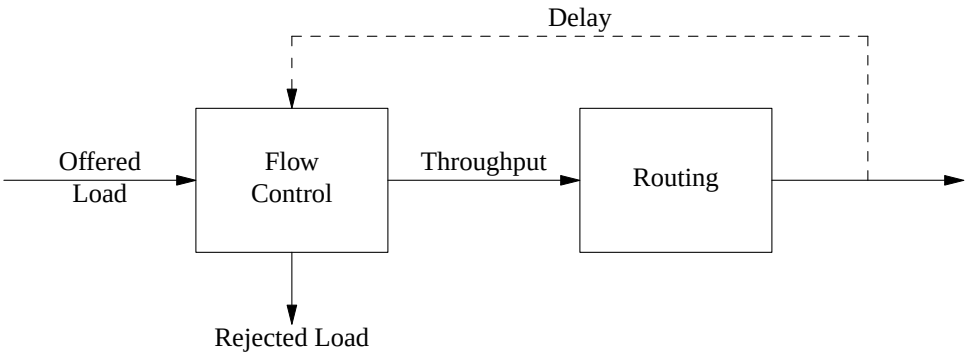
# Chapter VI: ROUTING

## 1. Introduction

1. Network Models
   A. Virtual circuit
      set up before first packet is transmitted
      every packet follows the same path
   B. Datagram
      each packet routed independently
      at each node select the next node
2. Objectives
   A. cope with link and node failures
   B. high performance - avoid congested areas
3. Performance measures
   A. Throughput - quantity of service
   B. Average packet delay - quality of service
   C. Peak delay - delay bounds for real time traffic

## 1.1 Interaction flow control and routing




Low load

traffic fully accepted into the network

throughput = offered load

Heavy load

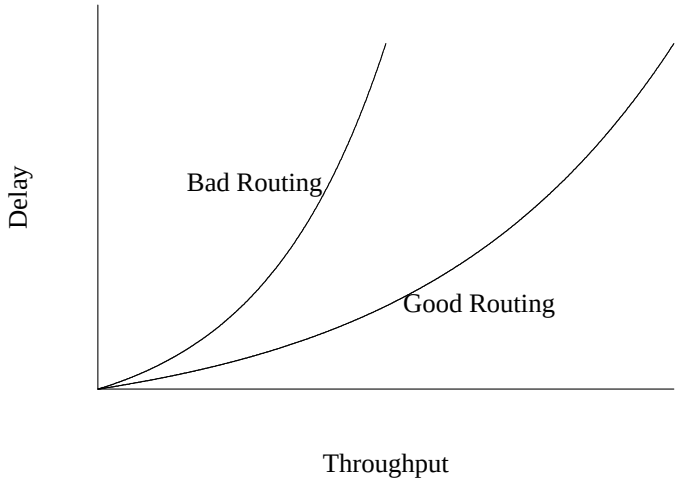
A portion of the traffic rejected by the flow control algorithm

throughput = offered load - rejected load

rejected traffic may just be queued outside the network

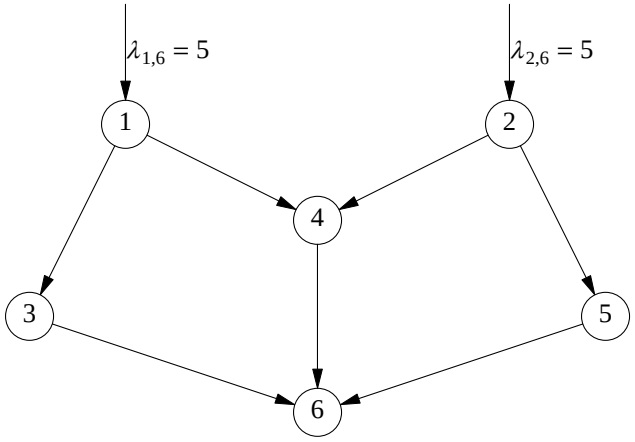
Flow control affected by routing

As the routing algorithm is more successful at keeping the delay low, the flow control algorithm lets more traffic into the network



## 1.2 Examples

Example 1




C = 10
For all Links

If 2->4->6 and 1->4->6, then $\rho_{4,6} = 1$ -> large delay

If 2->5->6 and 1->3->6 then $\rho_{max} = .5$ -> smaller delays.

Example 2

$\lambda_{1,6} = 5$, $\lambda_{2,6} = 15$

If routing is 1->4->6 and 2->4->6 then at least 10 units are rejected when $\rho_{4,6} = 1$.

If routing is 1->3->6 and 2->5->6 then at least 5 units of $\lambda_{2,6}$ is rejected when $\rho_{2,5} = \rho_{5,6} = 1$

if routing is 1->3->6 and 1/2 2->5->6 and 1/2 2->4->6, then all of the traffic may get through

If $\lambda_{1,6} = \lambda_{2,6} = 15$, then the throughput is upper bounded between 10 and 30 depending on the routing rule

Conclusion

Good routing increases the throughput for the same value of average delay under high load

Good routing decreases the average delay under low and moderate load

Objective: Route to keep the average packet delay as small as possible under alls loads

## 2. Shortest Path Algorithms Applied to routing

— Model a network as a graph of nodes and links

— Each link is assigned a weight

— The "best" route between two nodes is the minimum weight path

— The weights we assign to the links, depend upon what we consider to be "best"

— For instance:

1. If each link has weight 1, the minimum weight path has the smallest number of intermediate nodes
   This is important if our cost is dominated by the processing in routers
2. If we assign each link a weight equal to the distance between nodes, the shortest path is the minimum distance path.
   Until recently, private telephone connections were charged by the mile.
3. If we set the weight to the average delay - we pick the path with the smallest delay as the "best" route
   We can use Kleinrock's independence approximation to calculate the delay at each node
4. If we set the weight to 1/C we tend to stay off lower capacity paths
5. If we set the weight to 1/(remaining battery life) we tend to stay away from sensors that are running out of power and prolong the life of the sensor network.
6. If we set the weight to a measure of how congested the link is, we tend to stay away from congested regions in the network. $\left(\frac{1}{C_i - F_i}\right)^i$
7. Finally, we can set the weight to a weighted sum of any of the previous measures.

## 2.1 Dijkstra's algorithm

### 2.1.1 Idea:

Grow a tree from the destination

1. Initially the destination is the only node in the tree
2. At each step find the node, $n_i$, that is not part of the tree, and is closest to the destination along a link from that node and a node in the tree.
3. Connect $n_i$ to the tree.
4. This is the shortest possible path from node $n_i$ and the destination.
   - If there is a node that is closer to the destination it is connected first
   - If a path through that node is shorter, it is selected
5. Once $n_i$ is connected to the tree, it is permanently connected
   There is no shorter path that it can be connected to.

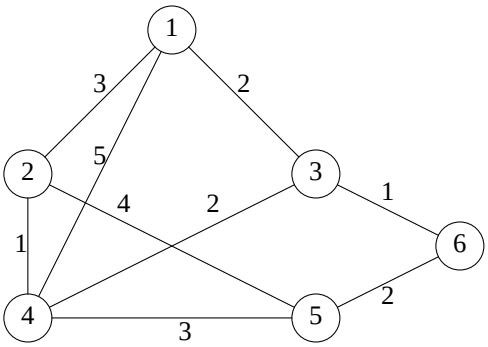
### 2.1.2 Algorithm:

- $P$ = set of nodes that are connected to the destination
  $\bar{P}$ = set of nodes that are not connected to the destination
- Initially:
  - $P = (n_{dest})$, where $n_{dest}$ is the destination node
  - $\bar{P} = (n_j)$, all $n_j \neq n_{dest}$
  - $n_{last} = n_{dest}$, $n_{dest}$ is the last node moved to $P$
  - $D_{last} = 0$, the destination is at distance 0 from itself.
  - $D_j = \infty$ for all $n_j \neq n_{dest}$

1. Calculate the minimum distances to $n_{dest}$ from each $n_j \in \bar{P}$ as.
   $D_j := min[D_j, d_{j,last} + D_{last}]$
2. Find the next closest node to $n_1$.
   Find $n_i \in \bar{P}$ such that $D_i = \min_{j \in \bar{P}} D_j$
3. Move $n_i$ from $\bar{P}$ to $P$.
   $n_{last} = n_i$
   $D_{last} = D_i$
4. Stop if $\bar{P}$ is empty, else go to 1.

**2.1.3 Example:**

$n_{dest} = n_6$




| | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $n_{last}$ | $D_{last}$ | $\bar{P}$ |
|---|---|---|---|---|---|---|---|---|---|
| Initial | ∞ | ∞ | ∞ | ∞ | ∞ | 0 | $n_6$ | 0 | $n_1, n_2, n_3, n_4, n_5$ |
| Step 1 | ∞ | ∞ | 1 | ∞ | 2 | - | $n_3$ | 1 | $n_1, n_2, n_4, n_5$ |
| Step 2 | 3 | ∞ | - | 3 | 2 | - | $n_5$ | 2 | $n_1, n_2, n_4$ |
| Step 3 | 3 | 6 | - | 3 | - | - | $n_1$ | 3 | $n_2, n_4$ |
| Step 4 | - | 6 | - | 3 | - | - | $n_4$ | 3 | $n_2$ |
| Step 5 | - | 4 | - | - | - | - | $n_2$ | 4 | empty |

$D_1 = 3,\ D_2 = 4,\ D_3 = 1,\ D_4 = 3, D_5 = 2$

This algorithm finds the minimum distance from each node to the destination.

To find the path from $n_i$ to $n_{dest}$:

- Initially, $n_{last} = n_i$, $Path = \{n_i\}$
- While $n_{last} \neq n_{dest}$,

  1. Find any $n_j$ such that $D_{last} = D_j + d_{last,j}$

  2. $Path := \{Path, n_j\}$

  3. $n_{last} = n_j$

To find the path from $n_2 \rightarrow n_6$:

- Initially, $n_{last} = n_2$, $Path = \{n_2\}$
- Step 1: $D_2 = D_4 + 1$, $Path = \{n_2, n_4\}$, $n_{last} = n_4$
- Step 2: $D_4 = D_3 + 2$, $Path = \{n_2, n_4, n_3\}$, $n_{last} = n_1$
- Step 3: $D_3 = D_6 + 1$, $Path = \{n_2, n_4, n_3, n_6\}$, $n_{last} = n_1$

What if the algorithm doesn't connect all of the nodes to the tree in less than $d = \infty$?

Notes:

1. This algorithm works for either directed or undirected graphs

2. The version discussed finds the distance from a node to the destination.
   The destination is the root node of the tree.

   We can find the distance from a source node to all other nodes.
   Start with the source node as the root of the tree.
   At each step, calculate the minimum distances from node 1 from each $n_j \in \bar{P}$ as.

$D_j := min[D_j, d_{last,j} + D_{last}]$

Instead of

$D_j := min[D_j, d_{j,last} + D_{last}]$

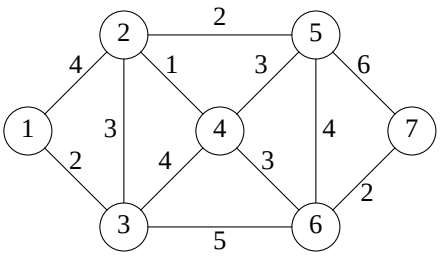
3. Relationship between MST and MDT in Dijkstra's algorithms

In the MST, the node connected to the tree is at distance 0 from the root node

In the MDT, the node connected to the tree is at the distance from the root node

**Home work**

Consider the network with bidirectional links. The links have the same cost in each direction.




A. Use Dijkstra's algorithm to find the Minimum Depth Tree rooted at Node 1.

B. What is the average cost to transmit from nodes 2,3,4,5,6,7 to Node 1?

## 2.2 The Bellman-Ford Algorithm

### 2.2.1 Centrallized Bellman-Ford Shortest Path Algorithm

*2.2.1.1 Idea*

- Initially,
  the destination node, $n_{dest}$ is at dist $D = 0$ from the destination, and all other nodes are at distance $D = \infty$ from the destination.
- At each step:
  - — Calculate the distance from each node, $n_i$, to the destination through each of its 1-hop neighbors, $n_j$ as:
    $D_{i,j,dest}$ = the distance from $n_i$ to its neighbor, $n_j$ plus the distance of $n_j$ to the destination
  - — Set the distance of $n_i$ to the destination as the minimum of $D_{i,j,dest}$, and connect $n_i$ to the node $n_j$ on the minimum distance path, as long as the distance isn't $\infty$.
- In each step a node can be moved to a path that reduces its distance to the destination.
- The algorithm terminates when none of the nodes can find a shorter path to the destination.
  - — If the node was attached to the destination at step k-1, but one of the other nodes changed its distance, and now provides a shorter distance to the destination, then the connection for the node is changed.
  - — Unlike the Dijkstra algorithm, the connections are not permanent.
  - — Since the path lengths are non-negative, we cannot create loops in which the distance is maller on every step.
  - — Since the distance to the destination for each node decreases or remains the same at each step, the algorithm converges to the minimum distance.

*2.2.1.2 Algorithm:*

1. Initially: $D_j(0) = \begin{cases} 0 & j = \text{destination} \\ \infty & j \neq \text{destination} \end{cases}$.

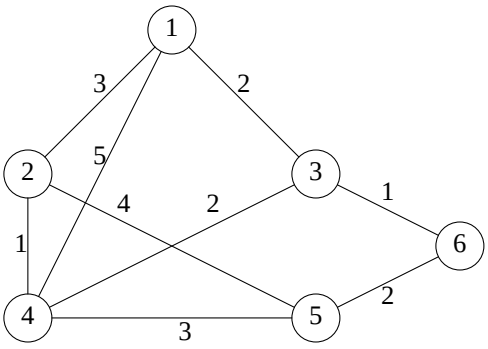
2. At step k:

   For each node "i" calculate $D_i(k) = \min_{j \neq i} \left( D_j(k-1) + d_{i,j} \right)$

   connect $n_i \rightarrow n_{j,min}$.

