Slides based on
Computer Networks by R.S. Chang, Dept. CSIE, NDHU
Chapter 5 The Network Layer

Routing: path selection
Different network translation
Congestion control
Network accounting
5. The Network Layer

5.1 Network Layer Design Issues

5.1.1 Services Provided to the Transport Layer
5. The Network Layer

5.1 Network Layer Design Issues

5.1.1 Services Provided to the Transport Layer

The network layer services have been designed with the following goals in mind:

1. The services should be independent of the subnet technology.
2. The transport layer should be shielded from the number, type, and topology of the subnets present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.
5. The Network Layer

5.1 Network Layer Design Issues

5.1.1 Services Provided to the Transport Layer

Two camps:
1. Connectionless services: Internet community (based on nearly 30 years of actual experience with a real, working computer network)
2. Connection-oriented services: telephone companies (based on 100 years of successful experience with the worldwide telephone system)

The argument between connection-oriented and connectionless service really has to do with where to put the complexity (the subnet or the host).
5. The Network Layer

5.1 Network Layer Design Issues

5.1.1 Services Provided to the Transport Layer

Supporters of connectionless service say:
1. User computing power has become cheap, so there is no reason not to put the complexity in the hosts.
2. The subnet is a major international investment that will last for decades, so it should not be cluttered up with features that may become obsolete quickly.
3. Some applications, such as digitized voice and real-time data collection may regard **speedy** delivery as much more important than **accurate** delivery.

For example, the Internet TCP/IP protocol
5. The Network Layer

5.1 Network Layer Design Issues

5.1.1 Services Provided to the Transport Layer

Supporters of connection-oriented service say:
1. Most users are not interested in running complex transport layer protocols in their machines.
2. Some services, such as real time audio and video are much easier to provide on top of a connection-oriented network layer.

For example: Asynchronous Transfer Mode networks
5. The Network Layer

5.1 Network Layer Design Issues

5.1.2 Internal Organization of the Network Layer

Virtual Circuits, in analogy with the physical circuits set up by the telephone system

Datagrams, in analogy with telegrams
<table>
<thead>
<tr>
<th>Issue</th>
<th>Datagram subnet</th>
<th>VC subnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit setup</td>
<td>Not needed</td>
<td>Required</td>
</tr>
<tr>
<td>Addressing</td>
<td>Each packet contains the full source and destination address</td>
<td>Each packet contains a short VC number</td>
</tr>
<tr>
<td>State information</td>
<td>Subnet does not hold state information</td>
<td>Each VC requires subnet table space</td>
</tr>
<tr>
<td>Routing</td>
<td>Each packet is routed independently</td>
<td>Route chosen when VC is set up; all packets follow this route</td>
</tr>
<tr>
<td>Effect of router failures</td>
<td>None, except for packets lost during the crash</td>
<td>All VCs that passed through the failed router are terminated</td>
</tr>
<tr>
<td>Congestion control</td>
<td>Difficult</td>
<td>Easy if enough buffers can be allocated in advance for each VC</td>
</tr>
</tbody>
</table>
5. The Network Layer

5.1 Network Layer Design Issues

5.1.2 Internal Organization of the Network Layer

<table>
<thead>
<tr>
<th>Upper layer</th>
<th>Datagram</th>
<th>Type of subnet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Datagram</td>
<td>Virtual circuit</td>
</tr>
<tr>
<td>Connectionless</td>
<td>UDP over IP</td>
<td>UDP over IP over ATM</td>
</tr>
<tr>
<td>Connection-oriented</td>
<td>TCP over IP</td>
<td>ATM AAL1 over ATM</td>
</tr>
</tbody>
</table>
5. The Network Layer

5.2 Routing Algorithms

**routing algorithm:** determine the route and maintain the routing table

desired properties for a routing algorithm:
1. **correctness**
2. **simplicity**
1. **robustness** with respect to failures and changing conditions
2. **stability** of the routing decisions
3. **fairness** of the resource allocation
4. **optimality** of the packet travel times
5. The Network Layer

5.2 Routing Algorithms

Fairness and optimality are often contradictory goals.
5. The Network Layer

5.2 Routing Algorithms

What is it that we seek to optimize?

