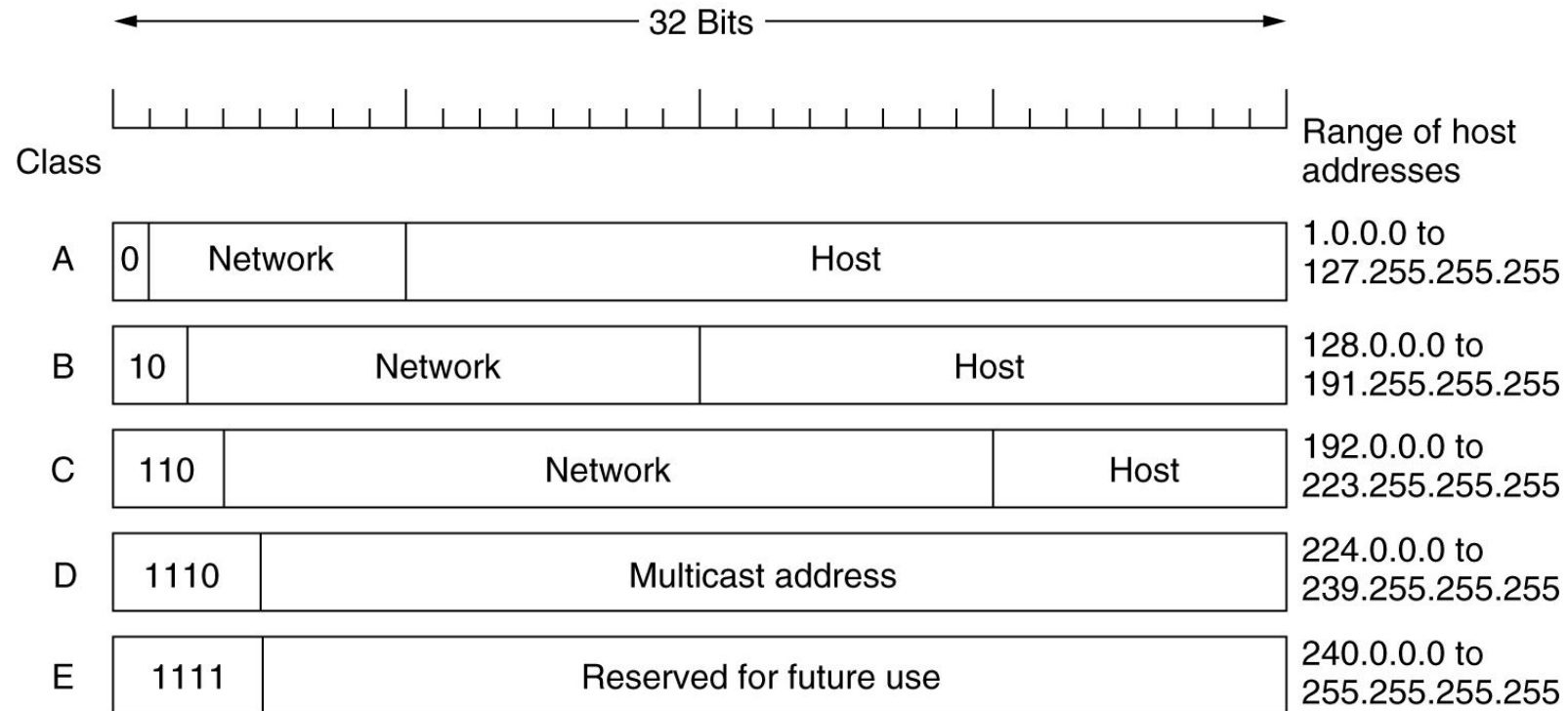
© 2017 Ashok Agrawala

September 18, 2018

Message, Segment, Packet, and Frame



IP Addresses



IP Address and a 24-bit Subnet Mask

Address

12

34

158

5

00001100	00100010	10011110	00000101
----------	----------	----------	----------

11111111	11111111	11111111	00000000
----------	----------	----------	----------

Mask

255

255

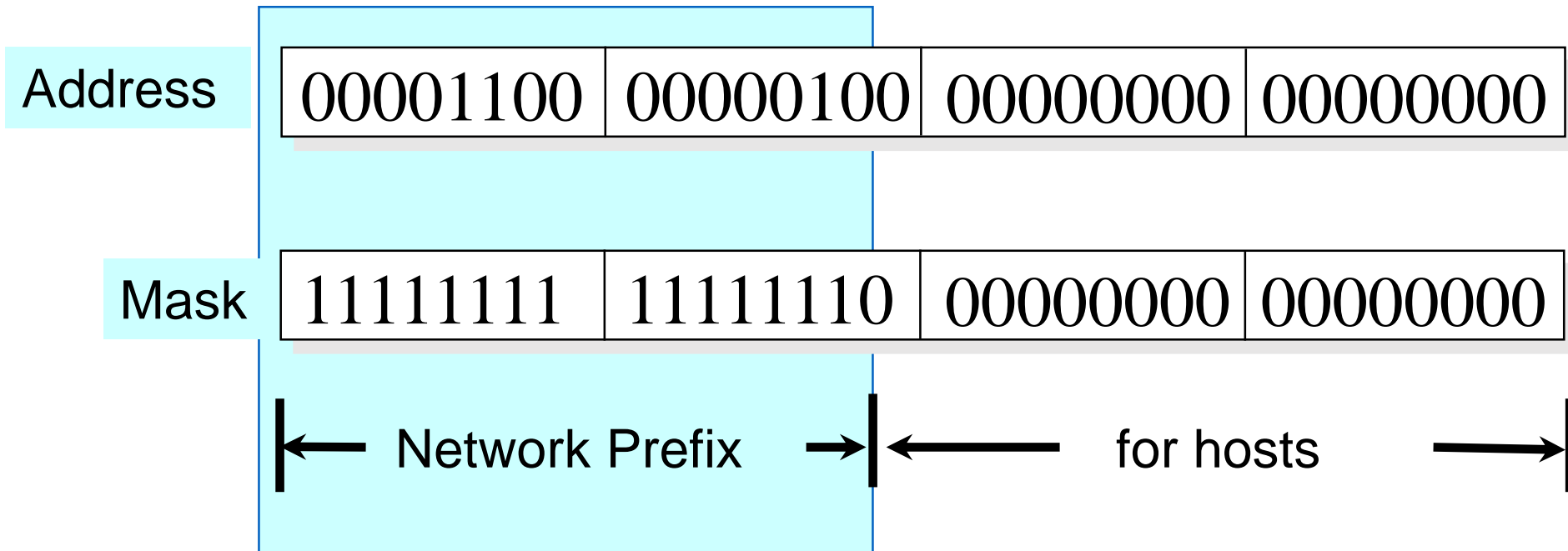
255

0

Classless Inter-Domain Routing (CIDR)

Use two 32-bit numbers to represent a network.
Network number = IP address + Mask

IP Address : 12.4.0.0 IP Mask: 255.254.0.0



Written as 12.4.0.0/15

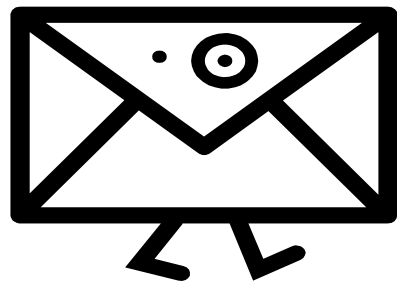
Routing

Routing Algorithms

- The Optimality Principle
- Shortest Path Routing
- Flooding
- Distance Vector Routing
- Link State Routing
- Hierarchical Routing
- Broadcast Routing
- Multicast Routing
- Routing for Mobile Hosts
- Routing in Ad Hoc Networks

What is Routing?

