Introduction to LAN/WAN

Network Layer

Topics

- ☞ Introduction (5 - 5.1) ←
- ☞ Routing (5.2) (*The core)*
- ☞ Internetworking (5.5)
- ☞ Congestion Control (5.3)

Network Layer Design Isues

- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets

Store-and-Forward Packet Switching

- Note: shaded circle indicates carrier’s equipment

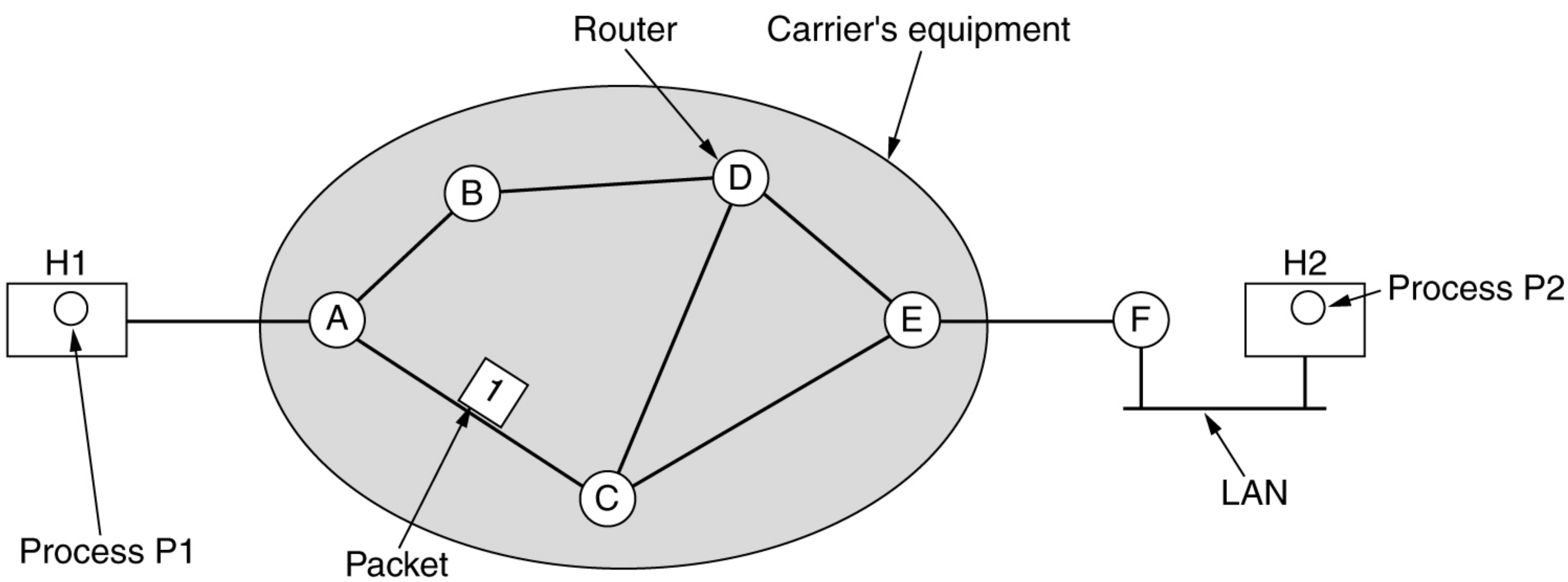


fig 5-1

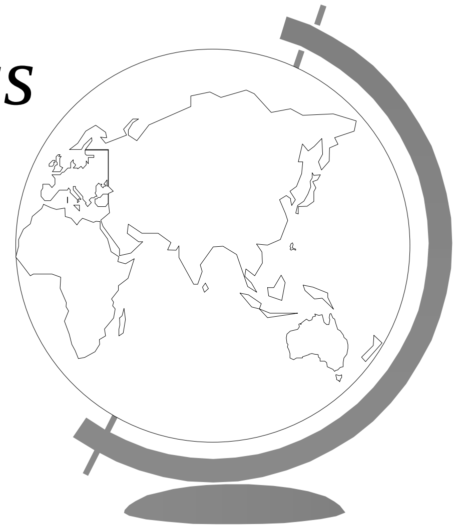
The environment of the network layer protocols.

Store-and-Forward Packet Switching

- ☞ Service to transport layer
- ☞ Getting packets from source to destination
  - – may require many hops
  - – data link layer from one end of wire to another
- ☞ Must know topology of subnet
- ☞ Avoid overloading routes
- ☞ Deal with different networks

Services to Transport Layer

- ☞ Depend upon services to Transport Layer
- ☞ Goals
  - – services independent of subnet technology
  - – shield transport layer from topology
  - – uniform number of network addresses, across LANs or WANS
- ☞ Lots of freedom, but two factions
  - – *connection-oriented* and *connectionless*



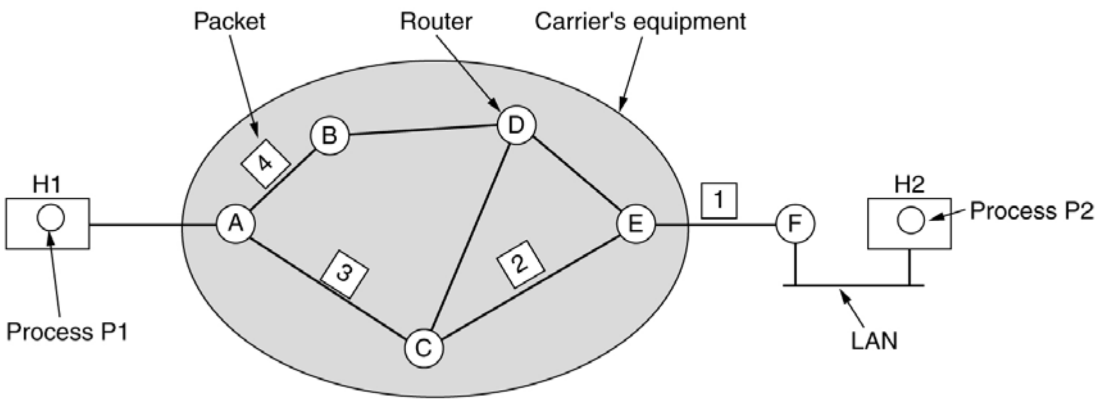
Connectionless

- Internet camp
  - 30 years of experience with real networks
  - subnet is unreliable, no matter how well designed
  - hosts should accept this and do error control and flow control
  - SEND_PACKET and RECV_PACKET
  - each packet full information on source, dest
  - no ordering or flow control since will be redundant with transport layer

Connection-Oriented

- ☞ Telephone company camp
  - – 100 years of international experience
  - – set up connection between end hosts
  - – negotiate about parameters, quality and cost
  - – communicate in both directions
  - – all packets delivered in sequence
    - ◆ some might still be lost
  - – flow control to help slow senders

