What is Routing?

- To ensure information is delivered to the correct destination at a reasonable level of performance
- Forwarding
  - Given a forwarding table, move information from input ports to output ports of a router
  - Local mechanical operations
- Routing
  - Acquires information in the forwarding tables
  - Requires knowledge of the network
  - Requires distributed coordination of routers

Viewing Routing as a Policy

- Given multiple alternative paths, how to route information to destinations should be viewed as a policy decision
- What are some possible policies?
  - Shortest path (RIP, OSPF)
  - Most load-balanced
  - QoS routing (satisfies app requirements)
  - etc

A
E
D
C
B
F
2
2
1
3
1
2
5
3
5
**Internet Routing**

- Internet topology roughly organized as a two level hierarchy
- First lower level – autonomous systems (AS’s)
  - AS: region of network under a single administrative domain
- Each AS runs an intra-domain routing protocol
  - Distance Vector, e.g., Routing Information Protocol (RIP)
  - Link State, e.g., Open Shortest Path First (OSPF)
- Possibly others
- Second level – inter-connected AS’s
  - Between AS’s runs inter-domain routing protocols, e.g., Border Gateway Routing (BGP)

- De facto standard today, BGP-4

---

**Example**

---

**Why Need the Concept of AS or Domain?**

- Routing algorithms are not efficient enough to deal with the size of the entire Internet
- Different organizations may want different internal routing policies
- Allow organizations to hide their internal network configurations from outside
- Allow organizations to choose how to route across multiple organizations (BGP)

- Basically, easier to compute routes, more flexibility, more autonomy/independence
Outline

• Two intra-domain routing protocols
• Both try to achieve the “shortest path” routing policy
• Quite commonly used
• OSPF: Based on Link-State routing algorithm
• RIP: Based on Distance-Vector routing algorithm

Intra-domain Routing Protocols

• Based on unreliable datagram delivery

• Distance vector
  – Routing Information Protocol (RIP), based on Bellman-Ford algorithm
  – Each neighbor periodically exchange reachability information to its neighbors
  – Minimal communication overhead, but it takes long to converge, i.e., in proportion to the maximum path length

• Link state
  – Will not cover; read book

Routing on a Graph

• Goal: determine a “good” path through the network from source to destination
  – Good often means the shortest path

• Network modeled as a graph
  – Routers → nodes
  – Link → edges
  – Edge cost: delay, congestion level, …
Distance Vector Routing (RIP)

- **What is a distance vector?**
  - Current best known cost to get to a destination
- **Idea:** Exchange distance vectors among neighbors to learn about lowest cost paths

<table>
<thead>
<tr>
<th>Node C</th>
<th>Dest.</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>3</td>
</tr>
</tbody>
</table>

Note no vector entry for C itself

At the beginning, distance vector only has information about directly attached neighbors, all other dests have cost $\infty$

Eventually the vector is filled

---

Distance Vector Routing Algorithm

- **Iterative:** continues until no nodes exchange info
- **Asynchronous:** nodes need not exchange info/iterate in lock steps
- **Distributed:** each node communicates only with directly-attached neighbors
- **Each router maintains**
  - Row for each possible destination
  - Column for each directly-attached neighbor to node
  - Entry in row Y and column Z of node X: best known distance from X to Y, via Z as next hop
- **Note:** for simplicity in this lecture examples we show only the shortest distances to each destination

---

Distance Vector Routing

- Each local iteration caused by:
  - Local link cost change
  - Message from neighbor: its least cost path change from neighbor to destination
- Each node notifies neighbors only when its least cost path to any destination changes
  - Neighbors then notify their neighbors if necessary

Each node:

wait for (change in local link cost or msg from neighbor)

recompute distance table

if least cost path to any dest has changed, notify neighbors
Distance Vector Algorithm (cont’d)

1 Initialization:
2 for all nodes \( V \) do
3 \( \text{if } \text{V adjacent to } A \)
4 \( \text{D}(A, \text{V}, \text{V}) = c(A, \text{V}); \) /* Distance from A to V via neighbor V */
5 else
6 * \( \text{D}(A, \text{V}, \ast) = \infty; \)
7 loop:
8 wait (until A sees a link cost change to neighbor V or until A receives update from neighbor V)
9 \( \text{if } (c(A, \text{V}) \text{ changes by } d) \)
10 for all destinations \( Y \) through \( \text{V} \)
11 \( \text{D}(A, Y, \text{V}) = \text{D}(A, Y, \text{V}) + d \)
12 else if (update \( \text{D}(\text{V}, Y) \) received from \( \text{V} \)) /* shortest path from \( \text{V} \) to some \( Y \) has changed */
13 \( \text{D}(A, Y, \text{V}) = c(A, \text{V}) + \text{D}(\text{V}, Y); \)
14 \( \text{if } (\text{there is a new minimum for destination } Y) \)
15 \( \text{send } \text{D}(A, Y) \text{ to all neighbors} \) /* \( \text{D}(A, Y) \text{ denotes the min } \text{D}(A, Y, \ast) */
16 forever

