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 influencing routing
  - static: topology
  - dynamic: load

Distance Vector

- Each node maintains a set of triples
  - (Destination, Cost, NextHop)
- Exchange updates directly connected neighbors
  - periodically (on the order of several seconds)
  - whenever its 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

- Example 1
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update to 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 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 reach E in 4 hops; advertises this to C
  - C decides that it can reach E in 5 hops...

Loop-Breaking Heuristics

- Set a small infinity
  - Uses a small number to represent infinity. Nodes can count to infinity faster!
- Split horizon
  - Don’t send routing information back to the node that you learnt it from in the first place
- Split horizon with poison reverse
  - Send negative routing information back to the node you learnt it from

Routing Loops

- Example 1
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update to 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 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 reach E in 4 hops; advertises this to C
  - C decides that it can reach E in 5 hops...

Link State

- Strategy
  - send to all nodes (not just neighbors)
    information about directly connected links (not entire routing table)
- Link State Packet (LSP) contains
  - id of the node that created the LSP
  - cost of the 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 stored LSP
  - discard when TTL=0

### 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 \( n \)

\[
N = \{s\}
\]

for each \( n \) in \( N - \{s\} \)

\[
C(n) = l(s, n)
\]

while \( (M \neq N) \)

- \( M = M \cup \{w\} \) such that \( C(w) \)
  is the minimum for all \( w \) in \( N - M \)

for each \( n \) in \( (N - M) \)

\[
C(n) = \text{MIN}(C(n), C(w) + l(w, n))
\]

### Open Shortest Path First (OSPF)

- **Uses the link state routing protocol**

- **Offers several additional features**
  - Authentication of routing messages
  - Allows further partitioning of a network into areas
  - Can load balance traffic over multiple routes to the same destination, if the routes have the same cost

### Metrics

- **Original ARPANET metric**
  - measures number of packets enqueued 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
  \[
  \text{Delay} = (DT - AT) + \text{Transmit} + \text{Latency}
  \]
  - on reliable links, reset \( DT \) to departure time on retransmission
  - link cost = average delay over some time period
  - Problem:
    - causes routing instability on heavy load

- **Fine Tuning**
  - compressed dynamic range
  - replaced delay with link utilization

© 1999, Scott F. Midkiff