Minimizing mean packet delay is an obvious candidate, but so is maximizing total network throughput. Furthermore, these two goals are also in conflict, since operating any queuing system near capacity implied a long queuing delay.

As a compromise, many networks attempt to minimize the number of hops a packet must make, because reducing the number of hops tends to improve the delay and also reduce the amount of bandwidth consumed, which tends to improve the throughput as well.
5. The Network Layer

5.2 Routing Algorithms

Static (nonadaptive) Routing
The routing table is not changed according to network conditions.

adaptive routing

centralized routing: one node calculates the routing table
isolated routing: do not exchange information with other node
distributed routing: node exchanges information and makes
routing decisions by itself
5. The Network Layer

5.2 Routing Algorithms

5.2.1 The Optimality Principle

The optimality principle states that if router $J$ is on the optimal path from router $I$ to router $K$, then the routes from $I$ to $J$ and from $J$ to $K$ are also optimal.

As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a sink tree.
5. The Network Layer

5.2 Routing Algorithms

5.2.1 The Optimality Principle

A sink tree for router B
5. The Network Layer

5.2 Routing Algorithms

5.2.1 The Optimality Principle

A sink tree does not contain any loops, so each packet will be delivered within a finite and bounded number of hops. In practice, life is not quite this easy. Links and routers can go down and come back up during operation, so different routers may have different ideas about the current topology.

Also, we have quietly finessed the issue of whether each router has to individually acquire the information on which to base its sink tree computation, or whether this information is collected by some other means.
5. The Network Layer

5.2 Routing Algorithms

5.2.2 Shortest Path Routing

To compute the shortest path from A to D: Dijkstra’s algorithm
5. The Network Layer

5.2 Routing Algorithms

5.2.2 Shortest Path Routing

To compute the shortest path from A to D
5. The Network Layer

5.2 Routing Algorithms

5.2.2 Shortest Path Routing

To compute the shortest path from A to D
5. The Network Layer

5.2 Routing Algorithms

5.2.2 Shortest Path Routing

To compute the shortest path from A to D
5. The Network Layer

5.2 Routing Algorithms

5.2.2 Shortest Path Routing

To compute the shortest path from A to D
To compute the shortest path from A to D
5. The Network Layer

5.2 Routing Algorithms

5.2.3 Flooding

Transmit a copy of each packet it receives on every one of its transmission links.

advantages: robust, simple, broadcasting, discovery

disadvantages: use too much resource

How to curb the flooding: 1. hop count
2. time stamp

A variation of flooding that is slightly more practical is selective flooding. In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction.
5. The Network Layer

5.2 Routing Algorithms

5.2.4 Flow-Based Routing

A subnet with line capacity shown in kbps
## 5. The Network Layer

### 5.2 Routing Algorithms

#### 5.2.4 Flow-Based Routing

The traffic in packets/sec and the routing matrix:

<table>
<thead>
<tr>
<th>Source</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
<td>AB</td>
<td>4</td>
<td>ABC</td>
<td>1</td>
<td>ABFD</td>
</tr>
<tr>
<td>B</td>
<td>9</td>
<td>BA</td>
<td>8</td>
<td>BC</td>
<td>3</td>
<td>BFD</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>CBA</td>
<td>8</td>
<td>CB</td>
<td>3</td>
<td>CD</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>DFBA</td>
<td>3</td>
<td>DFB</td>
<td>3</td>
<td>DCE</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>EA</td>
<td>3</td>
<td>EC</td>
<td>3</td>
<td>ECD</td>
</tr>
<tr>
<td>F</td>
<td>4</td>
<td>FEA</td>
<td>4</td>
<td>FB</td>
<td>2</td>
<td>FEC</td>
</tr>
</tbody>
</table>

