Routing

• WANs are made of store and forward switches.

WAN – Wide Area Networks

* A packet switch with two types of I/O connectors: one type is used to connect to other packet switches, and the other is used to connect to computers.

Packet Switches

* A small WAN formed by interconnecting packet switches. Connections between packet switches usually operate at a higher speed than connections to individual computers.

Packet Switch Operation

forall input lines {
    do forever {
        read a packet;
        if (valid) {
            output_line = table[destination_addr]
            if (queue[output_line] not full)
                put packet on queue[output_line]
        }
    }
}

Routing - Network Model

• The graph below models a network. Each node corresponds to a packet switch in the network.

Routing

* Each node corresponds to a packet switch, and each edge represents a connection between the corresponding packet switches.

Next-Hop Forwarding

• A given switch has information about the next place (hop) to send a packet so the packet will eventually reach its destination.

The table is the next-hop forwarding information found in switch 2. Each switch has different next-hop information.
Next Hop Routing Tables

Routing Tables with Alternatives
- Routing tables can also contain a second choice route
- In the event the first route is unavailable or congested, packets can be sent on the alternate interface.

Next Hop Routing Tables with Alternatives

Hierarchical Addressing
- Hierarchical addresses can be used to simplify routing.
  - Example of hierarchical addresses in a WAN.
  - Each address consists of two parts:
    1. Identifies a packet switch
    2. Identifies a computer connected to the switch.

Hierarchical Routing

Route Generation
- Global routing is done by local decisions.
- Routing table can be created:
  - By a central system and distributed to nodes
  - By sending node for each packet
  - “Learned” by each node from neighbors
Optimal Routes

- There are several algorithms for computing the optimal path.
  - Dijkstra’s algorithm
  - Floyd’s algorithm
  - Distributed algorithm

Dijkstra’s Algorithm

(One of the more popular methods for computation of the optimal path)

- Solves the problem of finding the shortest path from a point in a graph (the source) to a destination.
- Dijkstra’s algorithm computes shortest paths in a graph by using weights on edges as a measure of distance. A path with the fewest number of edges may not be the path with least weight.

* A graph with weights assigned to edges. The shortest path between nodes 4 and 5 is shown darkened. The distance along the path is 19, the sum of the weights on the edges.

Floyd’s Algorithm

Start with a table giving the distance from each source to each destination. Immediate neighbors use the link distance. Other nodes have an infinite distance.

<table>
<thead>
<tr>
<th>Source</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Floyd’s Algorithm

For each non-infinite link, see if that is a shorter way to other destinations. If shorter, replace the current distance with the sum of the distance to an intermediate plus the distance from the intermediate to the destination.

<table>
<thead>
<tr>
<th>Source</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Floyd’s Algorithm

Node 1 can get to node 3 in 4 time units. Node 3 can get to node 2 in 5 time units. The time to get from node 1 to 3 to 2 is 4+5=9.

<table>
<thead>
<tr>
<th>Source</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Floyd’s Algorithm

Repeat for each source node, for each intermediate and for each destination.
Floyd’s Algorithm

Iterate until no more changes are made.

<table>
<thead>
<tr>
<th>Source</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>1</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Distributed Routing

- Each node computes the time to send a packet to its neighbors.
- Periodically, each node shares its routing table with its neighbors.
- Use the minimum( current path, (time to neighbor + neighbor to destination))

Connection Oriented and Connectionless Systems

<table>
<thead>
<tr>
<th>Issue</th>
<th>Connection Oriented</th>
<th>Connectionless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial setup</td>
<td>required</td>
<td>not necessary</td>
</tr>
<tr>
<td>Destination address</td>
<td>only needed during</td>
<td>needed every</td>
</tr>
<tr>
<td></td>
<td>initial setup</td>
<td>packet</td>
</tr>
<tr>
<td>Packet sequencing</td>
<td>guaranteed</td>
<td>not guaranteed</td>
</tr>
<tr>
<td>Option negotiation</td>
<td>possible at setup</td>
<td>not available</td>
</tr>
<tr>
<td>Overhead</td>
<td>moderate</td>
<td>low</td>
</tr>
</tbody>
</table>

Connectionless Examples

- Ethernet
- Token Ring
- FDDI
- Internet Protocol
- User Datagram Protocol

Connection Oriented Examples

- ATM
- Transmission Control Protocol (TCP)