3. Repeat step 2 until none of the $D_i(k)$ changes.

*2.2.1.3 Example:*
$n_{dest} = n_6$




At step k:

$D_1(k) = \min [\quad D_2(k-1)+3,\quad D_3(k-1)+2,\quad D_4(k-1)+5 \quad ]$
$D_2(k) = \min [\quad D_1(k-1)+3,\quad D_4(k-1)+1,\quad D_5(k-1)+4 \quad ]$
$D_3(k) = \min [\quad D_1(k-1)+2,\quad D_4(k-1)+2,\quad D_6(k-1)+1 \quad ]$
$D_4(k) = \min [\quad D_1(k-1)+5,\quad D_2(k-1)+1,\quad D_3(k-1)+2,\quad D_5(k-1)+3 \quad ]$
$D_5(k) = \min [\quad D_2(k-1)+4,\quad D_4(k-1)+3,\quad D_6(k-1)+2 \quad ]$

| Step | $D_1$ | Conn. | $D_2$ | Conn. | $D_3$ | Conn. | $D_4$ | Conn. | $D_5$ | Conn. | $D_6$ | Conn. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial | ∞ | - | ∞ | - | ∞ | - | ∞ | - | ∞ | - | 0 | $n_6$ |
| 1 | ∞ | - | ∞ | - | 1 | $n_6$ | ∞ | - | 2 | $n_6$ | 0 | $n_6$ |
| 2 | 3 | $n_3$ | 6 | $n_5$ | 1 | $n_6$ | 3 | $n_3$ | 2 | $n_6$ | 0 | $n_6$ |
| 3 | 3 | $n_3$ | 4 | $n_4$ | 1 | $n_6$ | 3 | $n_3$ | 2 | $n_6$ | 0 | $n_6$ |
| 4 | 3 | $n_3$ | 4 | $n_4$ | 1 | $n_6$ | 3 | $n_3$ | 2 | $n_6$ | 0 | $n_6$ |

At each step, a node need only calculate its distance based on the distance of the nodes that it is connected with, which is the basis of the distributed algorithm

### 2.2.2 Distributed Bellman-Ford Algorithm

Reference 1 sec. 5.2.4

*Objective:* To find the shortest path from each node $i$ to each destination, $k$.

- The shortest paths are the solution of Bellman's equation: $D_{i,k} = \min_j[d_{i,j} + D_{j,k}]$ for $i \neq k$

Constraints:

1. The distances are > 0
2. If link (i,j) exists then link (j,i) exists in order to spread the $D_{i,k}$ to the nodes that may use a link.

*Characterisitics of the Bellman-Ford algorithm that make it well suited to a distributed implementation:*

1. Minimizing $D_{i,k}$ is performed independently at each node.
    - In the Dijkstra algorithm the decision is based on a minimization over all nodes.
2. Each node only sends its calculation of $D_{i,k}$ to its 1-hop neighbors.
    - All nodes that are not 1-hop neighbors are at distance $\infty$, and remain at that distance.
3. The algorithm can start assuming that the shortest path is through any of a nodes neighbors and will eventally connect it to its neighbor that provides the minimum distance path
4. The algorithm can operate asynchronously
    - Thus far we have assumed that each node excutes the changes at step $i$. The algorithm also works if each node performs the calculations and informs its neighbors whenever it wants.

*Asynchronous Bellman-Ford algorithm*

- From time to time, each node executes:
  $D_{i,k} := \min_j[d_{i,j} + D_{j,k}]$ using the last estimate of $D_{j,k}$ received from $j$ and its own estimate of $d_{i,j}$.
- Node $i$ transmits its estimate of $D_{i,k}$ to its neighbors
- There is no synchronization for calculating $D_i$, or transmitting that estimate

*Application of the Distributed Belman-Ford Algorithm to finding minimum delay paths.*

- $d_{i,j}$ is the delay from node $i$ to node $j$, which is dependent on the flows through the nodes..
- The $d_{i,j}$ are continuously changing, dependent on the statistical arrivals and the current routing.
- The algorithm doesn't terminate, but modifies the paths as the load on the network changes, and the selected paths changes the flows through the nodes..
- This is the adaptive algorithm that was used in the original APRA-net.

*Implementation:*

1. Each node coneccted to $i$ tells it their expected delay to all destinations.

   For each destination $k$, node $i$ has:

   - $D_{j,k}$ = an estimate of the shortest delay from node $j$ to the destination $k$, and,
   - $d_{i,j}$ = the delay on the link from i to j
     for all nodes $j$ that are connected to node $i$.

2. Node $i$ calculates its minimum distance to $k$ as $D_{i,k} := \min_j \left( d_{i,j} + D_{j,k} \right)$.

3. At some later time, node i sends its estimate $D_{i,k}$ to its connected neighbors

   - The reporting is either done periodically, or aperiodically, when a significant change occurs in its expected delay to a destination.

**Home work**

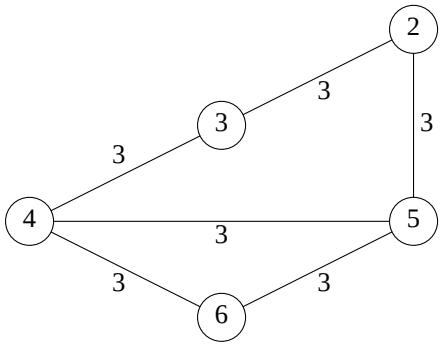
Distributed Bellman Ford Algorithm

— The link costs are $Cost_{i,j} = \begin{cases} 1.1 + \dfrac{F_{i,j}}{C_{i,j} - F_{i,j}} & \text{for } 0 \le F_{i,j} < C_{i,j} \\ \infty & \text{elsewhere} \end{cases}$

a combination of minimum hops and minimum network delay. Where $F_{i,j}$ and $C_{i,j}$ are the flow and capacity on link $i, j$.

— At each time 1, 2, 3, ... each node calculates its minimum cost path to each destination and reports the value to each of its neighbors. Its neighbors use these values in the next round. Any flows that arrive at a node do not modify the cost calculations until the next round.

— The network has bidirectional links with the capacity shown in each direction.




A. Before time t=0, there are no flows on the network.

i. At each node, except node 2, what is the cost to node 2 on each link from the node?

ii. What are the best links from each node to node 2?

B. At time $t = 0 + \varepsilon$ a flow, $\gamma_{6,2} = 1$ from node 6 to node 2 is inserted into the network.

i. What path does $\gamma_{6,2}$ follow, and what does node 6 believe is the cost of that path, $Cost(\gamma_{6,2})$, to deliver the flow to the destination?

ii. Recalculate the path costs, best links, and $Cost(\gamma_{6,2})$ after the application of the distributed Bellman-Ford algorithm at times t=1 and t=2.

C. At time $t = 2 + \varepsilon$ flow, $\gamma_{5,2} = 1$, is inserted into the network

i. What path does $\gamma_{5,2}$ follow, and what does node 5 believe is the cost, $Cost(\gamma_{5,2})$ to deliver the flow to the destination?

ii. Recalculate the path costs, best links, $Cost(\gamma_{5,2})$, $Cost(\gamma_{6,2})$, and the path for $\gamma_{6,2}$ at times t=3, t=4 and t=5.

# 3. Routing Before the ARPAnet - 1960's

Reference 1, section 5.1.2, and reference 2

**1960's** Processing and storage were much more expensive

## 3.1 Outline of Techniques

### 3.1.1 Directory Procedures

Reference 3

*3.1.1.1 Tables at each node*

- The table contains the next node on the path to the destination
- The tables entries can be pruned to save storage for infrequently used paths,
  but the table entries must be re-established when the path is requested
- Infrequent packets, to destinations not in the table, can be sent to a nearby node that knows paths to all destinations
- Used in the telephone network, the ARPAnet, and the current Internet

*3.1.1.2 Source routing*

- The source node obtains a path from a directory
- The path is transmitted in each packet, instead of being maintained at intermediate nodes.
- Different packets to the same destination can take different paths at an intermediate node
- Used in the ARPAnet. This mechanism is still part of the Internet standard, but is not being implemented by any of the major router manufacturers.

### 3.1.2 Routing to Survive Nuclear Attack

Paths are uncertain

*3.1.2.1 Flooding*

Implementation

- Transmit on all links from the source node and each intermediate node on the path.
- hop count limits
  forward up to diameter of the network
- don't transmit on link received
- nodes can keep list of recent messages xmitted and don't forward messages that have been forwarded
  each message has a unique id, that is the same for all copies of the packet

Applications

- Broadcasting information on the Internet
- Discovering paths in mobile ad hoc networks
  packets carry path used by this copy of the message

*3.1.2.2 Random Routing*

Reference 4

— At each hop select next link at random

— Extremely inefficient, but can be modified to be almost as efficient as directory procedures

### 3.1.3 Early adaptive techniques

Reference 5.

*3.1.3.1 Hot Potato Routing*

- variable size messages
- Don't queue the messages at the intermediate nodes
  - — If the desired path is busy take another path
    *The message is deflected*
  - — When the transmission rates of all of the links are the same, and the in-degree = out-degree at every node, no messages are lost because there is no storage
    Whenever a messages arrives at a node, at least one output is available
  - — In a network with bi-directional links, the in-degree always equals the out-degree
- Routing without storage is important in all optical networks, where storage is expensive.

*3.1.3.2 Backward Learning*

- Messages carry hop count information from the nodes it has visited, to each of the destinations it can track.
- The current node uses this information to determine its shortest paths.
- Early approximation to Bellman-Ford
- Pruned table based routing
  Tables are maintained by messages that are passing through the network

Back ward learning is used today in Ethernet Bridges

- Objective: To be able to connect an Ethernet terminal anywhere and have other terminals find it and communicate with it, without maintaining a central directory.
- Ethernet LAN's are connected together with bridges.
  - — Each bridge has 2 or more LAN's connected to it.
- The bridges form a spanning tree
  - — All LAN's can reach all other LAN's, but there are no loops.
- Until a terminal sends a message, none of the bridges know its location.
  - — A bridge forwards a message for this terminal on all of the LAN's, except for the LAN on which it received the message
  - — The message is broadcast on all LAN's
- When a terminal transmits a message, a bridge hears which LAN the message is received on, and does not forward future messages for this terminal on the other LAN's connected to the bridge.
  - — When $T_A$ transmits to $T_B$ for the frst time, the message may be broadcast on all LAN's
  - — When $T_B$ responds to $T_A$, the bridges heard the message from $T_A$, and only forward the message on the LAN's needed to reach $T_A$.
  - — Subsequent messages from $T_A$ to $T_B$ only follow the LAN's needed for the connection, since the bridges on the path have heard the response.
- Memory in the bridges times out and terminal direction information is removed so that the terminals can be moved to other parts of the network.

- There is increasing interest in "Carrier Grade Ethernets", which use Ethernet bridges and Ethernet connection protocols on global networks.
  - Ethernet bridges are much simpler than routers

## 3.2 Development of Select Techniques

### 3.2.1 Random Routing

1. Efficiency

- $N$ = number of nodes in the network
- $P_A$ = Prob of arriving at the destination on the next transmission
  $P_A = \frac{1}{N-1}$
- $\bar{L}$ = average number of hops to reach destination
  $P_i = (1-P_A)^{i-1}P_A$ = Prob of arriving at the destination on the $i^{th}$ transmission

$$\bar{L} = \sum_{i=1}^{\infty} i(1-P_A)^{i-1}P_A$$

$$= P_A \frac{d}{d(1-P_A)}\left(\sum_{0}^{\infty}(1-P_A)^i\right)$$

$$= P_A \frac{d}{d(1-P_A)}\left(\frac{1}{1-(1-P_A)}\right) = \frac{1}{P_A}$$

$$= N-1 \approx N$$

2. What if we know our neighbors?

- each time we move to a neighboring node we learn a set of nodes, instead of the single node we've moved to. We can move to one of those nodes directly.
- Most networks have distance < 10, knowing nodes to dist 2 or 3 makes the average path length in random routing similar to that in directory routing

*3.2.1.1 Example: Each node knows its 1-hop neighbors*

— Given:

- $N$ nodes in the network
- The destinations is equally likely to be any of the remaining $N-1$ nodes.
- Each node is connected to $K$ nodes
- Random Networks
  The $K$ nodes that a node is connected to are picked at random,
  - — Random networks are often used in comparisons rather than networks with a specific design.
  - — random networks are not a realistic model of many older networks, because nodes were usually connected to nodes that are closer in distance.
  - — Random networks are a better model of modern, DWDM (dense wavelength division multiplexing) networks with wavelenght routers, because each of the wavelenghts is routed to different, more distant, destinations
- Each time a packet is forwarded to the next node, $K-1$ new nodes are exposed.
  There are no cycles of length ≤ 3 in the network that would reduce the number of new nodes that are exposed.

— When a node enters at the source, the probability that the source is 1-hop from the destination is $P_1 = \frac{K}{N-1}$. (The path lenght is 1 with probability $P_1$.)

— Given that a node is not within 1-hop of the destination after the $(i-1)^{th}$ transmission, the probability that it is within 1-hop of the destination after the $i^{th}$ transmission is $P_A = \frac{K-1}{N-(K+1)}$

- K-1 new nodes are exposed when we move to the next node
- The previous node and its $K$ 1-hop neighbors are not one of the $K-1$ new nodes

— The average path length is:

$$\begin{aligned}
\bar{L} &= 1 * P_1 + 2 * (1-P_1)P_A + 3 * (1-P_1)(1-P_N)P_A + 4 * (1-P_1)(1-P_A)^2 P_A + \cdots \\
&= P_1 + (1-P_1)\sum_{i=1}^{\infty}(i+1)(1-P_A)^{i-1}P_A \\
&= P_1 + (1-P_1)P_A\sum_{j=2}^{\infty} j(1-P_A)^{j-2} \\
&= P_1 + \frac{1-P_1}{1-P_A}\left(P_A\sum_{j=0}^{\infty} j(1-P_A)^{j-1} - P_A\right) \\
&= P_1 + \frac{1-P_1}{1-P_A}\left(\frac{1}{P_A} - P_A\right) \\
&= P_1 + (1-P_1)\frac{1+P_A}{P_A} \\
&= \frac{K}{N-1} + \left(1-\frac{K}{N-1}\right)\frac{N-2}{K-1}
\end{aligned}$$

— When $N \gg K$, $\bar{L} \approx \frac{N}{K-1}$, and the average path length is reduced in proportion to the number of neighbors that are exposed on each step.

