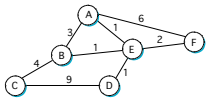
Routing

Outline

- Distance Vector
- Link State

Overview

- Forwarding vs Routing
  - forwarding: to select an output port based on destination address and routing table
  - routing: process by which routing table is built
- Network as a Graph

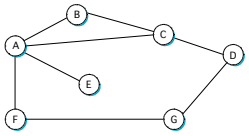


- Problem: Find lowest cost path between two nodes
- Factors
  - static
  - dynamic

Distance Vector

- Each node maintains a set of triples
  - `(Destination, Cost, NextHop)`
- Directly connected neighbors exchange updates
  - periodically (on the order of several seconds)
  - whenever table changes (called *triggered* update)
- Each update is a list of pairs:
  - `(Destination, Cost)`
- Update local table if receive a "better" route
  - smaller cost
  - came from next-hop
- Refresh existing routes; delete if they time out

Example



Destination	Cost	NextHop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

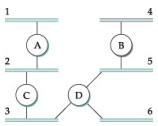
Routing Loops

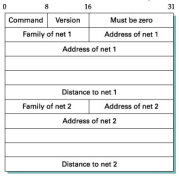
- Example 1
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update t o A
  - A sets distance to G to infinity since it uses F to reach G
  - A receives periodic update from C with 2-hop path to G
  - A sets distance to G to 3 and sends update to F
  - F decides it can reach G in 4 hops via A
- Example 2
  - link from A to E fails
  - A advertises distance of infinity to E
  - B and C advertise a distance of 2 to E
  - B decides it can reach E in 3 hops; advertises this to A
  - A decides it can read E in 4 hops; advertises this to C
  - C decides that it can reach E in 5 hops…

Loop-Breaking Heuristics

- Set infinity to 16
- Split horizon
- Split horizon with poison reverse

Routing Information Protocol (RIP)





- Distributed along with BSD Unix
- Straightforward implementation of DV
- Updates sent every 30 seconds
- Link costs constant at 1 (16 = infinity)

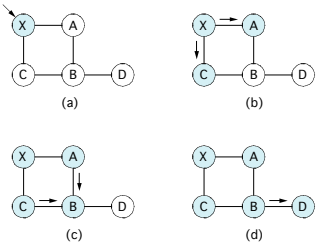
Link State

- Strategy
  - send to all nodes (not just neighbors) information about directly connected links (not entire routing table)
- Link State Packet (LSP)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet

Link State (cont)

- Reliable flooding
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new LSP periodically
    - increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each LSP
    - discard when TTL=0

Link State (cont)



Route Calculation

- Dijkstra's shortest-path algorithm
- Let
  - N denotes set of nodes in the graph
  - $l(i, j)$ denotes non-negative cost (weight) for edge (i, j)
  - s denotes this node
  - M denotes the set of nodes incorporated so far
  - $C(n)$ denotes cost of the path from s to node n

```
M = {s}
for each n in N - {s}
   C(n) = l(s, n)
while (N != M)
   M = M union {w} such that C(w) is the minimum for
       all w in (N - M)
   for each n in (N - M)
      C(n) = MIN(C(n), C (w) + l(w, n ))
```


OSPF

- Open Shortest Path First Protocol
- Authentication
- Additional hierarchy
- Load balancing

Metrics

- Original ARPANET metric
  - measures number of packets queued on each link
  - took neither latency or bandwidth into consideration
- New ARPANET metric
  - stamp each incoming packet with its arrival time (**`AT`**)
  - record departure time (**`DT`**)
  - when link-level ACK arrives, compute

```
Delay = (DT - AT) + Transmit + Latency
```

  - if timeout, reset **`DT`** to departure time for retransmission
  - link cost = average delay over some time period
- Revised ARPANET metric
  - compressed dynamic range
  - replaced **`Delay`** with link utilization
- Practice
  - static metrics (e.g., 1/bandwidth)

Routing for Mobile Hosts

- Mobile IP:
  - home agent, home address, foreign agent
  - triangle routing problem