Packet Switching

Raj Jain
Professor of CIS

Raj Jain is now at
Washington University in Saint Louis
Jain@cse.wustl.edu
http://www.cse.wustl.edu/~jain/
- Routing algorithms
- ARPAnet routing
Rooting or Routing

- **Rooting** is what fans do at football games, what pics do for truffles under oak trees in the Vaucluse, and what nursery workers intent on propagation do to cuttings from plants.

- **Routing** is how one creates a beveled edge on a table top or sends a corps of infantrymen into full scale, disorganized retreat

Ref: Piscitello and Chapin, p413
Routeing or Routing

- Routeing: British
- Routing: American

Since Oxford English Dictionary is much heavier than any other dictionary of American English, British English generally prevails in the documents produced by ISO and CCITT; wherefore, most of the international standards for routing standards use the routeing spelling.

Ref: Piscitello and Chapin, p413
Rooting or Routing

- **Rooting** is what fans do at football games, what pics do for truffles under oak trees in the Vaucluse, and what nursery workers intent on propagation do to cuttings from plants.

- **Routing** is how one creates a beveled edge on a table top or sends a corps of infantrymen into full scale, disorganized retreat.

Ref: Piscitello and Chapin, p413
Routeing or Routing

- Routeing: British
- Routing: American

Since Oxford English Dictionary is much heavier than any other dictionary of American English, British English generally prevails in the documents produced by ISO and CCITT; wherefore, most of the international standards for routing standards use the routeing spelling.

Ref: Piscitello and Chapin, p413
Routing Techniques Elements

- **Performance criterion:** Hops, Distance, Speed, Delay, Cost
- **Decision time:** Packet, session
- **Decision place:** Distributed, centralized, Source
- **Network information source:** None, local, adjacent nodes, nodes along route, all nodes
- **Routing strategy:** Fixed, adaptive, random, flooding
- **Adaptive routing update time:** Continuous, periodic, topology change, major load change
Distance Vector vs Link State

- **Distance Vector**: Each router sends a vector of distances to its neighbors. The vector contains distances to all nodes in the network. Older method. Count to infinity problem.

- **Link State**: Each router sends a vector of distances to all nodes. The vector contains only distances to neighbors. Newer method. Used currently in internet.
Dijkstra’s Algorithm

- Goal: Find the least cost paths from a given node to all other nodes in the network

- Notation:
  \[ d_{ij} = \text{Link cost from } i \text{ to } j \text{ if } i \text{ and } j \text{ are connected} \]
  \[ D_n = \text{Total path cost from } s \text{ to } n \]
  \[ M = \text{Set of nodes so far for which the least cost path is known} \]

- Method:
  - Initialize: \( M = \{s\}, D_n = d_{sn} \)
  - Find node \( w \notin M \), whose \( D_n \) is minimum
  - Update \( D_n \)
Example

\[ M = \{1\} \]
\[ D_2 = 2 \quad D_3 = 5 \]
\[ D_4 = 1 \]

\[ M = \{1, 4\} \]
\[ D_2 = 2 \quad D_3 = 4 \]
\[ D_4 = 1 \]
\[ D_5 = 2 \]

\[ M = \{1, 2, 4\} \]
\[ D_2 = 2 \quad D_3 = 4 \]
\[ D_4 = 1 \]
\[ D_5 = 2 \]

\[ M = \{1, 2, 4, 5\} \]
\[ D_2 = 2 \]
\[ D_3 = 3 \]
\[ D_4 = 1 \]
\[ D_5 = 2 \]
\[ D_6 = 4 \]
Example (Cont)

<table>
<thead>
<tr>
<th>M</th>
<th>D2</th>
<th>Path</th>
<th>D3</th>
<th>Path</th>
<th>D4</th>
<th>Path</th>
<th>D5</th>
<th>Path</th>
<th>D6</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 {1}</td>
<td>2</td>
<td>1-2</td>
<td>5</td>
<td>1-3</td>
<td>1</td>
<td>1-4</td>
<td>∞</td>
<td>-</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>2 {1,4}</td>
<td>2</td>
<td>1-2</td>
<td>4</td>
<td>1-4-3</td>
<td>1</td>
<td>1-4</td>
<td>2</td>
<td>1-4-5</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>3 {1,2,4}</td>
<td>2</td>
<td>1-2</td>
<td>4</td>
<td>1-4-3</td>
<td>1</td>
<td>1-4</td>
<td>2</td>
<td>1-4-5</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>4 {1,2,4,5}</td>
<td>2</td>
<td>1-2</td>
<td>3</td>
<td>1-4-5-3</td>
<td>1</td>
<td>1-4</td>
<td>2</td>
<td>1-4-5</td>
<td>4</td>
<td>1-4-5-6</td>
</tr>
<tr>
<td>5 {1,2,3,4,5}</td>
<td>2</td>
<td>1-2</td>
<td>3</td>
<td>1-4-5-3</td>
<td>1</td>
<td>1-4</td>
<td>2</td>
<td>1-4-5</td>
<td>4</td>
<td>1-4-5-6</td>
</tr>
<tr>
<td>6 {1,2,3,4,5,6}</td>
<td>2</td>
<td>1-2</td>
<td>3</td>
<td>1-4-5-3</td>
<td>1</td>
<td>1-4</td>
<td>2</td>
<td>1-4-5</td>
<td>4</td>
<td>1-4-5-6</td>
</tr>
</tbody>
</table>
Dijkstra's Routing Algorithm

- Apply to the following network and compute paths from node 1.

![Network Diagram]

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>D2 Path</th>
<th>D3 Path</th>
<th>D4 Path</th>
<th>D5 Path</th>
<th>D6 Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra's routing algorithm

- Apply to the following network and compute paths from node 1.