*Note: The routing matrix shows the traffic in packets/sec between different nodes.*
5. The Network Layer

5.2 Routing Algorithms

5.2.4 Flow-Based Routing

The delay for a 800 bits packet is given by:

\[
T = \frac{1}{\mu C - \lambda} \frac{\lambda_i}{\sum \lambda_i}
\]

<table>
<thead>
<tr>
<th>i</th>
<th>Line</th>
<th>( \lambda_i ) (pkts/sec)</th>
<th>( C_i ) (kbps)</th>
<th>( \mu C_i ) (pkts/sec)</th>
<th>( T_i ) (msec)</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AB</td>
<td>14</td>
<td>20</td>
<td>25</td>
<td>91</td>
<td>0.171</td>
</tr>
<tr>
<td>2</td>
<td>BC</td>
<td>12</td>
<td>20</td>
<td>25</td>
<td>77</td>
<td>0.146</td>
</tr>
<tr>
<td>3</td>
<td>CD</td>
<td>6</td>
<td>10</td>
<td>12.5</td>
<td>154</td>
<td>0.073</td>
</tr>
<tr>
<td>4</td>
<td>AE</td>
<td>11</td>
<td>20</td>
<td>25</td>
<td>71</td>
<td>0.134</td>
</tr>
<tr>
<td>5</td>
<td>EF</td>
<td>13</td>
<td>50</td>
<td>62.5</td>
<td>20</td>
<td>0.159</td>
</tr>
<tr>
<td>6</td>
<td>FD</td>
<td>8</td>
<td>10</td>
<td>12.5</td>
<td>222</td>
<td>0.098</td>
</tr>
<tr>
<td>7</td>
<td>BF</td>
<td>10</td>
<td>20</td>
<td>25</td>
<td>67</td>
<td>0.122</td>
</tr>
<tr>
<td>8</td>
<td>EC</td>
<td>8</td>
<td>20</td>
<td>25</td>
<td>59</td>
<td>0.098</td>
</tr>
</tbody>
</table>
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

Distance vector routing algorithms operate by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which line to use to get there. These tables are updated by exchanging information with the neighbors.

E.g.: Routing table for Router A

<table>
<thead>
<tr>
<th>Destination</th>
<th>cost(delay, distance, …)</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>10</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>● ● ●</td>
<td></td>
</tr>
</tbody>
</table>
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP (Routing Information Protocol) and in early versions of DECnet and Novell’s IPX. AppleTalk and Cisco routers use improved distance vector protocols.

Once every T msec each router sends to each neighbor a list of its estimate delays to each destination. It also receives a similar list from each neighbor.
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

Vectors received from J's four neighbors

New estimated delay from J

New routing table for J
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

The count-to-infinity problem

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>Initially</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>After 1 exchange</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>After 2 exchanges</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>$\infty$</td>
<td>After 3 exchanges</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>After 4 exchanges</td>
</tr>
</tbody>
</table>

Then A comes up. The good news spreads quickly.
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

The **count-to-infinity** problem

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>∞</td>
</tr>
</tbody>
</table>

Initially

After 1 exchange

After 2 exchanges

After 3 exchanges

After 4 exchanges

After 5 exchanges

After 6 exchanges

Then A comes down.
The bad news travels slowly.

A is up
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

The count-to-infinity problem
It should be clear why bad news travels slowly: no router ever has a value more than one higher than the minimum of all its neighbors. Gradually, all the routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity. For this reason, it is wise to set infinity to the longest path plus 1 (if using hop count as metric). If the metric is time delay, there is no well-defined upper bound, so a high value is needed to prevent a path with a long delay from being treated as down.
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

The **Split Horizon Hack**

Many ad hoc solutions to the count-to-infinity problem have been proposed in the literature, each one more complicated and less useful than the one before it. We will describe just one of them and tell why it, too, fails.

