The Routing Problem

- Routing messages between pairs of processors is a primary activity of any distributed network of processors.
- The mechanism responsible for performing this activity in the network is a subsystem called a *routing scheme*.
- This mechanism consists of a collection of message forwarding protocols and information tables.
- The quality of the routing scheme is important to the overall performance of the network.
- The design of efficient routing schemes is the subject of much recent study.
Store and Forward

• We consider one common type of routing method, known as “store and forward” packet routing.

• A store and forward routing scheme is based on handling each message individually, by forwarding it from the sender to the destination through a chain of intermediate vertices.

• Each of these intermediate sites stores the message locally, decides on the next edge on which to forward the message and places the message in the queue of outgoing messages associated with that edge.

Routing Schemes

• A routing scheme RS for an n-processor network $G = (V, E)$ is a mechanism specifying for each pair of vertices $u, v \in V$ a path in the network connecting $u$ to $v$.

• In particular, it has to specify, in each step along the way, the header attached to the message and the port through which it needs to be transmitted.
Labels and Headers

We define the following two notions.

- First, the routing scheme requires us to refer to vertices by special routing labels, i.e., it makes use of a labeling assignment \( \text{Label} \) on \( v \).
  \[ \text{Labels} = (\text{Label}(v_1), \ldots, \text{Label}(v_n)) \]

- Second, we will formally denote the collection of allowable message headers by \( \text{Headers} \).

Data structures

The data structures required for maintaining a routing scheme are composed of the following three types of functions. Each vertex \( v \) stores one function of each type.

1. An initial header function, \( I_v: \text{Labels} \rightarrow \text{Headers} \)

2. A header function, \( H_v: \text{Header} \rightarrow \text{Headers} \)

3. A port function, \( F_v: \text{Headers} \rightarrow [1, \deg_G(v)] \)
Forwarding Protocol (originator)

Suppose the vertex $u$ wishes to send a message MSG to a destination $v$ whose label is $Label(v)$. Then $u$ has to perform the following two steps:

1. Prepare a header $h$ and attach it to the message. Such a header typically consists of (all or portions of) the destination’s label, $Label(v)$, plus some additional routing information. It is computed using the initial header function $I_u$, i.e., by setting $h = I_u(v)$.

2. Decide on the exit port $i$ and load the message onto this port. This port is determined by consulting the port function $F_u$, i.e., by setting $i = F_u(h)$.

Forwarding Protocol (intermediate)

If a message $MSG$ with a header $h'$ arrives at some vertex $w$. $w$ reads $h'$ and check whether it is the final destination of $MSG$. If so, the routing process ends. Otherwise, in has to forward the message further by performing two steps:

1. Prepare a new header $h$ for the message and replace the old header $h'$ attached to the message by the new one. The new header is computed using the header function $H_w$, i.e., by setting $h = H_w(h')$.

2. Compute the exit port $i$ and load the message onto this port. This is again done by using the port function and setting $i = F_w(h)$. 
Route

• For every pair of vertices $u, v$, the scheme RS implicitly specifies a route
  
  $Y(RS, u, v) = (u = w_1, w_2, ..., w_j = v)$

  consisting of the vertices through which a message passes from origin $u$ to destination $v$.

• We denote the length of this path by $|Y(RS, u, v)|$.

• An obvious necessary requirement of any routing scheme is that for every pair of vertices $u, v$, the route defined in this way indeed reaches $v$.

Adaptive Schemes

• A major factor in evaluating the efficiency of a routing scheme is the delivery speed of messages.

• A good routing scheme should be able to estimate the expected delays along different paths and select the better paths for its use.

• These delays change dynamically and are not fixed over time.

• The most significant feature of an adaptive routing scheme is its ability to sense changes in the traffic distribution and the load conditions throughout the network and modify the routes accordingly, so that messages in transition avoid congested or disconnected areas of the network.

• The decisions made by the routing scheme itself directly affect future congestion, and hence delays.
Updates in Scheme

• The routing subsystem of the network performs periodic updates in the routing scheme.
• Such an update operation involves two main steps.
  1. Collect some information about the network state. The collected data is used to compute the new edge weights.
  2. Decide on the new routes and setting up the information tables accordingly.
• We concentrate on the second step of setting up a routing scheme once the costs of using each edge are determined.
• Our routing schemes must be able to handle arbitrary edge costs (as well as arbitrary network topologies).