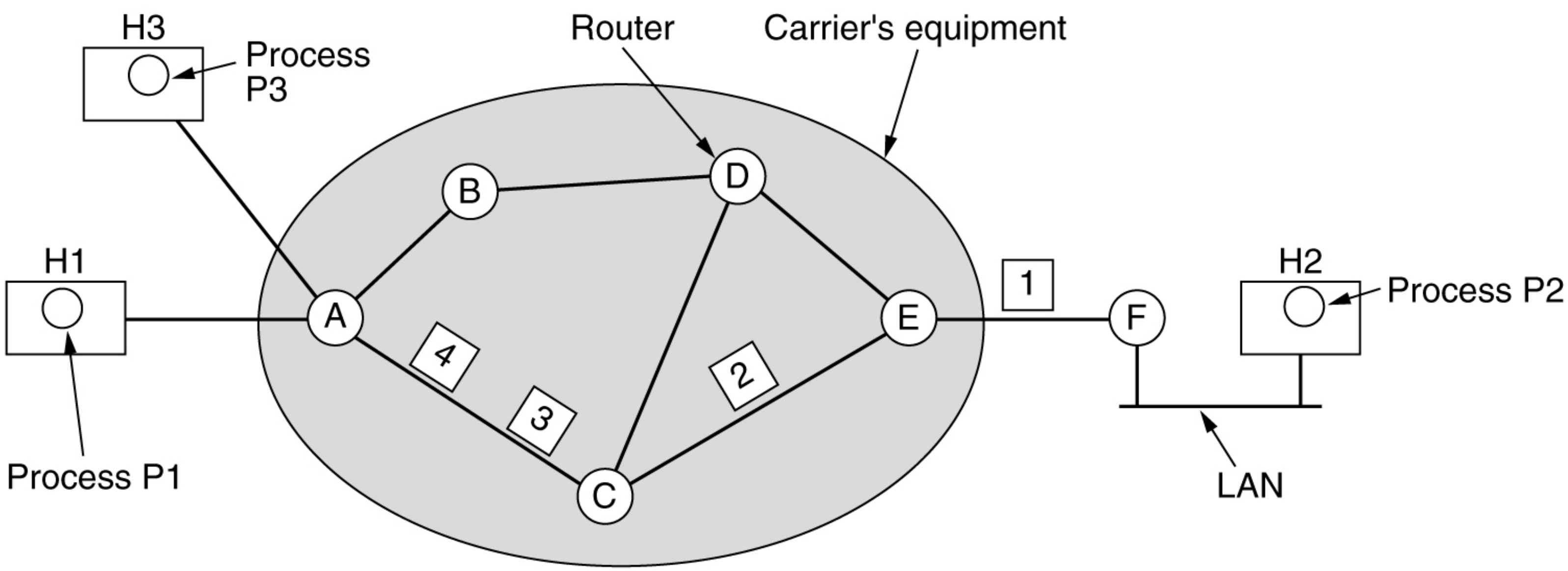
Implementation of Connectionless Service



A's table initially: Dest.	Line	A's table later: Dest.	Line	C's table: Dest.	Line	E's table: Dest.	Line
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	D	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F

Routing within a diagram subnet.

Implementation of Connection-Oriented Service



A's table

In		Out	
H1	1	C	1
H3	1	C	2

C's table

In		Out	
A	1	E	1
A	2	E	2

E's table

In		Out	
C	1	F	1
C	2	F	2

Routing within a virtual-circuit subnet.

Internal Organization

- *Virtual Circuit*
  - do not choose new route per packet
  - establish route and re-use
  - terminate route when terminate connection
- *Datagrams*
  - no advance routes
  - each packet routed independently
  - more work but more robust

Connected Vs Connectionless

- Really, where to put the complexity
  - transport layer (connectionless)
    - computers cheap
    - don't clutter network layer since relied upon for years
    - some applications don't want all those services
  - subnet (connected)
    - most users don't want complex protocols on their machines
      - embedded systems don't
    - real-time services much better on connected
- (Un) Connected, (Un) Reliable
  - 4 classes, but two are the most popular

Summary Comparison

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Topics

- ☞ Introduction (5 - 5.1)
- ☞ Routing (5.2) ←
- ☞ Internetworking (5.5)
- ☞ Congestion Control (5.3)

Routing Algorithms

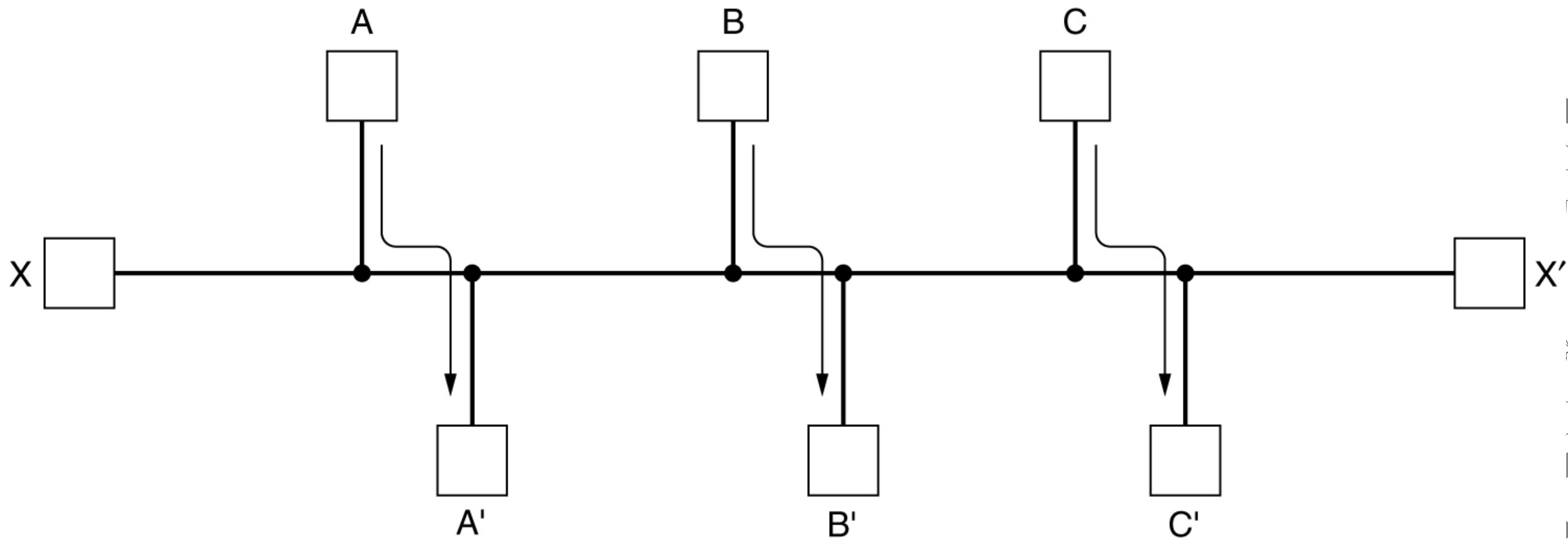
☞ *correctness* and *simplicity* (obviously)

☞ *robustness*

- parts can fail, but system should not
- topology can change

☞ *stability*

☞ *fairness* and *optimality* conflict!