- A famous quotation from RFC 791
“A *name* indicates what we seek.
An *address* indicates where it is.
A *route* indicates how we get there.”
-- Jon Postel

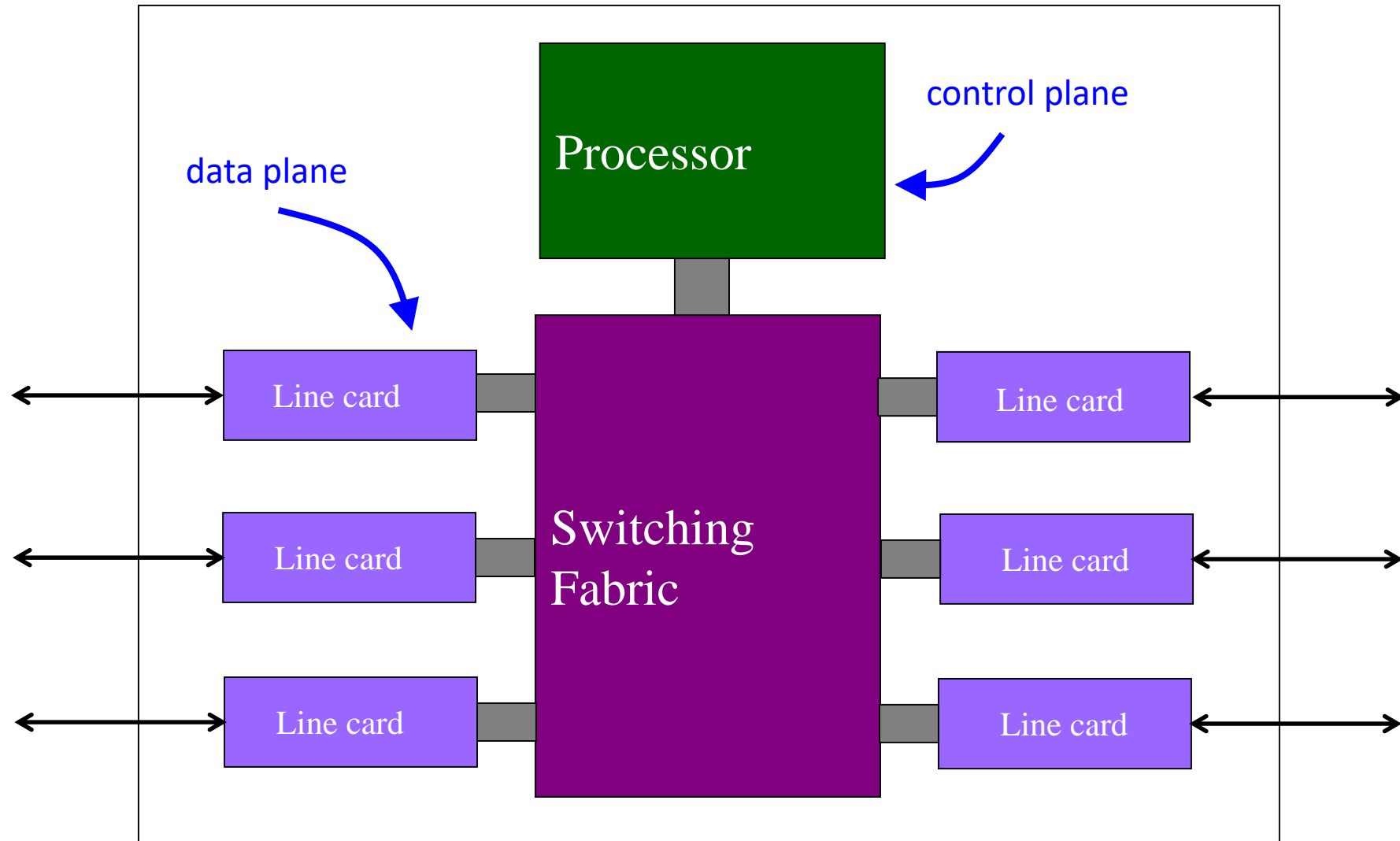


Routing vs. Forwarding

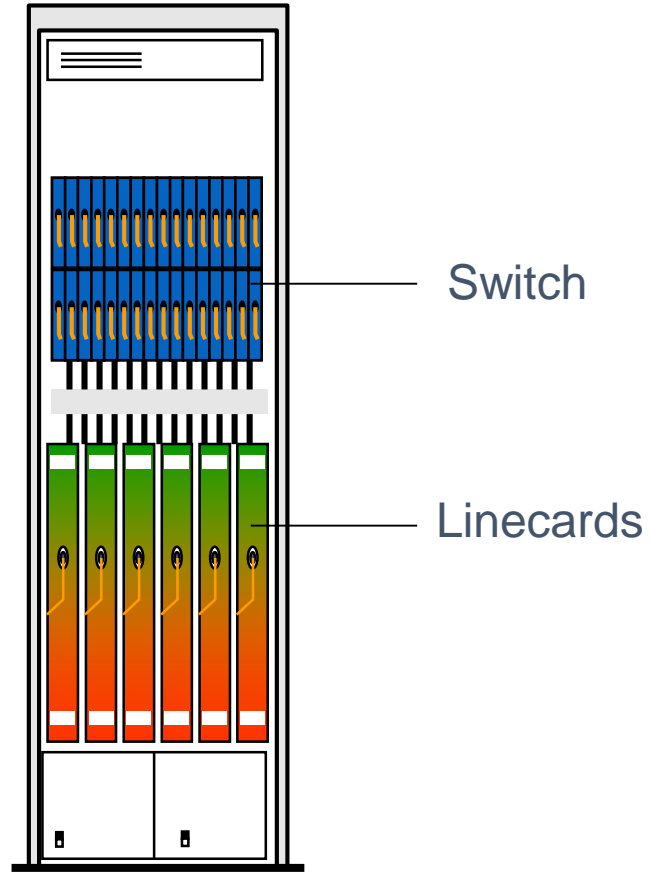
- Routing: control plane
 - Computing paths the packets will follow
 - Routers talking amongst themselves
 - Individual router *creating* a forwarding table
- Forwarding: data plane
 - Directing a data packet to an outgoing link
 - Individual router *using* a forwarding table



Data and Control Planes



Router Physical Layout



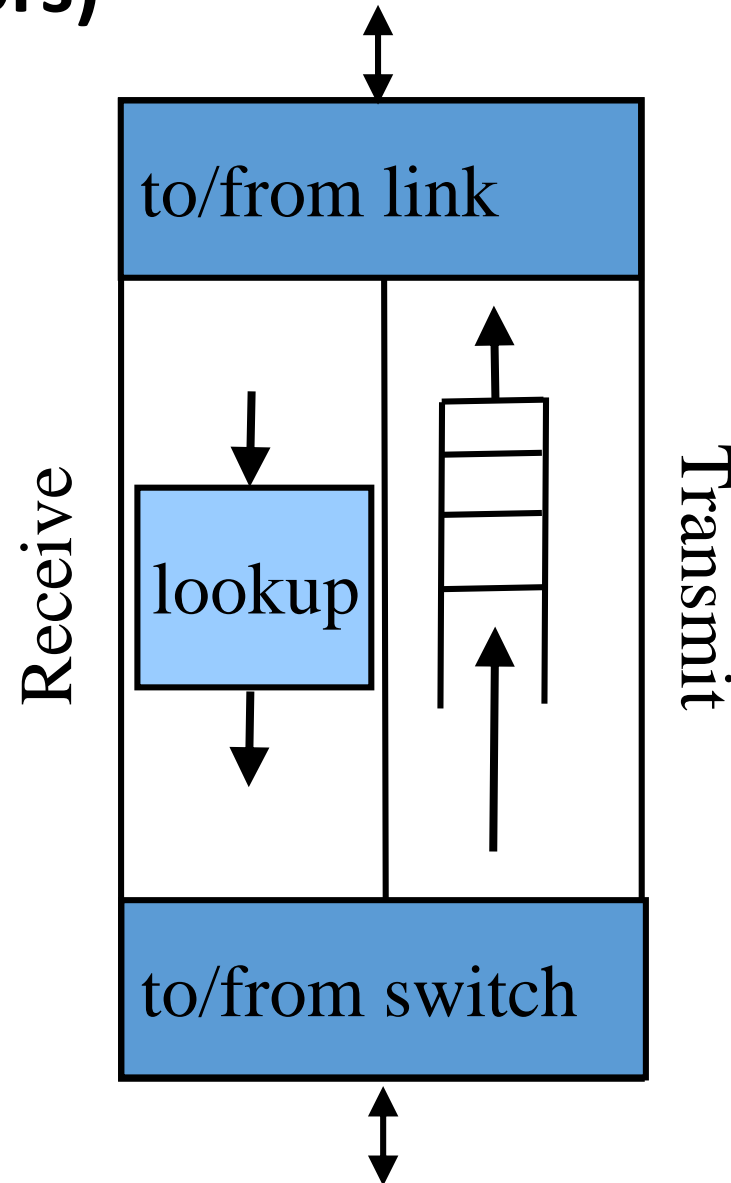
Juniper T series



Cisco 12000

Line Cards (Interface Cards, Adaptors)