The **split horizon** algorithm works the same way as distance vector routing, except that the **distance to X is not reported on the line that packets for X are sent on** (actually, it is reported as infinity).
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

The **Split Horizon Hack**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>inf</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>inf</td>
</tr>
<tr>
<td>2</td>
<td>inf</td>
<td>inf</td>
<td>3</td>
<td>4</td>
<td>inf</td>
</tr>
<tr>
<td>3</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>4</td>
<td>inf</td>
</tr>
<tr>
<td>4</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
</tr>
</tbody>
</table>

Initially

After 1 exchange

After 2 exchanges

After 3 exchanges

After 4 exchanges

inf = infinity
5. The Network Layer

5.2 Routing Algorithms

5.2.5 Distance Vector Routing

The **Split Horizon Hack**

When CD line goes down. A thinks it has a path to D through B and B thinks it has a path to D through A. A and B will count to infinity.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. Two primary reasons caused its demise.

First, since the delay metric was queue length, it did not take line bandwidth into account when choosing routes.

Second, the algorithm often took too long to converge, even with tricks like split horizon.

For these reasons, it was replaced by an entirely new algorithm now called link state routing.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

The idea behind link state routing is simple and can be stated as five parts. Each router must:
1. Discover its neighbors and learn their network addresses.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Distance vector routing differs significantly from the link state routing. With link state algorithms, routers share only the identity of their neighbors, but they flood this information through the entire network. Distance vector algorithms adopt an opposite approach. Routers periodically share knowledge of the entire network, but only with their neighbors.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Learning about the Neighbors

When a router is booted, its first task is to learn who its neighbor are. It accomplishes this goal by sending a special **HELLO packet** on each point-to-point line. The router on the other end is expected to send back a reply telling who it is.

When two or more routers are connected by a LAN, the situation is slightly more complicated. One way to model the LAN is to **consider it as a node** itself.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Learning about the Neighbors
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Measuring Line Cost

The link state routing algorithm requires each router to know, or at least have a reasonable estimate, of the delay to each of its neighbors.

The most direct way to determine this delay is to send a special ECHO packet over the line that the other side is required to send back immediately.

By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Measuring Line Cost

An interesting issue is whether or not to take the load into account when measuring the delay.

To factor the load in, the round-trip timer must be started when the ECHO packet is queued.

To ignore the load, the timer should be started when the ECHO packet reaches the front of the queue.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Measuring Line Cost

Including queuing cost: can use the best line, but may lead to routing table oscillating.

Same bandwidth on the two links
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Building Link State Packets

Building the link state packets is easy. The hard part is determining when to build them. 1. Periodically 2. When some significant event occurs, such as a line or neighbor going down or coming back up.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Distributing the Link State Packets

The trickiest part of the algorithm is distributing the link state packets reliably. As the packets are distributed and installed, the routers getting the first ones will change their routes.

Consequently, the different routers may be using different versions of the topology, which can lead to inconsistencies, loops, unreachable machines, and other problems.

The fundamental idea is to use flooding to distribute the link state packets.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Distributing the Link State Packets

To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent. Routers keep track of all the (source router, sequence) pairs they see.

When a new link state packet comes in, it is checked against the list of packets already seen.
1. If new: forward on all lines except the one it arrived on
2. If duplicate or old packet: discard
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Distributing the Link State Packets

Problems:

If the sequence numbers wrap around, confusion will reign. The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around.

If a router crashes, it will lose track of its sequence number.

If a sequence number is ever corrupted and 65540 is received instead of 4 (a 1-bit error), packets 5 through 65540 will be rejected as obsolete.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Distributing the Link State Packets

The solution to all these problems is to include the age of each packet after the sequence number and **decrement it once per second**. When the age hits zero, the information from that router is discarded.

The age field is also decremented by each router during the initial flooding process, to make sure no packet can get lost and live for an indefinite period of time.
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Distributing the Link State Packets

To guard against errors on the router-router lines, all link state packets are acknowledged.

