Lecture 10: Intra-Domain Routing

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

IP Forwarding

- The Story So Far…
  - IP addresses are structure 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 \( d(x,y) = \text{sum of link costs} \)

Routes from Node A

- 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

<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>
Ways to Compute Shortest Paths

- **Centralized**
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables

- **Distance-vector**
  - No one has copy of graph
  - Nodes construct their own tables iteratively
  - Each sends information about its table to neighbors

- **Link-state**
  - Every node collects complete graph structure
  - Each computes shortest paths from it
  - Each generates own routing table

Outline

- **Distance Vector**
  - Link State
  - Routing Hierarchy

Distance-Vector Method

<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>∞</td>
<td>–</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>–</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>F</td>
</tr>
</tbody>
</table>

- **Initial Table for A**

Distance-Vector Update

- **Update(x,y,z)**
  d ← 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), 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) ← Update(x,y,z)
- Until converge
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: Link Cost Changes

Link cost changes:
- Good news travels fast
- Bad news travels slow - "count to infinity" problem!

Distance Vector: Split Horizon

If Z routes through Y to get to X:
- Z does not advertise its route to X back to Y

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)
- Eliminates some possible timeouts with split horizon
- Will this completely solve count to infinity problem?
Poison Reverse Failures

Routing Information Protocol (RIP)

- Earliest IP routing protocol (1982 BSD)
- Current standard is version 2 (RFC 1723) which includes CIDR
- Features
  - Every link has cost 1
  - "Infinity" = 16
  - Limits to networks where everything reachable within 15 hops
- Sending Updates
  - Every router listens for updates on UDP port 520
  - RIP message can contain entries for up to 25 table entries

RIP Updates

- Initial
  - When router first starts, asks for copy of table for every neighbor
  - Uses it to iteratively generate own table
- Periodic
  - Every 30 seconds, router sends copy of its table to each neighbor
  - Neighbors use to iteratively update their tables
- Triggered
  - When every entry changes, send copy of entry to neighbors
    - Except for one causing update (split horizon rule)
    - Neighbors use to update their tables

RIP Staleness / Oscillation Control

- Small Infinity
  - Count to infinity doesn’t take very long
- Route Timer
  - Every route has timeout limit of 180 seconds
    - Reached when haven’t received update from next hop for 6 periods
    - If not updated, set to infinity
  - Soft-state refresh ➔ important concept!!!
- Behavior
  - When router or link fails, can take minutes to stabilize
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


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


- 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...

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

- 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 (LSA)
  - 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
- Cases
  - No entry present
    - Add entry, propagate to all neighbors other than Y
  - Entry present with sequence number p < q
    - Update entry, propagate to all neighbors other than Y
  - Entry present with sequence number p > q
    - Send entry back to Y
    - To tell Y that it has out-of-date information
  - Entry present with sequence number p = q
    - Ignore it

**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 O(E)

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

Speed of Convergence
- **LS**: Complex computation
  - But...can forward before computation
  - may have oscillations
- **DV**: convergence time varies
  - may be routing loops
  - count-to-infinity problem
  - (faster with triggered updates)

Robustness: what happens if router malfunctions?
- **LS**: node can advertise incorrect link cost
  - each node computes only its own table
- **DV**: DV node can advertise incorrect path cost
  - each node’s table used by others
  - errors propagate thru network
- Other tradeoffs
  - Making LSP flood reliable

Outline

- Distance Vector
- Link State
- Routing Hierarchy
Routing Hierarchies

- Flat routing doesn’t scale
  - Storage: Each node cannot be expected to store routes to every destination (or destination network)
  - Convergence times increase
  - Communication: 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
Path Sub-optimality

- Can result in sub-optimal paths

Next Lecture: BGP

- How to connect together different ISPs

RIP Table Processing

- RIP routing tables managed by application-level process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated

EXTRA SLIDES

The rest of the slides are FYI
Dijkstra’s Algorithm

1. Initialization:
   1. $N = \{A\}$
   2. for all nodes $v$
   3. if $v$ adjacent to $A$
   4. then $D(v) = c(A,v)$
   5. else $D(v) = \infty$

2. Loop
   1. find $w$ not in $N$ such that $D(w)$ is a minimum
   2. add $w$ to $N$
   3. update $D(v)$ for all $v$ adjacent to $w$ and not in $N$:
      1. $D(v) = \min(D(v), D(w) + c(w,v))$
      2. /* new cost to $v$ is either old cost to $v$ or known shortest path cost to $w$ plus cost from $w$ to $v$ */
   4. until all nodes in $N$

Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>start N</th>
<th>D(B),p(B)</th>
<th>D(C),p(C)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2,A</td>
<td>5,A</td>
<td>1,A</td>
<td>\infty</td>
<td>\infty</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2,A</td>
<td>4,D</td>
<td>2,D</td>
<td>\infty</td>
<td>\infty</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2,A</td>
<td>3,E</td>
<td>4,E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3,E</td>
<td>4,E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>