![Graph with nodes and edges](attachment:image.png)

<table>
<thead>
<tr>
<th>M</th>
<th>D2 Path</th>
<th>D3 Path</th>
<th>D4 Path</th>
<th>D5 Path</th>
<th>D6 Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 {1}</td>
<td>1-2</td>
<td>∞</td>
<td>1-4</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>2 {1,2}</td>
<td>1-2</td>
<td>4</td>
<td>1-4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3 {1,2,3}</td>
<td>1-2</td>
<td>4</td>
<td>1-4</td>
<td>2</td>
<td>1-2-5</td>
</tr>
<tr>
<td>4 {1,2,3,5}</td>
<td>1-2</td>
<td>4</td>
<td>1-2-5-1</td>
<td>2</td>
<td>1-2-5</td>
</tr>
<tr>
<td>5 {1,2,3,4,5}</td>
<td>1-2</td>
<td>4</td>
<td>1-2-5-1</td>
<td>2</td>
<td>1-2-5</td>
</tr>
<tr>
<td>6 {1,2,3,4,5,6}</td>
<td>1-2</td>
<td>4</td>
<td>1-2-5-1</td>
<td>2</td>
<td>1-2-5</td>
</tr>
</tbody>
</table>

The Ohio State University

Raj Jain
Bellman-Ford Algorithm

- Notation:
  \( h = \) Number of hops being considered
  \( D^{(h)}_n = \) Cost of \( h \)-hop path from \( s \) to \( n \)

- Method:
  Find all nodes 1 hop away
  Find all nodes 2 hops away
  Find all nodes 3 hops away

- Initialize:
  \( D^{(h)}_n = \infty \) for all \( n \neq s \); \( D^{(h)}_n = 0 \) for all \( h \)

- Find \( j \)th node for which \( h+1 \) hops cost is minimum
  \( D^{(h+1)}_n = \min_j [D^{(h)}_j + D_{jn}] \)
Example

Fig 9.23b
## Example (Cont)

<table>
<thead>
<tr>
<th>h</th>
<th>$D(h_2)$ Path</th>
<th>$D(h_3)$ Path</th>
<th>$D(h_4)$ Path</th>
<th>$D(h_5)$ Path</th>
<th>$D(h_6)$ Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\infty$ -</td>
<td>$\infty$ -</td>
<td>$\infty$ -</td>
<td>$\infty$ -</td>
<td>$\infty$ -</td>
</tr>
<tr>
<td>1</td>
<td>2 1-2</td>
<td>5 1-3</td>
<td>1 1-4</td>
<td>$\infty$ -</td>
<td>$\infty$ -</td>
</tr>
<tr>
<td>2</td>
<td>2 1-2</td>
<td>4 1-4-3</td>
<td>1 1-4</td>
<td>2 1-4-5</td>
<td>10 1-3-6</td>
</tr>
<tr>
<td>3</td>
<td>2 1-2</td>
<td>3 1-5-4-3</td>
<td>1 1-4</td>
<td>2 1-4-5</td>
<td>4 1-4-5-6</td>
</tr>
<tr>
<td>4</td>
<td>2 1-2</td>
<td>3 1-5-4-3</td>
<td>1 1-4</td>
<td>2 1-4-5</td>
<td>4 1-4-5-6</td>
</tr>
</tbody>
</table>

Table 9.4b

The Ohio State University

Raj Jain
Flooding
Flooding

- Uses all possible paths
- Uses minimum hop path
  Used for source routing

Fig 9.7
ARPAnet Routing (1969-78)

- Features: Cost=Queue length,
- Each node sends a vector of costs (to all nodes) to neighbors. Distance vector
- Each node computes new cost vectors based on the new info using Bellman-Ford algorithm
### ARPAnet Routing Algorithm

#### Destination Delay Next node

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Ń</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### D1 S1

#### Destination Delay Next node

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### D2 D3 D4

#### Destination Delay Next node

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Ń</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 1 1,2 = 2 1,3 = 5 1,4 = 1

#### (a) Node 1’s routing table before update

#### (b) Delay vectors sent to neighbor nodes

#### (c) Node 1’s routing table after update and link costs

---

Fig 9.9

The Ohio State University

Raj Jain

5-21
ARPAnet Routing (1979-86)

- Problem with earlier algorithm: Thrashing (packets went to areas of low queue length rather than the destination), Speed not considered
- Solution: Cost=Measured delay over 10 seconds
- Each node floods a vector of cost to neighbors. Link-state. Converges faster after topology changes.
- Each node computes new cost vectors based on the new info using Dijkstra’s algorithm

Fig 9.10
Problem with 2nd Method: Correlation between delays reported and those experienced later: High in light loads, low during heavy loads
⇒ Oscillations under heavy loads
⇒ Unused capacity at some links, over-utilization of others, More variance in delay more frequent updates
More overhead

Fig 9.11
Routing Algorithm

- Delay is averaged over 10 s
- Link utilization = \( r = \frac{2(s-t)}{(s-2t)} \)
  where \( t \) = measured delay,
  \( s \) = service time per packet (600 bit times)
- Exponentially weighted average utilization
  \( U(n+1) = \alpha U(n) + (1-\alpha)r(n+1) \)
  \( = 0.5 U(n) + 0.5 r(n+1) \) with \( \alpha = 0.5 \)
- Link cost = \( f_n(U) \)

Fig 9.12
Connection-oriented and Connectionless
Routing: Least-cost, Flooding, Adaptive
Dijkstra’s and Bellman-Ford algorithms
ARPAnet
Homework

- Exercise 9.4 (in b assume a unidirectional single loop), 9.10, 9.15
- Due: Next class