**Home work**

Random Routing:

- Each node knows its 1-hop and 2-hop neighbors
- The network has $N$ nodes
- Each node is connected to $K$ other nodes at random with the constraint that there are no cycles that reduce the number of neighbors exposed on each step.

A. What is the average path length?

B. What is the approximate average path length when $N \gg K$.

### 3.2.2 Deflections in Hot Potato Routing

Deflections decrease the network throughput

— All links have the same capacity

— The link capacity is 1 unit

1. Consider a store-and forward network with K links, and an average shortest path length of $\bar{L}$ links.

   - Without deflections, the maximum network throughput is $S_{\max} = \frac{K}{\bar{L}}$.
   - $S_{\max}$ occurs when the traffic can be distributed so that $\rho = 1$ on all of the links
   - In a store-and-forward network, the queue length, and the average delay $\rightarrow \infty$ at $\rho = 1$

2. Consider a network with hot potatoe routing, and no storage

   A. Deflected packets take a longer path, which increases the utilization of each link and decreases the throughput.

   - If each deflection increases the path length by an average of $\bar{D}$ links, and the probability of deflection is $P_D$, then the average path length, increases to:

     $$\bar{L}' = \bar{L} + P_D\bar{D}\bar{L}'$$
     $$= \frac{\bar{L}}{1 - P_D\bar{D}}$$

     In order for $L' < \infty$, $P_D < 1/\bar{D}$.

   - Increasing the average path length increases the average link utilization to $\rho' = \frac{\bar{L}'}{\bar{L}}\rho$, and $\rho = (1 - P_D\bar{D})\rho'$
   - Since, $\rho' \leq 1$, $\rho \leq (1 - P_D\bar{D})$,
   - The maximum throughput decreases to $S'_{\max} = \frac{K}{\bar{L}'} = (1 - P_D\bar{D})S_{\max}$.

   B. Deflections tend to equalize the utilization of the links

   — When a link is busy, we take another link rather than waiting for the busy link

   — Deflections tend to move traffic from more busy links to less busy links

*3.2.2.1 Example 1:*

Consider a network with

- $j$ inputs and outputs at each node,
- a utilization of $\rho'$ on each link, and
- a unique shortest path to each destination at each node

a. $P_D$ is proportional to $\rho'$

- The probability that $i$ of the other $j-1$ inputs are busy when a packets arrives is $P_i = \binom{j-1}{i}(\rho')^i(1-\rho')^{j-1-i}$
- When $i$ inputs are busy, $i$ of the outputs are also busy.
- The probability that an incoming packet is deflected given $i$ outputs are busy when it arrives is $P_{D/i} = \frac{i}{j}$,

- and, $P_D = \sum_{i=0}^{j-1} \frac{i}{j} P_i = \frac{j-1}{j} \rho'$
- As packets are deflected, $\rho'$ increases, which increases $P_D$.

b. The maximum link utilization, $\rho_{\max}$, which does not include the additional traffic caused by deflections, is $\frac{j}{4(j-1)\bar{D}}$

- $\rho_{\max}$ occurs at $\rho' =< 1$
- $\rho$ is related to $\rho'$ as:

$$\rho = (1 - P_D \bar{D})\rho'$$
$$= \left(1 - \frac{j-1}{j} \bar{D}\rho'\right)\rho'$$

- The maximum value of $\rho$ occurs at

$$\frac{d\rho}{d\rho'} = 1 - \frac{2(j-1)}{j} \bar{D}\rho' = 0$$
$$\rho' = \frac{j}{2(j-1)\bar{D}}$$

- And,

$$\rho_{\max} = \left(1 - \frac{1}{2}\right)\frac{j}{2(j-1)\bar{D}}$$
$$= \frac{j}{4(j-1)\bar{D}}$$

c. Hot potato routing reults in a throughput $S'_{\max}$ that is significantly less than store-and-forward throughput $S_{\max}$

i. If j=4, and $\bar{D} = 6$

- $\rho_{\max} = \frac{4}{4 * 3 * 6} = .056$, instead of 1
- And, $S'_{\max} = .056 S_{\max}$

ii. If $j = 2$, as in the Manhattan Street Network, and $\bar{D} = 4$,

- $\rho_{\max} = \frac{2}{4 * 1 * 4} = .125$
- And, $S'_{\max} = .125 S_{\max}$

*3.2.2.2 Example 2:*

- In many networks there are equal length shortest paths

  Consider a 2-connected network, where the probability that both paths to the destination have equal length is $P_{EQ}$.

- When there are several paths that a packet can take, the probability that a packet is deflected decreases

  In the 2-connectred network, the probability of deflection is reduced from $\frac{\rho'}{2}$ to $\frac{(1-P_{EQ})\rho'}{2}$.

  $$\rho = \left(1 - \frac{(1-P_{EQ})\bar{D}}{2}\,\rho'\right)\rho'$$

- Taking the derivative with respect to $\rho'$, $\rho_{\max}$ occurs at $\rho' = \frac{1}{(1-P_{EQ})\bar{D}}$

- If $(1-P_{EQ})\bar{D} < 1$, $\rho' > 1$, and $\rho_{\max}$ occurs at $\rho' = 1$.

- Therefore,

$$\rho_{\max} = \begin{cases} \dfrac{1}{2(1-P_{EQ})\bar{D}} & \text{if } P_{EQ} \le 1 - \dfrac{1}{\bar{D}} \\ 1 - \dfrac{(1-P_{EQ})\bar{D}}{2} & \text{if } P_{EQ} > 1 - \dfrac{1}{\bar{D}} \end{cases}$$

- In Example 1, if $\bar{D} = 4$, and $P_{EQ} = \frac{1}{3}$, $S'_{\max}$ increases from $.125S_{\max}$ to $.1875S_{\max}$.

# 4. Internet Era Networks

1970's-1990's - includes special topologies, and ATM
The cost of storage and processing decreased.
Early days of fiber opitcs - communications capacity is still limited

## 4.1 Adaptive routing based on Global Information

### 4.1.1 Basis for adaptation

A. topology - shortest paths
uses least network resources

B. expected delay - shortest delay paths

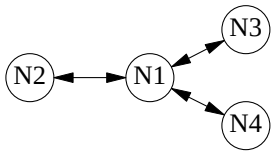
C. combination of delay and path length
Find low delay while limiting resources that are used

### 4.1.2 Duration of adaptation

A. Packet by packet adaptation

- Datagram networks - original ARPAnet
- Each node reports its expected delay to each destination as the
  min ( delay on path i + avg queueing delay to path i )
  for all paths i connected to the node.

  A node calculates its minimum delay to each destination




For each destination:

1. Nodes 2,3,4 send $d_{2,dest}$, $d_{3,dest}$, $d_{4,dest}$ to Node 1
   Their minimum delay to the destination
2. Node 1 calculates its delay to the destination as $d_{1,dest} = \min_{i=2,3,4}(d_{1,i} + d_{i,dest})$
3. Node 1 sends Nodes 2,3,4 $d_{1,dest}$, and they perform the calculation
4. Eventually, every node knows its minimum delay to the destination.
5. If one of the links breaks it may take several rounds of communications for nodes to find the proper path to a destination.

*This is a Synchronous Version of the Distributed Bellman Ford Algorithm*

- The packets in a message may have to be resequenced at the destination
- This technique was adequate for the 19 node ARPAnet, but did not scale as the number of nodes increased
- There are research efforts to find an hierarchical approach that will work in the current Internet

B. All of the packets in a message

The Beeforth protocol has been used to guarantee packet ordering on multi-hop paths with hop-by-hop retranmissions

C. Until topology changes or congestion problem occurs

OSPF weights in Internet

Path can change within a message -> resequencing

D. All messages in a session

Virtual circuits

ATM networks

no resequencing

### 4.1.3 Other Issues

A. Reporting frequency

1. Periodic - how often?
   - Too frequent -> wasted capacity
   - Too infrequent -> stale information
2. Asynchronous - when a significant change occurs at a particular node
   - This is the Asynchronous, distributed Bellman-Ford Algorithm
   - What is a *significant* change?
   - Changes percolate through the network until the change at the starting node does not affect the decisions at distant nodes

B. Decisions

1. centralized
   - network control center
   - make decisions for all nodes with knowledge of other decisions
   - best decisions for available information
2. distributed
   - use more current information - closer to the point where the information is applied
   - ie. - a node knows its current queue length, while the central controller does not

## 4.2 Hot potato derivatives

### 4.2.1 Shortest Q + Bias

- shortest Path unless long wait
- The path length is the bias
- hot-potato with packet storage

### 4.2.2 Delta routing

- least delay path unless long wait
- least delay is slowly varing estimate - not actual network delay
- delta is adjusted to pick actual shortest delay based upon the delay at the current node
- ie: At node A:

  the average delay on path 1 is $D_1$,

  the average delay on path 2 is $D_2 > D_1$

  $\delta = D_2 - D_1$

  the current queueing delay for path 1 is $Q_1$

  the current queueing delay for path 2 is $Q_2$

  *Rule*: Use path 1 if $Q_1 - Q_2 < \delta$,
  Otherwise use path 2

### 4.2.3 Deflection Routing

- Fixed size, lined up cells - more efficient than hot-potato
  - — not deflected by a packet that is almost done using a path
  - — make decisions on all of the packets at a node simultaneously to take better advantage of packets that can take either path
- particularly useful on networks where

  1. The network has many equal length paths to a destination

  2. the length of the next best path is only slightly longer than the best path

- Random selection of equal length paths
  Random selection of packet that is deflected

  — Prevents deadlocks or livelocks caused by unexpected traffic distribution

  — Prevents deadlock caused by inaccurate information about the topology following a link or node failure

- We will compare the throughput of deflection routing and hot-potato routing by example
- Effect of small amount of storage

  — With fixed size packets, we can use an optical fiber delay line in all optical networks

  — Consider the 2-connected network with a single buffer

  - When the buffer is empty, and 2 packets arrive, that must use the same link, one is stored in the buffer and neither is deflected
  - When the buffer is full, and 1 packet arrives that requires the same link, that packet is inserted in the buffer, as the packet in the buffer is forwarded. Neither packet is deflected.

- When the buffer is full, and 2 packets arrive that don't both require the same link as the packet in the buffer, one of the packets is forwarded and the other is stored in the buffer. No packets are deflected.
- The only time that a packet is deflected is when the 2 arriving packets and the packet in the buffer all require the same link.
- It has been shown that in the Manhattan Street network, with 1 buffer and $P_{EQ} \approx .33$, $S'_{\max} > .9S_{\max}$.

## 4.3 Alternatives to adaptive routing

Spread messages over entire network to try to load it up uniformly, rather than picking a single path that may become overloaded
These techniques can adapt to network conditions

### 4.3.1 Selective flooding

- entire message on several paths
- Increases load

### 4.3.2 Proportional Routing

- Send a packet on each path from a node with assigned probabilities
- If paths are equal length, there is no increase in load
- Gallager defined the problem of finding the proper proportions as the "Optimal Routing" problem

### 4.3.3 Dispersity routing

Non-Redundant

| $C_1$ | $C_2$ | $C_3$ | $C_4$ |
|---|---|---|---|
| $b_0$ | $b_1$ | $b_2$ | $b_3$ |
| $b_4$ | $b_5$ | $b_6$ | $b_7$ |
| $b_8$ | $b_9$ | $b_{10}$ | $b_{11}$ |
| ... | ... | ... | ... |
| $b_{4i+0}$ | $b_{4i+1}$ | $b_{4i+2}$ | $b_{4i+3}$ |

Redundant

| $C_1$ | $C_2$ | $C_3$ | $C_4$ |
|---|---|---|---|
| $b_0$ | $b_1$ | $b_2$ | $p_1$ |
| $b_3$ | $b_4$ | $b_5$ | $p_2$ |
| $b_6$ | $b_7$ | $b_8$ | $p_3$ |
| ... | ... | ... | ... |
| $b_{3i+0}$ | $b_{3i+1}$ | $b_{3i+2}$ | $p_i$ |

Xmit: $p_i = \sum_{j=1}^{3} C_{j,i} \bmod 2$

Rcv: $C_{k,i} = \sum_{j \neq k} C_{j,i} \bmod 2$

1. Non-redundant - 4,4 - smaller delay than proportional routing

- $D = \dfrac{1}{\mu C(1-\rho)}$
- if $\dfrac{1}{\mu}$, the average packet length is reduced to $\dfrac{1}{4\mu}$,
- but the number of packets increases by 4, so that the utilization $\rho$ remains the same,
- The $D$ on each path decreases by the degree of the dispersion ( 4 in this case.)

- The delay from the source to destination does not decrease by 4.
  At the destination we must wait for the packet on the slowest path before reconstructing the original message
  If all paths have the same distribution of delays, waiting for the slowest path is equivalent to waiting for the largest of 4 order statistics.
- Would we have the same result if we divided each packet into 4 parts and sent each part on the same link - one after the other?
  No. Why?

2. Selective flooding - 4,1

- The entire message is transmitted on all 4 paths
- The link utilizations increases, so that the delay on each path increases
- We only wait for the first of the 4 messages to arrive at the destination
  If the 4 paths have the same delay distribution, this is the first of 4 order statistics.
  If 3 of the 4 paths fail, we still receive the packet.

