Chapter 12

Wide Area Networks (WANs), Routing, and Shortest Paths

Motivation

- Connect multiple computers
- Span large geographic distance
- Cross public right-of-way
  - Streets
  - Buildings
  - Railroads
**Building Blocks**

- Point-to-point long-distance connections
- Packet switches

**Packet Switch**

- Hardware device
- Connects to
  - Other packet switches
  - Computers
- Forwards packets
- Uses addresses
Illustration Of A Packet Switch

- Special-purpose computer system
  - CPU
  - Memory
  - I/O interfaces
  - Firmware

Building A WAN

- Place one or more packet switches at each site
- Interconnect switches
  - LAN technology for local connections
  - Leased digital circuits for long-distance connections
Illustration Of A WAN

Interconnections depend on
- Estimated traffic
- Reliability needed

Store And Forward

- Basic paradigm used in packet switched network
- Packet
  - Sent from source computer
  - Travels switch-to-switch
  - Delivered to destination
- Switch
  - “Stores” packet in memory
  - Examines packet’s destination address
  - “Forwards” packet toward destination
Addressing In A WAN

❖ Need
  – Unique address for each computer
  – Efficient forwarding
❖ Two-part address
  – Packet switch number
  – Computer on that switch

Illustration Of WAN Addressing

❖ Two-part address encoded as integer
  – High-order bits for switch number
  – Low-order bits for computer number
Next-Hop Forwarding

- Performed by packet switch
- Uses table of routes
- Table gives next hop

Forwarding Table

Abbreviations

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, anything)</td>
<td>interface 1</td>
</tr>
<tr>
<td>(3, anything)</td>
<td>interface 4</td>
</tr>
<tr>
<td>(2, anything)</td>
<td>local computer</td>
</tr>
</tbody>
</table>

- Many entries point to same next hop
- Can be condensed (default)
- Improves lookup efficiency
Source Of Routing Table Information

- **Manual**
  - Table created by hand
  - Useful in small networks
  - Useful if routes never change

- **Automatic routing**
  - Software creates/ updates table
  - Needed in large networks
  - Changes routes when failures occur

Relationship Of Routing To Graph Theory

- **Graph**
  - Node models switch
  - Edge models connection
**Shortest Path Computation**

- Algorithms from graph theory
- No central authority (distributed computation)
- A switch
  - Must learn route to each destination
  - Only communicates with directly attached neighbors

**Illustration Of Minimum Weight Path**

- Label on edge represents ‘‘distance’’
- Possible distance metric
  - Geographic distance
  - Economic cost
  - Inverse of capacity
- Darkened path is minimum 4 to 5
Algorithms For Computing Shortest Paths

- Distance Vector (DV)
  - Switches exchange information in their routing tables
- Link-state
  - Switches exchange link status information
- Both used in practice

Distance Vector

- Periodic, two-way exchange between neighbors
- During exchange, switch sends
  - List of pairs
  - Each pair gives (destination, distance)
- Receiver
  - Compares each item in list to local routes
  - Changes routes if better path exists
Given:
- a local routing table, a weight for each link that connects to another switch, and an incoming routing message

Compute:
- an updated routing table

Method:
- Maintain a distance field in each routing table entry;
- Initialize routing table with a single entry that has the destination equal to the local packet switch, the next-hop unused, and the distance set to zero;
- Repeat forever
  - wait for the next routing message to arrive over the network from a neighbor;
  - Let \( N \) be the sending switch;
  - for each entry in the message {
    - Let \( V \) be the destination in the entry and let \( D \) be the distance;
    - Compute \( C \) as \( D \) plus the weight assigned to the link over which the message arrived;
    - Examine and update the local routing table:
      - if (no route exists to \( V \)) {
        - add an entry to the local routing table for destination \( V \) with next-hop \( N \) and distance \( C \);
      } else if (a route exists that has next-hop \( N \)) {
        - replace the distance in existing route with \( C \);
      } else if (a route exists with distance greater than \( C \)) {
        - change the next-hop to \( N \) and distance to \( C \);
      }
  }

Distance Vector Algorithm

Distance Vector Intuition

- Let
  - \( N \) be neighbor that sent the routing message
  - \( V \) be destination in a pair
  - \( D \) be distance in a pair
  - \( C \) be \( D \) plus the cost to reach the sender

- If no local route to \( V \) or local route has cost greater than \( C \), install a route with next hop \( N \) and cost \( C \)

- Else ignore pair
**Example Of Distance Vector Routing**

- Consider transmission of one DV message
- Node 2 sends to 3, 5, and 6
- Node 6 installs cost 8 route to 2
- Later 3 sends update to 6
- 6 changes route to make 3 the next hop for destination 2

**Link-State Routing**

- Overcomes instabilities in DV
- Pair of switches periodically
  - Test link between them
  - Broadcast link status message
- Switch
  - Receives status messages
  - Computes new routes
  - Uses Dijkstra’s algorithm
Example Of Link-State Information

- Assume nodes 2 and 3
  - Test link between them
  - Broadcast information
- Each node
  - Receives information
  - Recomputes routes as needed

Dijkstra’s Shortest Path Algorithm

- Input
  - Graph with weighted edges
  - Node, n
- Output
  - Set of shortest paths from n to each node
  - Cost of each path
- Called Shortest Path First (SPF) algorithm
Dijkstra’s Algorithm

Given:
- a graph with a nonnegative weight assigned to each edge and a designated source node
Compute:
- the shortest distance from the source node to each other node and a next-hop routing table
Method:
Initialize set S to contain all nodes except the source node;
Initialize array D so that D[v] is the weight of the edge from the source to v if such an edge exists, and infinity otherwise;
Initialize entries of R so that R[v] is assigned v if an edge exists from the source to v, and zero otherwise;
while (set S is not empty) {
    choose a node u from S such that D[u] is minimum;
    if (D[u] is infinity) {
        no path exists to nodes in S; quit;
    }
    delete u from set S;
    for each node v such that (u,v) is an edge {
        if (v is still in S) {
            c = D[u] + weight(u,v);
            if (c < D[v]) {
                R[v] = u;
                D[v] = c;
            }
        }
    }
}

Algorithm Intuition

- Start with self as source node
- Move outward
- At each step
  - Find node u such that it
    - Has not been considered
    - Is “closest” to source
  - Compute
    - Distance from u to each neighbor v
    - If distance shorter, make path from u go through v
Result Of Dijkstra’s Algorithm

Example routes from node 6
- To 3, next hop = 3, cost = 2
- To 2, next hop = 3, cost = 5
- To 5, next hop = 3, cost = 11
- To 4, next hop = 7, cost = 8

Early WAN Technologies

- **ARPANET**
  - Historically important in packet switching
  - Fast when invented; slow by current standards
- **X.25**
  - Early commercial service
  - Still used
  - More popular in Europe
Recent WAN Technologies

- SMDS
  - Offered by phone companies
  - Not as popular as Frame Relay
- Frame Relay
  - Widely used commercial service
  - Offered by phone companies
- ATM
  - Offered by phone companies
  - Designed as both WAN and LAN

Summary

- Wide Area Networks (WANs)
  - Span long distances
  - Connect many computers
  - Built from packet switches
  - Use store-and-forward
- WAN addressing
  - Two-part address
  - Switch/computer
**Summary (continued)**

- **Routing**
  - Each switch contains routing table
  - Table gives next-hop for destination

- **Routing tables created**
  - Manually
  - Automatically

- **Two basic routing algorithms**
  - Distance vector
  - Link state

**Example WAN technologies**

- ARPANET
- X.25
- SMDS
- Frame Relay
- ATM