Activity – Routing Algorithms/Protocols

Implementation
Initialization
- Select a person for each router for each exercise
- Select Floater (Professor)
- Write “Forwarding Table” and “Router i” across the board for routers
- Initialize forwarding table based on direct link costs.

General Procedures and Roles
- When a router gets new advertisement, router updates its table and re-advertises if necessary.
- Process updates until the no new update message.
- Floater might generate new actions as the network is stabilizing.
- During all actions, Floater stops routers as needed to explain process. After two or three trials, Floater calls on audience members to explain what just happened, or what would happen next given the present condition.
- During all actions, audience must graph the routing and record updates of forwarding tables.

Exercise 1 – Routing Information Protocol

- Distance vector algorithm
  - *Distance metric*: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message
  - *Each advertisement*: list of up to 25 destination subnets (in IP addressing sense)

- $D_x(y) =$ estimate of least cost from $x$ to $y$
  - $x$ maintains distance vector $D_x = [D_x(y): y \in N ]$

- Node $x$:
  - Knows cost to each neighbor $v$: $c(x,v)$
  - Maintains its neighbors’ distance vectors.
    - For each neighbor $v$, $x$ maintains $D_v = [D_v(y): y \in N ]$
  - When $x$ receives new DV estimate from neighbor, it updates its own DV using Belmond-Ford equation:
    - $D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\}$ for each node $y \in N$
Exercise 2 – Hierarchical – OSPF (*Open Shortest Path First*)

- Uses link state algorithm
  - Route computation using Dijkstra’s algorithm
- Multiple same-cost paths allowed
- *Two-level hierarchy*: local area, backbone.
  - Link-state advertisements only in area
  - Each node has detailed area topology;
    - Only know direction (shortest path) to nets in other areas.
- *Area border routers*: “summarize” distances to networks in own area, advertise to other Area Border routers.
- *Backbone routers*: run OSPF routing limited to backbone.

**Dijkstra’s Algorithm**

```plaintext
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5        then D(v) = c(u,v)
6      else D(v) = \infty
7
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N:
12     D(v) = \min( D(v), D(w) + c(w,v) )
13     /* new cost to v is either old cost to v or known
14        shortest path cost to w plus cost from w to v */
15   until all nodes in N'
```
Exercise 3 – *mcast forwarding tree*
- Tree of shortest path routes from source to all receivers
  - Dijkstra’s algorithm

Exercise 4 – *Controlled flooding (broadcast)*
- Node only broadcasts packet if it hasn’t broadcast the same packet before
- Node keeps track of packet ids already broadcasted

Exercise 5 – *DVMRP (Distance Vector Multicast Routing Protocol)*
- *Flood and prune*: reverse path forwarding, source-based tree
  - *Reverse path forwarding (RPF)*: only forward packet if it arrived on shortest path between node and source
  - RPF tree based on DVMRP’s own routing tables
  - Initial datagram to mcast group flooded everywhere via RPF
  - Routers not wanting group messages: send upstream *prune* msgs

Exercise 6 – *Center-based trees (multicast):*
- Single delivery tree shared by all
- One router identified as “center” of tree
- To join:
  - Edge router sends unicast *join-msg* addressed to center router
  - *join-msg* “processed” by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - Path taken by *join-msg* becomes new branch of tree for this router

Exercise 7 – *PIM (Protocol Independent Multicast) – sparse mode:*
- No membership until routers explicitly join
- Receiver-driven construction of mcast tree
  - For instance, center-based
    - Router sends *join msg* to rendezvous point (RP)
    - Intermediate routers update state and forward join
      - *join-msg* either hits existing branch for this RP, or arrives at RP
  - *sender(s):* unicast data to RP, which distributes down RP-rooted tree

*dense mode:*
- Group membership by routers assumed until routers explicitly prune
- Data-driven construction on mcast tree (e.g., RPF)