Optimality vs. Fairness

- What to optimize?
  - Minimize delay
  - Maximize network throughput
  - But basic queuing theory says if system near capacity then long delays!
- Compromise: minimize hops (common metric)
  - Improves delay
  - Reduces bandwidth, so usually increases throughput

Two Classes of Routing Algorithms

- *Non-Adaptive algorithms*
  - decisions not based on measurements
  - routes computed offline in advance
  - also called *Static Routing*
- *Adaptive algorithms*
  - change routes based on topology and traffic
  - info: locally, adjacent routers, all routers
  - freq: every ΔT seconds, load change, topology change
- Metric?
  - distance, number of hops, transit time

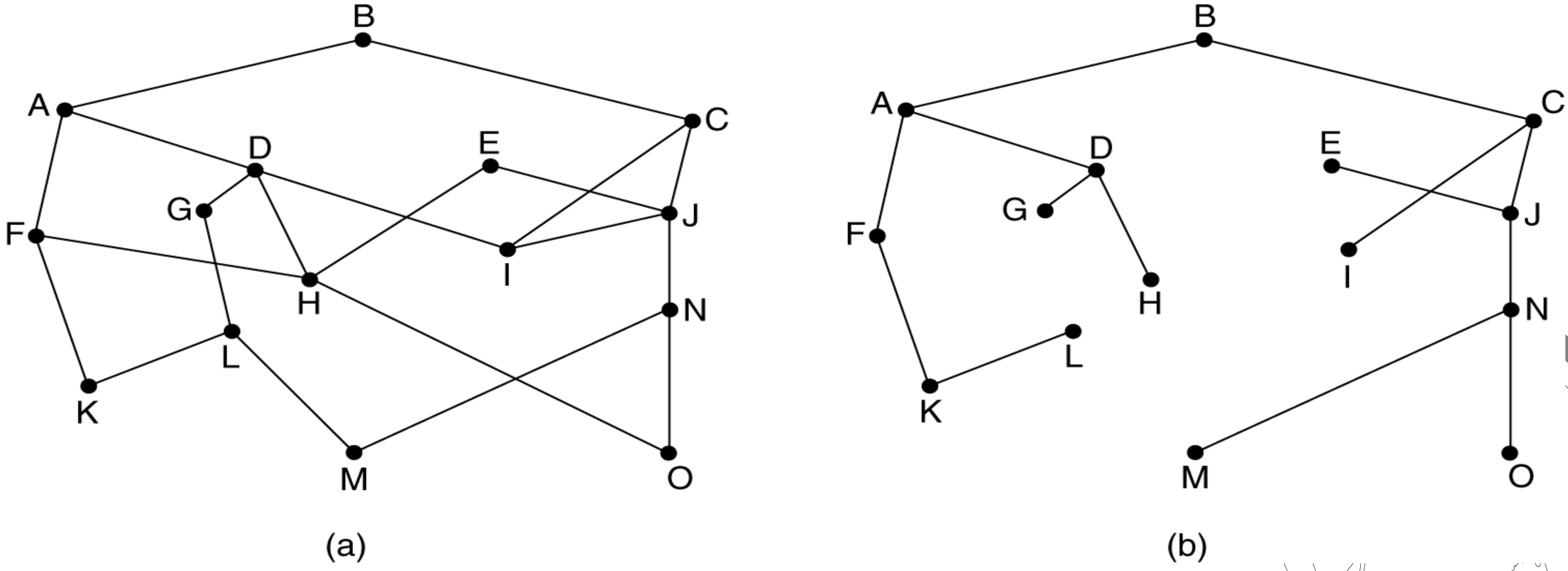
Optimality Principal

"If *J* is on optimal path from *I* to *K*, then optimal path from *J* to *K* is also on that path"

- Explanation by contradiction:
  - Call *I* to *J*, r_1 and *J* to *K*, r_2
  - Assume *J* to *K* has a route better than r_2, say r_3
  - Then $r_1 r_3$ is shorter than $r_1 r_2$
    - contradiction!
- Useful when analyzing specific algorithms

Sink Tree

- Set of optimal routes to a given destination
- Not necessarily unique
- Routing algorithms want sink trees



(a) A subnet. (b) A sink tree for router B.

Sink Trees

- No loops
  - each packet delivered in finite time
  - well, routers go up and down and have different notions of sink trees
- How is sink tree information collected?
  - we'll talk about this later
- Next up: *static routing algorithms*
- On deck: *adaptive algorithms*

Static Routing - Start Simple

- *Shortest path routing*
- How do we measure shortest?
- Number of hops
- Geographic distance
- Mean queuing and transmission delay (hourly tests)
- Combination of above

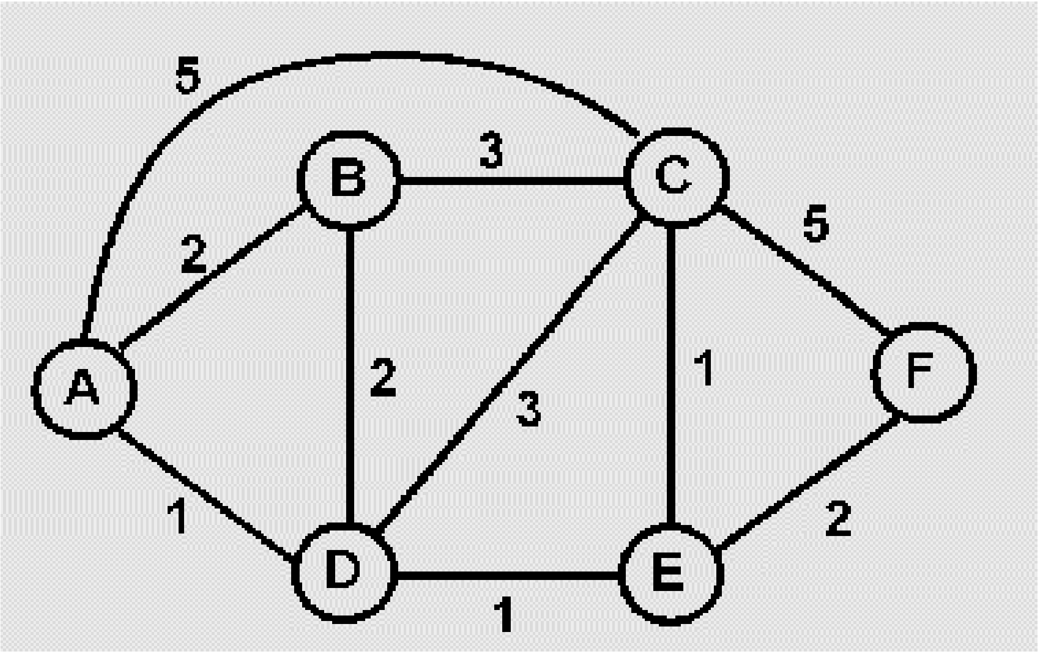
Computing the Shortest Path

- Dijkstra's Algorithm (1959)
- Label each node with distance from source
  – if unknown, then ∞
- As algorithm proceeds, labels change
  – tentative at first
  – permanent when "added" to tree

