Main Objective

- Data delivery from source to destination
Issues at the Network Layer

- Addressing
  - IP addresses on the Internet

- Switching
  - Datagram packet switching
  - Virtual circuit packet switching

- Routing
  - How to calculate a path from source to destination?

Models of the Network Service

- The “post office” model (IP)
  - Simple, unreliable, unordered delivery service
  - Different letters between the same parties are handled independently (connectionless model)
  - Stateless (no record of who’s corresponding with whom)
  - Intelligent end-user

- The “phone service” model (ATM)
  - Concept of “connections” between conversing users
  - Reliable delivery (of voice)
  - State is maintained regarding active connections
  - Dumb end-device (phone)
Connectionless vs. Connection-Oriented Service

- Network Layer Solutions
  - Connection-Oriented Service (ATM)
  - Connectionless Service (IP)
- Transport Layer Solutions (assume a connectionless network)
  - Connection-Oriented Service (TCP)
  - Connectionless Service (UDP)

Connectionless Service

- Each packet is transmitted independently
Connection-oriented Service

- Logical connection is established

Datagram Packet-Switching

- Unreliable, connectionless service
- Each packet is routed independently
Virtual-Circuit Packet Switching

- Connection-oriented service
- All packets of a VC follow the same route

Network Switches

- A network switch (or router) is a device that forwards data in the direction of the destination

Ethernet Switch (Link layer device)
Cisco Router (Network layer device)
Forwarding Tables

- Datagram packet switching

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next hop (port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
</tbody>
</table>

Packet Forwarding of Datagrams

- When a packet with destination node arrives at an incoming link, ...
  1. The router looks up the routing table
  2. The routing table lookup yields the address of the next node (next hop)
  3. The packet is transmitted onto the outgoing link that goes to the next hop

**Good:** The router does not need to know about flows

**Bad:** Size of the routing table can grow very large
Forwarding Tables

- Virtual circuit packet switching

<table>
<thead>
<tr>
<th>In Port</th>
<th>In VCI</th>
<th>Out Port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Packet Forwarding of Virtual Circuits

- When a packet with $V_{C_{in}}$ in header arrives from router $n_{in}$, ...
  1. The router looks up the routing table for an entry with $(V_{C_{in}}, n_{in})$
  2. The routing table lookup yields $(V_{C_{out}}, n_{out})$
  3. The router updates the VC# of the header to $V_{C_{out}}$ and transmits the packet to $n_{out}$

**Good:** Routing table is small (how small?)
**Bad:** Changing the route is complicated.
Routing table changes for each virtual circuit
VCI Establishment in Virtual Circuit Packet Switching

A establishes connection to D

Port0, VCI1 \rightarrow Port2, VCI5

Port3, VCI5 \rightarrow Port1, VCI7

Port4, VCI7

Source Routing

- Source determines the entire route of the packet
- Packet header carries complete route information
- Switches follow directions in packet header
- Problems?
Comparison

IP Model (connectionless, unreliable)
- No connection establishment
  - favors small transactions
- Complex transport layer
- Stateless network core
- Scalable
- Robust
- No reservation mechanism
- No QoS guarantees

ATM Model (connection-oriented, reliable)
- Signalling delay
  - favors large data transfers
- Complex network layer
- Per-connection state
- Scalability is a tough issue
- Vulnerable to router failure
- Easy reservation
- QoS

IP Protocol Stack

- IP (Internet Protocol) is a Network Layer Protocol

Copyright by Jorg Liebeherr 98,99
**IP Overview**

- IP is the highest layer protocol that is implemented at both routers and hosts.
- Offers unreliable connectionless service (send and deliver primitives).