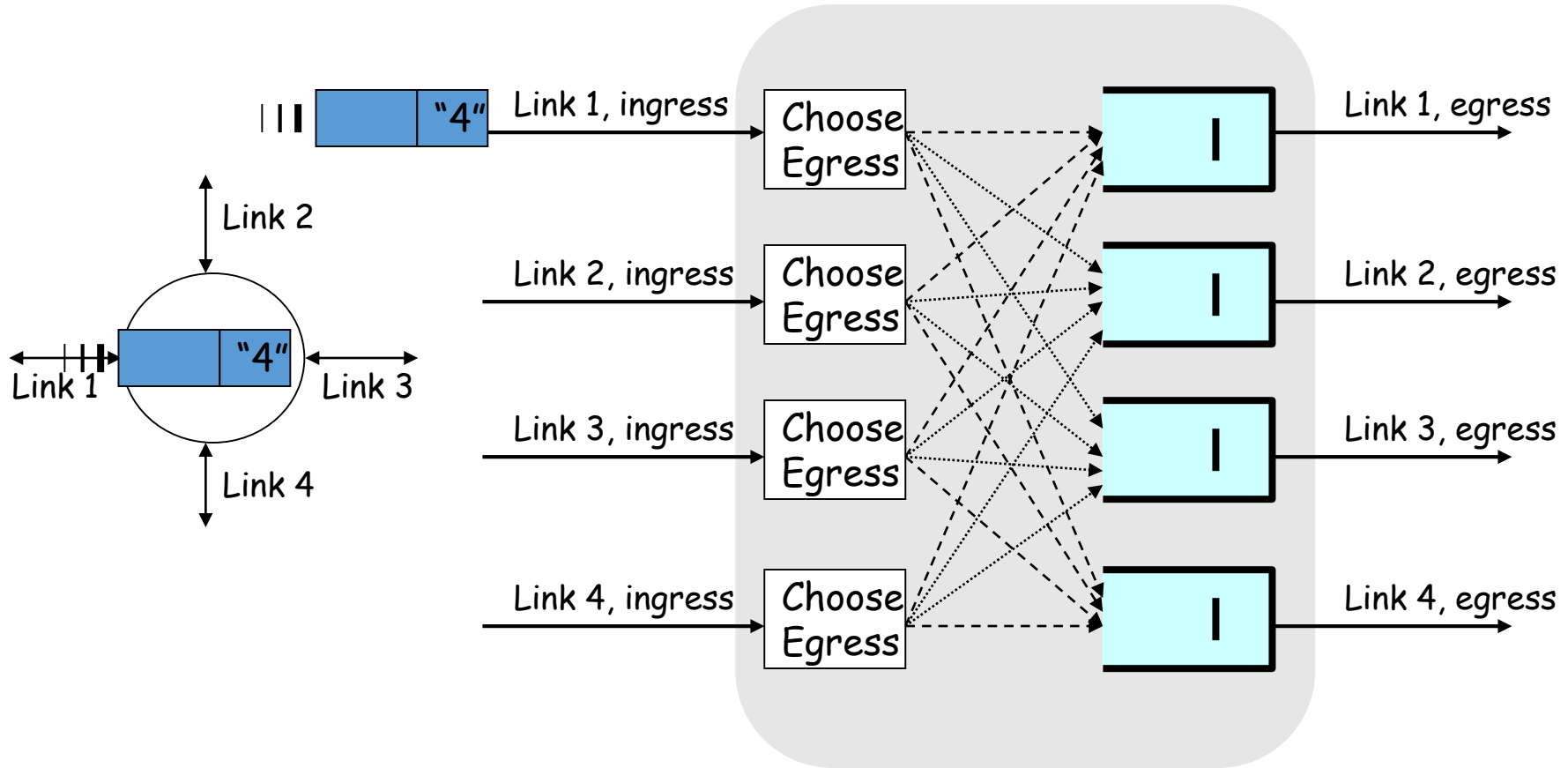
- Interfacing
 - Physical link
 - Switching fabric
- Packet handling
 - Packet forwarding
 - Decrement time-to-live
 - Buffer management
 - Link scheduling
 - Packet filtering
 - Rate limiting
 - Packet marking
 - Measurement



Switching Fabric

- Deliver packet inside the router
 - From incoming interface to outgoing interface
 - A small network in and of itself
- Must operate very quickly
 - Multiple packets going to same outgoing interface
 - Switch scheduling to match inputs to outputs
- Implementation techniques
 - Bus, crossbar, interconnection network, ...
 - Running at a faster speed (e.g., 2X) than links
 - Dividing variable-length packets into fixed-size cells

Packet Switching



Router Processor

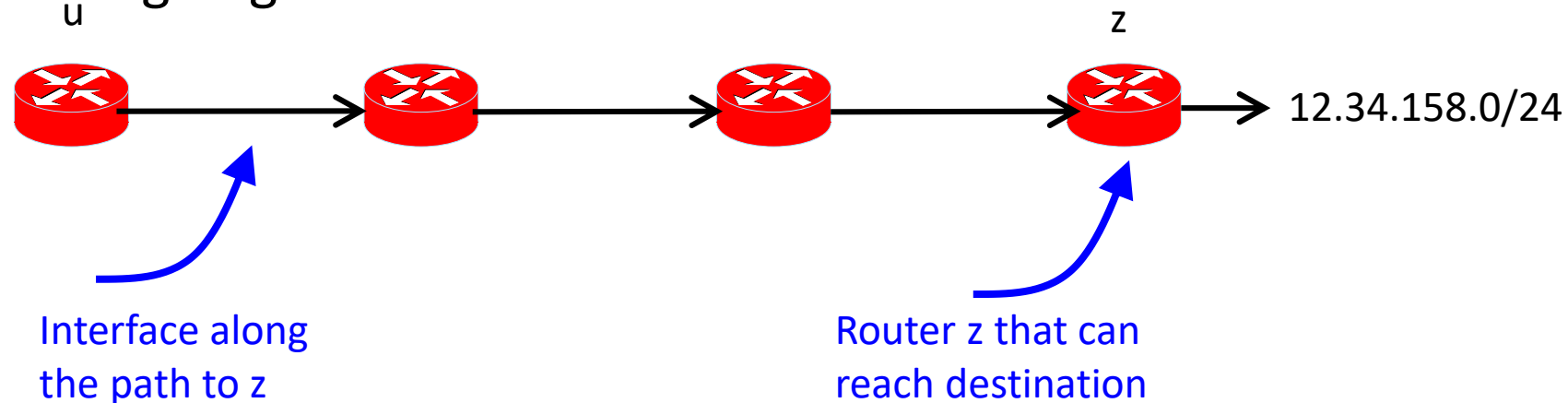
- So-called “Loopback” interface
 - IP address of the CPU on the router
- Interface to network administrators
 - Command-line interface for configuration
 - Transmission of measurement statistics
- Handling of special data packets
 - Packets with IP options enabled
 - Packets with expired Time-To-Live field
- Control-plane software
 - Implementation of the routing protocols
 - Creation of forwarding table for the line cards

Where do Forwarding Tables Come From?