Dijkstra Example

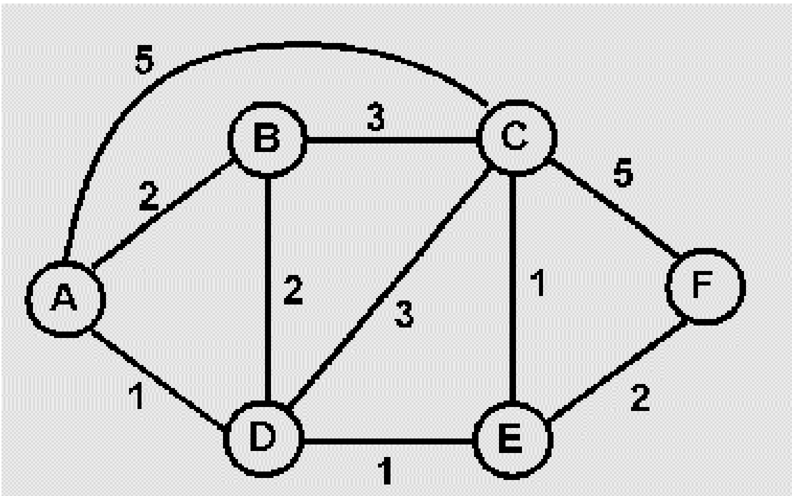
- Make A the source below
- Compute shortest paths from A, build table

N	B	C	D	E	F



Dijkstra Example Solution

N	B	C	D	E	F
A	2,A	5,A	1,A	∞	∞
AD	2,A	4,D		2,D	∞
ADE	2,A	3,E			4,E
ADEB		3,E			4,E
ADEBC					4,E
ADEBCF					

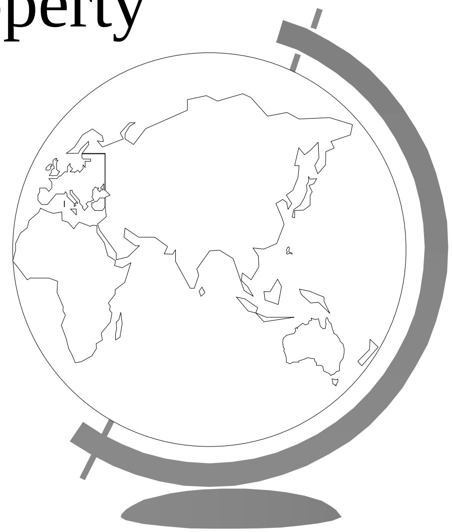


Flooding

- Send every incoming packet on every outgoing link
  - problems?
- Vast numbers of duplicate packets
  - infinite, actually, unless we stop. How?
- Hop count: decrease each hop
- Sequence number:
  - Track packets seen from any other host
  - don’t flood duplicates twice
- *Selective flooding*: send only in about the right direction

Uses of Flooding

- Military applications
  - redundancy is nice
  - routers can be blown to bits
- Distributed databases
  - multiple sources
  - update all at once
- Wireless
  - Channel is broadcast in nature, so use this property
- Baseline for comparing other algorithms
  - flooding always chooses shortest path
  - compare other algorithm to flooding



Topics

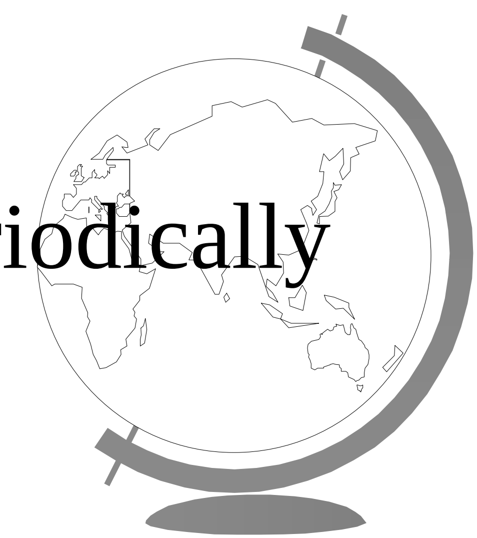
- ☞ Introduction
- ☞ Routing (5.2)
  - static
  - adaptive ←
- ☞ The Internet (5.5, brief)
- ☞ Congestion Control (5.3)

Modern Routing

- Most of today's computer networks use dynamic routing
- Distance vector routing
  - Original Internet routing algorithm
- Link state routing
  - Modern Internet routing algorithm

Distance Vector Routing

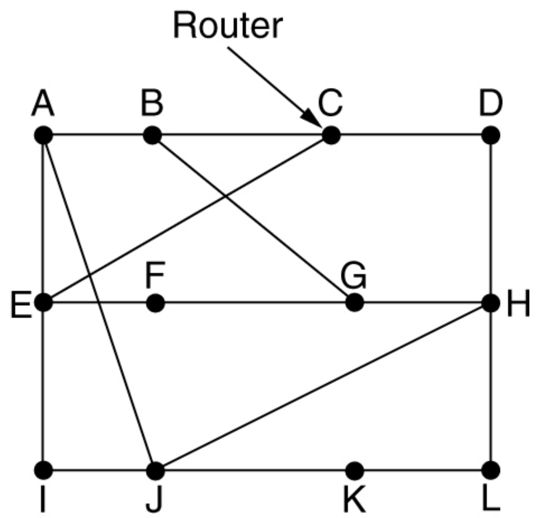
- Router table entries per destination:
  - preferred outgoing line
  - estimate of “distance” to get there
- Assume knows “distance” to each neighbor
  - if hops, just 1 hop
  - if queue length, measure the queues
  - if delay, can send PING packet
- Exchange tables with neighbors periodically



