The Internet Protocol (IP)

What problem are we trying to solve?

Since there are numerous DL technologies and protocols, an internetwork is going to need to pass data between subnetworks with different:

- protocols
- addressing schemes
- speeds
- ...

How can we manage these problems efficiently in large internets?
What is IP?

- Most widely applied internetworking protocol
- The L3 protocol of the Internet
- Packet-oriented communication
- Best-effort (“unreliable”)

Two versions we care about:
- IPv4 -- the version currently in use (mostly)
- IPv6 -- the next version
### IPv4 packet format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP version</td>
<td>Identifies the IP version (4 for IPv4)</td>
</tr>
<tr>
<td>Header Length (in 4B wds)</td>
<td>Total length of packet in bytes</td>
</tr>
<tr>
<td>Type of Service</td>
<td>Options for how IP will treat the packet (will discuss when we get to QoS)</td>
</tr>
<tr>
<td>Message ID</td>
<td>Identifies this packet with a particular message between the source and destination. The combination of Source_address, Dest_address, Message_ID, Protocol, and Fragment_number identify this packet uniquely.</td>
</tr>
<tr>
<td>Flags</td>
<td>Only 2 of 3 bits defined. Used to support fragmentation (later chart).</td>
</tr>
<tr>
<td>Fragment offset</td>
<td>Used to ensure that packets will eventually die if not delivered. Originally intended to measure life in seconds; is processed as a hop count (every router decrements TTL until it reaches 0).</td>
</tr>
<tr>
<td>TTL</td>
<td>Identifies the Transport-level protocol (usually TCP or UDP).</td>
</tr>
<tr>
<td>Protocol</td>
<td>Used by the sender to request network services (padded to be a multiple of 32 bits)</td>
</tr>
<tr>
<td>Data</td>
<td>The total packet length including header and options can be 64KB.</td>
</tr>
</tbody>
</table>

### Notes on some IPv4 header fields

**Header Length:** Measured in 32-bit wds. Minimum is 5.

**Type of service:** Options for how IP will treat the packet (will discuss when we get to QoS)

**Message ID:** Identifies this packet with a particular message between the source and destination. The combination of Source_address, Dest_address, Message_ID, Protocol, and Fragment_number identify this packet uniquely.

**Flags:** Only 2 of 3 bits defined. Used to support fragmentation (later chart).

**TTL:** Used to ensure that packets will eventually die if not delivered. Originally intended to measure life in seconds; is processed as a hop count (every router decrements TTL until it reaches 0).

**Protocol:** Identifies the Transport-level protocol (usually TCP or UDP).

**Options:** Used by the sender to request network services (padded to be a multiple of 32 bits)

**Data:** The total packet length including header and options can be 64KB.
IP in the protocol stack

Message passed from L4 (24B)

In an IP packet (44B)

DA SA L/T H1 H2 ▼ ... ▼ M20 M1 M2 ▼ ... ▼ M24 Pad Pad CRC

In an Ethernet Frame (64B)

IPv4 addresses

General format:

Class ID Network number Host number

32 bits

How they are usually written and talked about:

Dotted decimal notation: Express each byte as its equivalent in decimal.

Example:

11000000 00101001 00000110 00010100

192 . 41 . 6 . 20
IPv4 address formats
(“classful” addressing)

A

| 7 bits | 24 bits |
|--------|
| Network number | Host number |

B

| 14 bits | 16 bits |
|---------|
| Network number | Host number |

C

| 21 bits | 8 bits |
|---------|
| Network number | Host number |

D

<table>
<thead>
<tr>
<th>28 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicast Address</td>
</tr>
</tbody>
</table>

Note: Class E (“11110”) is reserved for future use.

IPv4 addresses

<table>
<thead>
<tr>
<th>Class</th>
<th>Format (when reading in dotted decimal)</th>
<th>Range of Unreserved Addresses</th>
<th>Approximate number of networks/hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N.H.H.H</td>
<td>1.0.0.0 to 126.255.255.255</td>
<td>126 / 16M</td>
</tr>
<tr>
<td>B</td>
<td>N.N.H.H</td>
<td>128.0.0.0 to 191.255.255.255</td>
<td>16K / 64K</td>
</tr>
<tr>
<td>C</td>
<td>N.N.N.H</td>
<td>192.0.0.0 to 223.255.255.255</td>
<td>2M / 256</td>
</tr>
</tbody>
</table>

Some special reserved addresses:

- All zeroes: This host
- Network=0 w/ host #: The indicated host on this network
- All ones: Broadcast on this network.
- Network # w/ host=all ones: Broadcast on the indicated network
- Network=127 Loopback
Hierarchical addressing in IP networks

Network 192.12.*
Network 192.3.4.*
Network 192.3.*
Network 192.126.4.*
Network 192.126.*
Network 192.*

Rx
R1
R2
R3
R4

How hierarchical addressing streamlines routing (1)

Traffic for any address starting with 192 will be sent to R1.

