Routing Problem Formulation (1/9)

- When source and destination are connected by a direct link (either or shared or dedicated), data packets may be simply addressed to the destination and transmitted over the link
- Q: How is data forwarded from source to destination over a potentially multi-hop path?

Routing Problem Formulation (2/9)

- **Connection-oriented service**: An end-to-end to “virtual channel” needs to be established between source and destination before data can be transferred

Routing Problem Formulation (3/9)

- **Connectionless service**: No session setup is required as data forwarding decisions are made on a, real-time hop-by-hop basis

Routing Problem Formulation (4/9)

- **Typical Switch/Router**:
  - Incoming links
  - CPU and buffer
  - Output links
  - Output link buffers

Routing Problem Formulation (5/9)

- **Routing function implementations**:
  - **Table-driven routing**: Each node maintains a table that maps a packet’s ID to an outgoing link and each incoming packet must be compared with a corresponding table entry
  - **Table-free routing**: No table look-up required (to reduce processing time) as the outgoing link for the incoming packet is specified either explicitly or implicitly in the packet header
Routing Problem Formulation (6/9)
Routing is regarded as layer 3 (network layer) function

Routing Problem Formulation (7/9)
• Path selection
  – Each end-to-end path is associated with a path cost, e.g.,
    • Total number of hops in path
    • Total end-to-end path delay
  – For each hop in the end-to-end path there is an associated link cost, e.g.,
    • Cost of 1 per hop
    • Link cost that is inversely proportional to the link transmission rate
    • Average total delay to deliver a new packet arrival to the next node in the path
    • Two costs: Low cost for when the queue length is below a threshold and a high cost otherwise

Routing Problem Formulation (8/9)
• Link delay metric
  – An important link cost metric in many networks is the average delay associated with forwarding a packet on a particular link
  – Components of a link delay metric
    • Packet transmission delay
      (There may also be delays associated with contending for media access on shared (multi-access) links)
    • Queuing delay while a packet waits its turn for access to the link transmission buffer
    • Estimation of link propagation delay
    • Processing delay at a switch (e.g., table look-up)

Routing Problem Formulation (9/9)
• Classification of routing schemes
  – Static versus dynamic:
    • Dynamic: Best-path computation is updated frequently based on (near) real-time network measurements
    • Static: Best-path information is updated infrequently based on long-term projected network conditions
  – Centralized versus distributed
    • Centralized: A single node or site is responsible for all best-path computations
    • Distributed: All network nodes are involved in the best-path computations

Shortest Path Procedures
• To compute the best path between a pair of nodes, shortest-path (i.e., least-cost-path) procedures are often used
• Link costs are inputted to some algorithm (either centralized or distributed) and routing table information is outputted
• Notation
  – $c_{vw}$ = Cost of the link connecting nodes $v$ and $w$ ($c_{vw} = \infty$ if no link exists)
  – $C_{vw}$ = Cost of the best-known path between $v$ and $w$ ($C_{vw} = \infty$ if no path is currently known)
  – $P_{vw}$ = Order list of nodes comprising the best-known shortest path from $v$ to $w$

Network Graphs (1/5)
• It is often useful to discuss networks in terms of a graph
• $G = (V,E)$ denotes a graph consisting of a set of vertices $V$ (i.e., nodes) inter-connected by a set of edges $E$ (i.e., links)
• An edge $e \in E$ of a directed graph is represented as an ordered pair $(u,v)$, where $u \in V$ and $v \in V$
  – Here, $u$ is the initial vertex and $v$ is the terminal vertex
  – For the purposes here, it is assumed that $u \neq v$
Network Graphs (2/5)

- Example, an "un-weighted" and directed graph:

\[
E = \begin{bmatrix}
\infty & 1 & \infty & \infty \\
\infty & 1 & 1 & \infty \\
\infty & \infty & \infty & \infty \\
1 & \infty & \infty & \infty \\
\end{bmatrix}
\]

- \(V = \{1,2,3,4\}, E = \{(1,2),(2,3),(2,4),(4,1),(4,2)\}\)
- Although \(E\) is un-weighted, uniform weights might be assigned to the edges
- Thus, \(E\) may be represented by a weight, or cost, matrix \(c\), where \((u,v) \notin E \leftrightarrow c(u,v) = \infty\)

Network Graphs (3/5)

- Example, an un-weighted and undirected graph:

\[
\begin{align*}
V &= \{1,2,3,4\} \\
E &= \{(1,2),(1,4),(2,3),(2,4)\} \\
&\leftrightarrow \{(2,1),(4,1),(3,2),(4,2)\}
\end{align*}
\]

- An edge \(e \in E\) in this example is an unordered pair \((u,v)\) \leftrightarrow \((v,u)\)
- An undirected graph yields a symmetric cost matrix \(c\)

Network Graphs (4/5)

- Example, a directed graph with non-uniform costs:

\[
E = \{(1,2),(2,3),(2,4),(4,1),(4,2)\}
\]

- \(V = \{1,2,3,4\}\)
- \(E\) is un-weighted, uniform weights might be assigned to the edges
- Thus, \(E\) may be represented by a weight, or cost, matrix \(c\), where \((u,v) \notin E \leftrightarrow c(u,v) = \infty\)

Network Graphs (5/5)

