**IP Datagrams**

- **IP** is the network layer
  - packet delivery service (host-to-host).
  - translation between different data-link protocols.

**IP Addresses**

- **IP addresses** are not the same as the underlying data-link (MAC) addresses. **WHY?**
- **IP** is a network layer - it must be capable of providing communication between hosts on different kinds of networks (different data-link implementations).
- The address must include information about what **network** the receiving host is on. This is what makes routing feasible.

**IP Addresses**

- **IP addresses** are *logical* addresses (not physical)
- **32 bits.**
  - IP Addresses are usually shown in *dotted decimal* notation: 1.2.3.4
  - Includes a network ID and a host ID.
  - Every host must have a unique IP address.
  - IP addresses are assigned by a central authority (*American Registry for Internet Numbers* for North America).
  - [http://www.iana.org/ipaddress/ip-addresses.htm](http://www.iana.org/ipaddress/ip-addresses.htm)
**The four formats of IP Addresses**

<table>
<thead>
<tr>
<th>Class</th>
<th>NetID</th>
<th>HostID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td>Multicast Address</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>128 possible network IDs, over 16M host IDs per network ID</td>
</tr>
<tr>
<td>B</td>
<td>16K possible network IDs, 64K host IDs per network ID</td>
</tr>
<tr>
<td>C</td>
<td>Over 2 million possible network IDs, about 256 host IDs per network ID</td>
</tr>
</tbody>
</table>

**Network and Host IDs**

- A Network ID is assigned to an organization by a global authority.
- Host IDs are assigned locally by a system administrator.
- Both the Network ID and the Host ID are used for routing.

**IP Addresses**

- IP Addresses are usually shown in *dotted decimal* notation:
  1.2.3.4
  cse.sc.edu is 129.252.138.8

**Special IP addresses**

- An IP broadcast address has a host ID of all 1s.
- An IP address that has a host ID of all 0s is called a *network address* and refers to an entire network.
- localhost: 127.0.0.1
Subnet Addresses

- An organization can subdivide its host address space into groups called subnets.
- The subnet ID is generally used to group hosts based on the physical network topology.

Subnetting

- Subnets can simplify routing.
- IP subnet broadcasts have a host ID of all 1s.
- It is possible to have a single wire network with multiple subnets?

Subnet Mask

- How do we know how many bits are allocated for subnet ID, how many bits are allocated for host ID?
- Example: subnet mask: 255.255.255.0
- Question: what is the subnet ID for 129.252.138.8 if subnet is:
  - 255.255.255.0
  - 255.255.255.192
  - 255.255.127.0

Subnet mask

- Question: if the subnet mask is 255.255.255.128, are 129.252.138.8 and 129.252.138.127 on the same subnet?
  - Calculate the subnet ID
  - If the subnet IDs are the same => Yes
  - Otherwise => No

IP Routing

- Q: How do you get a packet from one network to another?
Routing table

- Destination IP address. Either host address or a network address.
- IP address of the next hop Router.
- Flags.
- Network interface.

```
/sbin/route
```

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Flags</th>
<th>Metric</th>
<th>Ref</th>
<th>Use</th>
<th>Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>129.252.138.0</td>
<td>*</td>
<td>255.255.255.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>eth1</td>
</tr>
<tr>
<td>loopback</td>
<td>*</td>
<td>255.0.0.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>lo</td>
</tr>
<tr>
<td>default</td>
<td>SWG130.cse.sc.edu</td>
<td>0.0.0.0</td>
<td>UG</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>eth1</td>
</tr>
</tbody>
</table>

Mapping IP Addresses to Hardware Addresses

- IP Addresses are not recognized by hardware.
- If we know the IP address of a host, how do we find out the hardware address?
- The process of finding the hardware address of a host given the IP address is called Address Resolution.

ARP

- The Address Resolution Protocol is used by a sending host when it knows the IP address of the destination but needs the Ethernet (or whatever) address.
- ARP is a broadcast protocol - every host on the network receives the request.
- Each host checks the request against its IP address - the right one responds.

```
Ethernet D. addr Ethernet S. addr Type
Sender Ethernet addr Sender IP addr Target Ethernet addr Target IP addr
```

Example – Proxy ARP

```
<table>
<thead>
<tr>
<th>Ethernet D. addr</th>
<th>Ethernet S. addr</th>
<th>Type</th>
<th>Sender Ethernet addr</th>
<th>Sender IP addr</th>
<th>Target Ethernet addr</th>
<th>Target IP addr</th>
</tr>
</thead>
</table>
```

```
Routing table @ R2
```