This means that routing tables in other parts of the network can handle the entire 192.* network with one entry.

If this was not the case, every router in the Internet would need a routing table with millions of entries.
How hierarchical addressing streamlines routing (2)

R1's routing table
- 192.3.* 2
- 192.12.* 1
- 192.126.* 0
- default 3

R2's routing table
- 192.12.0.* 1
- 192.12.1.* 0
- 192.12.44.* 3
- default 2

Rx's routing table
- 192.* 0

The default route is used when no other route applies, meaning the packet is addressed to some other network or subnet.

Mapping IP addresses to L2 devices
IP addressing over MAC addresses

- IP addresses are “virtual” addresses assigned to a device. They do not relate to the device’s “real” address (its MAC address).

- When an IP packet arrives at its destination subnetwork, it needs to be delivered to the connected host having the specified IP address. But in most multidrop subnetworks (e.g., Ethernet), we need to know the MAC address -- the IP address does no good.

- This means that the subnetwork needs a system for translating IP addresses into MAC addresses.

The Address Resolution Protocol (ARP)

- Each host on the multidrop subnetwork maintains a table of the IP address and MAC address of each node on the subnetwork.

- When a host receives an IP packet, it:
  - Checks to see if the MAC address is in the table. If it is, it forwards the IP packet to the indicated MAC address.
  - If not, it broadcasts an ARP query on the subnetwork asking which node has the IP address in question.
  - The node with the IP address broadcasts its MAC address, allowing all nodes to update their tables.
IP Fragmentation

IP Packet Fragmentation

- Assume we send an IP packet through a subnetwork in which the frame payload size is smaller than the packet size.

- We could design to do either:
  1. **L2 Fragmentation:** Divide the IP packet among frames when it enters the subnetwork, then recombine them when it leaves the subnetwork.
     - Problems:
       - May introduce high delay by repeatedly fragmenting and re-assembling the same packet in different subnetworks.
       - Have to wait for all frames at the exit of each subnetwork.
  2. **L3 Fragmentation:** When entering the subnetwork, divide the packet into smaller IP-formatted packets. Re-assembly is done at the receiver.
Header fields supporting fragmentation

- Source Address
- Destination Address
- Message ID

<table>
<thead>
<tr>
<th>Flags:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF (if set, Don’t Fragment this packet)</td>
</tr>
<tr>
<td>MF (if set, More Fragments follow this fragment)</td>
</tr>
</tbody>
</table>

- Fragment Offset -- The offset (in bytes) of the data in this fragment packet referenced to the start of the data in the original packet

Fragmentation Example

- Original Packet 2000B
  - Frag1 1500B O=0 MF=1
  - Frag2 500B O=1500 MF=0

- FDDI MTU=4900
  - Frag1 1500B O=0 MF=1
  - Frag2 500B O=1500 MF=0

- PPP MTU=532
  - Frag1A 532B O=0 MF=1
  - Frag1B 532B O=532 MF=1
  - Frag1C 436B O=1064 MF=1

- Reassembled Packet 2000B
IP Tunneling

Sometimes, we want to set up a virtual point-to-point link across an IP internet:
- Make a virtual “Direct Connection”
- Redirect traffic to other addresses
- Use non-IP protocols
- Security

Protocol “p”
Network

IP hosts encapsulate the Protocol “p” packets inside normal IP packets

15.3.3.6

to: 1.2

abcde

to: 15.3.3.6

xxxxxx

xxxxxx

IP hosts encapsulate the Protocol “p” packets inside normal IP packets

15.3.3.6

to: 1.2

abcde

to: 15.3.3.6

xxxxxx

xxxxxx

IP hosts encapsulate the Protocol “p” packets inside normal IP packets

15.3.3.6

to: 1.2

abcde

to: 15.3.3.6

xxxxxx

xxxxxx