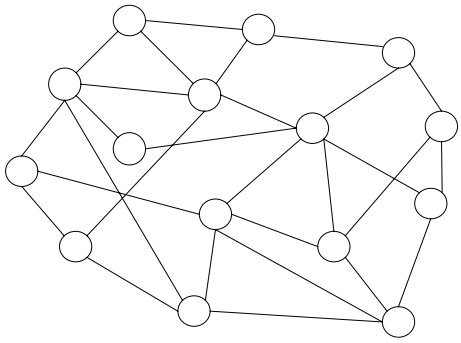
Review

- Layers
  - physical (hardware stuff, we won't go into)
  - data link (neighbor to neighbor...for now)
    1. delimit packets in stream of bits
    2. checksum to detect bit errors
    3. message numbers and acks and timeouts to notice lost messages and retransmit
    4. some examples: Bisync, ARPANET
    5. comparison with reliable layer 4
  - network (routing, addressing)
  - Transport
  - Session (irrelevant)
  - Presentation (not a protocol, won't cover)
  - Application (file transfer, telnet, ...)

New Topic: Routing Algorithms

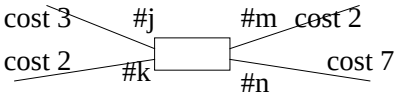


Types of Routing

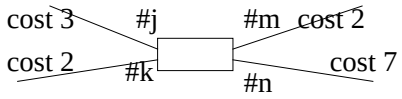
- Fully connected topologies
- Fixed topologies (LANs)
- Trust someone else (VANs)
- Static Routing
- Local information -- flooding, hot potato
- Centralized routing
- Source routing
- Distributed adaptive routing
  - distance vector (also called Bellman/Ford)
  - link state (also called SPF for Shortest Path First)

Distance Vector Routing

- You know the following:
  - your own ID
  - how many cables hanging off your box
  - the cost of going through that cable to whatever's at the other end



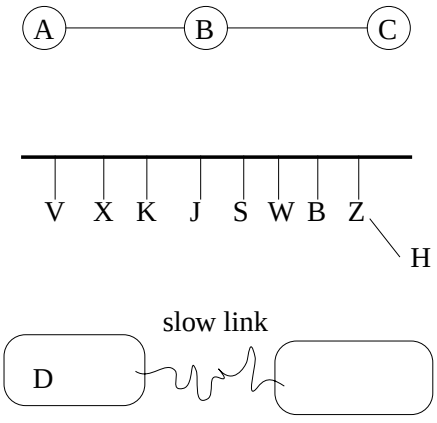
- Purpose of routing algorithm: come up with *forwarding database*, telling you which neighbor to send to for each possible destination
- Do this by exchanging *distance vector*s, which tells transmitters distance to each destination



You are destination #4

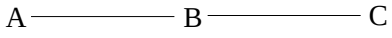
dest #	1	2	3	4	5	6	7	8	9	10	11
distance vector received from cable j											
3	12	3	15	3	12	5	6	18	0	7	15
distance vector received from cable k											
2	5	8	3	2	10	7	4	20	5	0	15
distance vector received from cable m											
2	0	5	3	2	19	9	5	22	2	4	7
distance vector received from cable n											
7	6	2	0	7	8	5	8	12	11	3	2
your own calculated distance vector											
	2	6	5	0	12	8	6	19	3	2	9
your forwarding table											
	m	j	m	0	k	j	k	n	j	k	n

