Lecture 11: Intra-Domain Routing

RIP (Routing Information Protocol) & OSPF (Open Shortest Path First)

IP Forwarding

- The Story So Far...
  - IP addresses are structured to reflect Internet structure
  - IP packet headers carry these addresses
  - When Packet Arrives at Router
    - Examine header to determine intended destination
    - Look up in table to determine next hop in path
    - Send packet out appropriate port
- This/next lecture
  - How to generate the forwarding table

Graph Model

- Represent each router as node
- Direct link between routers represented by edge
  - Symmetric links = undirected graph
  - Edge “cost” $c(x,y)$ denotes measure of difficulty of using link
    - delay, $\$, cost, or congestion level
- Task
  - Determine least cost path from every node to every other node
    - Path cost $d(x,y) = $ sum of link costs

Routes from Node A

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>E</td>
</tr>
</tbody>
</table>

- Properties
  - Some set of shortest paths forms tree
  - Shortest path spanning tree
  - Solution not unique
    - E.g., A-E-F-C-D also has cost 7
Ways to Compute Shortest Paths

- Centralized
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- Link-state
  - Every node collects complete graph structure
  - Each computes shortest paths from it
  - Each generates own routing table
- Distance-vector
  - No one has copy of graph
  - Nodes construct their own tables iteratively
  - Each sends information about its table to neighbors

Outline

- Distance Vector
- Link State
- Routing Hierarchy

Distance-Vector Method

- Idea
  - At any time, have cost/next hop of best known path to destination
  - Use cost $\infty$ when no path known
- Initially
  - Only have entries for directly connected nodes

Distance-Vector Update

- Update($x$, $y$, $z$)
  
  \[
  d \leftarrow c(x, z) + d(z, y)
  \]

  # Cost of path from $x$ to $y$ with first hop $z$

  if $d < d(x, y)$
    # Found better path
    return $d$, $z$
  # Updated cost / next hop
  else
    return $d(x, y)$, $\text{nexthop}(x, y)$

  # Existing cost / next hop
Algorithm

- Bellman-Ford algorithm
- Repeat
  For every node x
    For every neighbor z
      For every destination y
        $d(x,y) \leftarrow \text{Update}(x,y,z)$
- Until converge

**Start**

Optimum 1-hop paths

<table>
<thead>
<tr>
<th>Dist</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dist</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
</tr>
</tbody>
</table>

Optimum 2-hop paths

<table>
<thead>
<tr>
<th>Dist</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dist</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>F</td>
</tr>
</tbody>
</table>

Optimum 3-hop paths

<table>
<thead>
<tr>
<th>Dist</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dist</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>F</td>
</tr>
</tbody>
</table>
Distance Vector: Link Cost Changes

Link cost changes:
- Node detects local link cost change
- Updates distance table
- If cost change in least cost path, notify neighbors

“good news travels fast”

Distance Vector: Poison Reverse

If Z routes through Y to get to X:
- Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z)
- Immediate notification of unreachability, rather than split horizon timeout waiting for advertisement
- Will this completely solve count to infinity problem?
Poison Reverse Failures

Routing Information Protocol (RIP)

RIP Updates

RIP Staleness / Oscillation Control
Outline

- Distance Vector
- Link State
- Routing Hierarchy

Link State Protocol Concept

- Every node gets complete copy of graph
  - Every node "floods" network with data about its outgoing links
- Every node computes routes to every other node
  - Using single-source, shortest-path algorithm
- Process performed whenever needed
  - When connections die / reappear

Sending Link States by Flooding

- X Wants to Send Information
  - Sends on all outgoing links
- When Node Y Receives Information from Z
  - Send on all links other than Z

Dijkstra’s Algorithm

- Given
  - Graph with source node $s$ and edge costs $c(u,v)$
  - Determine least cost path from $s$ to every node $v$
- Shortest Path First Algorithm
  - Traverse graph in order of least cost from source
Dijkstra's Algorithm: Concept

- **Node Sets**
  - **Done**
    - Already have least cost path to it
  - **Horizon**
    - Reachable in 1 hop from node in Done
  - **Unseen**
    - Cannot reach directly from node in Done