3. Redundant Dispersity Routing

— In the example we use a 4,3 system
  This is a simple parity check code
  We could use any *erasure correcting* code

— We don't wait for last packet at the receiver

- Not waiting for the last packet may reduce the average and variance of delay
- The average packet length is reduced by 1/3,
- However, the number of packets is increased by 4
  The total traffic increases
  The average delay at a node increases
  If the utilization of the links is greater than 1, the average delay becomes infinite
  In the (4,3) system, what is the maximum link utilization in the non-redundant system?
- We only wait for the first 3 of the 4 packets to receive the message

  — When the delay distribution on each path is independent and identically distributed(iid), the delay is the 3rd largest of 4 order statistics

  — In general, when the delay density on each path is f(y),

  (The delay distribution is $F(y) = \int_0^y f(x)dx$),

  the delay density of the $k^{th}$ largest of $n$ order statistics is

$$g(y) = \left( \frac{n!}{(k-1)!\,(n-k)!} \right) [F(y)]^{k-1} [1 - F(y)]^{n-k} f(y).$$

- (4,3) dispersity routing can survive single link failures and avoids waiting for the longest of the 4 paths when the delay distributions on each path are different.

4. Dispersity routing is more secure than conventional packet routing

- Intercepting the packet on a path only provides a fraction of the bits in the entire message
- When security is a concern, we should use redundancy reduction on the message before applying dispersity routing so that the missing bits cannot be determined from the intercepted bits.

**Home work**

Routing

— In each of the following problems assume that Kleinrock's independence assumption holds,

— calculate

a. The utilization of each of the 8 links,

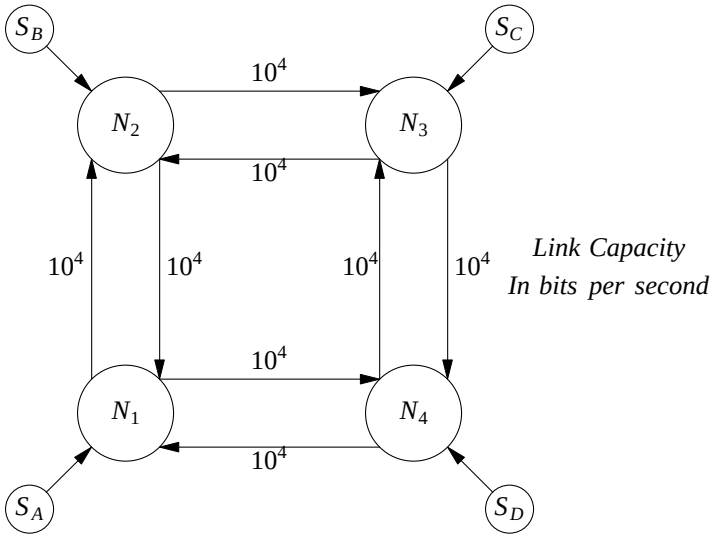
b. The average delay on each of the 8 links

c. The average delay on each route from the 4 sources to their destination.

d. And the average network delay.

— the messages are exponentially distributed with an average message lenght of $10^3$ bits

— use the following network




— each source presents a Poisson arrival process of 4.5 messages per second.
the destination for $S_A$ is $S_C$, the destination for $S_B$ is $S_D$, the destination for $S_C$ is $S_A$, and the destination for $S_D$ is $S_B$

A. Directory routing

$S_A$ follows the path $N_1 \rightarrow N_2 \rightarrow N_3$

$S_B$ follows the path $N_2 \rightarrow N_3 \rightarrow N_4$

$S_C$ follows the path $N_3 \rightarrow N_4 \rightarrow N_1$

$S_D$ follows the path $N_4 \rightarrow N_3 \rightarrow N_2$

B. Proportional Routing

For $S_A$ half of the packets follow the path $N_1 \rightarrow N_2 \rightarrow N_3$ and
half of the packets follow the path $N_1 \rightarrow N_4 \rightarrow N_3$

For $S_B$ half of the packets follow the path $N_2 \rightarrow N_3 \rightarrow N_4$ and
half of the packets follow the path $N_2 \rightarrow N_1 \rightarrow N_4$

$S_C$ half of the packets follow the path $N_3 \rightarrow N_4 \rightarrow N_1$ and
half of the packets follow the path $N_3 \rightarrow N_2 \rightarrow N_1$

$S_D$ half of the packets follow the path $N_4 \rightarrow N_1 \rightarrow N_2$ and
half of the packets follow the path $N_4 \rightarrow N_3 \rightarrow N_2$

C. Flooding

One copy of each message is transmitted on each of the paths specified in part B.

i. Is the average delay from each source to each destination <, =, or > than the average delay on the two paths? Why?

D. Dispersity Routing

Each source divides each message into 2 equal lenght sub-messages that are half the length of the original message, and sends each sub-message on each of the paths specified in part B.

i. Is the average delay from each source to each destination <, =, or > than the average delay of the sub-messages? Why?

## 4.4 Development of Select Techniques

### 4.4.1 Distributed Asynchronous Bellman-Ford Algorithm Applied to Minimum Delay

Reference 1 sec. 5.2.4

Used in original ARPANET.
Objective: To find the shortest delay path to the destination

- Find the shortest delay, $D_i$, from each node to a destination node $n_{dest}$.
- The distances are the solution of Bellman's equation:
  $D_i = \min_{j \in N}[d_{i,j} + D_j]$ for $i \neq dest$.

Constraints:

1. The delays are > 0; this is always true.

2. If link (i,j) exists then link (j,i) exists, but the link may have different delays in each direction.

Characterisitics:

- The nodes don't need to know the network topology, but only their 1-hop neighbors
  - — $D_i(k+1)$ is calculated at each node independently at each step, then each node sends this information to its 1-hop neighbors for the next iteration.
- every node forms a separate delay estimate to each destination.
  - — If there are $N$ nodes in the network, there are $N-1$ separate instances of the Bellman-Ford Algorithm running.
- The link $\rho$'s change as the algorithm changes the paths.
  - — Since the algorithm can start assuming that a node routes data to the destination through any link, if the utilization and the delay changes, then the algorithm changes the connection to the link that provides the minimum delay
- Since the algorithm can operate asynchronously, it can update its neighbors only when it has a significant change in its delay calculation - percolation of updates
  - Update neighbors when the delay on a path changes based on information from a neighbor
  - Update neighbors when the delay on a path that it uses increases because another node selects or deselects this node to forward data or because the data from the local source increases.
  - Don't update neighbors when an update from a neighbor does not change the delays to destinations because the change is not on a path that this node has selected.

#### *4.4.1.1 Stability of Adaptive Shortest path Routing Algorithms*

Reference 1 sec. 5.2.5

**Oscillations between 2 paths in a datagram network**

- Path 1 is congested, so traffic is routed along path 2.
- In the next cycle, all of the traffic is routed along path 2, so it becomes congested, and all of the traffic is routed along path 1.
- In the following cycle, all of the traffic is routed along path 1, so it becomes congested. ···

- The result is that traffic is always routed along the congested path.

**This is demonstrated on a bidirectional ring network:** Reference 1 fig. 5.38, pg 412-3

- The network starts with a reasonably good routing rule, that is slightly unbalanced. The counter clockwise direction has slightly more traffic.
- At the next step, some of the traffic is re-routed along the less congested path - clockwise direction
- This causes the clockwise direction to become congested, so on the next cycle, more of the traffic travels in the counter clockwise direction.
- Eventually, on alternate cycles, all of the traffic travels in the clockwise direction on one cycle and in the counterclockwise direction on the next cycle.
- The suggested solution is to give each link a cost $\alpha$ in addition to the delay. This discourages longer paths and stabilizes the algorithm.
- The problem is that the traffic that we re-route changes the delays and we need to stabilize the algorithm.
  - — As $\alpha$ becomes large the shortest path is selected and congestion plays almost no part.
  - — In the Bellman-Ford algorithm, the cost of a path is the weighted sum of the congestion and the path length.
  - — In the current Internet, the OSPF weights are the bias factor. OSPF weights are selected by system administrators. Most of the time the weights are selected to eliminate the effect of congestion. The B-F algorithm is only used to bypassing failed links.
- In most real networks, there are a large number of destinations and re-routing a single source does not change the network. All of the destinations are not changed at the same time.
- An alternate way to stabilize the network, when a single destination is significant is to use proportional routing and just change a fraction of the traffic routed along each path. This gets away from the Bellman-Ford algorithm.

### 4.4.2 Comparison of Deflection Routing and Hot-potato Routing

**Example**

— Two-connected network

— A packet can only be deflected if

i. It cares which path it takes, $(1 - P_{EQ})$

ii. A packet arrives on the other incoming link, $\rho'$,

iii. The other packet cares which path it takes, $(1 - P_{EQ})$, and,

iv. The other packet selects the same path, $\frac{1}{2}$

— When two packets select the same path, we flip a coin to decide which is deflected, therefore, the probability that a packet that can be deflected is deflected, is $\frac{1}{2}$.

— The probability of deflection is $P_D = \frac{1}{4}(1 - P_{EQ})^2 \rho'$

— The utilization relationship becomes:

$$\rho = \left(1 - \frac{(1 - P_{EQ})^2}{4} \bar{D} \rho'\right) \rho'$$

— $\rho_{\max}$ occurs at $\rho' = \frac{2}{(1 - P_{EQ})^2 \bar{D}}$

— When $P_{EQ} > 1 - \sqrt{\frac{2}{\bar{D}}}$, $\rho' > 1$, and $\rho_{\max}$ occurs at $\rho' = 1$.

— Therefore,

$$\rho'_{\max} = \begin{cases} \dfrac{1}{(1 - P_{EQ})^2 \bar{D}} & \text{if } P_{EQ} \le 1 - \sqrt{\dfrac{2}{\bar{D}}} \\ 1 - \dfrac{1}{4}(1 - P_{EQ})^2 \bar{D} & \text{if } P_{EQ} > 1 - \sqrt{\dfrac{2}{\bar{D}}} \end{cases}$$

— If $\bar{D} = 4$, and $P_{EQ} = 0$, $S'_{\max} = .25 S_{\max}$, as compared with $.125 S_{\max}$ with hot potato routing.

- This shows the advantage of a slotted system.

— If $\bar{D} = 4$, and $P_{EQ} = \frac{1}{3}$, $S'_{\max} = .55 S_{\max}$, as compared with $.1875 S_{\max}$ in hot-potato routing

(note $P_{EQ} > 1 - \sqrt{\frac{2}{\bar{D}}}$)

- Which shows that we can take better advantage of packets that have multiple equal length paths.

**Home work** Deflection Routing: Consider a 2-connected network with 1 buffer, in which $P_{EQ} = 0$.

A. Find the probability that the buffer is empty or full as a function of $\rho'$

B. Find the probability of deflection, as a function of $\rho'$

C. Find $\rho_{\max}$ when $\bar{D} = 4$.

### 4.4.3 Optimal Proportional Routing

Reference 1 section 5.4

Model:

- There are several source-destination (S-D) pairs in a network
- Each S-D pair has a small number (possibly 1) route that it can use, and must decide how to split its flow over the designated routes

We will first consider the routing problem:

- We assume that the network is able to transfer all of the flows between the S-D pairs, and our objective is to minimize a network cost function

In the flow control section, we will then jointly consider the routing and flow control problem

- The flows from the S-D pairs may be greater than the network is able to transfer, and we must decide both how to split the flows on the allowed paths and also how much to limit the flows from the different S-D pairs.
- We will show that the optimal routing and flow control problem is actually the same as the optimal routing problem.

#### *4.4.3.1 Optimal Routing Equations*

We are given

1. A set of S-D pairs
   W = { w }
2. The set of flows required by each *w*
   R = { $r_w$: where $r_w$ is the required flow on *w* }
3. The set of paths that *w*, can use.
   $P_w$ = { $p_w$: where $p_w$ is a directed path from the source to the destination for *w* }

Problem

- In the optimal routing problem, our objective is to place *all* of the flows on the paths in a way that minimizes a network cost function.
- The flow $r_w$ is split among the allowed paths $P_w$ with flow $x_{p_w}$ transmitted on path $p_w \in P_w$, such that
  - $\sum_{p_w \in P_w} x_{p_w} = r_w$, the entire flow is assigned to the allowed paths, and,
  - $x_{p_w} \geq 0$ for all $p \in P_w$, the flows are not negative.
- The flow, $F_{i,j}$, on a directed link (i,j) is the sum of the flows on paths that traverse the link
  $$F_{i,j} = \sum_{w} \sum_{\substack{all\ paths\ p_w \\ containing\ (i,j)}} x_{p_w}$$
- A cost $D_{i,j}(F_{i,j})$ is associated with each link
  - The cost is a monotonically increasing function of the flow $F_{i,j}$
  - The cost function for each link $(i, j)$ can be different.
    In particular, it can take into account the characteristics of the link, such as its capacity $C_{i,j}$ or its propagation delay $d_{i,j}$.
- The cost function, that we want to minimize, is
  $$C_N = \sum_{(i,j)} D_{i,j}(F_{i,j})$$

- Example
  - Let

$$D_{i,j} = \begin{cases} \dfrac{F_{i,j}}{C_{i,j} - F_{i,j}} + d_{i,j}F_{i,j} & \text{for } F_{i,j} \le C_{i,j} \\ \infty & \text{for } F_{i,j} > C_{i,j} \end{cases}$$

    — This is the average delay on a link when we use Kleinrock's independence approximation

    — When the propagation delay is negligible $d_{i,j} = 0$

    — The feasible flows are $F_{i,j} \le C_{i,j}$

  - $C_N = \sum_{(i,j)} \left( \dfrac{F_{i,j}}{C_{i,j} - F_{i,j}} + d_{i,j}F_{i,j} \right) = \gamma \bar{D}_N$, for feasible flows.
    where

    — $\bar{D}_N$ is the average network delay using Kleinrock's approximation, and

    — $\gamma = \sum_w r_w$ is the total throughput of the network

    $\gamma$ is constant for all distributions of the flows $r_w$ on the links $P_w$ in the optimal routing problem.

