Overview

- Routing Tables & static routing
- Dynamic routing (inter- and intra-domain)
- Distance vector vs Link state routing
- RIP, RIPv2
- OSPF
- Refs: Chap 9, 10.
- Books: "Routing in Internet" by Huitema, "Interconnections" by Perlman

Routing vs Forwarding

- Fig 9.1
- Routing table used by IP forwarding. Can display routing table using command "netstat -rn"
- Route Table setup by:
  a) 'route' command
  b) routing daemon (eg: 'routed')
  c) ICMP redirect message.
Routing Table structure

- **Fields:** destination, gateway, flags, ...
- **Destination:** can be a host address or a network address. If the ‘H’ flag is set, it is the host address.
- **Gateway:** router/next hop IP address. The ‘G’ flag says whether the destination is directly or indirectly connected.
- **U** flag: Is route up?
- **G** flag: router (indirect vs direct)
- **H** flag: host (dest field: host or n/w address?)

Static Routing

- **Upon booting,** default routes initialized from files. Eg: /etc/rc.net in AIX, /etc/netstart in BSD, /etc/rc.local in SUN/Solaris
- **Use ‘route’ command to add new routes** eg: route add default sun 1
- **ICMP redirect:** sent to host by router when a “better” router exists on the same subnet.
- **Alt:** router discovery ICMP messages
  - **Router solicitation request from host**
  - **Router advertisement messages from routers**

Dynamic routing

- Internet organized as “autonomous systems” (AS).
- **Interior Gateway Protocols (IGPs)** within AS. Eg: RIP, OSPF, HELLO
- **Exterior Gateway Protocols (EGPs)** for AS to AS routing. Eg: EGP, BGP-4
- **Reality:** most of internet uses default routes (which is allowed within dynamic routing). Serious dynamic routing starts near core of AS and from one AS to another.
Dynamic routing methods

- Source-based: chart route at source.
- Link state routing: Get map of network (in terms of link states) and calculate best route (but specify only a signpost: i.e. the next-hop)
- Distance vector: Set up signposts to destinations looking at neighbors’ signposts.
- Key: to make it a “distributed” algorithm?

Distance Vector routing

- “Vector” of distances (signposts) to each possible destination at each router.
- How to find distances?
  - Distance to local network is 0.
  - Look in neighbors’ distance vectors, and add link cost to reach the neighbor
  - Find which direction yields minimum distance to particular destination. Turn signpost that way.
  - Keep checking if neighbors change their signposts and modify local vector if necessary.
  - And that’s it!
  - Called the “Bellman-Ford algorithm”

Routing Information Protocol

- Uses hop count as metric
- Tables (vectors) “advertised” to neighbors every 30 s.
  - Robustness: Entries reinitialized (as 16 or infinity) if no refresh for 180 s.
  - Efficiency: Triggered updates used to inform neighbors when table changes.
- Protocol details:
  - Runs over UDP.
  - Init: send request message asking for vectors
  - Format can carry upto 25 routes (within 512 bytes)
  - RIPv1 does not carry subnet masks => many networks use default of 255.255.255.0
RIP problems

- Counting-to-infinity problem:
  - Simple configuration A->B->C. If C fails, B needs to update and thinks there is a route through A. A needs to update and thinks there is a route thru B.
  - No clear solution, except to set “infinity” to be small (e.g., 16 in RIP)
  - Split-horizon: If A’s route to C is thru B, then A advertises C’s route (only to B) as infinity.

- Slow convergence after topology change:
  - Due to count to infinity problem
  - Also information cannot propagate thru node until it recalculates routing info.

RIP problems (contd)

- Black-holes:
  - If one node goes broke and advertises route of zero to several key networks, all nodes immediately point to it.

- How to install a fix in a distributed manner??
  - Require protocol to be “self-stabilizing” i.e., even if some nodes are faulty, once they are isolated, the system should quickly return to normal operation

- Broadcasts consume non-router resources
- Does not support subnet masks (VLSMs)
- No authentication

RIPv2

- Why? Installed base of RIP routers
- Provides:
  - VLSM support
  - Authentication
  - Multicasting
  - “Wire-sharing” by multiple routing domains,
  - Tags to support EGP/BGP routes.
- Uses reserved fields in RIPv1 header.
- First route entry replaced by authentication info.
Link State protocols

- Create a network “map” at each node.
  - For a map, we need inks and attributes (link states), not of destinations and metrics (distance vector)
  - 1. Node collects the state of its connected links and forms a “Link State Packet” (LSP)
  - 2. Broadcast LSP => reaches every other node in the network.
  - 3. Given map, run Dijkstra’s shortest path algorithm => get paths to all destinations
  - 4. Routing table = next hops of these paths.

Dijkstra’s algorithm

- A.k.a “Shortest Path First” (SPF) algorithm.
- Idea: compute shortest path from a “root” node to every other node: “Greedy method”:
  - $P$ is a set of nodes for which shortest path has already been found.
  - For every node $o$ outside $P$, find shortest one-hop path from some node in $P$.
  - Add that node “o” which has the shortest of these paths to $P$. Record the path found.
  - Continue till we add all nodes (&paths) to $P$

- $P$: (ID, path-cost, next-hop) triples.
  - ID: node id.
  - Path-cost: cost of path from root to node
  - Next-hop: ID of next-hop on shortest path from the root to reach that node
  - $P$: Set of nodes for which the best path cost (and next-hop from root) have been found.

- $T$: (ID, path-cost, next-hop): Set of candidate nodes at a one-hop distance from some node in $P$.
  - Note: there is only one entry per node. In the interim, some nodes may not lie in $P$ or $T$.

- $R$: Routing table: (ID, next-hop) to be created
Dijkstra’s algorithm

1. Put root i.e., (myID, 0, 0) in P & (myID, 0) to R.
2. If node N is just put into P, look at N’s links (i.e. its LSP).
   a. For each link to neighbor M, add cost of the root-to-N path to the cost of the N-to-M link (from LSP) to determine a new cost: C.
   b. The “next-hop” corresponds to the next-hop ID in N’s tuple (or N if M is the root itself):
   c. If M not in T (or P) with better path cost, add (M, C, h) to T.
3. If T = empty, terminate. Else, move the min-cost triple from T to P, and add (M, h) to R. Go to step 2.

Topology dissemination

aka LSP distribution

1. Flood LSPs on links except incoming link
   a. Require at most 2E transfers for n/w with E edges
2. Sequence numbers to detect duplicates
   a. Why? Routers/links may go down/up
   b. Problem: wrap-around ⇒ have large seq # space
3. Age field (similar to TTL)
   a. Periodically decremented after acceptance
   b. Zero ⇒ discard LSP & request everyone to do so
   c. Router awakens ⇒ knows that all its old LSPs would have been purged and can choose a new initial sequence number

Link state vs Distance vector

Advantages:
- More stable (aka fewer routing loops)
- Faster convergence than distance vector
- Easier to discover network topology, troubleshoot network.
- Can do better source-routing with link-state
- Type & Quality-of-service routing (multiple route tables) possible
- Caveat: With path-vector-type distance vector routing, these arguments don’t hold
**OSPF**
- OSPF runs directly on top of IP (not over UDP)
- It can calculate a separate set of routes for each IP type of service (⇒ multiple routing entries)
- Dimensionless cost (eg: based on throughput, delay)
- Load balancing: distributing traffic equally among routes
- Supports VLSMs: subnet mask field in header
- Supports multicasting, authentication, unnumbered networks (point-to-point).

**Summary**
- Route Tables
- Distance vector, RIP, RIPv2
- Link state, OSPF.