- Routers have forwarding tables that map IP prefix to outgoing links
- Entries can be statically configured (e. g. “map 12.34.158.0/24 to Serial0/0.1”)
- But, this doesn’t adapt
 - To failures
 - To new equipment
 - To the need to balance load
- That is where routing protocols come in

Computing Paths Between Routers

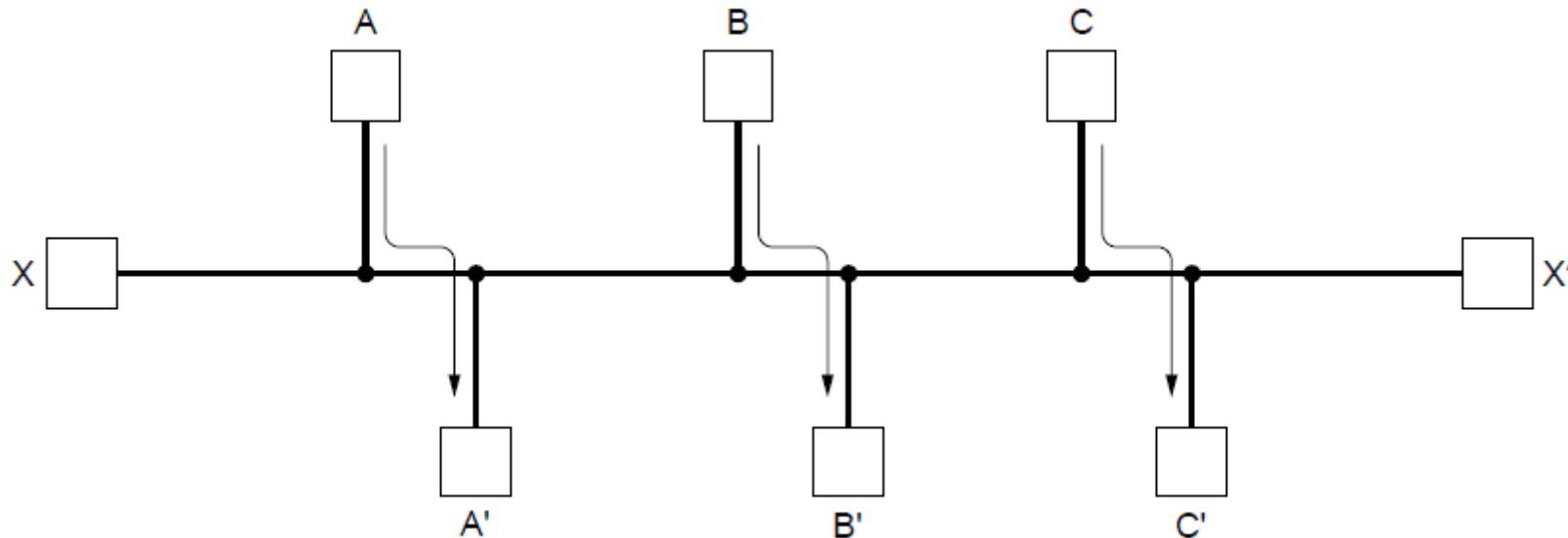
- Routers need to know two things
 - Which router to use to reach a destination prefix
 - Which outgoing interface to use to reach that router



How do you know how to forward packets toward z?

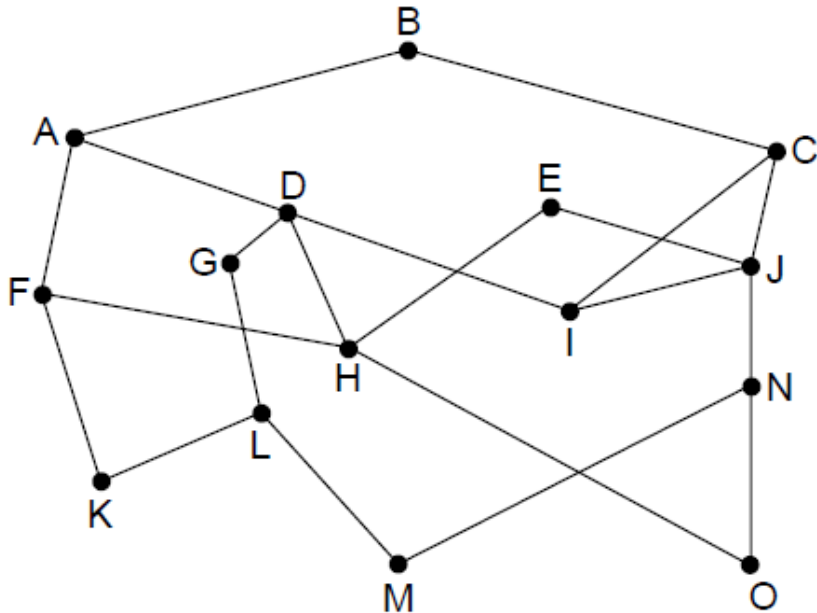
Fairness vs. Efficiency

Network with a conflict between fairness and efficiency.

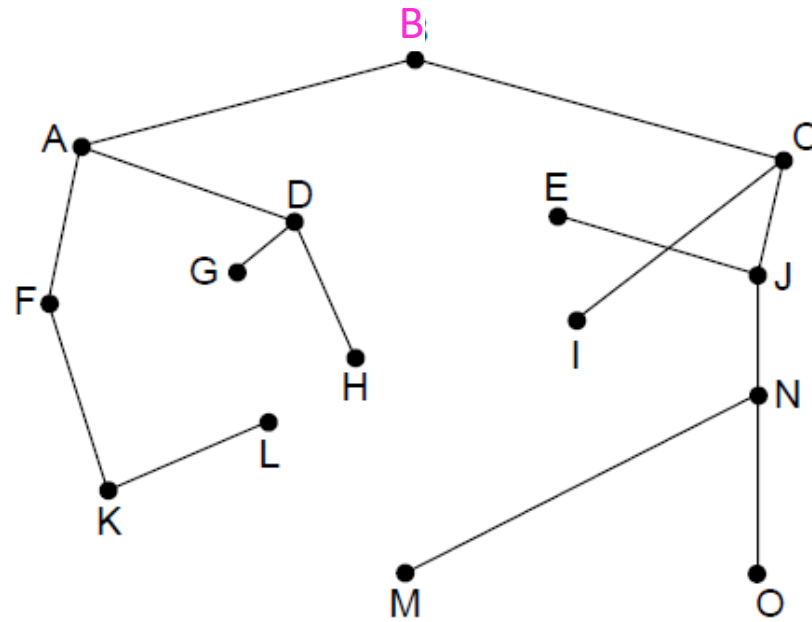


The Optimality Principle

Each portion of a best path is also a best path; the union of them to a router is a tree called the **sink tree**



Network



Sink tree of best paths to router B

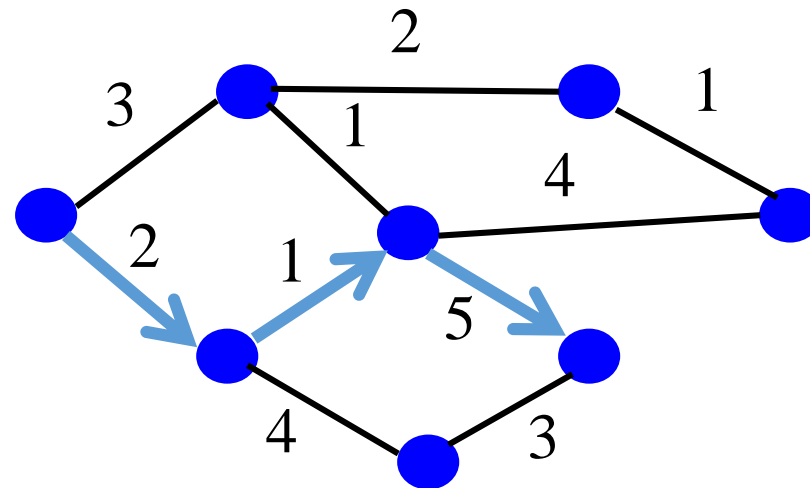
Best means
fewest
hops in the
example

Computing the Shortest Paths

(assuming you already know the topology)

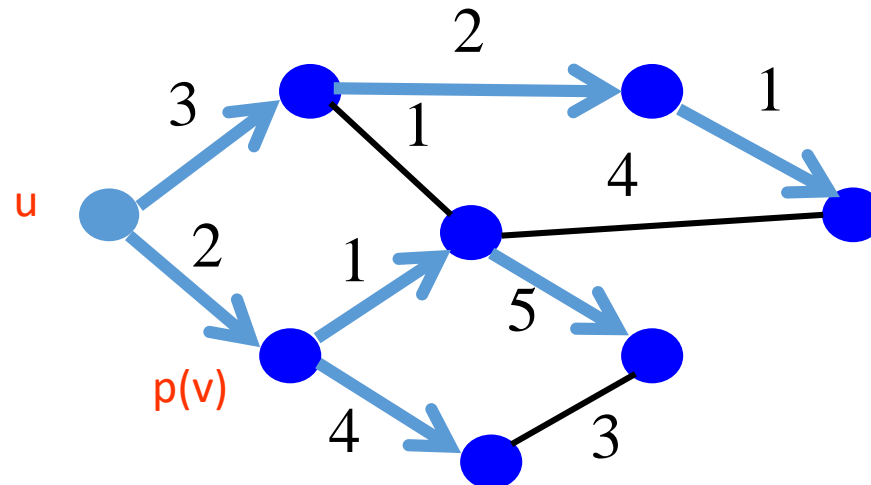
Shortest-Path Routing

- Path-selection model
 - Destination-based
 - Load-insensitive (e.g., static link weights)
 - Minimum hop count or sum of link weights



