Short review of routing algorithm types

- There are two general classes of routing algorithms
  - Distance Vector (RIP1, RIP2, BGP*)
  - Link State (OSPF, IS-IS)

*BGP uses a variant of the DV algorithm called the Path Vector algorithm

Distance Vector Routing

- The idea:
  - Each router knows the cost to each of its immediate neighbors
  - Each router builds a “distance vector” that contains the total cost of the best-known route to every destination (initial costs = ∞)
  - At intervals, each router sends its DV to all neighbors
  - When a router R receives a DV from a neighbor N, R scans the table to see if there are any cases where, for Destination D:
    \[ N's \text{ cost to get to } D + R's \text{ cost to get to } N < R's \text{ current cost to get to } D \]
    If there are any such cases, R updates its table so that future traffic for D is sent to N.

Example:
- Assume router R knows a path to D with a cost of 25.
- R’s neighbor N knows a path to D with a cost of 20.
- If the cost from R to N is 3, then R can get to D through N with a total cost of 23.
- Since this is less than the current cost, R will update its routing table so that all traffic for D goes to N.
A Distance Vector example

- The tables below show each router’s DV after all routers have exchanged DVs.
- Bold entries show changes since last exchange

<table>
<thead>
<tr>
<th>Initial</th>
<th>After first exchange</th>
<th>After second exchange</th>
<th>After third exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>0-</td>
<td>99-</td>
<td>99-</td>
</tr>
<tr>
<td>y</td>
<td>99-</td>
<td>99-</td>
<td>0-</td>
</tr>
<tr>
<td>z</td>
<td>99-</td>
<td>99-</td>
<td>99-</td>
</tr>
<tr>
<td>x</td>
<td>99-</td>
<td>99-</td>
<td>99-</td>
</tr>
</tbody>
</table>

```
"Router x knows a path to A through router w with cost 1."
```

The Link State routing algorithm

- The idea:
  - Each router discovers cost to immediate neighbors
  - At intervals, this info is flooded to all other routers in a "Link State Packet". This gives all routers a map of the network and link costs.
  - Each router runs a part-finding algorithm (e.g, Dijkstra’s Shortest Path Algorithm) to calculate least-cost paths.
Dijkstra’s Shortest Path Algorithm

To find the least-cost path from node S to node D:

1. Call S the “working node”. Tentatively label all other nodes (∞, -).
2. Tentatively label each neighbor of the working node (W, x) where W is the name of the working node and x is the total cost from the source node.
3. Examine the entire graph and find the tentatively-labeled node with the smallest cost in its label. Change this to the permanent label of the node. This is the new working node.
4. Repeat 2-4 until D is permanently labeled
5. Record the name of D. Call D the copying node.
6. Record the name W, where (W,x) is the label of the copying node.
7. W is the new copying node. If W <> S, repeat 5-6.
8. The least-cost path is the reverse order of the recorded node names.

Dijkstra’s Shortest Path Example

Find the least-cost path from A (w) to D (z)

w is the starting node
Calc total costs to x and y. x is low cost, so it is new working node.
The path to y through x is lower cost, so we replace y’s label.
z is permanently labeled, so stop

Recorded labels are: z y x w, so the path is w x y z