The Network Layer (Part II):
Routing Protocols

Lecture 23
6.02 Fall 2007
December 3, 2007

Hari Balakrishnan

Layering in the Internet

HTTP, FTP, SMTP, ...
Application

TCP, UDP
Transport

Ethernet, WiFi, ...
Data Link

End-to-End Layer
Reliability, integrity, ordering, jitter ctrl, congestion response, ...

Forwarding & routing (and addressing)
Framing, coding, [limited] rxmits, channel access Modulation/demodulation

Today
IP
Network
Network Layer Functions

- **Main goal**: ensure best-effort end-to-end connectivity
- **Addressing**: How to name nodes?
- **Forwarding**: What should each switch do to each packet?
- **Routing**: How to build routing tables to ensure that forwarding is correct?
  - Link-state protocols
  - Vector routing protocols

Link-State Routing

- Conceptually, a two-step process
- Each node disseminates information about its links to its neighbors; they re-send to their neighbors, etc.
  - This process allows each node to discover every other node and link
- Each node then runs the same local shortest path computation over its version of graph
  - If each node implements computation correctly and each node has the same graph, then resulting forwarding will work correctly
  - Each node makes *local* decision about next hop
- Distributed dissemination, centralized computation (at each node, but each node has entire graph)
First Step: Disseminating Link-State

- Each node sends periodic HELLO to all its neighbors
- Periodically send LSA (Link-State Advertisement) [node seq#, {<nbhr1, cost1>, <nbhr2, cost2>, ...}] to all neighbors
- If seq > last_heard, incorporate info in local version of graph, and rebroadcast LSA to your neighbors
  - LSAs aren’t reliable messages, so periodic
- Periodic messages help handle dynamism: state in each node is “soft” and times out if not refreshed

Second Step: Shortest Path Computation

- Many algorithms (Dijkstra’s in particular)
- Key property of shortest paths:
  
  Suppose shortest path from X to Y goes through Z. Then, the path from X to Z must be a shortest path.
Greedy Path Computation

- Find paths to all other nodes in order of non-decreasing path cost
  - Start with link with smallest cost (clearly no better way to get to that neighbor node)
  - Maintain nodes_done list: nodes for which we’ve found shortest path
  - Maintain spcost(v), shortest path cost to node v
- Consider all nodes z not in nodes_done that are connected to some node y in nodes_done
  - Find \( z^* = \text{MIN} \) (over z) of \( \text{spcost}(y) + \text{cost}(y,z) \)
  - Add \( z^* \) to nodes_done and set spcost(z*)
  - Continue until all nodes are in nodes_done

Vector Routing Protocols

- Distributed route computation
  - Each node sends periodic route advertisements about its best path to all destinations to neighbors
  - Upon receiving info from neighbor, update best path info if necessary
  - Send update to neighbors

- What information must advertisement contain?
  - At least cost to destination, to help neighbors update their best path info
  - Protocols differ in what else is in a route advertisement
Distance Vector Routing

- Each node periodically announces a vector of 
  &lt;destination:cost&gt; pairs to all its neighbors
- On hearing advertisement, run “integration step”: 
  if (current cost to dest &gt; cost in advertisement) then 
    update cost, nexthop /* Bellman-Ford algorithm */

Distance Vector: Pros and Cons

- + Simple protocol
- + Works well for small networks
- - Works only on small networks

Suppose link AC fails. When A discovers failure, it sends E: cost = INFINITY to B. B sends ‘E: cost=2’ to A. A installs E: cost=3 in table.
Now suppose link BD fails. B discovers it, then installs E: cost = INFINITY in its table.
Sends info to A, A installs E: cost = INFINITY in its table.

But what if now A sends advert to B before B sends advert to A?
Distance Vector Cons (Cont.)

- Solving counting to infinity problem needs “infinity” to be small
  - Otherwise, convergence time too long

- One approach: split horizon advertisements
  - Don’t advertise route to X if you got the installed route from X
  - Only works for avoiding two-hop routing loops, though
  - Longer loops require a different approach

- Path vector routing
  - Send path, in addition to cost to destination, in each advertisement

Path Vector Routing

- For each advertisement, run “integration step”
  - E.g., pick shortest, cheapest, quickest, etc.
- Ignore advertisements with own address in path vector
  - Avoids long-running routing loops that “count to infinity”
Path Vector Routing

Path Vector Pseudocode
(Shortest-path Routing)

path $\leftarrow$ NULL; // path is current path to destination

procedure n.ADVERTISE() // ADVERTISE is called every T seconds
    for each link do send(n | path); // prepend n to path

procedure n.INTEGRATE(p) // just heard a route p to destination
    if n is in p then return;
    else if ( path == NULL OR
              first_hop(p) == first(path) OR
              length(p) < length(path) )
        path $\leftarrow$ p;

procedure n.TIMER() // called periodically to check nbhr liveness
    if haveNotHeardFrom(first(path)) then path $\leftarrow$ NULL

For each advertisement, run “integration step”
- E.g., pick shortest, cheapest, quickest, fattest, etc.
- Ignore advertisements with own address in path vector
Summary

- The network layer is the “glue layer” that achieves network-wide connectivity
- Forwarding: What a switch does on each packet
- Routing: How the switches build tables to help forwarding
- Next lecture: Reliable data delivery
- And then: Network scalability (engineering very large networks)