Shortest-Path Problem

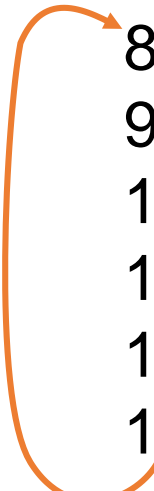
- Given: network topology with link costs
 - $c(x,y)$: link cost from node x to node y
 - Infinity if x and y are not direct neighbors
- Compute: least-cost paths to all nodes
 - From a given source u to all other nodes
 - $p(v)$: predecessor node along path from source to v



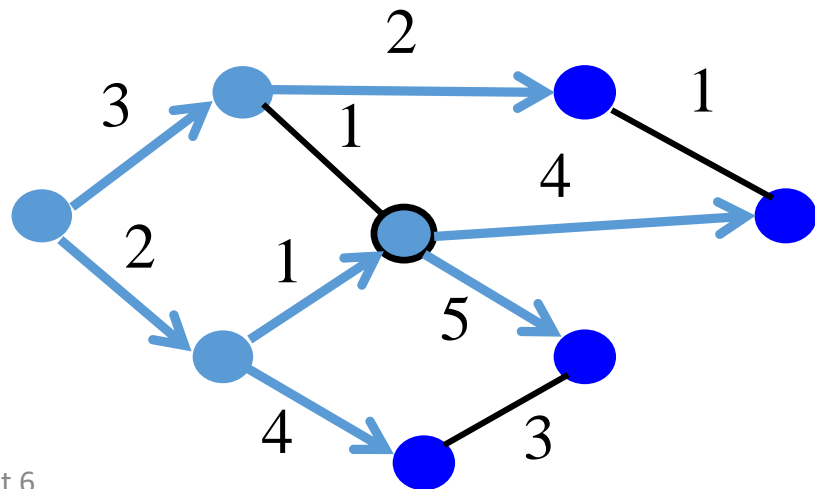
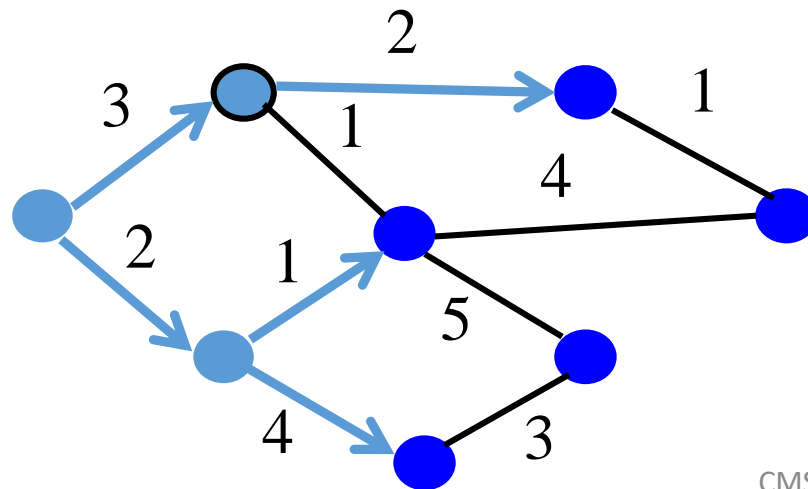
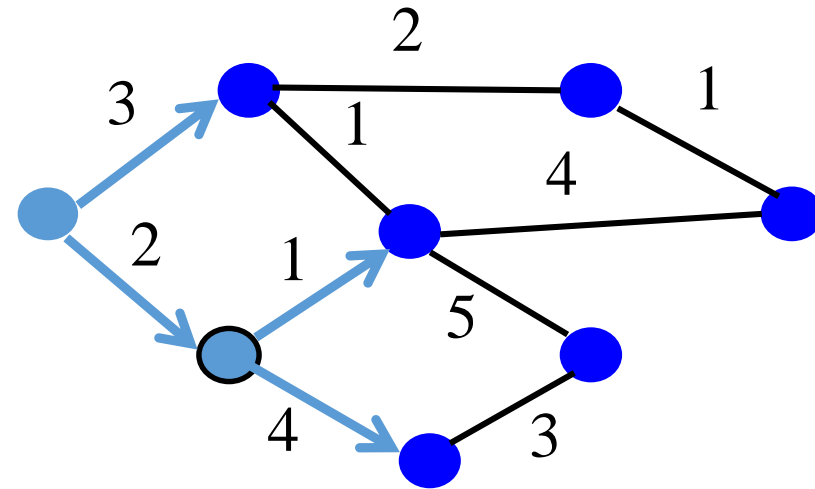
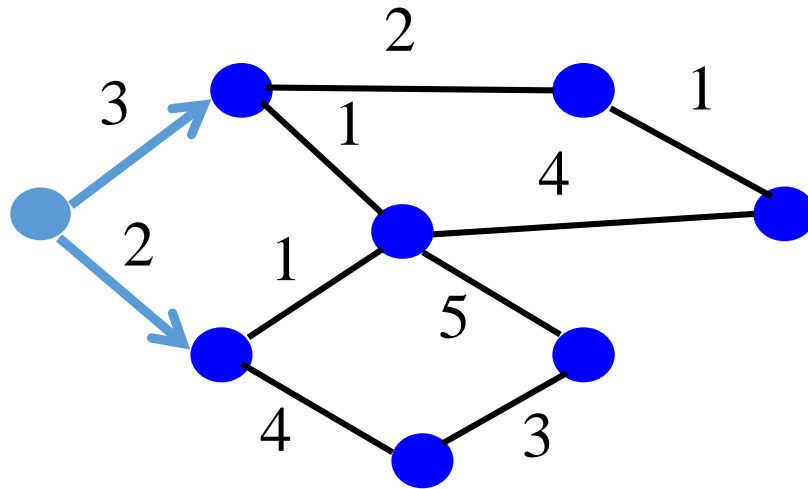
Dijkstra's Shortest-Path Algorithm

- Iterative algorithm: after k iterations, know least-cost path to k nodes
- **S**: nodes whose least-cost path definitively known
 - Initially, $\mathbf{S} = \{\mathbf{u}\}$ where u is the source node
 - Add one node to S in each iteration
- **D(v)**: current cost of path from source to node v
 - Initially, $\mathbf{D(v)} = \mathbf{c(u,v)}$ for all nodes v adjacent to u
 - ... and $\mathbf{D(v)} = \infty$ for all other nodes v
 - Continually update $D(v)$ as shorter paths are learned