Distance Vector Routing Computation

- ☞ Just got *Routing Table* from *X*
  - – X_i is estimate of time from *X* to *i*
- ☞ Delay to *X* is *m* msec
- ☞ Know distance to X (say, from ECHO's)
  - – Can reach router *i* via *X* in $X_i + m$ msec
- ☞ Do for all neighbors
- ☞ Closest to *i* as "preferred outgoing line"
- ☞ Can then make new routing table

Distance Vector Example



(a)

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

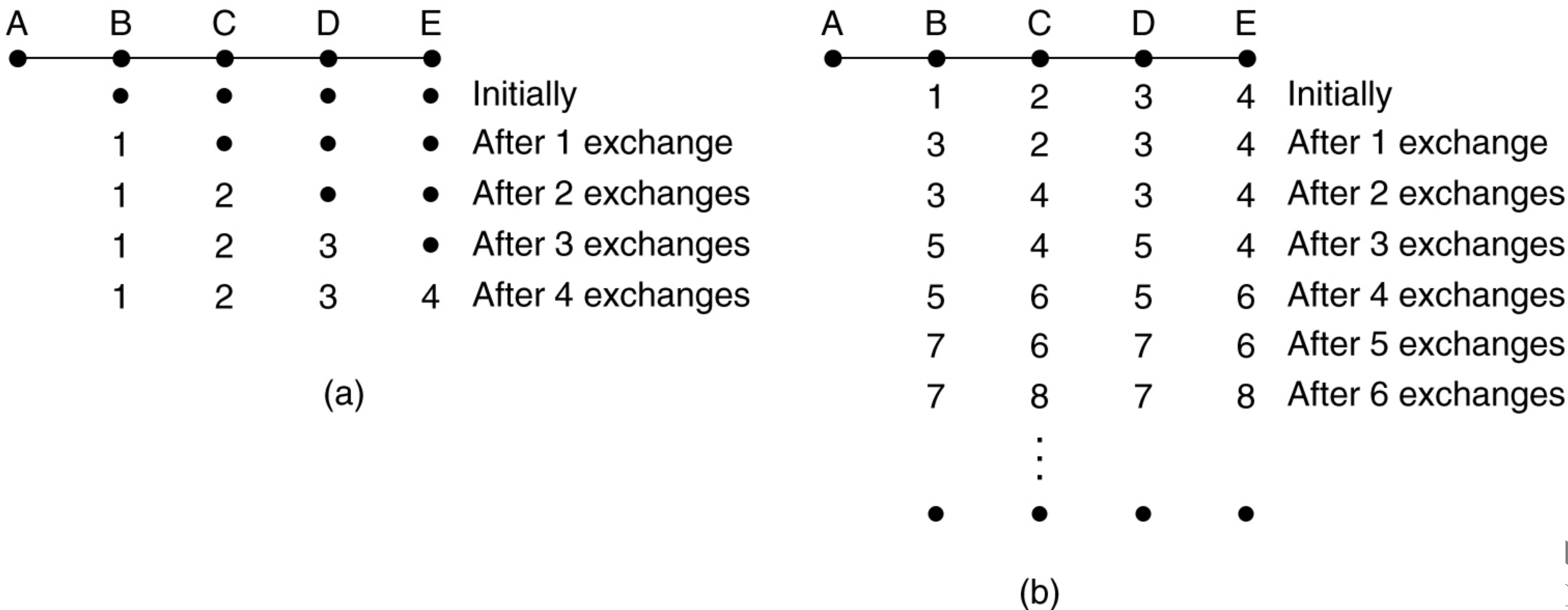
Vectors received from J's four neighbors

(b)

(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

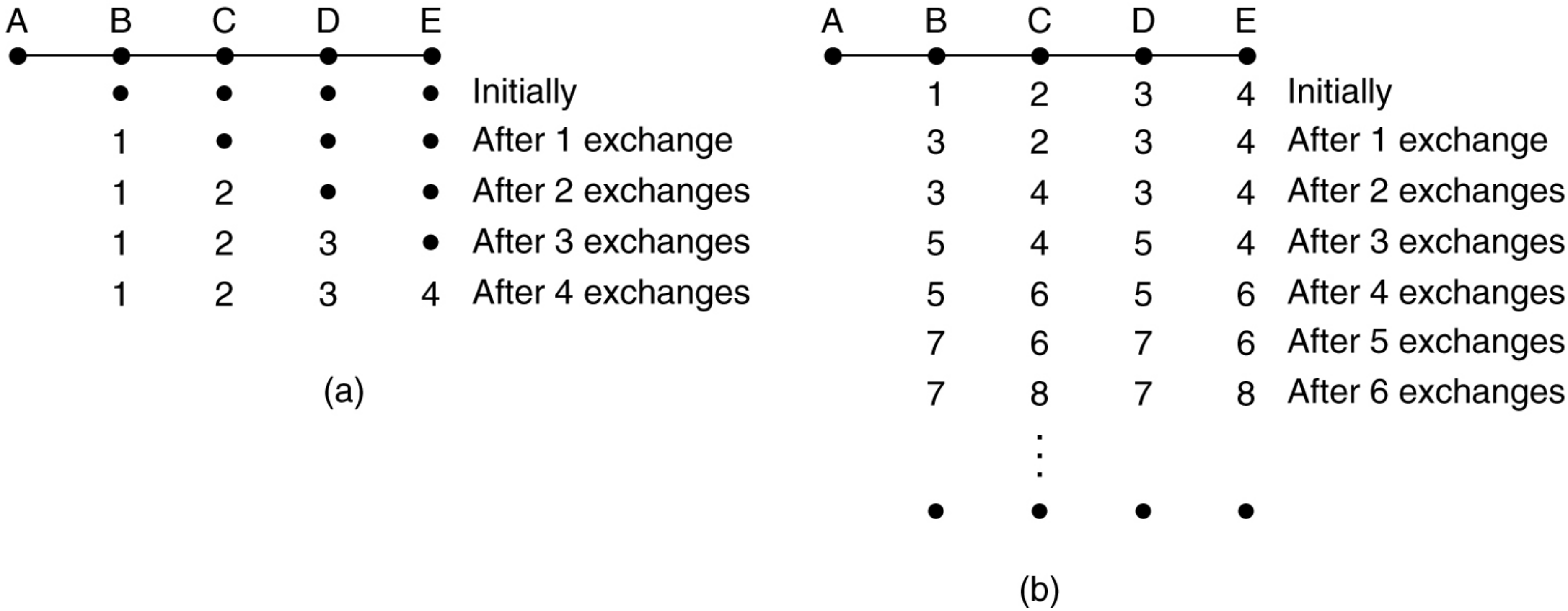
Good News Travels Fast

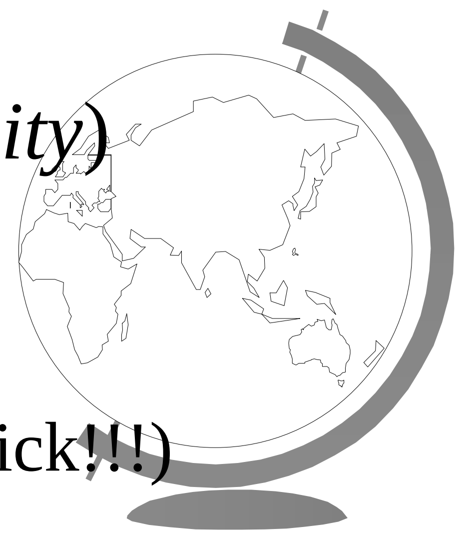
The count-to-infinity problem.



