Housekeeping

- Midterm exam Friday 10/12 – covering Chapters 1-4
- Homework for chapters 3&4 assigned 10/3 due 10/17

Quick Essay

- Do NOT put your name on your paper
- List the 3 most important things to learn from Project 1
- What is still most unclear about Project 1?
- I’ll report results by email tomorrow or in class Wednesday

Today

- Overall topic: Routing
- Distance-vector: loops
  - Preventing loops
- Link-state algorithms
  - Link-state forwarding
  - Route computation from link-state data

Distance Vector
also known as Bellman-Ford

- Each node maintains a set of triples
  - (Destination, Cost, NextHop)
- Exchange updates directly connected neighbors
  - 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**

Routing Table for node B

```
<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>A</td>
</tr>
</tbody>
</table>
```

**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 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 – ok for small networks
- Split horizon
  - Don’t send routes back to the peers they were learned from
  - If B has (E,2,A) in its routing table, it doesn’t send (E,2) as an update to A
- Split horizon with poison reverse
  - B sends (E,infinity) to A (and ONLY A)
- Still haven’t handled loops of >2 routers

**Internet RIP**

- Goal is to reach networks, not routers, so route table contains network identity, not router identity
  - route print
- Update every 30 seconds + triggered updates
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 stored LSP
    - discard when TTL=0

Link-state table

- At each node, link weight table and forwarding table:

<table>
<thead>
<tr>
<th>l(i,j) – “same” everywhere</th>
<th>Forwarding – node E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0 B 2 C 1 D 1 E 3</td>
<td>Dest NH Cost</td>
</tr>
<tr>
<td>A 0 2 1</td>
<td>A</td>
</tr>
<tr>
<td>B 2 0 1</td>
<td>B</td>
</tr>
<tr>
<td>C 1 1 0 1 3</td>
<td>C</td>
</tr>
<tr>
<td>D 1 0 1</td>
<td>D</td>
</tr>
<tr>
<td>E 3 1 0</td>
<td>E</td>
</tr>
</tbody>
</table>

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 \( n \)

  \[
  M = \{ s \} \\
  \text{for each } n \text{ in } N - \{ s \} \\
  C(n) = l(s, n) \\
  \text{while } (N != M) \\
  M = M \cup \{ w \} \text{ such that } C(w) \text{ is the minimum for all } w \text{ in } (N - M) \\
  \text{for each } n \text{ in } (N - M) \\
  C(n) = \text{MIN}(C(n), C(w) + l(w, n))
  \]
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
    \[ \text{Delay} = (DT - AT) + \text{Transmit} + \text{Latency} \]
  – if timeout, reset $DT$ to departure time for retransmission
  – link cost = average delay over some time period

• Fine Tuning
  – compressed dynamic range
  – replaced Delay with link utilization