  - Therefore, minimizing $C_N$ in this example is equivalent to minimizing the average network delay.

*4.4.3.2 Finding the Optimal Routes*

1. Rewrite $C_N$ explicitly in terms of the flows $x_{p_w}$ that we can move between paths:

$$C_N(\vec{x}) = \sum_{i,j} D_{i,j} \left( \sum_w \sum_{\substack{\text{all paths } p_w \\ \text{containing } (i,j)}} x_{p_w} \right)$$

   where $\vec{x}$ is the vector of the flows $x_{p_w}$ that are currently assigned to the network

2. Take the partial derivative of $C_N(\vec{x})$ wrt a particular flow $x_{p_w}$ as:

$$\frac{\partial C_N(\vec{x})}{\partial x_{p_w}} = \sum_{\substack{\text{all links } (i,j) \\ \text{on path } p_w}} \frac{\partial D_{i,j}}{\partial x_{p_w}}$$

   Note: The summation is only over the links that carry $p_w$, because the derivatives of the other links wrt $x_{p_w}$ are zero.
   The range of the summation will be useful when we consider a practical implementation

3. If we move a small amount of traffic, $\delta$, from path $p_w$ to $p'_w$, where $p_w, p'_w \in P_w$, the change in the cost is:

$$\delta \frac{\partial C_N(\vec{x})}{\partial x_{p'_w}} - \delta \frac{\partial C_N(\vec{x})}{\partial x_{p_w}}$$

4. If $\vec{x}$ is the optimal routing assignment, we cannot decrease $C_N(\vec{x})$ by moving some of the flow from $p_w$ to $p'_w$, therefore

$$\frac{\partial C_N(\vec{x})}{\partial x_{p'_w}} \ge \frac{\partial C_N(\vec{x})}{\partial x_{p_w}}$$

5. If $\vec{x}$ is the optimal routing assignment, we also cannot decrease $C_N(\vec{x})$ by moving some of the flow from $p'_w$ to $p_w$, therefore

$$\frac{\partial C_N(\vec{x})}{\partial x_{p'_w}} = \frac{\partial C_N(\vec{x})}{\partial x_{p_w}} \text{ for all } p_w, p'_w \in P_w \text{ for which } x_{p_w} \ne 0 \text{ and } x_{p'_w} \ne 0.$$

- At the optimal solution, the first derivative on all $p_w \in P_w$, for which $x_{p_w} > 0$, must be the same
- We may not use some of the available paths in $P_w$ if the flows on those paths, from other $w'$ congests those links.
- In this case, the first derivatives on the unused paths is greater than the first derivatives on the paths that we do use.

$$\frac{\partial C_N(\vec{x})}{\partial x_{p'_w}} \geq \frac{\partial C_N(\vec{x})}{\partial x_{p_w}} \text{ for all } p_w,\ p'_w \in P_w \text{ for which } x_{p_w} \neq 0 \text{ and } x_{p'_w} = 0.$$

6. If $\vec{x}$ is optimal, the equality and inequality in (5) must hold for all of the $w \in W$,

- The sum of the first derivatives for two flows, $w$ and $w'$, on the paths that these S-D pairs use may be different.

$$\frac{\partial C_N(\vec{x})}{\partial x_{p_w}} \neq \frac{\partial C_N(\vec{x})}{\partial x_{p_{w'}}} \text{ for } x_{p_w} \neq 0 \text{ and } x_{p_{w'}} \neq 0.$$

*4.4.3.3 A distributed implementation*

The gradient descent algorithm

- Each packet has 2 fields,
  - — When a source sends a packet on $p_w$ it inserts $x_{p_w}$ that it assigns to the path in the first field
  - — The second field is initially zero, and is incremented by $\frac{\partial D_{i,j}}{\partial x_{p_w}}$ at each node $i, j$ that the packet traverses.
  - — The value of the second field, $\sum_{path} \frac{\partial D_{i,j}}{\partial x_{p_w}}$, is returned to the source in the acknowledgement.
- The source
  - — Occasionally sends test packets on paths $p'_w \in P_w$ for which $x_{p'_w} = 0$
  - — The source moves flow from the paths with the largest $\sum_{path} \frac{\partial D_{i,j}}{\partial x_{p_w}}$ to the paths with the smallest $\sum_{path} \frac{\partial D_{i,j}}{\partial x_{p_w}}$, the minimum first derivative path (MFDL), to equalize the $\sum_{path} \frac{\partial D_{i,j}}{\partial x_{p_w}}$ and to remove flows from the worse paths.
- The nodes
  - — Keep track of the $x_{p_w}$ in the packet headers, and calculates $F_{i,j} = \sum_{p_w} x_{p_w}$.
  - — For any $p_w$ that passes through the node, $\frac{\partial D_{i,j}}{\partial x_{p_w}} = \frac{\partial D_{i,j}}{\partial F_{i,j}} \frac{\partial F_{i,j}}{\partial x_{p_w}} = \frac{\partial D_{i,j}}{\partial F_{i,j}}$
  - — When

$$D_{i,j} = \begin{cases} \frac{F_{i,j}}{C_{i,j} - F_{i,j}} + d_{i,j} F_{i,j} & \text{for } F_{i,j} \leq C_{i,j} \\ \infty & \text{for } F_{i,j} > C_{i,j} \end{cases}$$

$$\frac{\partial D_{i,j}}{\partial x_{p_w}} = \begin{cases} \frac{C_{i,j}}{(C_{i,j} - F_{i,j})^2} + d_{i,j} & \text{for } F_{i,j} \leq C_{i,j} \\ \infty & \text{for } F_{i,j} > C_{i,j} \end{cases}$$

— The node increments the second field in the packet by this amount

- Each S-D pair, $w$, moves it flows independent of each other to its MFDL paths
- Optimal routing only occurs when all of the flows for each $w$ are on MFDL paths.
- The algorithm continues to operate as the $r_w$ change and $w$ enter or leave the network.

*4.4.3.4 The Frank-Wolfe Flow Deviation Method*
Reference 1, section 5.6.1

A centrallized procedure that attempts to find the "best" step size, in order to converge to the optimum solution more quickly.

Start with a feasible path flow vector $\vec{x} = \{x_p\}$,
A "feasible" solution is a flow assignment for which $F_{i,j} \le C_{i,j}$ for all $i$, $j$

1. Find a MFDL path for each $w$.
   The first derivatives are evaluated at the flows, $F_{i,j}$, resulting from $\vec{x}$.

2. Let $\vec{x}^* = \{x_p^*\}$ be the flows if ALL of the flow for each $w$ is moved to its MFDL path.
   It is possible that $C_N(\vec{x}^*) > C_N(\vec{x})$.

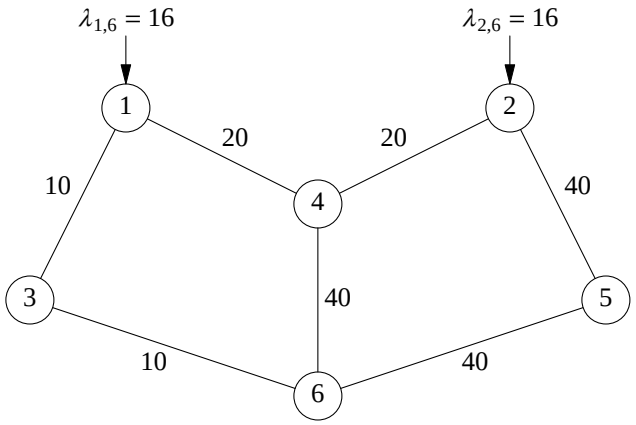
3. Move the part of the flow from $\vec{x}$ to $\vec{x}^*$ that results in the best solution.
   That is, pick $0 \le \alpha \le 1$ so that
   $$C_N[x + \alpha_{opt}(\vec{x}^* - \vec{x})] = \min_{0 \le \alpha \le 1} C_N[\vec{x} + \alpha(\vec{x}^* - \vec{x})]$$

4. The new flow is $\vec{x} := \vec{x} + \alpha(\vec{x}^* - \vec{x})$
   Go back to step 1

In this technique, the same step size is used for all $w$. The procedure can converge faster if we use different step sizes $\alpha_w$ for each $w$.

**Home work**

Optimal Routing: Consider the network, with the link capacities as shown:




The flow $\lambda_{1,6}$ can take path 1->4->6 or 1->3->6, and the flow $\lambda_{2,6}$ can take path 2->4->6 or 2->5->6.
Initially, each source splits its flow equally among the two available paths.

The cost of link i,j is $D(F_{i,j}) = \dfrac{F_{i,j}}{C_{i,j} - F_{i,j}}$

a. What is the network cost?

b. For each source find the MFDL path and indicate which direction we should shift the flow from that source

c. For each source, shift 1 unit of flow and recalculate the network cost.

## 4.5 Classification Techniques

Used to:

1. Explain relationships between routing algorithms

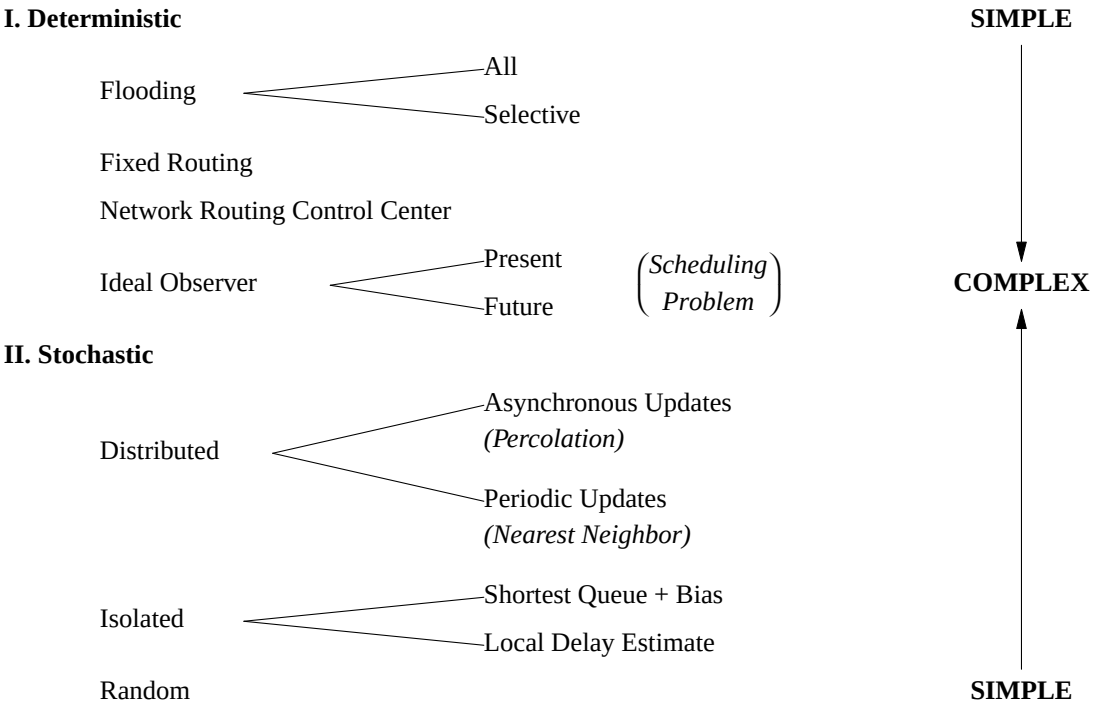
2. Show where a new mechanism fits in with existing mechanisms

### 4.5.1 Fultz and Kleinrock

Reference 6
Classification Categories

— Deterministic vrs Stochastic

— Complexity



### 4.5.2 Rudin - 3 dimensional plot

— to explain where delta routing fits in with other routing mechanisms

— NRC is a network routing center that obtains information from all of the nodes in the network. The information represents the average operation of the network, and is described as "lethargic" as the network changes

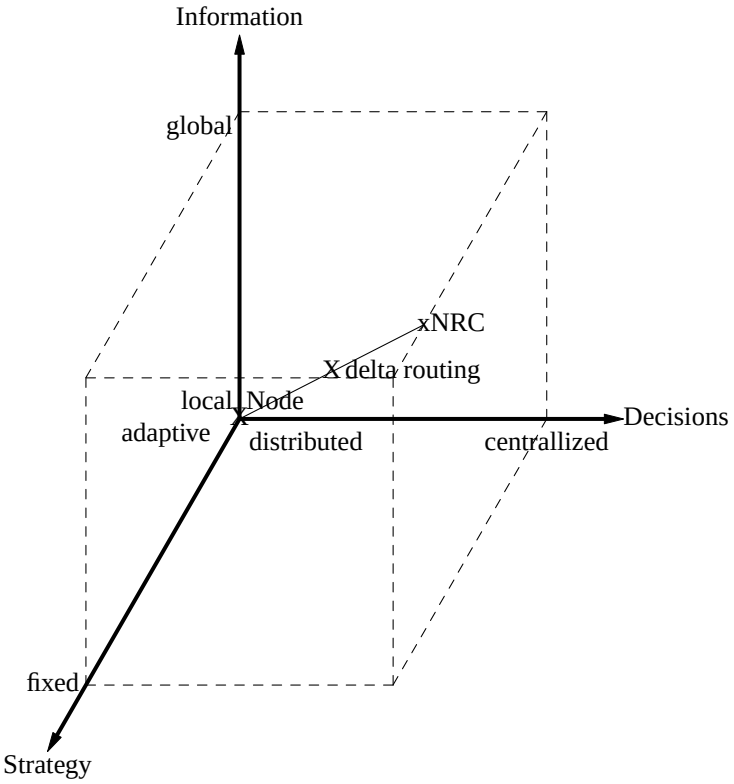
— "Node" are operations that are performed using local information at a node.

— The variable $\delta$ is the degree to which an algorithm depends on the local or global information to select a route.