- ☞ Consider figure (a)
- ☞ A is initially down
- ☞ Path to A updated every exchange
- ☞ Stable in 4 exchanges

Bad News Travels Slowly





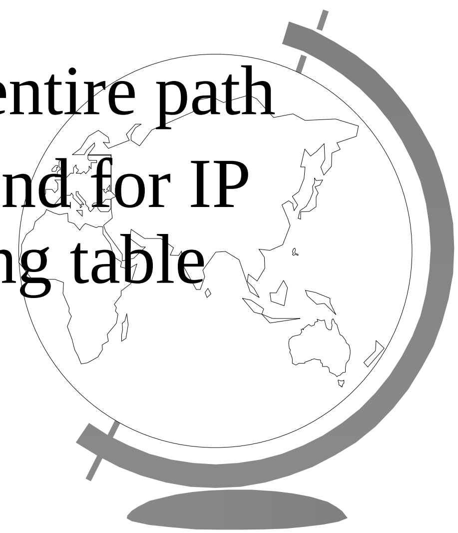
- ☞ Consider figure (b)
- ☞ Sloooowly converges to ∞ (*count to infinity*)
- ☞ Better to set infinity to max + 1
- ☞ Solution: split horizon hack
  - – Can inform neighbor of ∞ cost (bad news quick!!!)
  - – doesn’t always work

Topics

- ☞ Introduction
- ☞ Routing (5.2)
  - – static
  - – adaptive ←
- ☞ The Internet (5.5, brief)
- ☞ Congestion Control (5.3)

Review

- Each router
  - Has finite (small) no. of outgoing lines connected to it
  - Has to route to all hosts on the Internet
- Router view of the world
  - Sees world through routing table
  - Routing table: for any given host (IP address), which outgoing link router should pump it out on
  - Routing table tracks only outgoing line, not entire path
  - **Example:** when router receives a packet bound for IP address 178.157.277.191 in California, routing table says pump this out on line 6

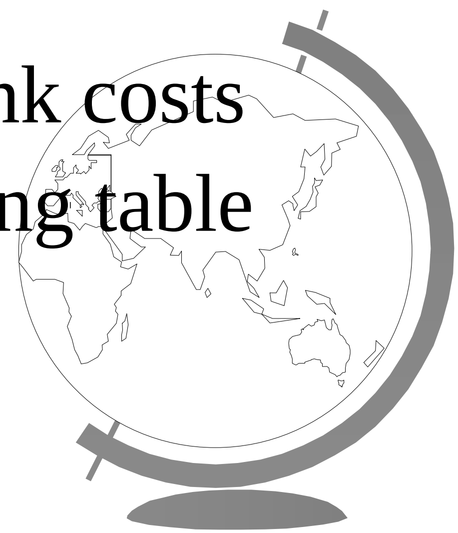


Review

- Routing algorithm: How to create entries on routing table
- Two classes: static and adaptive
- Static:
  - Each router collects costs of every link on the internet
  - Crunch once and determine shortest path to every node on the internet
  - Example: Dijkstra’s algorithm
  - Question: since each router has all the costs, why not just compute all possible paths and choose smallest?

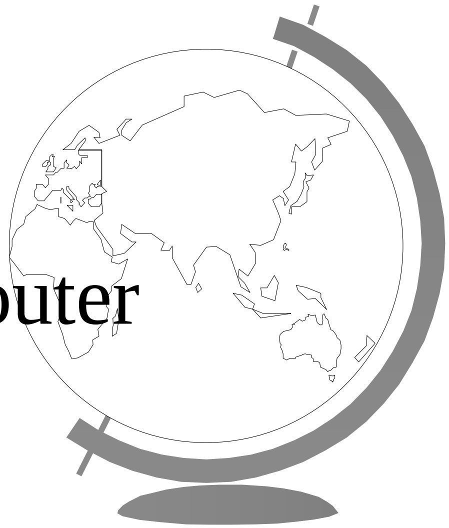
Review

- ☞ Adaptive:
  - – Periodically (say every 30 seconds) exchange routing tables with all neighbors
  - – Iteratively, learn about changes of costs
  - – Example: Distance Vector algorithm
- ☞ Major difference:
  - – Dijkstra’s algorithm sends neighbor link costs
  - – Distance vector exchanges entire routing table