Communication Cost

• The cost of using a link $e$ is represented via the weight function $w(e)$, reflecting the estimated link delay for a message transmitted over that link.
• The cost of routing a message along a given path is simply the sum of the costs of the transmissions performed during the routing.
• We denote the cost of routing a message from $u$ to $v$ by the scheme $RS$ by $\text{Comm}(RS, u, v)$.
• This communication cost can be thought of as represented by the weighted length of the route produced by the scheme, $|Y(RS, u, v)|$.
• More sophisticated schemes may be doing other things, e.g., consult certain databases for routing information, and the costs of these steps should be taken into account as well.
Shortest Path Routing Scheme

- Many common routing schemes insist on always routing messages along shortest paths, given the link weights. We refer to routing schemes imposing this requirement as *shortest path routing* schemes.
- Designing a shortest path routing scheme is composed of two components:
  (a) computing the shortest routes and
  (b) storing information at the different network sites enabling the processors to use these routes for message routing.

Full Tables Routing Schemes

- The most straightforward and commonly used approach for implementing a shortest-paths routing scheme is the *full tables* routing scheme, denoted by FTR, which operates as follows.

  **Data structures**

- Store a complete routing table in each vertex $v$ in the network.
- Given an $n$-processor network $G$, a full tables scheme for $G$ is constructed by specifying the port functions it uses.
- The port function $F_v$, stored at the vertex $v$, is specified as a complete table of $n-1$ entries, with one entry for each of the $n-1$ possible destinations $u \neq v$, listing the exit port to be used for forwarding a message to $u$. 
Forwarding Protocol

- Whenever a vertex $u$ needs to forward a message $MSG$ to a destination $v$ whose label is $Label (v)$, it decides on the exit port $i$ to be used by consulting the entry corresponding to $i$ in the table $Fu$.
- This approach guarantees shortest routes and hence optimal communication cost, namely,

$$Comm(RS,u,v) = |Y(RS,u,v)| = dist(u,v).$$

Example Network

<table>
<thead>
<tr>
<th>Dest</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
Compact Routing

• While message delivery speed is the most dominant performance parameter the routing scheme design should take into account, the space used for the routing tables is also a significant consideration.
• There are a number of reasons for minimizing the memory requirements of a routing scheme. We do not want memory requirements to grow fast with the size of the network.
• It is interesting to search for routing schemes that involve small communication cost and have low space requirements.

Quality Measures

• In order to enable us to evaluate and compare schemes, we need to define appropriate quality measures.
• The efficiency of a route provided by the routing scheme RS is measured in terms of its dilation factor, which takes the meaning of the ratio between the communication cost of routing a message from the origin $u$ to the destination $v$ using the scheme, $\text{Comm}(RS,u,v)$, and the cheapest possible cost for passing a message from this origin to this destination, namely, $\text{dist}(u,v)$.
• The dilation of the scheme is the maximum such ratio over all possible origin-destination pairs.
Routing Dilation

**Definition:** Given a routing scheme RS for an n-processor network $G = (V, E)$, we say that RS dilates the path from $u$ to $v$ (for every $u,v \in V$) by

$$Dilation(RS,u,v) = \frac{Comm(RS,u,v)}{dist(u,v)}$$

Define the **maximum dilation factor** of the routing scheme RS on the network G to be

$$Dilation(RS,G) = \max \{Dilation(RS, u, v)\}.$$ 

Memory Requirement

**Definition:** The **memory requirement** of a vertex $v$ labeled $Label(v)$ under the routing scheme RS, denoted $Mem(v,Rs)$, is the number of memory bits required to store the label $Label(v)$ and the functions $I_v$, $H_v$, and $F_v$ in the local memory of the processor.

The **total memory requirement** of the routing scheme RS is

$$Mem(RS) = \sum Mem(v,RS).$$
Communication Centers

• In many routing schemes, different vertices may play different roles and require different amounts of space.

• For instance, some vertices may be designated as communication centers, while others may just happen to be the crossing point of many routes which are maintained by the scheme.

• Such vertices may be required to store more information than others and consequently should be equipped with additional memory in advance.