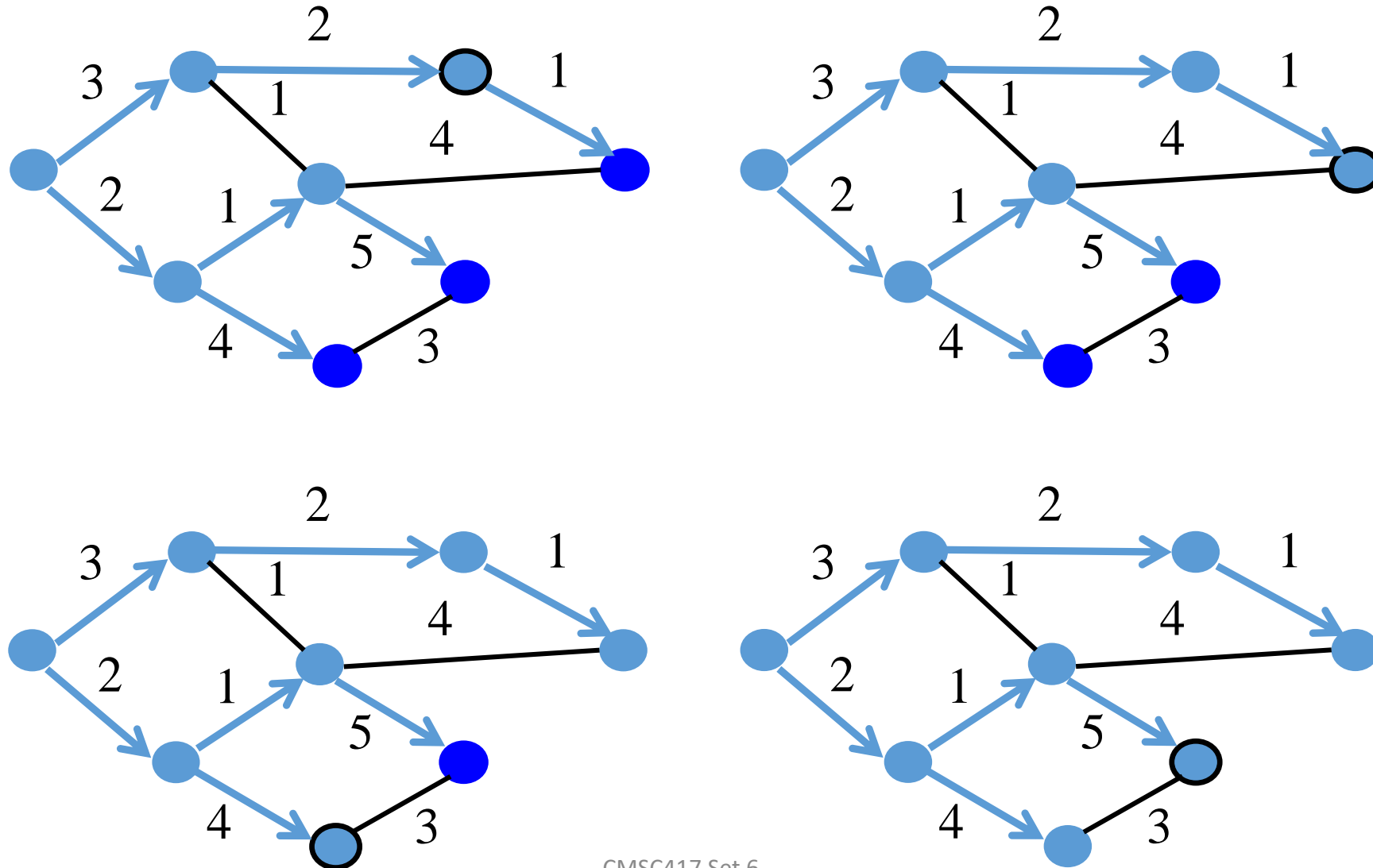
Dijkstra's Algorithm

- 1 *Initialization:*
 - 2 $S = \{u\}$
 - 3 for all nodes v
 - 4 if (v is adjacent to u)
 - 5 $D(v) = c(u,v)$
 - 6 else $D(v) = \infty$
 - 7
 - 8 *Loop*
 - 9 find w not in S with the smallest $D(w)$
 - 10 add w to S
 - 11 update $D(v)$ for all v adjacent to w and not in S :
 - 12 $D(v) = \min\{D(v), D(w) + c(w,v)\}$
 - 13 *until all nodes in S*
- 

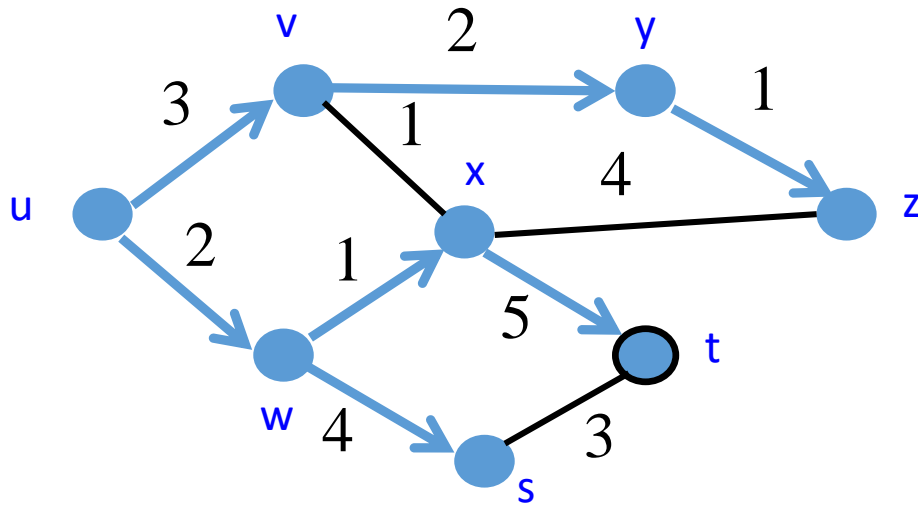
Dijkstra's Algorithm Example



Dijkstra's Algorithm Example



Shortest-Path Tree

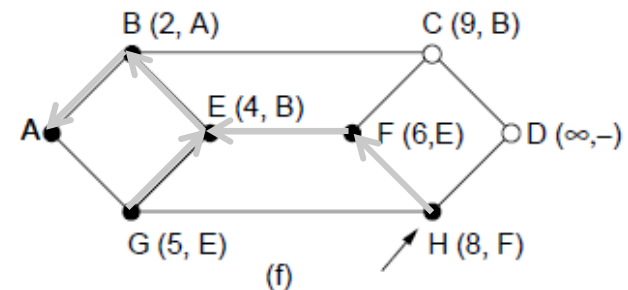
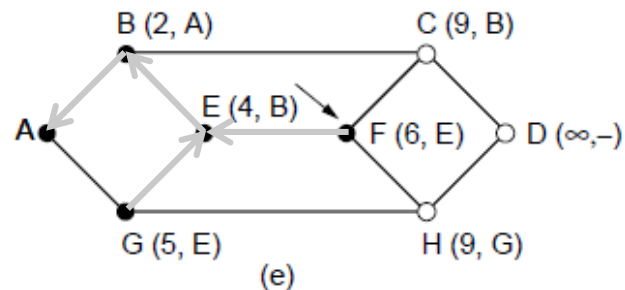
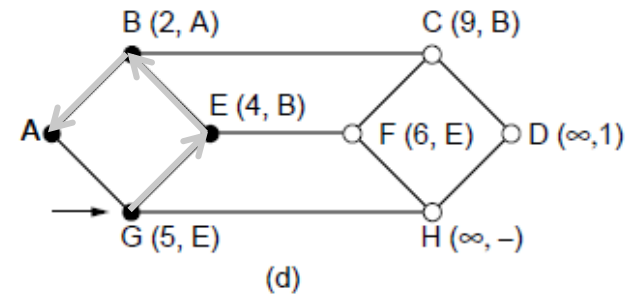
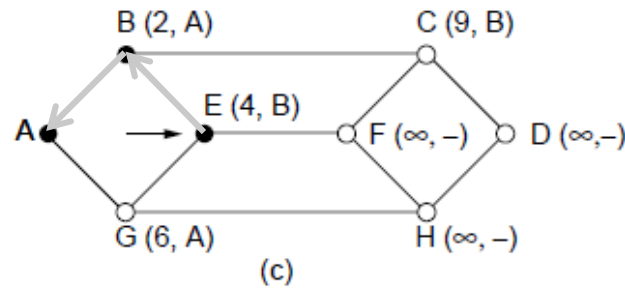
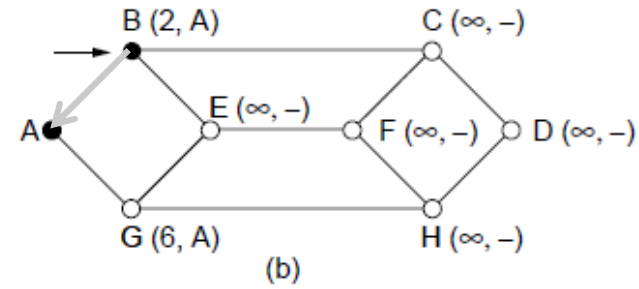
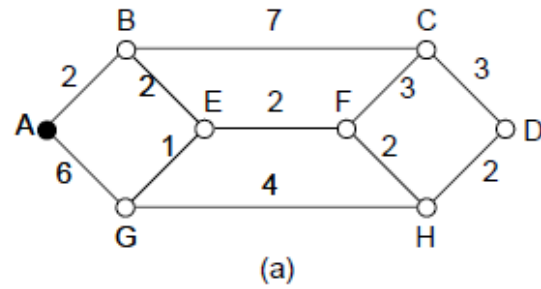


Shortest-path tree from u

Forwarding table at u

	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Shortest Path Algorithm



A network and first five steps in computing the shortest paths from A to D. Pink arrows show the sink tree so far.