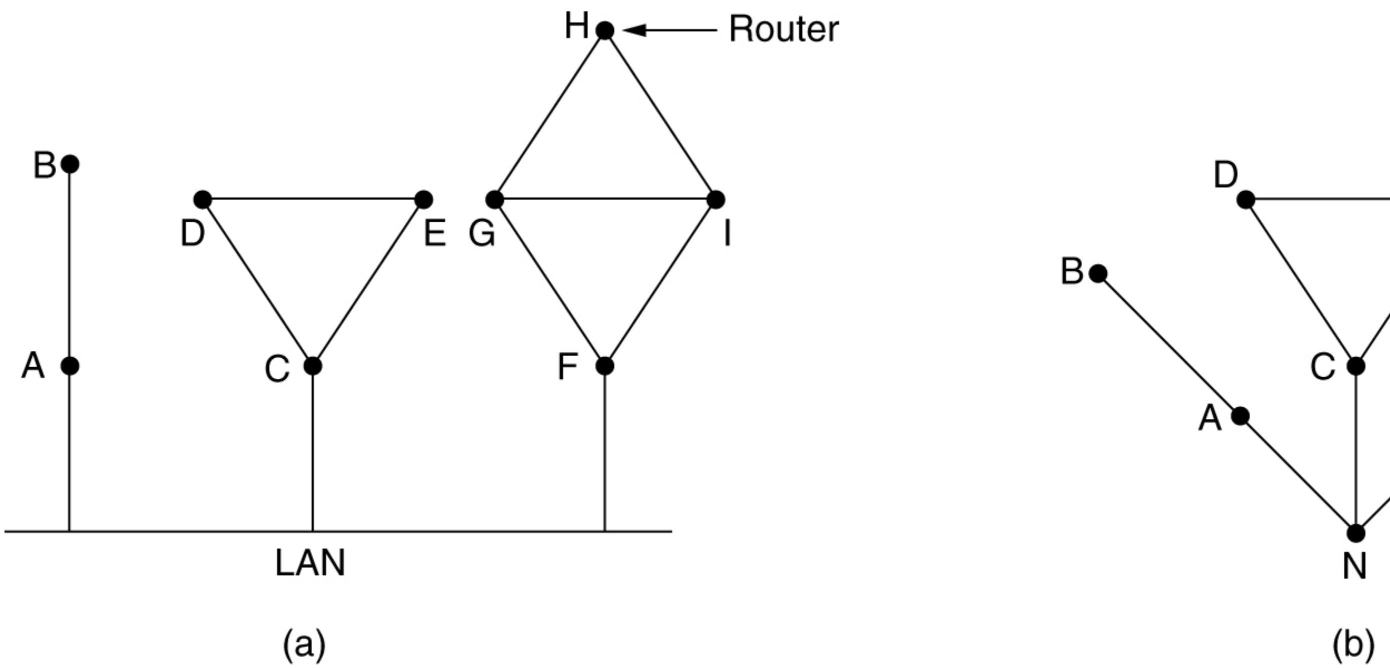
Link State Routing

- Used (w/variations) on Internet since 1979
- Basically
  - Experimentally measure distance
  - Use Dijkstra's shortest path
- Steps
  - Discover neighbors
  - Measure delay to each
  - Construct a packet telling what learned
  - Send to all other routers
  - Compute shortest path to every other router



Learning Neighbors

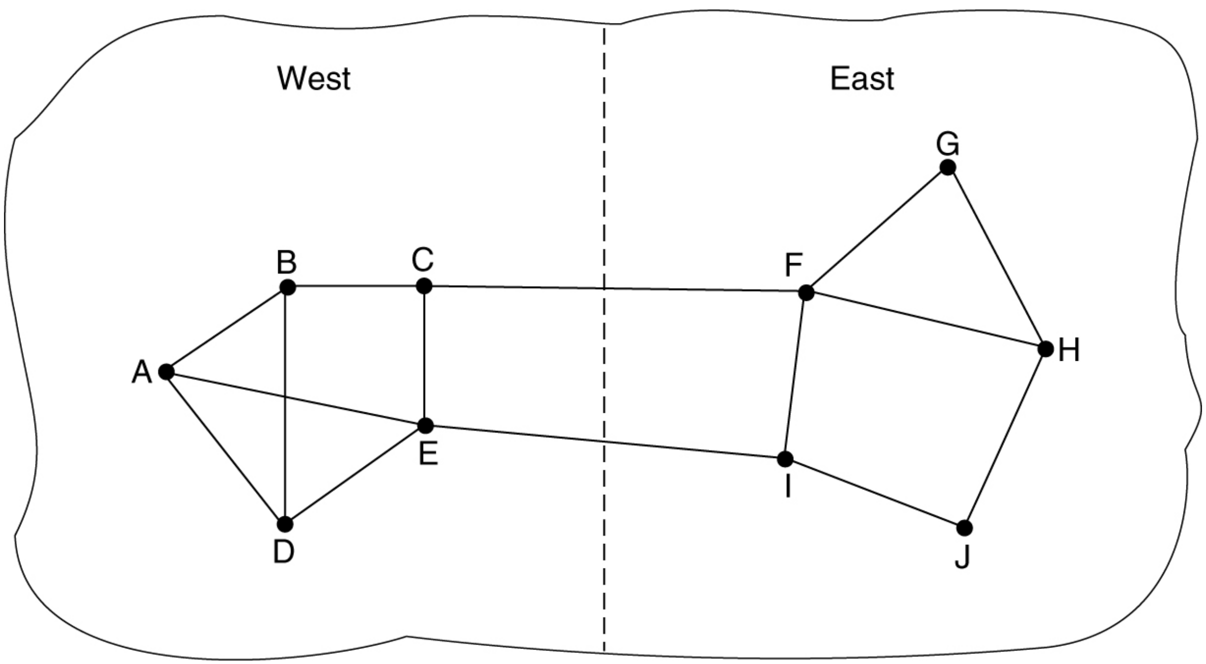
- Upon boot, send *HELLO* packet along point-to-point line
  - names must be unique
- Routers attached to LAN?



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

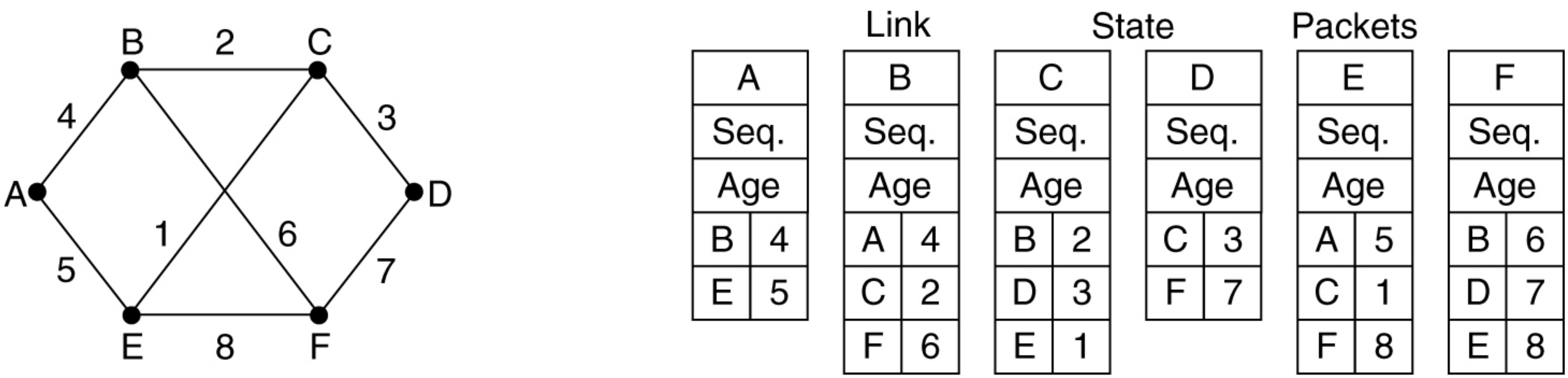
Measuring Line "Cost"

- Send ECHO packet, other router returns
  - delay
- Factor in load (queue length)?
  - Yes, if other distance equal, will improve perf
  - No, oscillating routing tables
  - Ex: Back and forth between C-F and E-I



Building Link State Packets

☞ Identity of sender, sequence number, age, list of (neighbors + distance)



(a) (b)

☞ *When* to send them?

– Periodically or when cost changes?