Example – routing

```
Routing table @ R2
```

```
Ethernet D. addr Ethernet S. addr Type | | | | | | | | Data | Ethernet header | IP header |
| Ethernet D. addr | Ethernet S. addr | Type | | | | | | Data | Ethernet header | IP header |
Reverse Address Resolution

- The process of finding out the IP address of a host given a hardware address is called Reverse Address Resolution.

- Reverse address resolution is needed by diskless workstations when booting (which used to be quite common).

RARP conversation

- HEY - Everyone please listen!
- My Ethernet address is 22:BC:66:17:01:75.
- Does anyone know my IP address?

- Hi Green! Your IP address is 128.213.1.17.

Services provided by IP

- **Connectionless** Delivery (each datagram is treated individually).
- **Unreliable** (delivery is not guaranteed).
- Fragmentation / Reassembly (based on hardware MTU).
- Routing.
- Error detection.

IP Datagram

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERS</td>
<td>1 byte</td>
</tr>
<tr>
<td>HL</td>
<td>1 byte</td>
</tr>
<tr>
<td>Service</td>
<td>1 byte</td>
</tr>
<tr>
<td>Datagram ID</td>
<td>1 byte</td>
</tr>
<tr>
<td>FLAG</td>
<td>1 byte</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>1 byte</td>
</tr>
<tr>
<td>TTL</td>
<td>1 byte</td>
</tr>
<tr>
<td>Protocol</td>
<td>1 byte</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>1 byte</td>
</tr>
<tr>
<td>Source Address</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Destination Address</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>0-4 bytes</td>
</tr>
<tr>
<td>Data</td>
<td>0-65432 bytes</td>
</tr>
</tbody>
</table>

IP Datagram Fragmentation

- Each fragment (packet) has the same structure as the IP datagram.
- IP specifies that datagram reassembly is done only at the destination (not on a hop-by-hop basis).
- If any of the fragments are lost - the entire datagram is discarded (and an ICMP message is sent to the sender).

IP Flow Control & Error Detection

- If packets arrive too fast - the receiver discards excessive packets and sends an ICMP message to the sender (SOURCE QUENCH).
- If an error is found (header checksum problem) the packet is discarded and an ICMP message is sent to the sender.
ICMP - Internet Control Message Protocol

- ICMP is a protocol used for exchanging control messages.
- Two main categories:
  - Query message
  - Error message
- Usage of an ICMP message is determined by type and code fields.
- ICMP uses IP to deliver messages.
- ICMP messages are usually generated and processed by the IP software, not the user process.

<table>
<thead>
<tr>
<th>IP header</th>
<th>ICMP Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 bytes</td>
<td></td>
</tr>
</tbody>
</table>

ICMP Message Format

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>code</td>
<td>checksum</td>
<td>payload</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ICMP Message Types

- Echo Request
- Echo Response
- Destination Unreachable
- Redirect
- Time Exceeded
- there are more ...

ICMP Address Mask Request and Reply

- intended for a diskless system to obtain its subnet mask.
- Id and seq can be any values, and these values are returned in the reply.
- Match replies with request type(17 or 18) code(0)

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>type(17 or 18)</td>
<td>code(0)</td>
<td>checksum</td>
<td>identifier</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>subnet mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transport Layer & TCP/IP

Q: We know that IP is the network layer - so TCP must be the transport layer, right?
A: No… well, almost.

TCP is only part of the TCP/IP transport layer - the other part is UDP (User Datagram Protocol).
**UDP User Datagram Protocol**

- UDP is a transport protocol
  - communication between processes
- UDP uses IP to deliver datagrams to the right host.
- UDP uses *ports* to provide communication services to individual processes.

**Ports**

- TCP/IP uses an abstract destination point called a protocol port.
- Ports are identified by a positive integer.
- Operating systems provide some mechanism that processes use to specify a port.

**UDP**

- Datagram Delivery
- Connectionless
- Unreliable
- Minimal

**UDP Datagram Format**

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>Length</th>
<th>Checksum</th>
<th>Data</th>
</tr>
</thead>
</table>

**TCP**

*Transmission Control Protocol*

- TCP is an alternative transport layer protocol supported by TCP/IP.
- TCP provides:
  - Connection-oriented
  - Reliable
  - Full-duplex
  - Byte-Stream
Connection-Oriented

- Connection oriented means that a virtual connection is established before any user data is transferred.
- If the connection cannot be established - the user program is notified (finds out).
- If the connection is ever interrupted - the user program(s) is finds out there is a problem.

Reliable

- Reliable means that every transmission of data is acknowledged by the receiver.
- If the sender does not receive acknowledgement within a specified amount of time, the sender retransmits the data.

Byte Stream

- Stream means that the connection is treated as a stream of bytes.
- The user application does not need to package data in individual datagrams (as with UDP).
  Somebody needs to do this since IP is delivering all the data, it's just that the application layer doesn't need to do this!

Buffering

- TCP is responsible for buffering data and determining when it is time to send a datagram.
- It is possible for an application to tell TCP to send the data it has buffered without waiting for a buffer to fill up.

Full Duplex

- TCP provides transfer in both directions (over a single virtual connection).
- To the application program these appear as 2 unrelated data streams, although TCP can piggyback control and data communication by providing control information (such as an ACK) along with user data.

TCP Ports

- Interprocess communication via TCP is achieved with the use of ports (just like UDP).
- UDP ports have no relation to TCP ports (different name spaces).
TCP Segments

The chunk of data that TCP asks IP to deliver is called a TCP segment.

Each segment contains:
- data bytes from the byte stream
- control information that identifies the data bytes

Addressing in TCP/IP

Each TCP/IP address includes:
- Internet Address
- Protocol (UDP or TCP)
- Port Number

NOTE: TCP/IP is a protocol suite that includes IP, TCP and UDP.

TCP vs. UDP

Q: Which protocol is better?
A: It depends on the application.

TCP provides a connection-oriented, reliable, byte stream service (lots of overhead).

UDP offers minimal datagram delivery service (as little overhead as possible).

TCP/IP Summary

- IP: network layer protocol
  - unreliable datagram delivery between hosts.
- UDP: transport layer protocol
  - unreliable datagram delivery between processes.
- TCP: transport layer protocol
  - reliable, byte-stream delivery between processes.
Hmmmm. TCP or UDP?
- Electronic commerce?
- Video server?
- File transfer?
- Email?
- Chat groups?
- Robotic surgery controlled remotely over a network?

Assignment & Next time
- Reading:
  - TI 3,4,5,6,17 **
  - TCP/IP FAQ
- Next Lecture:
  - Sockets Programming API