ELEN 602 Lecture 19

- Routing

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: topology
  - dynamic: load

Routing Algorithms

- Distance Vector
  - Each node constructs a vector of distances to its neighbors and transmits it to neighbors
  - Use this information in building routing tables
- Link State
  - Disseminate link state information of neighbors to everyone
  - Construct routing tables based on the sum of information from all the nodes

Distance Vector

- Each node maintains a set of triples
  - \((Destination, Cost, NextHop)\)
- Exchange updates with 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
**Initial Distances (Global View)**

<table>
<thead>
<tr>
<th>Node</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Inf</td>
<td>1</td>
<td>1</td>
<td>Inf</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>D</td>
<td>Inf</td>
<td>Inf</td>
<td>1</td>
<td>0</td>
<td>Inf</td>
<td>Inf</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>0</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>1</td>
<td>Inf</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Routing Table Updates**

- Update an entry if a new distance vector shows a shorter route to the destination.
- A shortest distance path to a destination consists of shortest distance paths between intermediate nodes.
  - \( D(I, J) = \min (D(I, K) + D(K, J)) \), over all \( K \)
- Bellman-Ford or Ford-Fulkerson Algorithm
- Periodic Updates based on timers
- Triggered updates --based on noticed changes in distance vector
- Recompute distances based on updates.

**Initial Routing Table at A**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>Inf</td>
<td>---</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>Inf</td>
<td>----</td>
</tr>
</tbody>
</table>
Example Network

- When A hears from C, it finds a distance 2 path to D through C - update routing table entry for D.
- When A hears from D, it finds a distance 3 path to G through D - updates routing table entry for G.
- When A hears from F, it finds a distance 2 path to G through F - updates routing table entry for G.

Final Routing Table at A

<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost</th>
<th>Nexthop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>B</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>1</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>F</td>
</tr>
</tbody>
</table>

Final Distances (Global View)

<table>
<thead>
<tr>
<th>Information at</th>
<th>Distance to Reach Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>A  B  C  D  E  F  G</td>
</tr>
<tr>
<td>A</td>
<td>0  1  1  2  1  1  2</td>
</tr>
<tr>
<td>B</td>
<td>1  0  1  2  2  2  3</td>
</tr>
<tr>
<td>C</td>
<td>1  1  0  1  2  2  2</td>
</tr>
<tr>
<td>D</td>
<td>2  2  1  0  3  2  1</td>
</tr>
<tr>
<td>E</td>
<td>1  2  2  3  0  2  3</td>
</tr>
<tr>
<td>F</td>
<td>1  2  2  2  2  0  1</td>
</tr>
<tr>
<td>G</td>
<td>2  3  2  1  3  1  0</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 through C; 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…
Routing Tables for Example Network

Example Network

Sample Network with link costs
New Network with link 3-6 broken

![Network Diagram]

Routing entries to Destination Node 6

<table>
<thead>
<tr>
<th>Update</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before break</td>
<td>(3,3)</td>
<td>(4,4)</td>
<td>(6,1)</td>
<td>(3,3)</td>
<td>(6,2)</td>
</tr>
<tr>
<td>1</td>
<td>(3,3)</td>
<td>(4,4)</td>
<td>(4,5)</td>
<td>(3,3)</td>
<td>(6,2)</td>
</tr>
<tr>
<td>2</td>
<td>(3,7)</td>
<td>(4,4)</td>
<td>(4,5)</td>
<td>(2,5)</td>
<td>(6,2)</td>
</tr>
<tr>
<td>3</td>
<td>(3,7)</td>
<td>(4,6)</td>
<td>(4,7)</td>
<td>(2,5)</td>
<td>(6,2)</td>
</tr>
<tr>
<td>4</td>
<td>(2,9)</td>
<td>(4,6)</td>
<td>(4,7)</td>
<td>(5,5)</td>
<td>(6,2)</td>
</tr>
<tr>
<td>5</td>
<td>(2,9)</td>
<td>(4,6)</td>
<td>(4,7)</td>
<td>(5,5)</td>
<td>(6,2)</td>
</tr>
</tbody>
</table>

Each entry = (Next node, distance)

Problems with link failures

![Network Diagram (a) and (b)]

Routing Table

<table>
<thead>
<tr>
<th>Update</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Break</td>
<td>(2,3)</td>
<td>(3,2)</td>
<td>(4,1)</td>
</tr>
<tr>
<td>After Break</td>
<td>(2,3)</td>
<td>(3,2)</td>
<td>(3,3)</td>
</tr>
<tr>
<td>1</td>
<td>(2,3)</td>
<td>(3,4)</td>
<td>(3,3)</td>
</tr>
<tr>
<td>2</td>
<td>(2,5)</td>
<td>(3,4)</td>
<td>(3,5)</td>
</tr>
<tr>
<td>3</td>
<td>(2,5)</td>
<td>(3,6)</td>
<td>(3,5)</td>
</tr>
<tr>
<td>4</td>
<td>(2,7)</td>
<td>(3,6)</td>
<td>(3,7)</td>
</tr>
<tr>
<td>.......</td>
<td>.......</td>
<td>.......</td>
<td>.......</td>
</tr>
</tbody>
</table>

