Routing is how you get to a destination from a source point.

Forwarding versus Routing

- **Forwarding**: at the data plane
  - Directing a packet to an outgoing link
  - Individual router using a forwarding table
- **Routing**: at the control plane
  - Computing paths the packet will follow
  - Involves routers talking to each other
  - Individual router creates a forwarding table
- **Routers have a forwarding table.**
  - Contains header value which contains header for packets.
  - Contains the output Link

Why does routing matter?

- **End-to-end performance**
  - Quality of the path affects user performance
  - Propagation delay, through output, packet loss
- **Use of network resources**
  - Balance of the traffic over routers and links
  - Avoid congestion by directing traffic to lightly loaded links
- **Transient disruptions during changes**
  - Failures, maintenance, load balancing
  - Limiting packet loss and delay during changes

Forwarding vs Routing

- **Forwarding table**
  - Used when a packet is being forwarded and so must contain enough information to accomplish the forwarding function
  - A row in the forwarding table contains the mapping from a network number to an outgoing interface and some MAC information, such as Ethernet Address of the next hop.
- **Routing table**
  - Built by the routing algorithm as a prefix/length and the next hop ip address
- **Forwarding table has prefix/length, interface, and next hop mac address.**

Routing: Network as a graph

- The basic problem of routing is to find the lowest-cost path between any two nodes.
  - Where the cost of a path equals the sum of the costs of all the edges that make up the path.
• For a simple network, we can calculate all shortest paths and load them into some nonvolatile storage on each node.
• Such a static approach has several shortcomings
  ○ It does not deal with node or link failures
  ○ It does not consider the addition of new nodes or links
  ○ It implies that edge costs cannot change
• The solution is a distributed and dynamic protocol
• Two main classes of protocols are available.

<table>
<thead>
<tr>
<th>Global or decentralized</th>
<th>Static or dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>○ Global</td>
<td>Static</td>
</tr>
<tr>
<td>■ All routers have complete topology and link cost info</td>
<td></td>
</tr>
<tr>
<td>■ &quot;link state&quot; algorithms</td>
<td></td>
</tr>
<tr>
<td>○ Decentralized</td>
<td>Dynamic</td>
</tr>
<tr>
<td>■ Routers knows about physically connected neighbors</td>
<td></td>
</tr>
<tr>
<td>■ Iterative process of computation, exchange of info with neighbors</td>
<td></td>
</tr>
<tr>
<td>■ &quot;distance vector&quot; algorithms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manual configuration</td>
</tr>
<tr>
<td></td>
<td>When routers very slowly over time after human intervention</td>
</tr>
</tbody>
</table>

Distance Vector
• Each node constructs a one dimensional array (a vector) containing the "distances" (costs) to all other nodes and distributes that vector to its immediate neighbors.
• Starting assumption is that each node knows the cost of the link to each of its directly connected neighbors

Distance Vector Algorithm
• The Bellman-Ford Equation
• Define \( d_x(y) = \min\{c(x,y) + d_v(y)\} \)
• Where \( v \) is a neighbor of \( x \), and \( x \) and \( y \) are the starting and destination nodes, respectively.

Bellman-Ford Example
• An example was given, see the slides online. The example tries to find the least cost path.

Distance Vector Algorithm
• Basic Idea
  ○ Each node periodically sends its own distance vector estimate to neighbors
  ○ When node a node \( x \) receives new distance vector estimate from it's neighbor, it updates its own DV using B-F equation:
  ○ Converges to lowest cost under simple, stable conditions (no additions/removals of hosts or nodes).
• Iterative, asynchronous:
Each local iteration caused by:
- Local link cost change
- DV update message from neighbor
- Distributed:
  - Each node notifies neighbors only when its DV changes
    - Neighbors then notifies their neighbors if necessary.

Distance Vector
- When a node detects a link failure it tries to fix it to find a new link.
- Once it is finished it will send the information to all of its neighbors.

Solutions to stop loops
- Solution 1: Holdowns
  - If cost metric increases, delay propagating the information
    - In our example, B delays advertisement
    - C eventually think B's route is gone, picks its own route
    - B then selects C as the next hop
  - Adversely affects convergence
- Other solutions
  - Split horizon
    - When a node sends a routing update to its neighbors, it does not send those
      routes it learned from each neighbor back to that neighbor
      - B does not advertise route to C
      - Split horizon can fail sometimes though.
    - Poisoned reverse
      - B advertises route to C with infinite distance
  - Works for two node loops
    - Does not work for loops with more nodes.
- Avoiding the bouncing effect
  - Select loop-free paths
    - Have each route advertisement carry the entire path information
      - If a router sees itself in a path, rejects the route
  - BGP does it this way
    - Space proportional to network diameter.
    - BGP propagates entire path, and policies can be employed
  - RIP & RIPv2:
    - Uses combination of Split Horizon and Poison Reverse algorithms