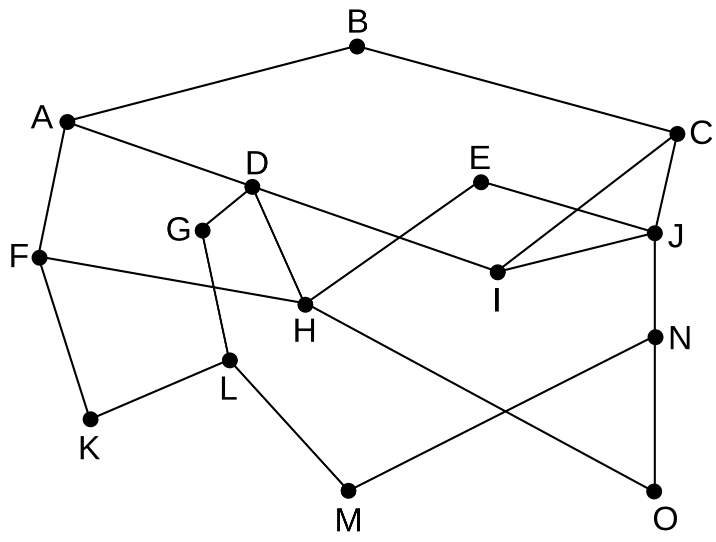
Announcements

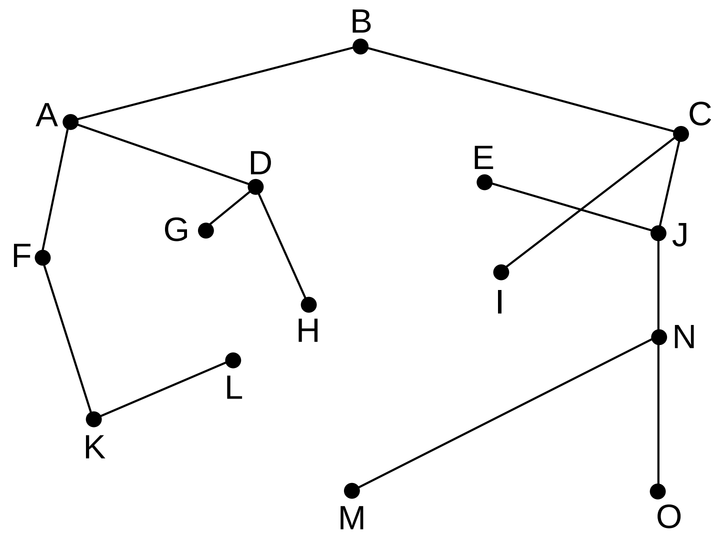
- Midterms
  - Mt #1 Tuesday March 6
  - Mt #2 Tuesday April 15
  - Final project design due **April 11**
- Project partner sign-up sheet
  - it was passed around

Optimality Principal

- If J is on the optimal route from I to K
  - then the optimal route from I to K shares the optimal route from J to K
- transitive result of this is a sink tree
  - can construct a tree from all nodes to a specific node



(a)



(b)