Balanced Memory Requirements

• In a dynamic routing scheme, vertices may be required to change roles from time to time.

• Ideally, it is preferable if these roles can be assigned to vertices on the basis of the current system load status, depending on the present edge costs.

• This forces every vertex to have sufficient memory for performing the most demanding role in the scheme, rendering the bound on average (or total) space meaningless.

• Therefore we will strive to achieve schemes ensuring balanced memory requirements, i.e., guaranteeing a bound on the worst-case (rather than average or total) memory requirements of each vertex.
Maximal Memory Requirement

**Definition:** The maximal memory requirement of the routing scheme RS is

$$\text{Max\_Mem}(RS) = \max\{\text{Mem}(v,RS)\}.$$  

- One basic goal in large-scale communication networks is the design of routing schemes that produce efficient routes and have low memory requirements.

Routing Strategy

**Definition:** A routing strategy is an algorithm that computes a routing scheme RS for every given (weighted) network $$G=(V,E,w)$$. A routing strategy has dilation factor $$k$$ if for every network $$G$$ it produces a scheme RS such that $$\text{Dilation}(RS, G) \leq k$$.

The memory requirement of a routing strategy (as a function of $$n$$) is the maximum, over all $$n$$-vertex networks, of the memory requirements of the routing schemes it produces.
Flooding

• There is a routing scheme based on *flooding*.
• In this scheme, instead of forwarding a message along a shortest path, the origin simply *broadcasts* it throughout the entire network.
• Clearly, this scheme requires no routing tables, hence it needs minimal $O(\log n)$ bit memory requirements.
• On the other hand, the communication cost of such a scheme may be significantly higher than optimal, since instead of using just one path, we may be using many (possibly expensive) links.
• Thus the dilation factor is unbounded.

Full Table Routing

• Consider the *FTR* scheme discussed in the previous section.
• This scheme uses shortest routes, and hence is communication-optimal (namely, $Dilation(FTR, G)=1$ for every network $G$).
• On the other hand, the disadvantage of the *FTR* scheme is that it may be too expensive for large systems, since it requires each vertex to store a very large routing table (of $\Theta(n \log n)$ memory bits).
• Thus these two example strategies set two extreme endpoints of a possible trade-off between communication cost and memory requirements.
Unweighted Ring

- Consider the unit cost $n$-vertex ring.
- A shortest-paths routing scheme can be implemented on the ring in a straightforward way by labeling the vertices consecutively with labels $0, \ldots, n-1$ and routing from $i$ to $j$ along the shorter of the two ring segments connecting them (which can be easily inferred from the labels $i,j$).
- This clearly yields dilation 1 (optimal routes) and requires only $2 \log n$ memory bits per vertex (i.e., each vertex needs to store its own label and $n$).

Name-Based Routing Schemes

- Our definition of a routing scheme allows the scheme to impose a routing label on each of the vertices of the network, and the processors are required to use these labels for routing purposes.
- Two alternatives:
  1. A potentially simpler approach is to rely on the preexisting processor identifiers supplied by the hardware level.
  2. More “user-friendly” approach would be to attempt to design the scheme in such a way that it allows the sender of a message to refer to the destination by its name, where site names may be chosen by each of the sites for themselves.
“Geography-Based” Mailing System

- The difficulty with these two approaches is that it might be much harder to devise a routing scheme if it is required to support such “name-based” mailing.
- The imposed routing labels are thus intended to serve as a means of introducing some order into the site name domain and to reflect the “geographical” structure of the network to some extent.
- Exploiting the labels to capture information on the network topology (and, specifically, on the location of each site within the network) may be used to support more efficient routing.
- We may think of this latter form of address specification as a “geography-based” mailing system.

Ring Example

- Suppose that the vertices of a ring are given arbitrary identifiers (say, by taking some permutation of \(\{1, n\}\)) and the routing scheme is required to allow “name-based” mailing.
- It is easy to see that if the routing scheme is additionally required to achieve a dilation factor of 1, then it is forced to use a full routing table at each vertex, hence its maximal memory complexity (per vertex) is \(\Omega(n \log n)\).
- Interestingly, constant dilation can be achieved with minimal memory requirements per vertex.
Deadlock

- Overcoming store-and-forward deadlocks is one of the major problems in the design of routing protocols for communication networks.
- A store-and-forward deadlock occurs at some set $N$ of vertices when all the buffers of these vertices are full and each of the messages occupying these buffers needs to be forwarded to some other vertex in the set $N$.
- All of these messages are locked in a circle, since the only way to forward a message is to have a free buffer in the next vertex on its route and the only way to release a buffer is to forward the message currently occupying it.
- Avoidance or fast resolution of such store-and-forward deadlocks is essential for efficient utilization of available network resources.