— Reference 7

— Plot

- x axis - decisions: distibuted -> centralized
- y axis - information: local-> global
- z axis - stategy: adaptive->fixed




— putting points on this graph is harder than it seems, and is subjective

— ie: there are different types of information used in routing - path, delay, ...

— hot-potato uses global information on the topology, but local information on the delay.

— random routing doesn’t have global information on the topology, and can only use local information on q lengths

— random routing with nearest neighbors is somewhere between the two.

# 5. Current Work on Routing

## 5.1 Moble, Ad hoc Networks - MANET's

Reference 8

### 5.1.1 Source Routing

— The path is contained in the packet

— Difference: Finding and maintaining paths

1. Find Complete Path
   - The source uses flooding to determine all routes to the destination
   - The "best" of the potential routes is selected by the destination from all flooding packets that arrive.
   - A new route must be selected when a node on the path moves
2. Find Partial Path (Dynamic Source Routing - DSR)
   - If a packet reaches a node that already has a path to the destination, that intermediate node terminates the flooding and responds to the source
   - The path specified by the intermediate node is a concatenation of its path to the destination and the "best" path it receives from a flood packet
   - The source may obtain several responses from the intermediate nodes and the destination.
   - The source picks the "best" route from the responses

### 5.1.2 Bellman-Ford Modifications

*5.1.2.1 Destination Sequenced Distance Vector (DSDV)*

- Each node uses the Distributed Bellman-Ford Algorithm with periodic updates to determine the paths with the smallest number of hops to a destination.
- Periodically, each node informs its 1-hop neighbors how long the path is to all other nodes.
- If a node was a direct neighbor, and moves, its distance is changed to $\infty$, until a new path is discovered.

*5.1.2.2 Wireless Routing Protocol - WRP*

- Same as DSDV except the updates are both periodic and asynchronous
- An asynchronous update occurs when there is a change in 1-hop neighbors

*5.1.2.3 On Demand Routing*

- Provides a means of scaling the Bellman-Ford algorithm to networks with a larger number of nodes but with limited storage and bandwidth.
- The tables are pruned when entries are not used.
  - — When paths to a destination are not used for a "significant" amount of time they are removed from the table at a node
  - — This reduces the storage at nodes to the entries that are used at that node, and reduces the bandwidth that is used to send updates between the nodes.
- The table entries are restored when a message requires a path to a destination that is not stored at the node.
  - — The node floods the network with a request to connect to the destination.

— Nodes with table entries to the destination, and the destination, send their estimates of the number of hops to the destination to all of their neighbors, and forces their neighbors to re-establish the table entries.

— The table netry for the destination is re-established at all of the nodes in the network

— Eventually, the table entry is once again pruned at the nodes that do not require a path to that destination.

*5.1.2.4 Fish-eye State routing (FSR)*

- Reduce the amount of data that is transmitted to update the tables without eliminating the table entries.

  The eye of a fish sees more detail near the focal point than further away from the focal point

- In a mobile network, the path we take toward a destination that is far away changes very slowly, even though the destination node is moving.
  Example: If a destination node is in a different city, we may use the same paths to get to that city, independent of whether or not the destination is moving from street to street.
- As we get closer to the destination, the final few links change more rapidly
- In the Bellman-Ford algorithm we don't update the table entries for nodes that are far away as frequently as we do for nodes that are nearby.

  References 9, 10

**5.1.3 Geographic Routing**

— Assume that nodes have GPS units and know their location

*5.1.3.1 Simple Geographical Routing*

- *Objective:* Survive Changing Paths that occur when nodes along the path between the source and destination move
- At each intermediate node forward the packet to the next node that makes the greatest forward progress toward the destination, instead of to a specific node
  Try to follow a straight line from the source and destination
- Geographic routing fails if at any node along the path there isn't a node that is closer to the destination in the transmission range, or if the destination moves.
- If the destination moves, we can

  — Use flooding techniques to discover the new location of the destination,

  — Have the destination register its new location with a network control center, or

  — Have the destination leave a forwarding direction

*5.1.3.2 Routing through intermediate forwarding nodes*

- *Problem:* A physical obstable, such as a building or a lake, blocks transmission toward a destination or a lack of forwarding nodes results in no neighbors being closer to the destination.
- Find a set of intermediate nodes that have geographic routing paths from the source to an intermediate node, between the intermediate nodes, and from an intermediate node to the destination.

  — The intermediate nodes can be found by using flooding and graph techniques
- The intermediate nodes are stored in the packet, as in source routing.

  — Geographic routing is used to send the packet from the source to the first intermediate node.

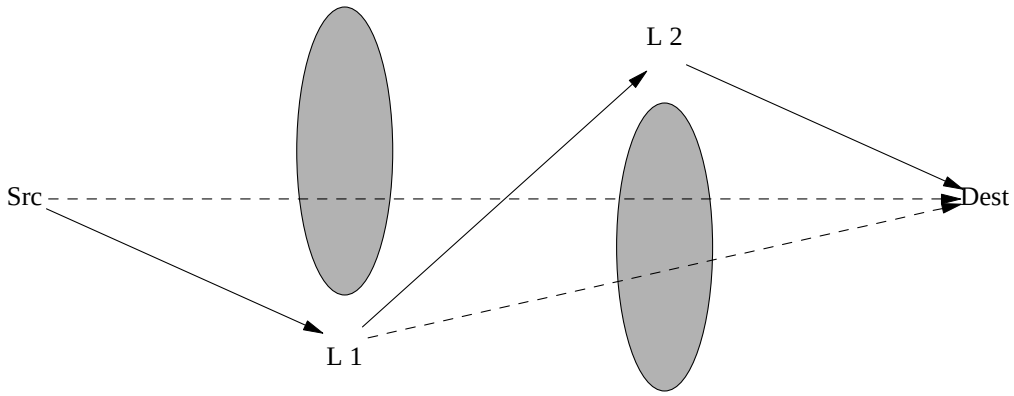
— At the frst intermediate node, the next intermediate node is removed from the packet, and geographic routing is used to send the packet to that node.

— The process is repeated until the packet reaches the destination.

— This technique survives node movement along each of the path segments, since we do not route through specific nodes.

- The forwarding node may be a specific node, or the location of a group of nodes.

— By using the location of a group of nodes, rather than a specific node, we can survive movements by individual forwarding nodes.




**5.1.4 Robotic Routing**

Robotic routing is used in conjuction with geographic routing to find a path when the packet cannot find a node closer to the destination.

*5.1.4.1 Description of Operation*

If a packet encounters an obstacle, it uses the right hand rule, a technique used in robotics, to find a path around the obstacle.

— In robotics, the robot places its right hand on the obstacle and moves around the obstacle.

- The robot can trace the perimeter of any obstacle, no matter how complex the shape.

— In a wireless network we cannot trace the perimeter of an obstacle, but must jump between forwarding nodes.

- The first time that we use the right hand rule, we point the fingers of our right hand toward the destination, and rotate our fingers until they point toward an intermediate forwarding node that we can reach.
- On successive applications of the right hand rule, we point our fingers toward the node that we arrived from and rotate our hand until it points toward another forwarding node. that we can reach.

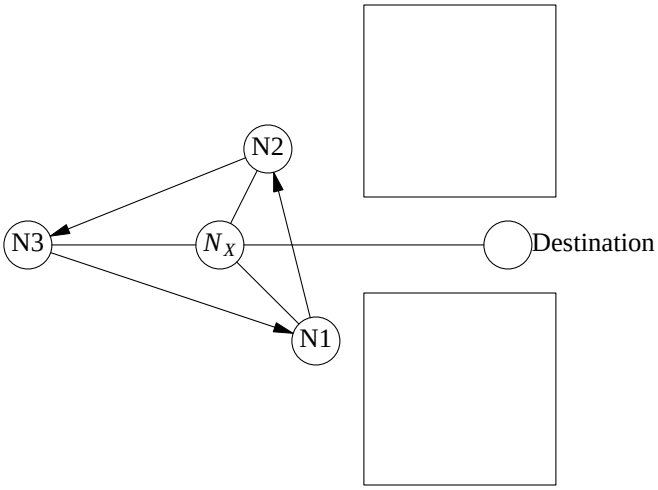
— In robotic routing we continue the procedure until we find a node that is closer to the destination than the starting node, and then resume geographic routing.

- If we return to the starting node, and select the first forwarding node as the next node, there is no path to the destination.

— Since the packet always progesses during geographic routing and is closer to the destination at the end of a robotic segment than at the beginning, the packet always progresses toward the destination.

*5.1.4.2 Problem:*

The rule may go into a routing loop and not find a path to the destination.




*5.1.4.3 Planar Graphs:*

- The bypassed nodes and cycles all have the characteristic that the paths in the cycle cross one another.
- There are no cycles in the route if and only if we restrict the set of nodes that we can forward to, so that the links between the forwarding nodes form a planar graph, and do not cross.
- There are distributed techniques that guarantee that the forwarding nodes form a planar graph. Unfortunately, the techniques require extensive communications between nodes
- Reference 11
- In the figure, a planar graph is formed if we do not allow transmission between nodes N1 and N2.

*5.1.4.4 A Simple Solution*

We can guarantee loop free operation by dividing the surface into a grid and using the right hand rule to forward between adjacent grid elements

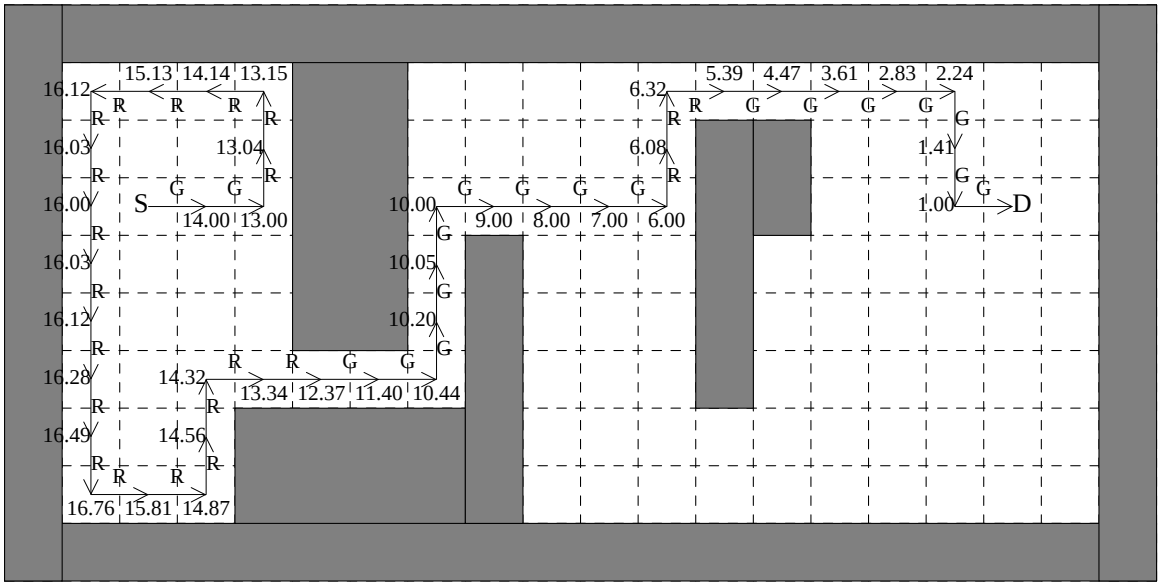
— Communications is used between nodes in a grid element so that all of the nodes in a grid element know which adjacent grid elements can be reached by any of the nodes in the grid element

— We perform robotic routing between grid elements, rather than specific nodes.

— If the nodes are in adjacent squares in the grid model, the grid elements selected always form a planar graph.
The adjacent squares are above, below, to the left and to the right of the current grid element.

— This technique also has the advantage that we avoid many small steps when tracing an obstacle and reduce the number of forwarding nodes

- Reference 12




— We can also allow communications between grid elements on the diagonal from the current grid element 8 possible choices for forwarding grid elements

- When the right hand rule is applied to the 8 choices, the resulting graph of the selected path is always planar, even though there are crossing edges.
- This results in fewer instances where a forwarding path cannot be found

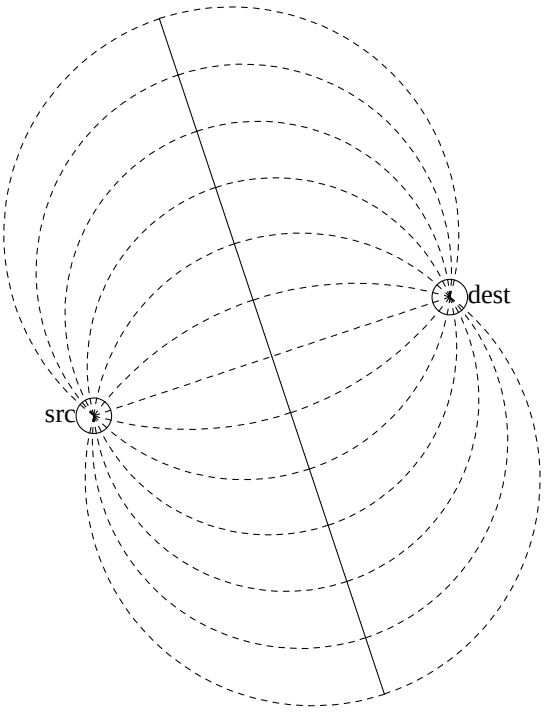
### 5.1.5 Dispersity Routing

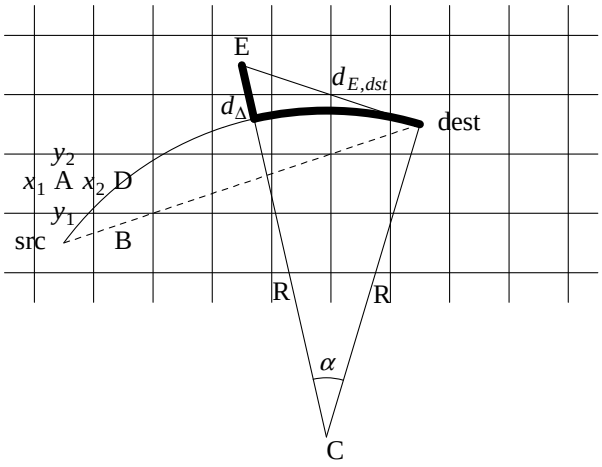
Difference: Finding and maintaining multiple paths
Flooding or Partial Flooding is used to find several disjoint paths to the destination

References 13, 14, 15.

