L3: Network Layer

- In the DL layer, we can combine "similar" networks to make larger ones
- What about networks that aren't similar?
  - Different connection models?
  - Different addressing schemes?
- Network layer forms a uniform "shell" around interconnected networks of all types, provides uniform address space
Routers:
• Connect networks at the Network Layer
• Both packet and circuit-switching varieties
• Have two primary jobs:
  • Routing: Maintaining tables that specify the router output to use for a particular destination address
  • Switching: Moving data from router inputs to outputs based on the routing table
• May also support other functions:
  • Congestion control
  • QoS

Routing tables

<table>
<thead>
<tr>
<th>Dest Addr</th>
<th>Out Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
</tr>
</tbody>
</table>
Review:
The communication models

- Connectionless, AKA packet switching
  - The “postal service” model
  - Messages are broken into small packets. Each packet is individually forwarded through the network to the destination.

- Connection-oriented, AKA circuit switching
  - The “phone-company” model
  - Communicating hosts set up a dedicated channel across the network. After the channel is set up, they can exchange data with only a small amount of protocol overhead.

The idea of Shortest or “Least-Cost” paths

A 3-hop path

A 4-hop path
Packet Switching

Source breaks message into individually-addressed packets

Message to 97.206.0.6: 
"Mary had a little lamb"

Source: 205.16.4.4
Dest: 97.206.0.6

<table>
<thead>
<tr>
<th>Packet</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Message</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>205.16.4.4</td>
<td>97.206.0.6</td>
<td>&quot;Mary had a little lamb&quot;</td>
<td>Forward</td>
</tr>
<tr>
<td>2</td>
<td>97.206.0.6</td>
<td>205.16.4.4</td>
<td>&quot;little lamb&quot;</td>
<td>Forward</td>
</tr>
</tbody>
</table>
Connectionless (2)

Packets are sent into the network one after the other.

Source
205.16.4.4
97.206.0.6
205.16.4.4/1 little lamb
97.206.0.6/205.16.4.4/1 Mary had a

Dest
97.206.0.6

Connectionless (3)

Each router’s routing table specifies which of its output ports is “best” to use for packets for a particular destination address.

Source
205.16.4.4
97.206.0.6/205.16.4.4/2 little lamb

Dest
97.206.0.6

address
port
97.206.0.6
205.16.4.4
3
1
2
3
1
Connectionless (4)

Ultimately, the packets reach the destination where they are checked, corrected, and ordered. Protocol headers are stripped to re-form the original message.

Message from 205.16.4.4: “Mary had a little lamb”

Connection-Oriented
Connection-oriented:
The history of a circuit (1)

Source builds a message requesting a Circuit to the destination

Source: 205.16.4.4
Dest: 97.206.0.6

“Call”
Dest = “97.206.0.6”
Circuit ID = “203”
QoS Req’d

Connection-oriented:
The history of a circuit (2)

When a router receives the message, it:
1. Determines which of its output ports gives the “best” route to the destination

Source: 205.16.4.4
Dest: 97.206.0.6

“Call”
Dest = “97.206.0.6”
Circuit ID = “203”
QoS Req’d

Output 4: 3 hops
Connection-oriented: The history of a circuit (3)

When a router receives the message, it:
1. Determines which of its output ports gives the “best” route to the destination
2. Records the Circuit ID and the input and output port in its circuit routing table.

Connection-oriented: The history of a circuit (4)

When a router receives the message, it:
1. Determines which of its output ports gives the “best” route to the destination
2. Records the Circuit ID and the input and output port in its circuit routing table
3. Forwards the request through the selected output port.
Connection-oriented:
The history of a circuit (5)

The request is ultimately forwarded to the destination

Source: 205.16.4.4
Dest: 97.206.0.6
Circuit ID: 203
QoS Req’d

Connection-oriented:
The history of a circuit (6)

The destination node decides if it will accept the circuit and sends an ACCEPT or REJECT message back to the source. (Often, the ACCEPT/REJECT message is sent back over the original path so that routers on the path can drop the circuit if it is rejected)
Connection-oriented: The history of a circuit (7)

When the source node receives the ACCEPT message, it knows it has a dedicated channel to the destination node. It can begin sending data which just needs to be marked with the circuit number. The destination node will respond over the same channel.

Connection-oriented: The history of a circuit (8)

When the source is finished using the circuit, it will send a teardown message along the circuit. Routers receiving the message will drop the circuit.
Thinking about connection-oriented

- From the programmer’s standpoint:
  - The dedicated end-to-end “pipe” paradigm is attractive for some applications:
    - After initial setup, there’s very little work needed to communicate
    - If we have a long-lived, steady stream of data (e.g., streaming video), low overhead per data item makes up for high setup overhead

- From the network designer’s standpoint:
  - Some good points:
    - Since channel is dedicated, easier to set up service guarantees (QoS)
    - Don’t have to worry about some packet-related problems (e.g., out-of-order packets)
  - Some bad:
    - If the traffic over a circuit is relatively sporadic (e.g., typical request/reply traffic), the utilization of the circuit will be low and we will waste network bandwidth.
    - If the lifetime of a connection is short, we may not gain enough in the low overhead per data item to make up for the high setup overhead.
    - Could send a packet and receive an ACK in the time it takes to merely set up a circuit.
    - If a device on the connection fails, do we have to re-build the whole path?

Thinking about packet switching

- From the programmer’s standpoint:
  - For short, quick messaging, the idea of just packing the data up and firing it off seems right

- From the network designer’s standpoint:
  - With packet switching, multiplexing is, in a sense, automatic – packets from multiple channels are mixed together on the links, sharing the total bandwidth
  - Since every packet is individually routed, network topology changes and failures can be handled (hard for circuit switching)
  - Packet switching routers only have one job to do – generally produces simpler, faster designs
A best-of-both-worlds approach (TCP/IP)

- Give the programmer both connection-oriented (TCP) and connectionless (UDP) paradigms
- For network efficiency, run everything over a connectionless infrastructure (IP)

A third way to route: Source routing

The route is pre-calculated by the sender and sent as part of the message.

Advantages:
- Very simple, fast routers

Disadvantages:
- Sender must know the entire network
- Messages can be long
- Practical only for small, stable networks