- **Label**
  - \( d(v) \) = path cost from s to v
- **Path**
  - Keep track of last link in path

Dijkstra's Algorithm: Initially

- No nodes done
- Source in horizon

Dijkstra's Algorithm: Initially

- \( d(v) \) to node A shown in red
- Only consider links from done nodes

Dijkstra’s Algorithm

- Select node \( v \) in horizon with minimum \( d(v) \)
- Add link used to add node to shortest path tree
- Update \( d(v) \) information
Dijkstra’s Algorithm

- Repeat...

Dijkstra’s Algorithm

- Update d(v) values
  - Can cause addition of new nodes to horizon

Dijkstra’s Algorithm

- Final tree shown in green

Link State Characteristics

- With consistent LSDBs*, all nodes compute consistent loop-free paths
- Can still have transient loops

*Link State Data Base

Packet from C→A may loop around BDC if B knows about failure and C & D do not
**OSPF Routing Protocol**

- Open
  - Open standard created by IETF
- Shortest-path first
  - Another name for Dijkstra’s algorithm
- More prevalent than RIP

**OSPF Reliable Flooding**

- Transmit link state advertisements
  - Originating router
    - Typically, minimum IP address for router
  - Link ID
    - ID of router at other end of link
  - Metric
    - Cost of link
    - Link-state age
      - Incremented each second
      - Packet expires when reaches 3600
    - Sequence number
      - Incremented each time sending new link information

**OSPF Flooding Operation**

- Node X Receives LSA from Node Y
  - With Sequence Number q
  - Looks for entry with same origin/link ID

**Flooding Issues**

- When should it be performed
  - Periodically
  - When status of link changes
    - Detected by connected node
- What happens when router goes down & back up
  - Sequence number reset to 0
  - Other routers may have entries with higher sequence numbers
  - Router will send out LSAs with number 0
  - Will get back LSAs with last valid sequence number p
  - Router sets sequence number to p+1 & resends
Adoption of OSPF

- RIP viewed as outmoded
  - Good when networks small and routers had limited memory & computational power
- OSPF Advantages
  - Fast convergence when configuration changes

Comparison of LS and DV Algorithms

**Message complexity**
- **LS**: with n nodes, E links, \( O(nE) \) messages
- **DV**: exchange between neighbors only

**Speed of Convergence**
- **LS**: Relatively fast
  - Complex computation, but can forward before computation
  - may have transient loops
- **DV**: convergence time varies
  - may have routing loops
  - count-to-infinity problem
  - faster with triggered updates

**Space requirements:**
- **LS**: maintains entire topology
- **DV**: maintains only neighbor state

**Robustness: router malfunctions**
- **LS**: Node can advertise incorrect link cost
  - Each node computes its own table
- **DV**: Node can advertise incorrect path cost
  - Each node’s table used by others (error propagates)

Outline

- Distance Vector
- Link State
- Routing Hierarchy

Routing Hierarchies

- Flat routing doesn’t scale
  - Storage \( \rightarrow \) Each node cannot be expected to store routes to every destination (or destination network)
  - Convergence times increase
  - Communication \( \rightarrow \) Total message count increases
- Key observation
  - Need less information with increasing distance to destination
  - Need lower diameters networks
- Solution: area hierarchy
Areas

- Divide network into areas
  - Areas can have nested sub-areas
- Hierarchically address nodes in a network
  - Sequentially number top-level areas
  - Sub-areas of area are labeled relative to that area
  - Nodes are numbered relative to the smallest containing area

Routing Hierarchy

- Partition Network into “Areas”
  - Within area
    - Each node has routes to every other node
  - Outside area
    - Each node has routes for other top-level areas only
    - Inter-area packets are routed to nearest appropriate border router
  - Constraint: no path between two sub-areas of an area can exit that area

Area Hierarchy Addressing

Path Sub-optimality

- Can result in sub-optimal paths

3 hop red path vs. 2 hop green path
Next Lecture: BGP

- How to connect together different ISPs