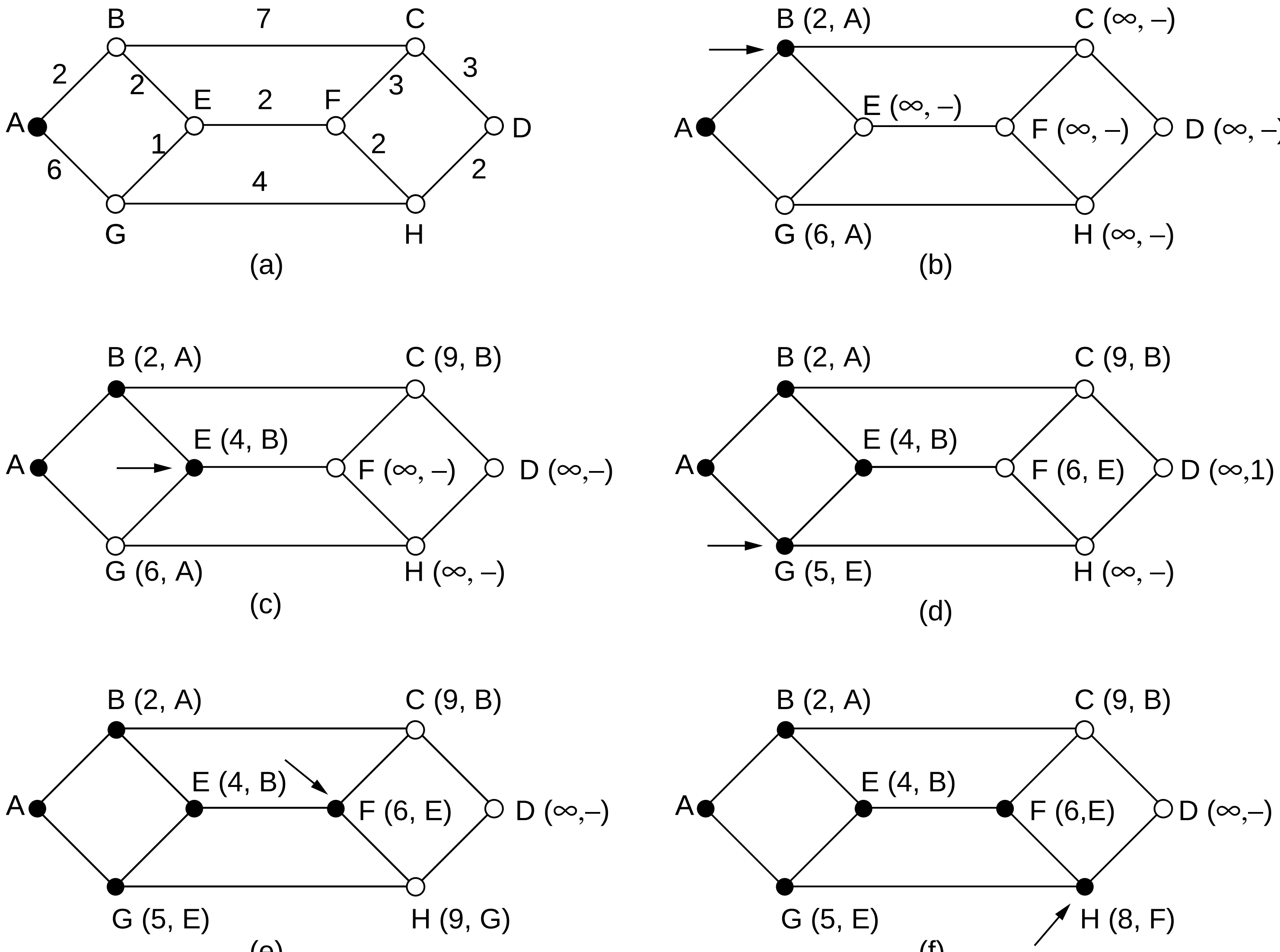
From: *Computer Networks*, 3rd Ed. by Andrew S. Tanenbaum, (c)1996 Prentice Hall.

Shortest Path Routing

- Graph Representation
  - nodes are routers
  - arcs are links
  - to get between two routes, select a the shortest path
  - need to decide metric to use for minimization
- Dijkstra's Algorithm

select source as current node
while current node is not destination
    foreach neighbor of current
        if route via current is better update its tentative route
        label node with <distance, current Node>
    find tentative node with shortest route
        mark a permanent
        make it current

Shortest Path Example



From: *Computer Networks*, 3rd Ed. by Andrew S. Tanenbaum, (c)1996 Prentice Hall.

Flood Routing

- Every Incoming packet is resent on every outbound link
- generates many duplicate packets
- potentially infinite packets unless they are damped
  – multiple paths to the same destination result in loops
  – can use a lifetime (max hops) to damp traffic
  – can also keep track in routers if the packet has been seen
- good metric to compare algorithms
  – flooding always chooses the shortest path
  – must ignore overhead and congestion due to flooding

Flow-Based Routing

- Compute optimal routes off-line if we know *in advance*:
  - link capacity
  - topology
  - traffic for foreach <src,dest> pair
- Testing a routing table:
  - given a tentative routing table
  - for each link we can compute mean delay

$$T = \frac{1}{\mu C - \lambda}$$

  - C is link capacity bps, $1/\mu$ is mean packet size, λ is actual traffic in packets/sec
  - then compute overall utilization (as mean or max of delays)
  - possible to exhaustively try all routing tables this way

Distance Vector Routing

- Also known as Bellman-Ford or Ford-Fulkerson
  - original ARPANET routing algorithm
  - early versions of IPX and DECnet used it too
- Each router keeps a table of tuples about all other routers
  - outbound link to use to that router
  - metric (hops, etc.) to that router
  - routers also must know “distance” to each neighbor
- Every T sec., each router sends it table to its neighbors
  - each router then updates its table based on the new info
- Problems:
  - fast response to good news
  - slow response to bad news
    - takes max hops rounds to learn of a downed host
    - known as count-to-infinity problem

Link State Routing

- Used on the ARPANET after 1979
- Each Router:
  - computes metric to neighbors and sends to **every** other router
  - each router computes the shortest path based on received data
- Needs to estimate time to neighbor
  - best approach is send an **ECHO** packet and time response
- Distributing Info to other routers
  - each router may have a different view of the topology
  - simple idea: use flooding
  - refinements
    - use age sequence number to damp old packets
    - use acks to permit reliable delivery of routing info