An alternative way to determine multiple paths is "Arc Routing" which uses geographic routing along an arc instead of a straight line.

- Robotic routing is used to find a path around obstacles
- The distance to the destination is measured as the distance to the arc "+" the distance along the arc to the destination
- This distance metric causes messages to move back to the original arc when they move away from the arc or after they circumvent an obstacle.




*5.1.5.1 Non-redundant Dispersity Routing*

- More secure in military networks
  Intercepting the data on one path does not provide information
- Uses cut-set of capacity to destination
  More capacity is available between a source and destination with dispersity routing than in a single path routing procedure
  This is particularly important in wireless networks
- After a failure: Redistribute load over remaining paths instead of looking for a new path
  Failures are common in ad hoc networks because a path fails when any of the intermediate nodes move.
- Greater Dispersion for anycast than point-to-point communications
  - — In anycast, a source routes to any of a set of destinations, rather than a specific destination
  - — This paradigm is used in wired networks when data is stored at several locations in the network, or when a processing resource exists in several locations
  - — The paradigm is used in wireless networks when the main objective is to connect with the wired network
    There are normally several gateways where the wireless network connects with the wired network

*5.1.5.2 Redundant Dispersity Routing*

- Continues to operate as paths fail - up to the redundancy of the system

**5.1.6 Hierarchical Routing**

— The number of nodes in a mobile network may be too large to use many of the table based approaches

*5.1.6.1 Cluster Switch Gateway Routing*

- Architecture

- — Nodes are organized into clusters
- — One node is designated as the cluster head
- — Adjacent clusters have gateways that have membership in the two clusters
- — This is similar to the piconets that are organized into scatternets in Bluetooth networks
- — Unlike Bluetooth, the clusterhead may not communicate directly with each node in the cluster, but may go through several intermediate nodes in the cluster.

- Operation
  - — The routing tables specify the route to the cluster that contains the destination, rather than the complete path to the destination.
  - — If a node moves within a cluster, the route to the cluster does not need to be changed.
  - — Routing between clusters may be
    - Gateway -> clusterhead -> gateway, as in Bluethooth scatternets, or
    - Gateway -> gateway
  - — DSDV is used to route between clusters
    - If the metric used in the Bellman Ford algorithm is 1 for each cluster, the path a packet traverses is the minimum number of clusters
    - If the metric is the number of hops between gateways, the path traversed is the minimum number of hops
  - — Mobility may cause the gateways and cluster heads to change
    - If the gateway changes, the cluster head determines a new gateway or removes the connection between adjacent clusters
    - One cluster may be split into 2 clusters if the nodes within a cluster move apart
    - Two clusters may be combined into a single cluster if 2 clusters move together

*5.1.6.2 Geographical Clusters*

- Route to location of clusters
- Use any available nodes

*5.1.6.3 Landmark Routing*

- Clusters are moving groups of nodes
- Application: Set of nodes that move as a group.
  ie. Tanks on a battlefield
- There is a group leader - the landmark - that reports its movement to the other groups
- Packets for a node are first routed to the landmark
  Accurate path information is maintained from the nodes in the group to the landmark.
- An hierarchical structure can be established in which the landmarks form a larger group with a single landmark
- References

  [1] Reference 16

### 5.1.7 Network Discovery

The locations of nodes and the path between them is continuously changing in mobile networks. Routing a packet from a source and destination requires locating the destination and determining the nodes along the path.

The information that must be determined depends on the routing mechanism:

1. In source routing, the source must determine a complete list of the intermediate nodes that will forward the packet.
   This is accomplished by finding the complete path to the destination or by finding a partial path to an intermediate node that knows a path to the destination.
2. In geographic routing, the source must determine the location of the destination, and the intermediate forwarding points if a packet cannot make forward progress at all of the intermediate nodes.
3. In table based routing, when unused table entries are pruned, the table entries on the path to the destination may have to be restored.
4. In dispersity routing, the source must determine a set of maximally disjoint paths to a destination.
5. In anycast, the source must locate any one of a set of receivers.

The following techniques are used to determine information:

1. Network Control Centers (NCC)
   - The nodes register with control centers that maintain a map of the network topology.
     A node notifies the control center when it moves, or its set of one-hop neighbors changes
   - A source contacts an NCC to determine the location of, or paths to, a destination
   - Similar to the Internet
   - Susceptible to failures or disconnects with the NCC's
   - Does not operate well in rapidly changing networks
2. Flooding
   This procedure is used in most ad hoc network applications, and will be expanded in the next section.
3. Robotic Discovery
   Used to determine intermediate forwarding points in geographic routing.
   - Flooding is used to locate a destination.
   - Robotic routing is used to circumnavigate and map obstacles.
   - The map of the obstacles encoutered are returned in the acknowledgement from the destination, intermediate forwarding points are identified that result in shorter paths on subsequent transmissions.
4. Seven Degrees of Separation
   - Each node knows a path to a number of destinations, those it has communicated with recently.
     A node knows its friends, rather than its neighbors.
   - If a node does not know a path to a destination, it uses partial flooding, or random selection, to find a node that knows a path to the destination.
     - The nodes it contacts can be to its neighbors or its friends.
     - ie. A node asks its friends if the destination is one of its friends, or it asks its neighbors if the destination is one of its friends
     - If a node knows that one of its friends has an extrodinarily large number of friends, this is the node it should try first.

- This is similar to random routing with knowledge of neighbors, only instead of knowing your nearest neighbors, you know the people whom you've communicated with recently

### 5.1.8 Flooding

*5.1.8.1 Techniques*

1. In source routing,
   - Each flood packet accumulates the list of intermediate nodes that it traverses, and possibly a metric for the path such as the utilization, transmission energy or battery life of the intermediate node.
   - The destination selects a path, from all of the copies of the flood packet that it receives, and returns the selected path to the source by using the reverse path.
2. In geographic routing without obstacles, and robotic routing,
   - The flood packet only contains the location of the source.
   - When the destination receives a flood packet from the source, it uses geographic routing to communicate with the source and returns its own location.
3. In geographic routing with obstacles,
   - The flood packet contains the location of the source and collects the locations of the intermediate nodes.
   - The receiver determines which of the intermediate nodes can be reached by geographic routing, and eliminates them from the list.
     The remaining intermediate nodes are forwarding points.
   - The receiver returns the list of forwarding nodes to the source.
4. In table based routing, with pruned tables,
   - The flood packet contains a hop count
   - When a packet with a table entry for the receiver, or the receiver, obtains a packet, it uses an asynchronous update in the Bellman Ford algorithm to extend the table out to the hop count

*5.1.8.2 Reduction of Flood Messages*

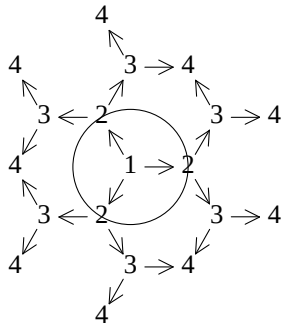
Flooding occurs frequently in ad hoc networks. A great deal of work has been done to reduce the number of messages generated during flooding.
Reference 17

*5.1.8.2.1 Local Techniques*

1. Only forward flood packets whose path length is less than the diameter of the network
2. When a node receives multiple copies of a flood packet, only forward one flood packet
3. If the receiver will decide on the path based on the path length or another metric, delay forwarding a flood packet until most of the copies of the flood packet from other nodes have been received.
   — At time $\Delta$ after the first flood packet is received, forward the flood packet with the smallest metric.
   — If a flood packet with a smaller metric is received after this time, it must also be forwarded.
4. An intermediate node that knows a path to the receiver can terminate the flood packet
   — The intermediate node sends a concatenation of the flood path and its path to the source
   — The source selects the best path received from the intermediate nodes

*5.1.8.2.2 Limit the number of forwarding nodes in dense networks*

- A flood packet is broadcast on the radio channel to all of the nodes in a transmission radius, the transmitting nodes 1-hop neighbors.
  - — If a node knows its one-hop neighbors, and one of its neighbors is the destination, it addresses the flood packet to the destination, and none of the other nodes in the neighborhood forward the packet.
- Only a few of the 1-hop neighbors are selected to retransmit a flood packet.
  - — The objective is to select the nodes that forward the packet so that every node in the network can receive at least one copy of the packet, and there are as few copies forwarded as possible.
- The fewest flood messages are forwarded when the selected nodes are at the maximum transmission distance, and are separated by $120^o$.
  - — With this pattern, every node in the network receives at least one copy of the flood message, so that we are certain to find the destination.
  - — The source sends the flood packet to the 3 neighbors that are $120^o$ apart.
  - — Fowarding nodes only send the flood packet to 2 neighbors.
    The node that they received the packet from is one of the $120^o$ spokes.
- The pattern of retransmissions of flood packets form the edges in a hexagonal array.
  The same hexagonal array arises in optimal frequency reuse cellular networks




- With the hexagonal retransmission plan the number of flood messages $\approx 2A_{net}/A_{hex}$
  - — Each node in the corners of the hexagon forwards a message once
  - — There are 6 nodes in the corners of each hexagon, and each node occurs in 3 hexagons. Therefore, the number of forwarding nodes is twice the number of hexagons.
  - — In dense networks the number of flood packets is a function of the area of the network, and not the number of nodes in the network.

*5.1.8.2.3 Zone Routing Protocol - ZRP*

- ZRP reduces the number of flood messages when a node also know its 2-hop or 3-hop neighbors.
  - — The neighbors that a node knows is the nodes zone
  - — When a node knows more than its one-hop neighbors, the area of the hexagon increases, and the number of flood messages decreases.
- When a forwarding node receives a flood message it checks the nodes in its zone.
  - — If the destination is in its zone, it forwards the message to the destination.
  - — If the destination isn't within its zone, it forwards the flood message to the nodes that are at the maximum distance away, and $120^o$ apart.
    These are the border nodes in its zone.

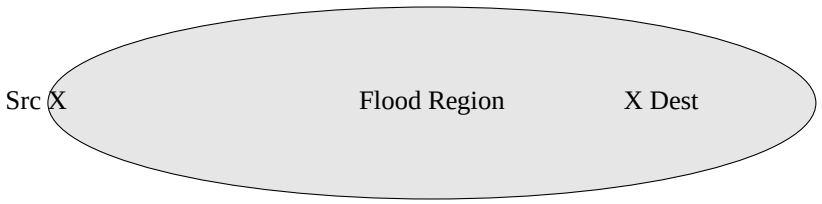
— If the border node has received this location request, it discards it, as in normal flooding

*5.1.8.2.4 Progressive Flooding based on hop count*

- First flood to all receivers that are i hops away,
  If we don't locate the receiver, flood to all receivers that are j>i hops away
  Continue increasing the hop count until we find the receiver, or reach the span of the network.
- Progessive flooding is particularly useful in geographic routing when a receiver moves.
  — It is likely that the node that moved is close to its last known location.
  — Use progressive flooding around the last known location


*5.1.8.2.5 Location Aided Routing - LARS*

- Assume that the source knows the approximate direction to the destination, and the approximate distance to the destination.
- Flood messages are constrained to a geometric area that most likely contains the destination.
- Flood messages that fall outside the geometric area are discarded to reduce the number of flooding messages
- If the destination is not located on the first attempt, the geometric shape can be enlarged until it contains the entire world
- Reference 18




**Home work**

Classification of Ad hoc routing

There are many types of ad hoc networks, with different characteristics.

A. Consider networks with different rates of changes, caused by different amounts of mobility. Go through the various routing methods. Order them according to the amount of mobility that they can sustain. Explain your reasons for the relative ordering of the routing mechanisms.

B. Repeat part 1 considering the importance of receiving the data quickly.

C. Suggest one additional metric for classifying the routing mechanisms, and repeat part 1 for your metric.

D. Invent one more routing mechanism and describe its characteristics.

## 5.2 Sensor Networks

### 5.2.1 Unique Characteristics

a. Data collection network:

- Data from all of the sensors is directed toward one, or a small number of data collection points
- The collection network is one or more trees with the collection nodes at the root of the tree,
- Conventional Networks have a large number of independent point-to-point connections

b. Severe Energy Constraint

- Many sensor networks are powered by small batterys that have a limited lifetime
  Some have large stationary sensors that are powered off of the power grid
- Conserving Energy is more important in sensor networks than is conventional networks

c. Data can be compressed during transmission

- Much of the information is redundant, and need not be forwarded to the destination
- When two or more sensors detect the same event, the amount of infomation is reduced in the network to conserve battery power
- Conventional networks do not modify the data in a packet

d. Event driven network.

- An event occurs that causes a large number of sensors to generate signals, or sensors collect data periodically
- There is an impulse of transmissions in the network
- Conventional networks have a Poisson arrival process because data arrives at the sources is independently.
- The impulse of arrivals makes random contention MAC protocols less efficient.
- Polling or hybrid polling/splitting is usually used instead of random access protoocls to conserve energy

### 5.2.2 Routing Mechanisms

*5.2.2.1 Energy Efficient Routing*

- Find the path that requires the least power to get from the sensor to the destination.
- Use the Bellman-Ford algorithm with the path weight proportional to the transmission power.
- Combine routing with MAC protocol

  — Control transmission energy to just reach the next node

  — Less transmission energy results in fewer collisions

*5.2.2.2 Tree structures to collect data*

a. Weight of path between nodes = transmission power, proportional to $d^i$, where $i = 2, 3, 4$ depending on antenna

b. If there is no data compression,

- Each packet that is received by the node must be forwarded to the destination
- The least energy tree is the minimum depth tree

- Each source uses the path that requires the least energy to reach the destination

c. If there is perfect compression

- When K packets arrive at a node, all of the information is redundant, and only one packet is forwarded
- The least energy tree is the spanning tree
  The sum of the link weights is minimized

d. In most networks there will be partial compression, since we are interested in characterizing the event, and not just detecting when it occurs.