Packet buffer for router B

<table>
<thead>
<tr>
<th>Source</th>
<th>Seq.</th>
<th>Age</th>
<th>A</th>
<th>C</th>
<th>F</th>
<th>A</th>
<th>C</th>
<th>F</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>21</td>
<td>60</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>21</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>21</td>
<td>59</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>21</td>
<td>59</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
5. The Network Layer

5.2 Routing Algorithms

5.2.6 Link State Routing

Computing the New Routes

Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented. Now Dijkstra’s algorithm can be run locally to construct the shortest path to all possible destinations.

The OSPF (Open Shortest Path First) protocol uses link state routing algorithm.
5. The Network Layer

5.2 Routing Algorithms

5.2.7 Hierarchical Routing

from U.S. send a packet to csie.ndhu.edu.tw

1. first send to domain tw (Taiwan)
2. then to subdomain MOE (edu)
3. then to subsubdomain ndhu
4. then to host csie

Advantage: simple, efficient, and saving routing table space

In a 1000 users network, each routing table needs 999 entries. If it is divided into 10 domains, then each table only needs 99+9=108 entries.
5. The Network Layer

5.2 Routing Algorithms

5.2.7 Hierarchical Routing
5. The Network Layer

5.2 Routing Algorithms

5.2.7 Hierarchical Routing

Unfortunately, the gain in routing table space are not free. There is a penalty to be paid, and this penalty is in the form of increased path length.

For example, the best route from 1A to 5C is via region 2, but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5.

When a single network becomes very large, an interesting question is: How many levels should the hierarchy have? Answer: the optimal number of levels for an \( N \) router subnet is \( \ln N \), requiring a total of \( e \ln N \) entries per router.
5. The Network Layer

5.2 Routing Algorithms

5.2.8 Routing for Mobile Hosts
5. The Network Layer

5.2 Routing Algorithms

5.2.8 Routing for Mobile Hosts

When a new user enters an area, either by connecting to it, or just wandering into the cell, his computer must register itself with the foreign agent there. The registration procedure typically works like this:

1. Periodically, each foreign agent broadcasts a packet announcing its existence and address. A newly arrived mobile host may wait for one of these messages, but if none arrives quickly enough, the mobile host can broadcast a packet saying: “Are there any foreign agents around?”
5. The Network Layer

5.2 Routing Algorithms

5.2.8 Routing for Mobile Hosts

2. The mobile host registers with the foreign agent, giving its home address, current data link layer address, and some security information.

3. The foreign agent contacts the mobile host’s home agent and says: “One of your hosts is over here.” The message from the foreign agent to the home agent contains the foreign agent’s network address. It also includes the security information, to convince the home agent that the mobile host is really there.
5. The Network Layer

5.2 Routing Algorithms

5.2.8 Routing for Mobile Hosts

4. The home agent examines the security information, which contains a time stamp, to prove that it was generated within the past few seconds. If it is happy, it tells the foreign agent to proceed.

5. When the foreign agent gets the acknowledgement from the home agent, it makes an entry in its tables and informs the mobile host that it is now registered.

Ideally, when a user leaves an area, that, too, should be announced to allow deregistration, but many users abruptly turn off their computers when done.
5. The Network Layer

5.2 Routing Algorithms

5.2.8 Routing for Mobile Hosts

Packet routing for mobile hosts

1. Packet is sent to the mobile host's home address

2. Packet is tunneled to the foreign agent

3. Sender is given foreign agent's address

4. Subsequent packets are tunneled to the foreign agent
5. The Network Layer

5.2 Routing Algorithms

5.2.9 Broadcast Routing

One broadcasting method that requires no special features from the subnet is for the source to simply send a distinct packet to each destination.

Waste bandwidth and require the source to have a complete list of all destinations.