---

17
18
19
20
Loop-Breaking Heuristics

(Split Horizon with Poisoned Reverse)

- Set infinity to 16
- Split horizon
  - Don’t send minimum cost to neighbor if neighbor is the NextHop
  - Send minimum cost to all neighbors, but set infinity to the NextHop neighbor
- Work satisfactorily in some cases

Routing Table Example

<table>
<thead>
<tr>
<th></th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>(2,3)</td>
<td>(3,2)</td>
<td>(4,1)</td>
</tr>
<tr>
<td>Update</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After</td>
<td>(2,3)</td>
<td>(3,2)</td>
<td>(4,1)</td>
</tr>
<tr>
<td></td>
<td>(2,3)</td>
<td>(2,3)</td>
<td>(1,Inf)</td>
</tr>
<tr>
<td></td>
<td>(1,Inf)</td>
<td>(1,Inf)</td>
<td>(1,Inf)</td>
</tr>
</tbody>
</table>

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
- Reliable flooding
  - discard when TTL=0

Link State Packet (LSP)

- 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

- send to all nodes (not just neighbors) information about directly connected links (not entire routing 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 $n$

\[
M = \{s\} \\
\text{for each } n \in N - \{s\} \\
\quad C(n) = l(s, n) \\
\text{while } (N \neq M) \\
\quad M = M \cup \{w\} \text{ such that } C(w) \\
\quad \text{is the minimum for all } w \in (N - M) \\
\quad \text{for each } n \in (N - M) \\
\quad C(n) = \text{MIN}(C(n), C(w) + l(w, n))
\]

Example Route Calculation

<table>
<thead>
<tr>
<th>Iteration</th>
<th>N</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>{1}</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>Inf</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>{1,3}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>Inf</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>{1,2,3}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>{1,2,3,6}</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>{1,2,3,4,6}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>{1,2,3,4,5,6}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Shortest-path tree from node 1

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
    \[
    \text{Delay} = (DT - AT) + \text{Transmit + Latency}
    \]
  - if timeout, reset $DT$ to departure time for retransmission
  - link cost = average delay over some time period
- Fine Tuning
  - compressed dynamic range
  - replaced dynamic with link utilization
How to Make Routing Scale

- Flat versus Hierarchical Addresses
- Inefficient use of Hierarchical Address Space
  - class C with 2 hosts (2/255 = 0.78% efficient)
  - class B with 256 hosts (256/65535 = 0.39% efficient)
- Still Too Many Networks
  - routing tables do not scale
  - route propagation protocols do not scale

Subnetting

- Add another level to address/routing hierarchy: subnet
- Subnet masks define variable partition of host part
- Subnets visible only within site

<table>
<thead>
<tr>
<th>Network number</th>
<th>Host number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B address</td>
<td></td>
</tr>
<tr>
<td>11111111111111111111111</td>
<td>00000000</td>
</tr>
<tr>
<td>Subnet mask (255.255.255.0)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network number</th>
<th>Subnet ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnetted address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subnet Example

- Forwarding table at router R1
- Subnet mask: 255.255.255.128
- Subnet number: 128.96.34.0

<table>
<thead>
<tr>
<th>Subnet Number</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.96.34.0</td>
<td>255.255.255.128</td>
<td>interface 0</td>
</tr>
<tr>
<td>128.96.34.128</td>
<td>255.255.255.128</td>
<td>interface 1</td>
</tr>
<tr>
<td>128.96.33.0</td>
<td>255.255.255.0</td>
<td>R2</td>
</tr>
</tbody>
</table>

Figure 7.27
### Forwarding Algorithm

\[
D = \text{destination IP address} \\
\text{for each entry } (\text{SubnetNum, SubnetMask, NextHop}) \\
\text{D1 = SubnetMask \& D} \\
\text{if D1 = SubnetNum} \\
\quad \text{if NextHop is an interface} \\
\quad \quad \text{deliver datagram directly to D} \\
\text{else} \\
\quad \quad \text{deliver datagram to NextHop} \\
\]

- Use a default router if nothing matches
- Not necessary for all 1s in subnet mask to be contiguous
- Can put multiple subnets on one physical network
- Subnets not visible from the rest of the Internet

### Supernetting

- Assign block of contiguous network numbers to nearby networks
- Called CIDR: Classless Inter-Domain Routing
- Represent blocks with a single pair \((\text{first_network_address, count})\)
- Restrict block sizes to powers of 2
- Use a bit mask (CIDR mask) to identify block size
- All routers must understand CIDR addressing

### Route Propagation

- Know a smarter router
  - hosts know local router
  - local routers know site routers
  - site routers know core router
  - core routers know everything
- Autonomous System (AS)
  - corresponds to an administrative domain
  - examples: University, company, backbone network
  - assign each AS a 16-bit number
- Two-level route propagation hierarchy
  - interior gateway protocol (each AS selects its own)
  - exterior gateway protocol (Internet-wide standard)