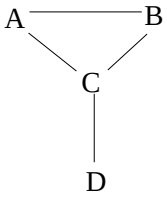
Looping Problem



Split Horizon

- Don't tell neighbor N you can reach D if you'd forward to D through N

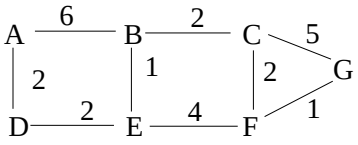




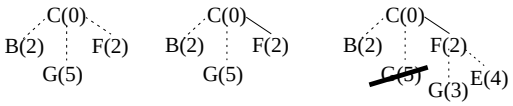
Link State Routing

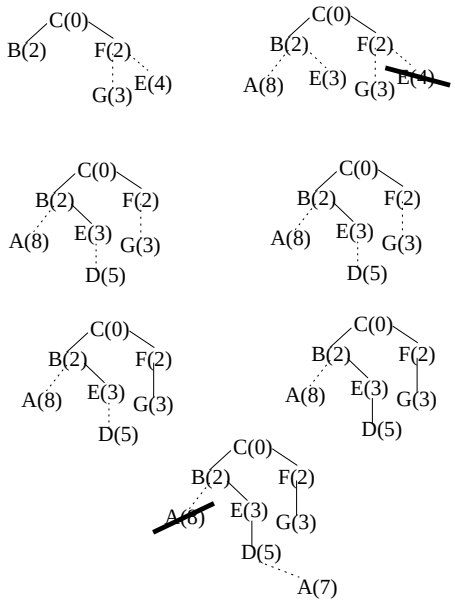
- Meet your neighbors
- Construct Link State Packet (LSP)
  - who you are
  - list of (neighbor, cost) pairs
- Broadcast the LSP to all routers
- Store latest LSP from every other node
- Compute routes
  - Edsgar Dijkstra's algorithm
    1. Put (SELF,0) on tree as Root
    2. Look at LSP of node just placed on tree. If for any node N the cost c is best path of any found so far, add (N,c) to tree under N with dotted line
    3. Make shortest dotted line solid. Go to 2.

Example Dijkstra Calculation



A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



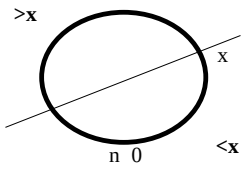


Broadcasting LSP

- Can't depend on routing database giving any useful information
- Basic idea is flooding — send to every neighbor except one from which LSP received
- But flooding is exponential. We can do better than that since we store the LSPs. Only flood an LSP if it's new
- How do you tell if it's new?
  - different from what's in database?
  - globally synchronized clocks?
  - local battery-backup clocks
  - sequence numbers — need age field too
    1. finite sized field
    2. restarts

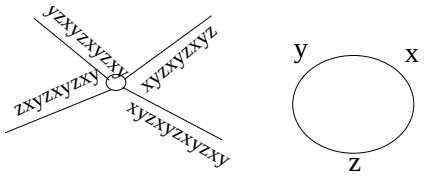
ARPANET Flooding Algorithm

- Two fields: sequence number, and age
- Suppose you have LSP from Alice in database, with sequence number *x*. You receive an LSP (via neighbor N) with source Alice, with sequence number *y*.
- As you'd expect, flood if y > x, else ignore
- But sequence number wraps around. Arithmetic in circular space is as follows:



ARPANET Flooding, cont'd

- Source sets age field to MAX-AGE (64 seconds, 3 bit field, units of 8 seconds)
- Decrement age in stored LSP after holding it for 8 seconds
- If age=0, then "too old" — don't propagate
- Generate new LSP within MAX-INT (60 seconds)
- When starting, wait RESTART-TIME (90 seconds)
- ARPANET incident — symptom: net didn't work



What now?

- Networks don't have on-off switches
- First crash and reload BBN router
- Still broken — examine core dump
- Realize the problem
- Created patched version of the code
- Load BBN router with patched version
- Tell it to tell neighbors to crash
- Eventually they get around to crashing, reload them with patched version
- After every router reloaded, reload one by one with "correct" code
- Hope it doesn't happen again by accident
- Hope it doesn't ever happen on purpose

New Improved LSP Flooding

- Require LSPs to age every time a router touches it
- Make sequence number large and linear
- Careful synchronization between neighbors of LSP database

src	nbr 1	nbr 2	nbr 3	nbr 4	nbr 5
A	ok	ack	ok	Xmit	ack
B	ok	ok	ok	ok	ok
C	ack	Xmit	Xmit	Xmit	Xmit
D	ok	Xmit	ok	ok	ok
E	Xmit	ok	ack	ok	ok

LSP sychronization rules

- round robin when link available
- if flag says "ack", send ack, and change flag to "ok"
- if flag says "Xmit", send LSP
- if receive new LSP from neighbor N, set "ack" for N, and "Xmit" for each other neighbor for that LSP, and transmit that LSP as soon as possible (don't wait first time for retransmit interval)
- if receive duplicate LSP from neighbor N, set flag to "ack"
- if receive ack from neighbor N for LSP in database, change flag to "ok"
- Still have age field. When it gets to be 0, and all neighbors have ack'd, then delete LSP

Distance Vector vs Link State

- Memory: distance vector wins, by a little
- CPU: debatable, probably distance vector wins, by a little
- Simplicity of coding: simple distance vector wins. Not sure about newfangled ones.
- Convergence speed: link state wins easily when comparing with RIP, but some claim distance vector can avoid looping. If so, link state would still win, by a little.
- Functionality: link state wins
  - fancy custom routes
  - troubleshooting
  - mapping the network
  - sabotage-proof routing