Flooding is another obvious candidate. But it generates too many packets and consumes too much bandwidth.
5. The Network Layer

5.2 Routing Algorithms

5.2.9 Broadcast Routing

Multidestination Routing

Each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, the router checks all the destinations to determine the set of output lines that will be needed.

The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line. In effect, the destination set is partitioned among the output lines.
5. The Network Layer

5.2 Routing Algorithms

5.2.9 Broadcast Routing

A fourth broadcast algorithm makes explicit use of the sink tree for the router initiating the broadcast, or any other convenient spanning tree for that matter.

This method makes excellent use of bandwidth, generating the absolute minimum number of packets necessary to do the job. The only problem is that each router must have knowledge of some spanning tree for it to be applicable.
5. The Network Layer

5.2 Routing Algorithms

5.2.9 Broadcast Routing

Reverse Path Forwarding

When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the line that is normally used for sending packets to the source of the broadcast. If so, forward it.
5. The Network Layer

5.2 Routing Algorithms

5.2.10 Multicast Routing

To do multicasting, group management is required. Some way is needed to create and destroy groups, and for processes to join and leaves groups. It is important that routers know which of their hosts belong to which groups.

Either hosts must inform their routers about changes in group membership, or routers must query their hosts periodically. Either way, routers learn about which of their hosts are in which groups. **Routers tell their neighbors**, so the information propagates through the subnet.
5. The Network Layer

5.2 Routing Algorithms

5.2.10 Multicast Routing

To do multicast routing, each router computes a spanning tree covering all other routers in the subnet.

A multicast tree for group 1
5. The Network Layer

5.2 Routing Algorithms

5.2.10 Multicast Routing

Various ways of pruning the spanning tree are possible. The simplest one can be used if *link state routing* is used, and each router is aware of the complete subnet topology, including which hosts belong to which groups.

Then the spanning tree can be pruned by *starting at the end of each path and working toward the root*, removing all routers that do not belong to the group in question.
5. The Network Layer

5.2 Routing Algorithms

5.2.10 Multicast Routing

With distance vector routing, whenever a router with no hosts interested in a particular group and no connections to other routers receives a multicast message for that group, it responses with a PRUNE message, telling the sender not to send it any more multicasts for that group.

**source-specific multicast** trees: scales poorly to large networks

$n$ groups, $m$ members: a total of $nm$ trees

**core-based tree** approach: each group has only one multicast tree

$n$ groups: $n$ trees
5. The Network Layer

5.3 Congestion Control Algorithms

![Diagram showing network throughput vs. packets sent]

- Perfect
- Desirable
- Congested

Maximum carrying capacity of subnet
5. The Network Layer

5.3 Congestion Control Algorithms

Congestion control vs. flow control

**Flow control:** a network with a capacity of 1000 gigabits/sec on which a supercomputer is trying to transfer a file to a personal computer at 1Gbps. Although there is no congestion (the network itself is not in trouble), flow control is needed to force the supercomputer to stop frequently to give the personal computer a chance to breathe.
5. The Network Layer

5.3 Congestion Control Algorithms

Congestion control vs. flow control

At the other extreme, consider a store-and-forward network with 1-Mbps lines and 1000 large computers, half of which are trying to transfer files at 100 kbps to the other half. Here the problem is not that of fast senders overpowering slow receivers, but simply that the total offered traffic exceeds what the network can handle.
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.1 General Principles of Congestion Control

**Open loop solution**: solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made. Tools for doing open-loop control include deciding **when to accept new traffic**, deciding **when to discard packets and which ones**, and **making scheduling decisions at various points in the network**.
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.1 General Principles of Congestion Control

Closed loop solutions:
1. Monitor the system to detect when and where congestion occurs.

Metrics for congestion measurement for routers or hosts: percentage of all packets discarded for lack of buffer space, the average queue length, the number of retransmitted packets, the average packet delay, and the standard deviation of packet delay.

