Dijkstra's Shortest-Path Algorithm

- Iterative algorithm
  - After k iterations, know least-cost path to k nodes
- S: nodes whose least-cost path definitively known
  - Initially, S = {u} where u is the source node
  - Add one node to S in each iteration
- D(v): current cost of path from source to node v
  - Initially, D(v) = c(u,v) for all nodes v adjacent to u
  - ... and D(v) = \infty for all other nodes v
  - Continually update D(v) as shorter paths are learned

Dijkstra's Algorithm Example

Dijkstra's Algorithm

1. Initialization:
   1. S = {u}
   2. for all nodes v
   3. if v adjacent to u {
   4. D(v) = c(u,v)
   5. else D(v) = \infty
   6. Loop
   7. find w not in S with the smallest D(w)
   8. add w to S
   9. update D(v) for all v adjacent to w and not in S:
   10. D(v) = min[D(v), D(w) + c(w,v)]
11. until all nodes in S

Shortest-Path Tree

- Shortest-path tree from u
- Forwarding table at u

<table>
<thead>
<tr>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v (u,v)</td>
</tr>
<tr>
<td>w (u,w)</td>
</tr>
<tr>
<td>x (u,w)</td>
</tr>
<tr>
<td>y (u,v)</td>
</tr>
<tr>
<td>z (u,v)</td>
</tr>
<tr>
<td>s (u,w)</td>
</tr>
<tr>
<td>t (u,w)</td>
</tr>
</tbody>
</table>
Distance Vector Algorithm

• \( c(x,v) \) = cost for direct link from \( x \) to \( v \)
  – Node \( x \) maintains costs of direct links \( c(x,v) \)

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

• Node \( x \) maintains its neighbors’ distance vectors
  – For each neighbor \( v \), \( x \) maintains \( D_v = [D_v(y) : y \in N] \)

• Each node \( v \) periodically sends \( D_v \) to its neighbors
  – And neighbors update their own distance vectors
  \( D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \) for each node \( y \in N \)

• Over time, the distance vector \( D_x \) converges