- Shortest path spanning tree (SPST) \(\equiv\) A tree \((T)\) rooted at a particular node, say \(s\), such that the minimum cost paths from \(s\) to each of the other network nodes is contained in \(T\)

Shortest Path Forward Tree (1/3)

- A tree consisting of the shortest (directed) paths from a source node \(s\) to all other nodes in the network
- May be computed by Dijkstra’s algorithm
  - Maintains a set \(S\) of nodes to which shortest paths have been found (initially \(S = \{s\}\))
  - At each step, find a node \(v \in V - S\) such that \(C_{s,v}\) is among nodes all nodes in \(V - S\)
  - Add \(v\) to \(S\): \(S = S \cup v\)
  - Nodes are added to \(S\) until \(S = V\)

Shortest Path Forward Tree (2/3)

Dijkstra Algorithm:

1. \(S = \{s\}; U = V - \{s\}\)
2. \(\text{For } v \in U \text{ do}\)
   - \(C_{s,v} = c_{s,v}\)
   - If \(C_{s,v} < \infty\) then \(P_{s,v} = \{s,v\}\)
3. \(\text{EndFor}\)
4. \(\text{While } U \neq \emptyset \text{ do}\)
   - Find \(v \in U\) for which \(C_{s,v} = \min_{w \in U} C_{s,w}\)
   - Add \(v\) to \(S\): \(S = S \cup v\)
   - For all \(v \in S\) do
     - \(C_{s,v} = \min_{w \in U} C_{s,w}\)
     - If \(C_{s,v} < \infty\) and \(C_{s,w} = C_{s,v} + c_{s,v}\) then \(P_{s,v} = \{s,w\} \cup P_{s,w}\)
   - EndIf
5. \(\text{EndWhile}\)
Shortest Path Forward Tree (3/3)

Dijkstra’s Algorithm Example

Network Graph

<table>
<thead>
<tr>
<th>Node</th>
<th>d</th>
<th>P_d</th>
<th>C_d</th>
<th>P_d</th>
<th>C_d</th>
<th>P_d</th>
<th>C_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(A,B)</td>
<td>3</td>
<td>15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>(A,B)</td>
<td>3</td>
<td>(A,B,C)</td>
<td>18</td>
<td>-</td>
<td>(A,E)</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>(A,B,E)</td>
<td>3</td>
<td>(A,B,E,C)</td>
<td>14</td>
<td>(A,E,D)</td>
<td>14</td>
<td>(A,E)</td>
</tr>
<tr>
<td>4</td>
<td>(A,B,E,C,E)</td>
<td>3</td>
<td>(A,B,E,D,C)</td>
<td>14</td>
<td>(A,E,D)</td>
<td>14</td>
<td>(A,E)</td>
</tr>
<tr>
<td>5</td>
<td>(A,B,C,D,E,F)</td>
<td>3</td>
<td>(A,B,C,D,E,F)</td>
<td>14</td>
<td>(A,E,D,C,F)</td>
<td>14</td>
<td>(A,E)</td>
</tr>
</tbody>
</table>

Shortest Backward Path Tree (1/3)

- Here, an algorithm computes the shortest path from all sources to a single destination
- \( d \) = Destination to which shortest paths must be computed
- \( L_d(v) = (n_v, C_{vd}) \) where
  - \( C_{vd} \) is as before and
  - \( n_v \) is the next node in shortest path from \( v \) to \( d \)

- For each cycle, each node \( v \) updates \( L_d(v) \) based on costs to \( d \) reported by its neighbors
- Algorithm terminates when \( L_d(v) \) is the same as from the previous cycle for all \( v \in V \)

Shortest Backward Path Tree (2/3)

**Shortest Backward Path Tree Algorithm:**

\[ L_d(d) = (-\infty, 0) \]

For all nodes \( v \neq d \) do \( L_d(v) = (-\infty, \infty) \);

\( L_d = \) Set of all node labels for destination \( d \);

Repeat

\[ L_{temp} = L_d \]

For all \( v \in V \) do

\[ A(v) = \) Set of neighbors of \( v \); \]

\[ C_{vd} = \min_{w_1 \in A(v)} \{ C_{vd} + C_{vw} \}; \]

\[ n_v \) that minimizes \( C_{vd} \)

EndFor

Until \( L_d = L_{temp} \)

**Algorithm complexity?**

Shortest Backward Path Tree (3/3)

**Shortest Backward Path Tree Example:**

Network Graph

<table>
<thead>
<tr>
<th>Cycle</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(-\infty)</td>
<td>(-\infty)</td>
<td>(-\infty)</td>
<td>(-\infty)</td>
<td>(-\infty)</td>
<td>(-\infty)</td>
</tr>
<tr>
<td>1</td>
<td>(-\infty)</td>
<td>(A,2)</td>
<td>(B,8)</td>
<td>(C,9)</td>
<td>(A,1)</td>
<td>(C,9)</td>
</tr>
<tr>
<td>2</td>
<td>(-\infty)</td>
<td>(A,2)</td>
<td>(B,8)</td>
<td>(E,6)</td>
<td>(A,1)</td>
<td>(C,9)</td>
</tr>
<tr>
<td>3</td>
<td>(-\infty)</td>
<td>(A,2)</td>
<td>(B,8)</td>
<td>(E,6)</td>
<td>(A,1)</td>
<td>(C,9)</td>
</tr>
</tbody>
</table>