We can also send probe packets out periodically to explicitly ask about congestion.
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.1 General Principles of Congestion Control

2. Pass this information to places where action can be taken.

The obvious way is for the router detecting the congestion to send a packet to the traffic source or sources, announcing the problem. Of course, these extra packets increase the load at precisely the moment that more load is not needed.

A router can set a bit in the packet to notify neighbors when congestion occurs.
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.1 General Principles of Congestion Control

3. Adjust system operation to correct the problem.

The hope is that the knowledge of congestion will cause the sources to take appropriate action to reduce the congestion. To work correctly, the time scale must be adjusted carefully.

React too quickly: system will oscillate
React too slowly: no real use
## 5. The Network Layer

### 5.3 Congestion Control Algorithms

#### 5.3.2 Congestion Prevention Policies

<table>
<thead>
<tr>
<th>Layer</th>
<th>Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td>• Retransmission policy</td>
</tr>
<tr>
<td></td>
<td>• Out-of-order caching policy</td>
</tr>
<tr>
<td></td>
<td>• Acknowledgement policy</td>
</tr>
<tr>
<td></td>
<td>• Flow control policy</td>
</tr>
<tr>
<td></td>
<td>• Timeout determination</td>
</tr>
<tr>
<td>Network</td>
<td>• Virtual circuits versus datagram inside the subnet</td>
</tr>
<tr>
<td></td>
<td>• Packet queueing and service policy</td>
</tr>
<tr>
<td></td>
<td>• Packet discard policy</td>
</tr>
<tr>
<td></td>
<td>• Routing algorithm</td>
</tr>
<tr>
<td></td>
<td>• Packet lifetime management</td>
</tr>
<tr>
<td>Data link</td>
<td>• Retransmission policy</td>
</tr>
<tr>
<td></td>
<td>• Out-of-order caching policy</td>
</tr>
<tr>
<td></td>
<td>• Acknowledgement policy</td>
</tr>
<tr>
<td></td>
<td>• Flow control policy</td>
</tr>
</tbody>
</table>
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.3 Traffic Shaping

A source promises the network that the traffic a flow sends into the network will *conform to a particular shape*. The network uses this information to:

- decide whether to accept the flow (CAC: Connection Admission Control)
- if accepted, how to manage the flow's traffic (UPC: Usage Parameter Control)

Three major purposes:

(1) The network knows what kind of traffic to expect.
(2) The network can determine if the flow should be allowed to send.
(3) The network can periodically monitor the flow's traffic and confirm that the flow is behaving as it promised.
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.3 Traffic Shaping

Traffic shaping is about regulating the average rate (and burstiness) of data transmission. In contrast, the sliding window protocols we studied earlier limit the amount of data in transmit at once, not the rate at which it is sent.

Monitoring a traffic flow is called traffic policing.

A good traffic shaping method should be easy to police.
5. The Network Layer
5.3 Congestion Control Algorithms

5.3.3 Traffic Shaping

The Leaky Bucket Algorithm

The Leaky Bucket Algorithm
5. The Network Layer
5.3 Congestion Control Algorithms

5.3.3 Traffic Shaping

The Leaky Bucket Algorithm
5. The Network Layer
5.3 Congestion Control Algorithms

5.3.3 Traffic Shaping

The Leaky Bucket Algorithm

Bucket capacity=1 MB
5. The Network Layer
5.3 Congestion Control Algorithms
5.3.3 Traffic Shaping

The Token Bucket Algorithm (allow some burstiness)

![Diagram]

- One token is added to the bucket every $\Delta T$
- The bucket holds tokens

Networks
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.3 Traffic Shaping

Calculate the length of the maximum rate burst

burst length = \( S \) seconds
token bucket capacity = \( C \) bytes
token arrival rate = \( \rho \) bytes/sec
maximum output rate = \( M \) bytes/sec

Therefore, \( C + \rho S = MS \). \( S = C / (M - \rho) \)
5. The Network Layer
5.3 Congestion Control Algorithms
5.3.3 Traffic Shaping

Token Bucket with Leaky Bucket Rate Control

Limit how long a token bucket can monopolize the network
1. Limit the size of token bucket size (too restrictive)
2. token bucket combined with a simple leaky bucket

\[ C \text{ should be substantially greater (faster) than } \rho. \text{ The maximum transmission rate at any time is } C. \]
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.4 Flow Specifications

Traffic shaping is most effective when the sender, receiver, and subnet all agree to it. To get agreement, it is necessary to specify the traffic pattern in a precise way. Such an agreement is called a flow specification.