Hierarchical Routing

- Routing grows more complex with more routers
  - takes more space to store routing tables
  - requires more time to compute routes
  - uses more link bandwidth to update routes
- Solution:
  - divide the world into several hierarchies
    - Do I really care that router z at foo U just went down?
  - only store info about
    - your local area
    - how to get to higher up routers
  - optimal number of levels for an N router network is ln N
    - requires a total of *e* ln N entries per router

Routing for Mobility

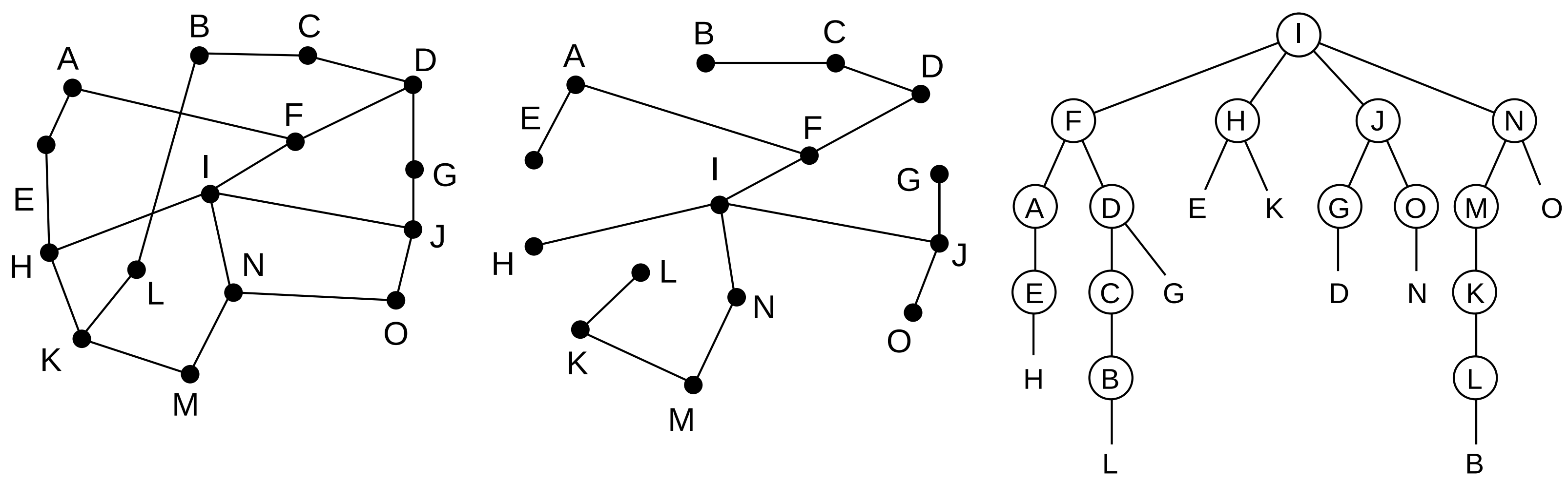
- Or What happens when computers move?
- Two types of mobility:
  - migratory: on the net in many locations but not while in motion
  - roaming: on the net while in motion
- Basic idea:
  - everyone has a home
    - you spend much of your time near home
    - when not at home, they know where to find you
  - home agents: know where you are (or that you are missing)
  - foreign agents: inform home agents of your location
    - informs users that future communication should be sent via them (this is a huge potential security hole)

Broadcast Routing

- Sometimes information needs to go to everyone
  - routing updates in link-state
  - stock data, weather data, etc.
- sender iterates over all destinations
  - wastes bandwidth
  - sender must know who is interested
- flooding
  - see routing updates for issues
- multi-destination routing
  - routers support having multiple destinations
  - routers copy output packets to correct link(s)
- spanning tree
  - contains subset of graph with no loops
  - efficient use of bandwidth
  - requires info to be present in routers (but it is for link state)

Routing Broadcast Traffic (cont.)

- Reverse path forwarding
  - check link a packet arrives on
  - if the inbound link is the one the router would use to the source, then
    - forward it out all other links
  - else
    - discard the packet
  - requires no special data sorted in each router



(a) (b) (c)

From: *Computer Networks*, 3rd Ed. by Andrew S. Tanenbaum, (c)1996 Prentice Hall.

Multicast Routing

- Specify a (relatively) small list of hosts to receive traffic
  - may need to exchange traffic as a group
  - must create/destroy group
- Using spanning trees
  - prune links that are have no members of mulicast group
  - for distance-vector use a variation on reverse path forwarding
    - when a router gets a message it doesn't need it send a prune message back
    - recursively prunes back un-needed subnets
- core-based trees
  - one tree for group not one per group member
  - hosts send to "core" and it multicasts it out