Distributing Link State Packets

- Tricky if topology changes as packets travel
  - routes will change "mid-air" based on new topology
- Basically, use *flooding* with checks
  - increment sequence each time new packet sent
- Forward all new packets
- Discard all duplicates
- If sequence number lower than max for sending station
  - then packet is obsolete and discard

Distribution Problems

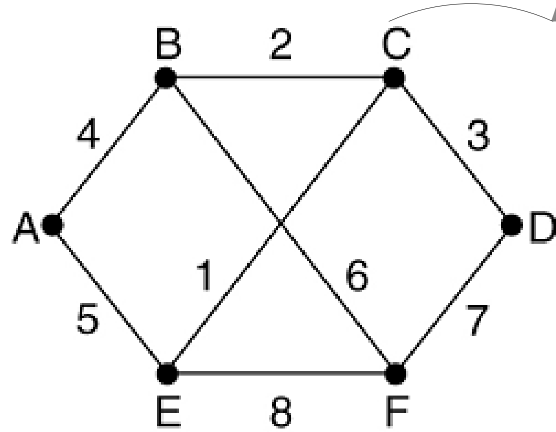
- Sequence numbers wrap around
  – use 32 bits and will take 137 years
- Router crashes … start sequence number at 0?
  – next packet it sends will be ignored
- Corrupted packet (65540)
  – packets 5 - 65540 will be ignored
- Use *age* field
  – decrement every second
  – if 0, then discard info for that router
- Hold for a bit before processing

Keeping Track of Packets

The packet buffer for router B in (Fig. 5-13).

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	

- A arrived
  - ack A
  - forward C and F
- F arrived
  - ack F
  - forward A and C

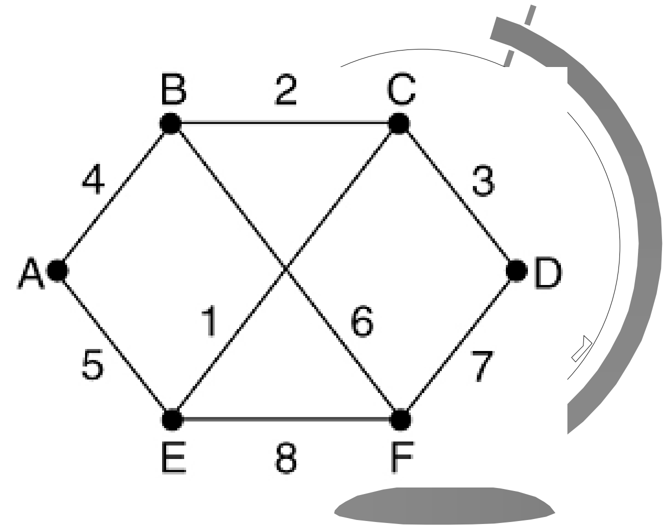


Keeping Track of Packets

Station B

Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
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	

- E arrived via EAB and via EFB
  - send only to C
- If C arrives via F before forwarded, updated bits and don’t send to F



Computing New Routes

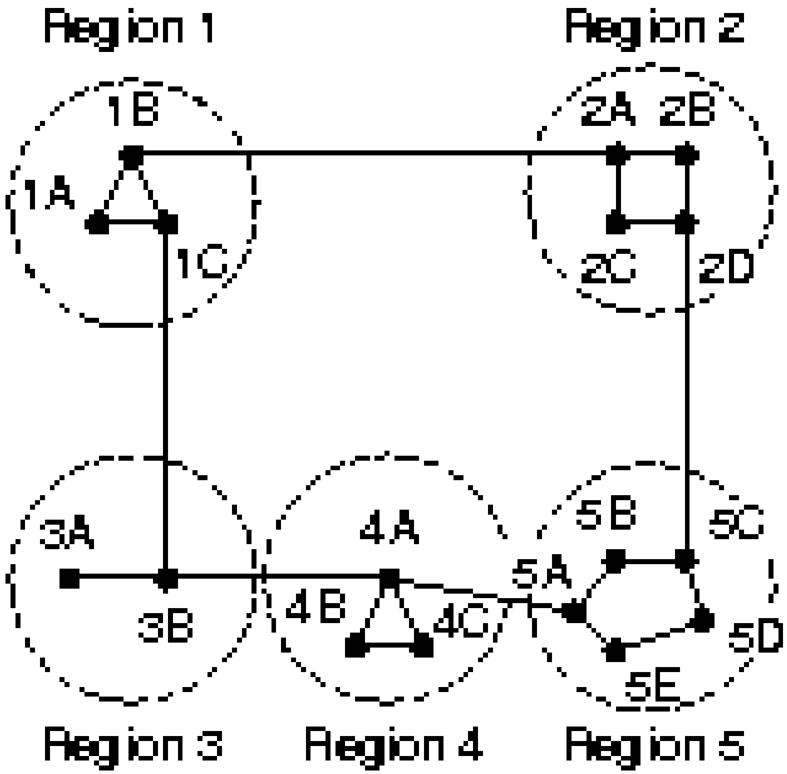
- Router has all link state packets
  - build subnet graph
- *N* routers degree *K*, O(*KN*) space
- Problems
  - router lies: forgets link, claims low distance
  - router fails to forward, or corrupts packets
  - router runs out of memory, calculates wrong
  - with large subnets, becomes probable
- Limit damage from above when happens

Link State Routing Today

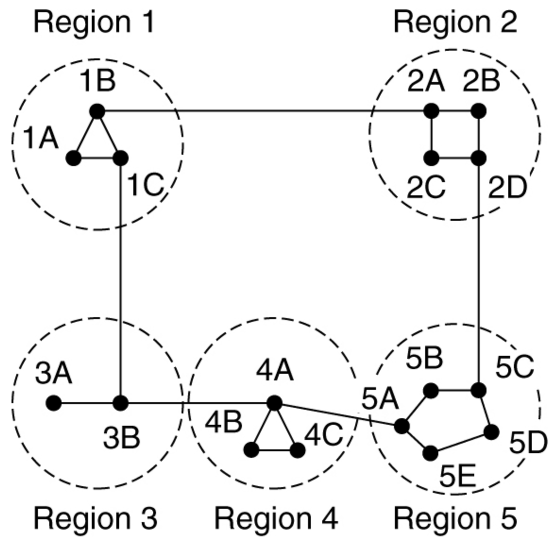
- ☞ Open Shortest Path First (OSPF) (5.6.4)
  - – used in Internet today
- ☞ Intermediate Sys Intermediate Sys (IS-IS)
  - – Designed initially for DECnet
  - – used in Internet backbones
  - – variant used for IPX in Novell networks
  - – carry multiple network layer protocols

Hierarchical Routing

- Global picture difficult for large networks
- Divide into regions
  - Router knows detail of its region
  - Routers in other regions reduced to a point



Reduced Routing Table



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

- Cost is efficiency
- Consider 1A to 5C
  - via 3 better for most of 5