Shortest path

```
#define MAX_NODES 1024          /* maximum number of nodes */
#define INFINITY 1000000000    /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                /* the path being worked on */
    int predecessor;           /* previous node */
    int length;                /* length from source to this node */
    enum {permanent, tentative} label; /* label state */
} state[MAX_NODES];

int i, k, min;
struct state *p;

for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
    p->predecessor = -1;
    p->length = INFINITY;
    p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t; /* k is the initial working node */
```

Start with the sink,
all other nodes are
unreachable

Dijkstra's algorithm to compute the shortest path through a graph.

Shortest path (2)

```
do {
    for (i = 0; i < n; i++)
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }
}
```

Relaxation step.
Lower distance to
nodes linked to
newest member of
the sink tree

```
/* Find the tentatively labeled node with the smallest label. */
k = 0; min = INFINITY;
for (i = 0; i < n; i++)
    if (state[i].label == tentative && state[i].length < min) {
        min = state[i].length;
        k = i;
    }
state[k].label = permanent;
} while (k != s);
```

Find the lowest
distance, add it to
the sink tree, and
repeat until done

```
/* Copy the path into the output array. */
i = 0; k = s;
do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
```

ugh a graph.

Learning the Topology

(by the routers talking among themselves)

Link-State Routing

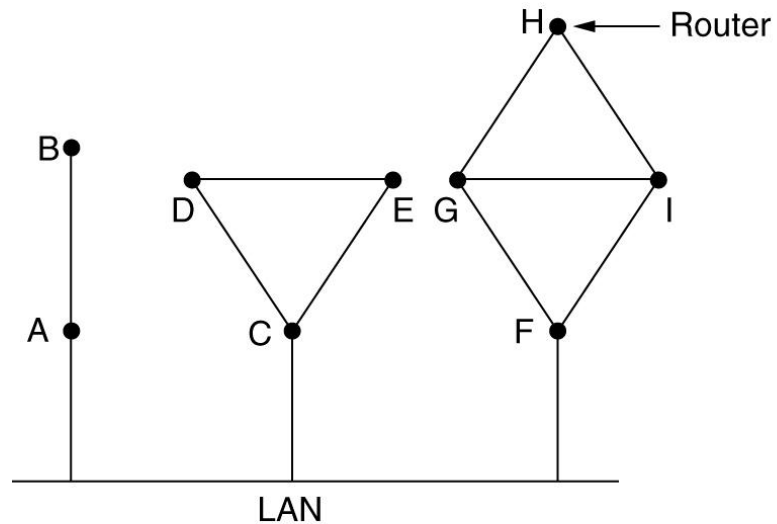
- Each router keeps track of its incident links
 - Whether the link is up or down
 - The cost on the link
- Each router broadcasts the link state to give every router a complete view of the graph
- Each router runs Dijkstra's algorithm to compute the shortest paths and construct the forwarding table
 - Open Shortest Path First (OSPF)
 - Intermediate System – Intermediate System (IS-IS)

Link State Routing

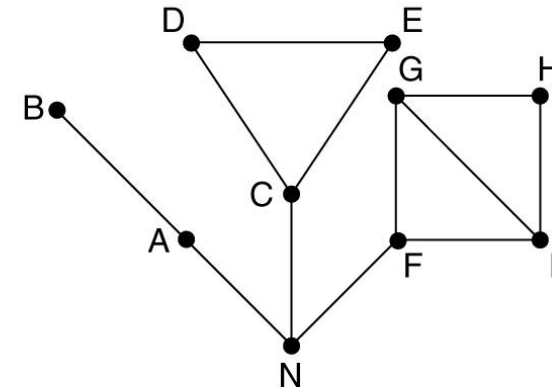
Each router must do the following:

1. Discover its neighbors, learn their network address.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

Learning about the Neighbors



(a)

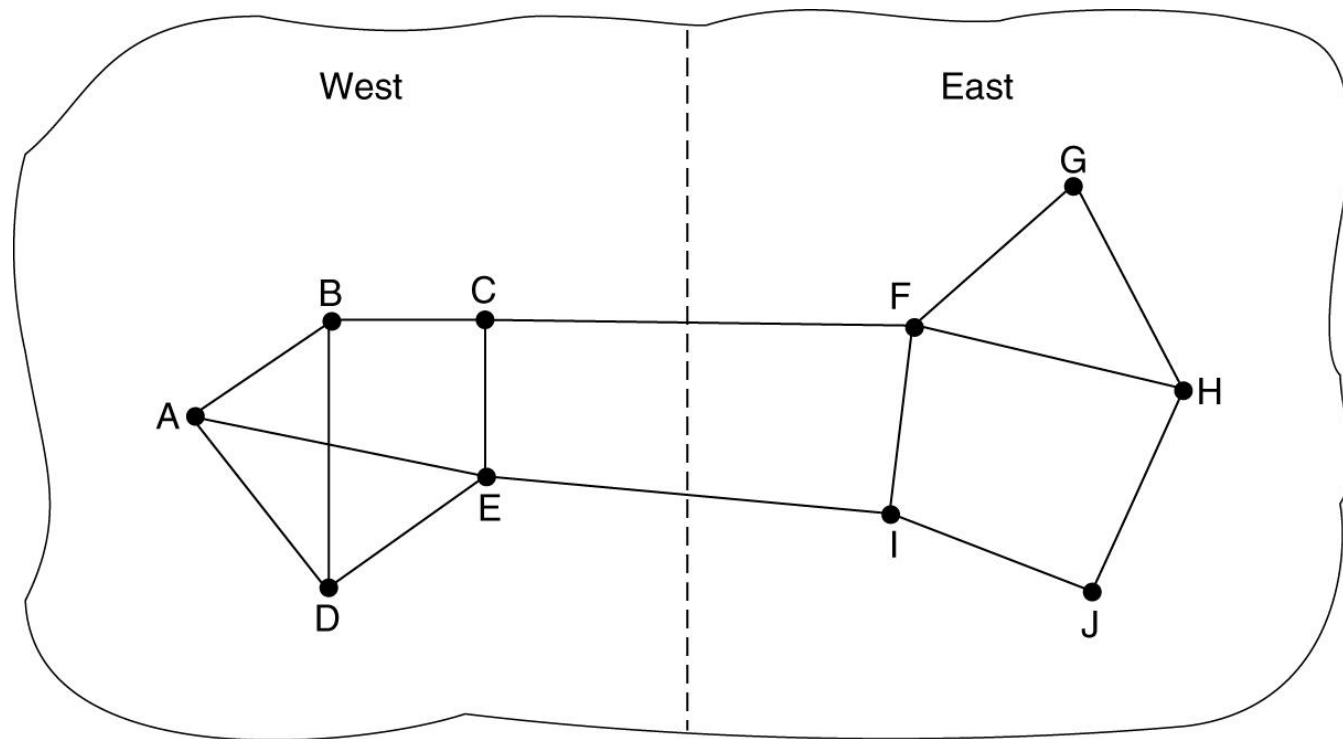


(b)

(a) Nine routers and a LAN. (b) A graph model of (a).

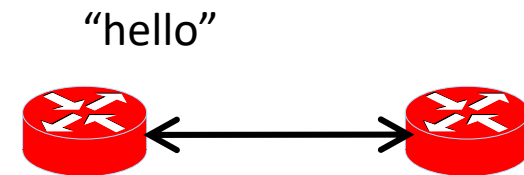
Measuring Line Cost

A subnet in which the East and West parts are connected by two lines.