Deadlock Example
Buffer-Class Approach

• One approach to deadlock prevention is known as the buffer-class approach.
• This approach involves solutions based on dividing the buffer pool into buffer classes and utilizing these classes so as to prevent cyclic waiting chains.
• Some of these solutions are based on restricting the family of allowed routes in order to avoid deadlocks.

Buffer-Routing Scheme

• We extend the definition of a routing scheme so that it specifies, not only the exit port to be used, but also the specific buffer $B_i(w)$ into which the message will be placed at $w$.
• The routing decision may possibly rely on the buffer $B_j(v)$ in which the message is currently placed at $v$.
• Such a scheme may be allowed to include also internal routing steps, in which the message is passed from one buffer to another in the same vertex $v$. 
Buffer-Routing Scheme (cont)

- The description of a route should be expanded to detail not only the identifiers of the intermediate vertices 
  \[(v = w_1, w_2, \ldots, w_j = u),\]
  but also the buffers used along the way,
  \[(B_1(w_1), \ldots, B_j(w_j)).\]
- We refer to this expanded version of a routing scheme as a buffer-routing scheme and denote it by $RSB$.

Buffer Graph

- Define the concept of the buffer graph corresponding to a given buffer-routing scheme $RSB$.
- Suppose that each vertex $v$ has $k$ buffers, denoted $B_1(v), \ldots, B_k(v)$.
- The graph $B(RSB) = (V_B, E_B)$ is a directed graph defined as follows.
- Its vertices are the buffers in the network 
  \[V_B = \{B_i(v) \mid v \in V, 1 \leq i \leq k\}.\]
- The edge set $E_B$ contains a directed edge from $B$ to $B'$ iff there is a route in $RSB$ in which a message is moved from $B$ to $B'$ in one of the steps along the route.

**Lemma:** If the graph $B(RSB)$ is acyclic, then the routing scheme $RSB$ is deadlock-free.
Expanding RS

- A promising way would be to come up with a general technique of starting with a known routing scheme $RS$ and expanding its specification into a buffer-routing scheme $RSB$ in such a way that the resulting scheme is guaranteed to be deadlock-free.
- In view of Lemma, one particular way of doing so would be to ensure that the corresponding buffer graph is acyclic.
- We next describe two general methods for transforming routing schemes into deadlock-free buffer-routing schemes.

Individual Buffer-Routing

- Let $RS$ be a shortest-paths routing scheme for the network $G=(V, E)$.
- Fix $k = n$ (i.e., each vertex $v$ will have $n$ buffers).
- Expand $RS$ into $RSB$ by requiring a message whose destination is vertex $v_i$ to use buffer $Bi(w)$ in every vertex $w$ along its route.
- All messages directed at $v_i$ will travel in “private” buffers dedicated to $v_i$.
- The fact that the routes of $RS$ use shortest paths ensures the following.
- **Lemma:** The resulting buffer graph $B$ is acyclic.
- **Corollary:** The individual routing scheme is deadlock-free.
Increasing class buffer-routing

- Let $RS$ be a shortest-paths routing scheme for the network $G$.
- Fix $k=Diam(G)+1$. Note that for every $u,v \in V$ the route $Y(RS,u,v) = (u=w_1, w_2, ..., w_j=v)$ consists of at most $Diam(G)$ links.
- Expand $RS$ into $RSB$ by requiring a message originated at vertex $v$ to start its way in buffer $B_1(v)$, and for every transition from $u$ to $w$ along its route, if the message is stored at $B_i(u)$ in $u$, then it must enter $B_{i+1}(w)$ in $w$.
- In other words, the buffers visited by any message routed on the network must belong to strictly increasing classes.

Observations

We make the following observation.
- **Lemma**: The resulting buffer graph $B$ is acyclic.
- **Corollary**: The increasing class routing scheme is deadlock-free.