Before a connection is established or before a sequence of datagrams are sent, the source gives the flow specification to the subnet for approval. The subnet can either accept it, reject it, or come back with a counterproposal.
5. The Network Layer
5.3 Congestion Control Algorithms
5.3.4 Flow Specifications

<table>
<thead>
<tr>
<th>Characteristics of the Input</th>
<th>Service Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum packet size (bytes)</td>
<td>Loss sensitivity (bytes)</td>
</tr>
<tr>
<td>Token bucket rate (bytes/sec)</td>
<td>Loss interval (μsec)</td>
</tr>
<tr>
<td>Token bucket size (bytes)</td>
<td>Burst loss sensitivity (packets)</td>
</tr>
<tr>
<td>Maximum transmission rate (bytes/sec)</td>
<td>Minimum delay noticed (μsec)</td>
</tr>
<tr>
<td>jitter</td>
<td>Maximum delay variation (μsec)</td>
</tr>
<tr>
<td></td>
<td>Quality of guarantee</td>
</tr>
</tbody>
</table>

(Loss sensitivity)/(loss interval) = maximum acceptable loss rate
5. The Network Layer
5.3 Congestion Control Algorithms
5.3.4 Flow Specifications

The quality of guarantee indicates whether or not the application really means it. One the one hand, the loss and delay characteristics might be ideal goals, but no harm is done if they are not met. On the other hand, they might be so important that if they cannot be met, the application simply terminates.

A problem inherent with any flow specification is that the application may not know what it really wants. For example, an application program running in New York might be quite happy with a delay of 200 msec to Sydney, but most unhappy with the same 200-msec delay to Boston.
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.5 Congestion Control in Virtual Circuit Subnets

Admission control: once congestion has been signaled, no more virtual circuits are set up until the problem has gone away.

Allow new VCs but carefully route all new VCs around problem areas.
Another strategy is to reserve all the resources needed for a virtual circuit when it is set up. In this way, congestion is unlikely to occur on the new virtual circuits because all the necessary resources are guaranteed to be available.

This kind of reservation can be done all the time as standard OS procedure, or only when the subnet is congested. A disadvantage of doing it all the time is that it tends to waste resources.
5. The Network Layer
5.3 Congestion Control Algorithms
5.3.6 Choke Packets

Each router monitors the utilization of its output lines and other resources, for example, by:

\[ u_{new} = a \times u_{old} + (1 - a) \times f \]

Where \( f \) is the sampling value

\( a \) determines how fast the router forgets recent history.

Whenever \( u \) moves above the threshold, the output line enters a “warning” state. Each newly arrived packet is checked to see if its output line is in warning state. If so, the router sends a **choke packet** back to the source, giving it the destination found in the packet.
5. The Network Layer
5.3 Congestion Control Algorithms

5.3.6 Choke Packets

The original packet is tagged (a header bit is turned on) so that it will not generate any more choke packets further along the path and is then forwarded in the usual way.

When the source gets the choke packet, it is required to reduce the traffic sent to the specified destination by $X$ percent.

Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets, the host should ignore choke packets referring to that destination for a fixed time interval.
5. The Network Layer

5.3 Congestion Control Algorithms

5.3.6 Choke Packets

After that period has expired, the host listens for more choke packets for another interval. If one arrives, the line is still congested, so the host reduces the flow still more. If no choke packets arrive during the listen period, the host may increase the flow again.

Usually, exponential decrease and additive increase.