Detecting Topology Changes

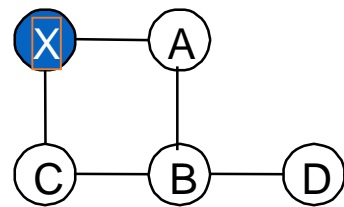
- Beaconing
 - Periodic “hello” messages in both directions
 - Detect a failure after a few missed “hellos”
- Performance trade-offs
 - Detection speed
 - Overhead on link bandwidth and CPU
 - Likelihood of false detection



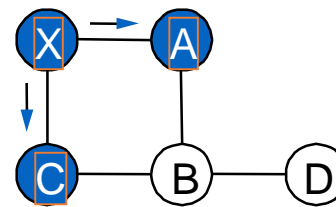
Broadcasting the Link State

Flooding

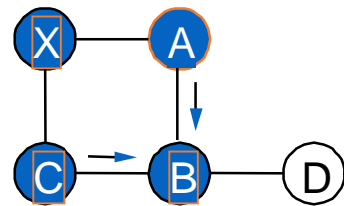
1. Node sends link-state information out its links
2. The next node sends out all of its links,
 - except the one where the information arrived



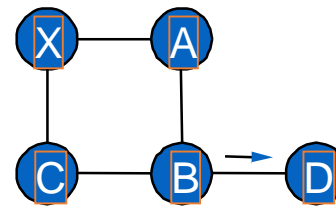
(a)



(b)



(c)

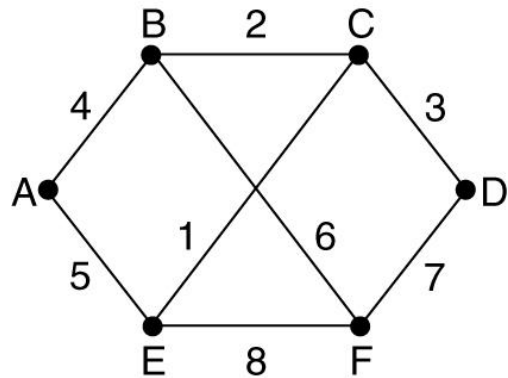


(d)

Broadcasting the Link State

- Reliable flooding: Ensure all nodes receive link-state information, and that they use the latest version
- Challenges
 - Packet loss
 - Out-of-order arrival
- Solutions
 - Acknowledgments and retransmissions
 - Sequence numbers
 - Time-to-live for each packet

Building Link State Packets



(a)

		Link		State		Packets					
A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

(b)

(a) A subnet. (b) The link state packets for this subnet.

Distributing the Link State Packets

The packet buffer for router B in the previous slide

Source	Seq.	Age	Send flags			ACK flags			Data
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

When to Initiate Flooding

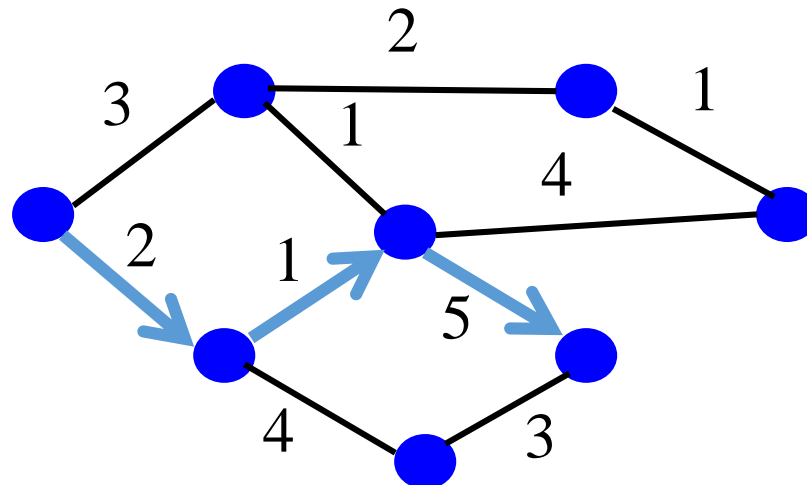
- Topology change
 - Link or node failure
 - Link or node recovery
- Configuration change, link cost change
- Periodically
 - Refresh the link-state information
 - Typically (say) 30 minutes
 - Corrects for possible corruption of the data

When the Routers Disagree

(during transient periods)

Convergence

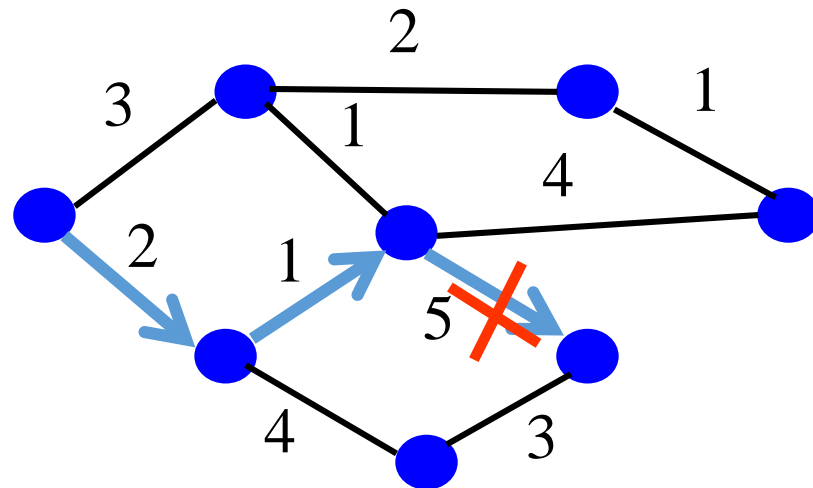
- Getting consistent routing information to all nodes (e. g. all nodes having the same link-state database)
- Consistent forwarding after convergence
 - All nodes have the same link-state database
 - All nodes forward packets on shortest paths
 - The next router on the path forwards to the next hop



Transient Disruptions

Detection delay

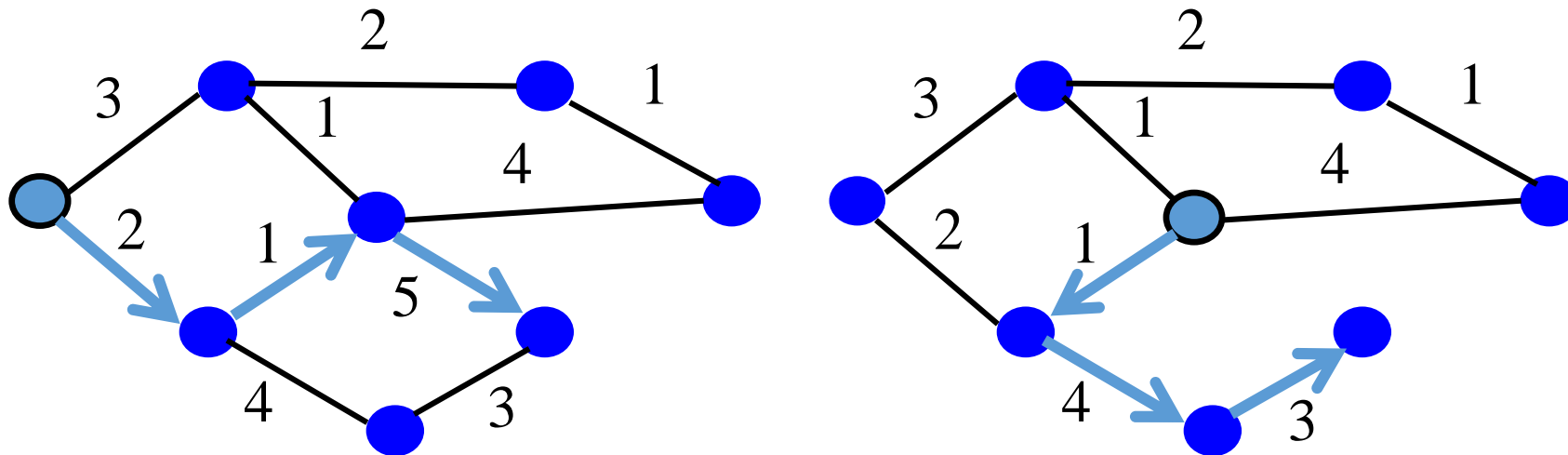
- A node does not detect a failed link immediately
- ... and forwards data packets into a “blackhole”
- Depends on timeout for detecting lost hellos



Transient Disruptions

Inconsistent link-state database

- Some routers know about failure before others
- The shortest paths are no longer consistent
- Can cause transient forwarding loops



Convergence Delay

- Sources of convergence delay
 - Detection latency
 - Flooding of link-state information
 - Shortest-path computation
 - Creating the forwarding table
- Performance during convergence period
 - Lost packets due to blackholes and TTL expiry
 - Looping packets consuming resources
 - Out-of-order packets reaching the destination
- Very bad for VoIP, online gaming, and video

Reducing Convergence Delay

- **Faster detection**
 - Smaller hello timers
 - Link-layer technologies that can detect failures
- **Faster flooding**
 - Flooding immediately
 - Sending link-state packets with high-priority
- **Faster computation**
 - Faster processors on the routers
 - Incremental Dijkstra's algorithm
- **Faster forwarding-table update via data structures that support incremental updates**

Scaling Link-State Routing

- Overhead of link-state routing
 - Flooding link-state packets throughout the network
 - Running Dijkstra's shortest-path algorithm
- Introducing hierarchy through “areas”