**IP Addresses**

- Each router or host on the Internet has a unique global address, called the IP address.
- An IP address is 32 bits long.
- Encodes a network number and a host number.
- IP addresses are written in a dotted decimal notation:
  - **128.238.42.112** means
    - 10000000 in 1st Byte
    - 11101110 in 2nd Byte
    - 00101010 in 3rd Byte
    - 01110000 in 4th Byte
IP Datagrams

- IP breaks data up into Datagrams limited to 64K bytes each
- Datagrams prevent long flows from monopolizing the network for a long time
- In future gigabit networks the 64K limit can be increased
  - Example: For how long will a 1M byte datagram tie up a T1 line (1.5Mbps)?
  - How about a 1 Gbps optical fiber?
- Datagrams can further be fragmented depending on packet size of the data link layer (e.g., Ethernet)

IP Datagram Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>4 bits</td>
</tr>
<tr>
<td>header length</td>
<td>8 bits</td>
</tr>
<tr>
<td>Type of Service/TOS</td>
<td>8 bits</td>
</tr>
<tr>
<td>Total Length (in bytes)</td>
<td>16 bits</td>
</tr>
<tr>
<td>Identification (16 bits)</td>
<td></td>
</tr>
<tr>
<td>flags (3 bits)</td>
<td></td>
</tr>
<tr>
<td>Fragment Offset (13 bits)</td>
<td></td>
</tr>
<tr>
<td>TTL Time-to-Live (8 bits)</td>
<td></td>
</tr>
<tr>
<td>Protocol (8 bits)</td>
<td></td>
</tr>
<tr>
<td>Header Checksum (16 bits)</td>
<td></td>
</tr>
<tr>
<td>Source IP address (32 bits)</td>
<td></td>
</tr>
<tr>
<td>Destination IP address (32 bits)</td>
<td></td>
</tr>
<tr>
<td>Options (if any, &lt;40 bytes)</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td></td>
</tr>
</tbody>
</table>
**IP Header**

- **Version:**
  - Coexistence of multiple versions (smooth transition)

- **Identification, Fragment offset:**
  - 64K datagrams, fragmentation

- **ToS:**
  - Allows prioritized or differentiated services

- **TTL:**
  - Loop suppression

**IP Header (Continued)**

- **Protocol:** Specifies the higher-layer protocol. Used for demultiplexing to higher layers.

- **Header checksum:** verifies correctness of header.
IP Header (Continued)

- **Options:**
  - Security restrictions
  - **Record Route:** each router that processes the packet adds its IP address to the header.
  - **Timestamp:** each router that processes the packet adds its IP address and time to the header.
  - (loose) **Source Routing:** specifies a list of routers that must be traversed.
  - (strict) **Source Routing:** specifies a list of the only routers that can be traversed.

- **Padding:** ensures that header ends on a 4-byte boundary

---

IP Fragmentation

- Host A sends large IP datagram to host B
- Datagram traverses multiple networks with different MTUs
  - Problem?

```
+-----------+-------------+---------+-----------+----------+
| Host A    | Router      | Ethernet| Host B    |
| FDDI: 4352|             |         | Ethernet: 1500 |

MTUs: FDDI: 4352 Ethernet: 1500
```

- IP Router performs fragmentation
What’s involved in Fragmentation?

The following fields in the IP header are involved:

- **version** (4 bits)
- **header length**
- **Type of Service/TOS**
- **Total Length (in bytes)**
- **Identification**
- **Flags**
- **Fragment Offset**
- **TTL Time-to-Live** (8 bits)
- **Protocol** (8 bits)
- **Header Checksum (16 bits)**

<table>
<thead>
<tr>
<th>headers</th>
<th>fields</th>
<th>type</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>header length</td>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>int</td>
<td>16</td>
</tr>
<tr>
<td>TTL Time-to-Live</td>
<td>Protocol</td>
<td>int</td>
<td>16</td>
</tr>
<tr>
<td><strong>…..</strong></td>
<td><strong>…..</strong></td>
<td><strong>…..</strong></td>
<td><strong>…..</strong></td>
</tr>
</tbody>
</table>

**Identification** is the same in all fragments.

**Flags** contains a “more fragments” bit

*(There is also a “don’t fragment bit” that can be set)*

**Fragment offset** contains the offset of current fragment in the original datagram

**Total length** is changed by fragmentation

---

Copyright by Jorg Liebeherr 98,99