6.033 - Routing
Lecture #9
Katrina LaCurts, lacurts@mit.edu

0. Intro
   - Today: link-state and distance-vector routing
   - Neither of these routing protocols is used to route across the entire Internet; let's find out why.

1. Routing protocols in general
   - Goal: For every node X, after the routing protocol is run, X's routing table should contain a *minimum-cost route* to every other reachable node.
   - Route vs. path
     - Path = full path packets will travel
     - Route = first hop of that path. Node really only needs to know the first hop
   - Link costs
     - Link costs can represent many things: delay, congestion, etc. Sometimes all link costs are just 1, so that the min-cost paths are the paths with the fewest hops.
     - Link costs can change
   - Once a routing table is set up, when a switch gets a packet, it can check the packet header for the destination address, look that address up in its routing table, and add the packet to the queue for that outgoing link.

2. Distributed routing protocols
   - Scale better than centralized ones
   - Three steps:
     1. Nodes learn about their neighbors via the HELLO protocol
     2. Nodes learn about other reachable nodes via advertisements
     3. Nodes determine minimum-cost routes
   - All of these steps happen periodically, which lets routing protocols adapt as link costs change, as advertisements get lost, as links fail, as nodes fail, etc.
     - HELLO protocol lets nodes discover link/node failures

3. Link-state Routing
   - Main idea: through advertisements, nodes disseminate information about the topology of the graph to all other nodes. Once all nodes have that information, they can run a shortest-path algorithm
   - Advertisement format: Each node's advertisement is a list of its neighbors and its link costs to those neighbors.
   - Each node sends advertisements to their neighbors, who forward to their neighbors, who forward to their neighbors, ... I.e., advertisements are flooded
   - After flooding, each node should have a complete map of the network (except in the case of very specific, very rare sequences

of advertisement loss). Nodes use this map to run Dijkstra's shortest path algorithm.

- Here's one way to do Dijkstra's algorithm, slightly different from the example I did in class (below, we assume that all nodes are initially known; you can make this a nodes-need-to-be-discovered based implementation with a few small changes).
- In each step of Dijkstra's algorithm, we keep track of W, the set of nodes we haven't processed yet. Initially, W is all nodes in the network.
- We also keep track of the current costs and routes to all of the nodes. Initially:

```
  routing_table[self] = Self; routing_table[anyone else] = ?
  cost_table[self] = 0; cost_table[anyone else] = infinity

While W is not empty:
1. u = the node in W we have the minimum cost to so far
2. Remove u from W
3. For every neighbor v of u:
  d = cost_table[u] + cost(u, v)
  if d < cost_table[v]
      cost_table[v] = d
      routing_table[v] = routing_table[u]
```

- Good things: Flooding makes link-state extremely robust to failure. Very unlikely that a node will miss so many advertisements that it has an incorrect view of the network.
- Bad thing: overhead of flooding is overwhelming. N nodes, L links => ~2NL advertisements total.

4. Distance-vector Routing

- In link-state, nodes calculate full shortest paths. They really only need the route (first-hop) to a destination. Let's exploit that.
- Advertisement format: Each node's advertisement is a lists of all the nodes it knows about, and its current costs to those nodes. Initially, this advertisement is just [(self, 0)].
- Nodes who receive an advertisement: Node X's advertisement will be received only by its neighbors.
- Integrate step: When node X receives an advertisement from its neighbor Y, this advertisement will be a list of [(dst, cost)] pairs. Each cost represents Y's cost to dst.

  For each (dst, cost) in the advertisement, X needs to check for two things:
  - If X is already using Y to get to dst, update the cost information (remember, costs can change!)
  - If X is not already using Y to get to dst, see if Y could provide a better path; if so, update the routing and cost information

- Good thing: Overhead is much better. N nodes, L links => 2L advertisements, not 2NL as in link-state
- Bad thing: Counting to INFINITY
  - When a node A has no route to destination B, it will advertise a cost of INFINITY to B. A cost of INFINITY B is interpreted as there being no route to B. So INFINITY must be larger than the longest path in the network.
    - But because the order in which advertisements are sent matters, sometimes nodes can incorrectly think there's a route when there isn't one. This can last for up to INFINITY steps (see slides)
    - Ironically, INFINITY is typically around 16 or 32

5. Summary
   - Link-state routing: nice, easy to reason about failures, but overhead prevents it from scaling. In practice, good for MIT-sized networks.
   - Distance-vector routing: significantly less overhead, but harder to reason about failures. In practice, good for very small networks (where we can reason about properties of routing loops)
   - As of right now, we can route data within MIT (via link-state), but not outside of MIT.