Example: Distance Vector Algorithm

<table>
<thead>
<tr>
<th>Node A</th>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>7</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>( \infty )</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node B</th>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node C</th>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node D</th>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( \infty )</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Example: 1st Iteration (C \( \rightarrow \) A)

<table>
<thead>
<tr>
<th>Node A</th>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>7</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>8</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node B</th>
<th>Dest.</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

D(A,D,C) = c(A, C) + D(C,D) = 7 + 1 = 8
(D(A,C), D(A,B), D(A,D))
Example: 1st Iteration (B→A, C→A)

```
Node A
Dest. | Cost | NextHop
B    | 2    | B
C    | 3    | B
D    | 5    | B

Dest. | Cost | NextHop
B    | 2    | A
C    | 3    | B
D    | 5    | B

D(A,D,B) = c(A,B) + D(B,D) = 2 + 3 = 5
D(A,C,B) = c(A,B) + D(B,C) = 2 + 1 = 3
```

Example: End of 1st Iteration

```
Node A
Dest. | Cost | NextHop
B    | 2    | B
C    | 3    | B
D    | 5    | B

Dest. | Cost | NextHop
B    | 2    | A
C    | 3    | B
D    | 5    | B

D(A,D,B) = c(A,B) + D(B,D) = 2 + 3 = 5
D(A,C,B) = c(A,B) + D(B,C) = 2 + 1 = 3
```

Example: End of 2nd Iteration

```
Node A
Dest. | Cost | NextHop
B    | 2    | B
C    | 3    | B
D    | 5    | B

Dest. | Cost | NextHop
B    | 2    | A
C    | 3    | B
D    | 5    | B

D(A,D,B) = c(A,B) + D(B,D) = 2 + 3 = 5
D(A,C,B) = c(A,B) + D(B,C) = 2 + 1 = 3
```

Alan Mislove
amislove@ccs.neu.edu
Northeastern University
### Example: End of 3rd Iteration

Node A

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>B</td>
</tr>
</tbody>
</table>

Node B

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>C</td>
</tr>
</tbody>
</table>

Node C

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
</tr>
</tbody>
</table>

Node D

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
</tbody>
</table>

Nothing changes \(\rightarrow\) algorithm terminates

### Distance Vector: Link Cost Changes

7 \(\Delta\) loop:
8 \(\Delta\) wait (until \(A\) sees a link cost change to neighbor \(V\))
9     or until \(A\) receives update from neighbor \(V\)
10 \(\Delta\) if \(\Delta A, V\) changes by \(d\)
11     for all destinations \(Y\) through \(V\) do
12         \(D(A, Y, V) = D(A, Y, V) + d\);
13 \(\Delta\) else if \(update\ D(V, Y)\) received from \(V\)
14     \(D(A, Y, V) = c(A, V) + D(V, Y)\);
15 \(\Delta\) if (there is a new minimum for destination \(Y\))
16     \(send\ D(A, Y)\) to all neighbors
17 \(\Delta\) forever

Algorithm terminates

### Distance Vector: Count to Infinity

**Problem**

7 \(\Delta\) loop:
8 \(\Delta\) wait (until \(A\) sees a link cost change to neighbor \(V\))
9     or until \(A\) receives update from neighbor \(V\)
10 \(\Delta\) if \(\Delta A, V\) changes by \(d\)
11     for all destinations \(Y\) through \(V\) do
12         \(D(A, Y, V) = D(A, Y, V) + d\);
13 \(\Delta\) else if \(update\ D(V, Y)\) received from \(V\)
14     \(D(A, Y, V) = c(A, V) + D(V, Y)\);
15 \(\Delta\) if (there is a new minimum for destination \(Y\))
16     \(send\ D(A, Y)\) to all neighbors
17 \(\Delta\) forever

Algorithm terminates

**Link cost changes here**

Node B

\[\text{DCN} \quad \text{DCN} \quad \text{DCN} \quad \text{DCN} \quad \text{DCN}\]

Node C

<table>
<thead>
<tr>
<th>(\Delta)</th>
<th>(\Delta)</th>
<th>(\Delta)</th>
<th>(\Delta)</th>
<th>(\Delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

**“bad news travels slowly”**

Link cost changes here: recall that \(B\) also maintains shortest distance to \(A\) through \(C\), which is \(6\). Thus \(D(B, A)\) becomes \(6\)!
Distance Vector: Poisoned Reverse

- If C routes through B to get to A:
  - C tells B its (C's) distance to A is infinite (so B won't route to A via C)
  - Will this completely solve count to infinity problem?

Algorithm terminates

Link cost changes here: 8 updates D(B, A) = 60 as C has advertised D(C, A) = \infty

Alan Mislove
amislove at ccs.neu.edu
Northeastern University