- The "best tree" will be between a minimum depth and a minimum spanning tree
- The "best tree" will depend on the route that provides the greatest compression
- We may be able to approximate good trees by weighting the link that is added more heavily than the dstance assigned to the node in Dykstra's minimum depth algorithm.

*5.2.2.3 Routing to extend the network lifetime*

- Battery powered sensor networks fail when the batterys in the nodes fail
- One objective of routing in sensor networks is to equalize, and limit the drain on the battery in order to extend the life of the network.
- Receiving drains the battery at a constant rate, and transmitting drains the battery in proportion to the flow of packets generated by or forwarded by the node.
  The energy in the battery in a node at time t, $E(t)$ is related to the energy at time $t'$ by $E(t) = E(t') - \alpha(t - t') - \beta f$, where $f$ is the flow through the node during the interval.
- Work is in progress on metrics to minimize in the Bellman-Ford algorithm to extend the battery life of the network.
- The objective is to route packets through nodes that have sufficient battery power.
- One approach is to minimize $\sum_i \frac{1}{E_i}$ on each path in the tree.
  As more flows select a particular path, the battery drains more quickly, and the path is changed

*5.2.2.4 Directed Diffusion*

- Performs both route discovery and routing
- Sensors collect data continuously, but don't know when to send it, or where to send it.
- Transmission is controlled by the receiver at the neck of the funnel.
- The receiver broadcasts a request for data - floods the network
- A sensor receives the request from several nodes.
- Initially, the sensor sends a fraction of the data to each of the nodes with equal probability
- Each time a sensor sends data, or forwards data from another sensor, it receives a hop-by-hop acknowledgment, from the next sensor that tells how "good" that path is.
  The indicator may quantify the battery power left on the path, or the amount of compression that can be performed on that path.
- The sensor modifies the probability that it forwards data to the set of nodes, so that it forwards data along the better paths more often.
- Eventually the data favors the best paths
- By continuing to use several paths, the best path may change with time - for instance, when taking the path wears down the battery.

### 5.2.3 The Bellman-Ford Algorithm applied to mobile networks

| Distance Metrics | Update Mechanism |
|---|---|
| Adapted From Wired Networks | |
| Number of Hops<br><br>Congestion/Delay | Periodic<br><br>When significant change occurs |
| Developed for Wireless Networks | |
| Transmission Energy<br>For $d^i$, as $i$ increases,<br>MDT-> MST<br><br>Energy left in battery | More frequent for nearby destination<br>Fish-eye state routing<br><br>Table updates are part of message ACK's<br>Directed Diffusion<br><br>Prune Unused Table Entries<br>Regrow on Flood Command |

**Home work**

Energy Efficient Routing:

— Consider a linear network with a node every 1 unit of length.

— A sensor and its destination are 10 units apart

— The transmission distance for a node is up to 3 units, but it adjusts its transmission power to just reach a node

— The transmission power is proportional to $d^2$.

Which nodes does the sensor forward its data through?

What is the energy cost to send a packet to the destination?

## 5.3 Routing in intermitently connected and delay tolerant networks

### 5.3.1 Deterministic Routing

Future movements and connections, and the entire network topology is known

— inter-planet communications or deep space probes using satellites around planets or moons.

— Planned on/off times in energy constrained sensor networks

*5.3.1.1 Space-time routing*

The graph of the network changes in a known way at certain times. Find path that

— Min. time to reach a destination

— Min hops to reach a destination.

*5.3.1.2 Tree*

— Build a tree from the source to the children nodes that they reach at the earliest time

— Starting at the time that the child is reached, build a tree of the nodes that it can reach, at the earliest time that the source reaches that node

— Dykstra-like minimum depth tree algorithm, where each next node in the tree is added at the earliest time that it can be reached after the time marked at the current node

— The result is a tree that shows the min. time that the source can reach each other node

*5.3.1.3 Modified Shortest path*

Use weights on the links that include time and hop count in a deep space network, or time and energy for a sensor network.

### 5.3.2 Stochastic Routing

*5.3.2.1 Epidemic/random spray*

— no knowledge about network topology or movement

— spread the message to some or all of the nodes that you come in contact with

— the first to come in contact with the destination delivers the message

— uses flooding or partial flooding

— There is a relationship between the delay to deliver a message and the fraction of the nodes infected. Infecting more neighbors uses more energy.

*5.3.2.2 Predictions*

- Based on one-hop information from the next nodes that are met
  ie: forward to nodes that are moving around quickly
- Based on end-to end information
  ie: forward to nodes that are known to meet the destination often, or travel to the destination quadrant

*5.3.2.3 Model-based*

Use trajectories of nodes, rather than assuming random motion
motion of vehicles on a highway

*5.3.2.4 Coding-based*

- Erasure coding and network coding
- Send part of message, rather than complete information, to neighbors
- Some messages are linear combinations of other messages to avoid waiting for all of the pieces to arrive at the destination.

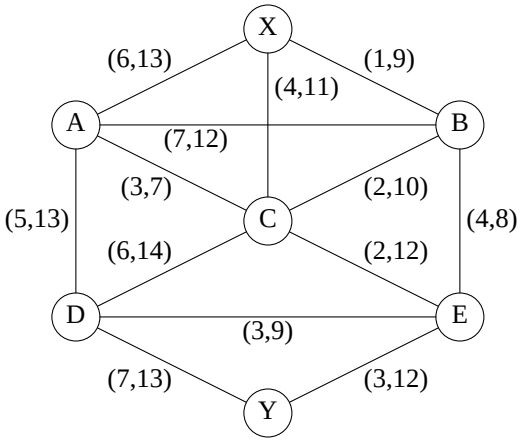
**Home work**

Delay Tolerant Networks

Nodes in a network are only able to communicate at specific times.

The links are labled with the times that nodes can communicate in parenthesis.

Messages are forwarded between nodes, then forwarded when the links become available




A message arrives at node X at time t=2 that is destined for node Y.

a. Draw a tree that shows the earliest time that the message from node X can arrive at each of the nodes in the network.

b. What path from X to Y gets the message to the destination at the earliest possible time?

c. What time does the message arrive at the destination?

## 6. Oblivious Routing

**Problem:**

- We know how much traffic enters and leaves at each node in the network, but don't know the detailed traffic matrix.
  - This is known as the *Hose Model*
  - The traffic arriving at node $N_i$ from outside the network is $A_i$.
  - The traffic leaving the network at $N_i$ is $B_i$.
    - The values of $A_i$ and $B_i$ are known for each $N_i$.
  - The traffic matrix for the network is $T$, where $t_{ij}$ is the flow from $N_i$ to $N_j$.
    - The individual $t_{ij}$ are not known,
- We want to set up routes from each node $N_i$ to each node $N_j$ that will guarantee bandwidth, or minimize delay

**Technique**: Two Phase Routing [19]

1. When traffic arrives in the network, an intermediate node is selected at random, with a pre-specified probability, and the traffic is forwarded to that node.
   - When traffic arrives at node, $N_i$, it is forwarded to an intermediate node $N_k$, with probability $\alpha_{ik}$, without regard for the packet's ultimate destination.
   - The only constraint on the random distribution is that $\sum_k \alpha_{ik} = 1$.
2. The intermediate node then sends the traffic to the destination.
   - The packets that arrive at intermediate $N_k$ are routed to their destination $N_j$.

If each node sends the same fraction of its traffic to an intermediate node, we can determine the complete traffic matrix.

- $\alpha_{ik} = \alpha_k$ for all $i$

Our objective is to determine the flows, $z_{kj}$, from each node $k$ to each node $j$ as a function of $A = \{A_i\}$ and $B = \{B_i\}$, the flows entering and leaving nodes, but independent of the actual traffic $T = \{t_{ij}\}$ between nodes.

**At each node $N_k$,**

A. **In phase 1:** The traffic that enters the network at node $N_k$ that is sent to intermediate node $N_j$ is

$x_{kj} = \alpha_j A_k$,

$x_{kj}$ is independent of $T$.

B. **In phase 2:**

The traffic from node $N_i$ that has been sent to node $N_k$ in phase 1 is:
$\alpha_k A_i$

The traffic from node $N_i$ that was sent to node $N_k$ in phase 1, and must be forwarded to node $N_j$ in phase 2 is
$\alpha_k t_{ij}$, since:

$A_i = \sum_j t_{ij}$ and each arriving packet is sent to $N_k$ with the same probability $\alpha_k$

This *IS* a function of $T$

The traffic from all of the nodes that was sent to $N_k$ in phase 1, and must be forwarded to $N_j$ in phase 2 is:

$y_{kj} = \sum_i \alpha_{ik} t_{ij} = \alpha_k \sum_i t_{ij} = \alpha_k B_j$,

since, $B_j = \sum_i t_{ij}$.

Therefore, $y_{kj}$ *IS* independent of $T$

C. The total traffic that must be routed from $N_k$ to $N_j$ is

$z_{kj} = x_{kj} + y_{kj} = \alpha_j A_k + \alpha_k B_j$, which is independent of $T$.

The new traffic matrix is $Z$, which is a function of $A$ and $B$, but is independent of $T$.

- The traffic matrix, $Z$ is a function of the fractions of the traffic, $\alpha_k$, sent to each intermediate node.
- A routing mechanism finds the "best" way to place the flows in $Z$ on the network.
- By adjusting the $\alpha_k$, we can change the traffic matrix to better use the network.

The average path length increases, and hence the load on the network is increased by oblivious routing.

Two phase routing makes the traffic between the nodes on the network only a function of the $A_i$ and $B_j$, independent of the individual $t_{ij}$.

Therefore,

1. Routing is oblivious to traffic variations. The traffic does not depend on the matrix $T$, and

2. the bandwidth or interference on a path does not change as the $T$ changes.

At this point we can apply any of the fair or optimal routing and/or flow control procedures to the fixed flow.

Web traffic in the Internet is an example of where oblivious routing can be used.
The Web servers can control the total flow that they insert into the Internet but do not control where they

send their data. The destinations are continuously changing.
The users have a limit on the flow that they receive from the Internet, but do not control where the flows originate. The servers that send them data continuously change.

*REFERENCES*

[1] D. Bertsekas, R. G. Gallager, **Data Networks**, Prentice-Hall Inc, 1992.

[2] N. F. Maxemchuk, M. El Zarki, "Routing and Flow Control in High Speed Networks," Proceedings of the IEEE, Jan. 1990, vol.78, no. 1, pp. 204-221.

[3] R. T. Prosser, "Routing Procedures in Communications Networks- Part II: Directory Procedures," IRE Trans. on Commun. Syst., Dec. 1962, pp 329-335.

[4] R. T. Prosser, "Routing Procedures in Communications Networks- Part I: Random Procedures," IRE Trans. on Commun. Syst., Dec. 1962, pp 322-329.

[5] P. Baran, "On Distributed Communications Networks," IEEE Trans. on Comm. Sys., vol cs-12, no.1, pp. 1-9, Mar 1964.

[6] G. L. Fultz, L. Kleinrock, "Adaptive Routing Techniques for Store-and-Forward Computer-Communications Networks," Proc. ICC ’71, pp. 39.1-8.

[7] H. Rudin, "On Routing and ’Delta-Routing’: A Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks" IEEE Transaction on Communications, Vol. COM-24, No. 1, January 1976, pp. 43-59.

[8] C.-K. Toh, "A novel distributed routing protocol to support ad-hoc mobile computing," Proc. IEEE 15th Pheonix Conf. Comp. and Commun. Mar. 1996, pp. 480-6.

[9] A. Iwata, C-C. Chiang, G. Pei, M. Gerla, T-W. Chen, "Scalable routing strategies for ad hoc wireless networks," IEEE JSAC, Aug. 1999, vol. 17, no. 8, pp. 1369-79.

[10] Guangyu Pei M. Gerla, Tsu-Wei Chen, "Fisheye state routing: a routing scheme for ad hoc wireless networks," IEEE ICC 2000, vol. 1, pp. 70 -74.

[11] B. Karp, H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," Mobicom 2000, Boston, Ma., Aug 6-11, 2000, pp. 243-254.

[12] D. Kim, N. F. Maxemchuk, "Simple Robotic Routing in Mobile Ad Hoc Networks," IEEE Int. Conf. on Network Protocols 2006, Nov. 6-9, 2005, Boston, Ma., pp. 159-168.

[13] S. J. Lee, M. Gerla, "Split multipath routing with maximally disjoint paths," Proc. ICC 2001,

[14] Y. Birk, N. Bloch, "Improving network performance with prioritized dispersal," IEEE INFOCOM 2000, vol. 3, pp. 817 -1826.

[15] M. R. Pearlman, Z. J. Haas, P. Sholander, S. S.Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc network," Mobile and Ad Hoc Networking and Computing, 2000, Aug 11, 2000, Boston, Ma., Page(s): 3 -10.

[16] M. Gerla, X. Hong, G. Pei, "Landmark Routing for Large Ad Hoc Wireless Networks," IEEE Globecom 2000, San Francisco, Ca., Nov. 28-30, 2000.

[17] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," Mobicom’99, Aug. 1999.

[18] Y. B. Ko, N. H. Vaidya, "Location-Aided Routing (LAR) in mobile ad hoc networks," Wireless Networks, 2000, v6, n4, pp 307-321.

[19] M. Kodialam, T. V. Lakshman, S. Sengupta, "Traffic-Oblivious Routing for Guaranteed Bandwidth Performance," IEEE Communications Magazine, vol. 45, Iss. 4, Apr. 2007, pp.:46-51.