Last Time

- **Focus:**
  - What to do when one shared LAN isn't big enough?

- **Interconnecting LANs**
  - Bridges and LAN switches
  - But there are limits …
This Time: Internetworks

- Set of interconnected networks, e.g., the Internet
  - Scale and heterogeneity

```
Network 2 (Ethernet)
H4
H5
H6
```

```
Network 1 (Ethernet)
R1
R2
```

```
Network 3 (FDDI)
```

```
Network 4 (point-to-point)
```

The Protocol Stack

- Thinking about roles:
  - Transport: Process to Process
    - Example: TCP
      Reliable bytestream
  - Network: Host to Global Host
    - Example: IP
      Unreliable datagram
  - Data Link/Physical: Host to Local Host
    - Example: Ethernet
      Pretty reliable frame delivery
As a picture

- IP is the network layer protocol used in the Internet
- Routers are network level gateways
- Packet is the term for network layer protocol data units (PDUs)

Packet formats: encapsulation

- View of a packet on the (Ethernet) wires
  - Ethernet Header, IP Header, Higher layer headers and Payload
- Routers work with IP header, not higher
  - Higher would be a "layer violation"
- Routers strip and add link layer headers
Network Layer Goals

• Run over heterogeneous Link/Physical layers
  – Motivates minimizing promises about the service
    • End-to-end argument

• Global delivery
  – Must be scalable
  – This requires a new addressing scheme (IP addresses)
    • Want address of remote host to give clue to direction to send packet

• Low overhead switching
  – Minimal processing of IP packet
    • E.g., don’t have to rewrite IP header (much…)
  – “Fast path” processing

• Network control / diagnosis
  – If I’m having trouble communicating, what’s wrong?
    • Routers have IP addresses, just like everyone else
    • Ping / traceroute

Review: Network Service Models

• Datagram delivery: postal service
  – connectionless, best-effort or unreliable service
  – Network can’t guarantee delivery of the packet
  – Each packet from a host is routed independently
  – Example: IP

• Virtual circuit models: telephone
  – connection-oriented service
  – Signaling: connection establishment, data transfer, teardown
  – All packets from a host are routed the same way (router state)
  – Example: ATM, Frame Relay, X.25
Internet Protocol (IP)

- IP (RFC791) defines a datagram "best effort" service
  - May be loss, reordering, duplication, and errors!
  - Currently IPv4 (IP version 4), IPv6 on the way

- Routers forward packets using periodically updated routes
  - Routing protocols (RIP, OSPF, BGP) run between routers to maintain routes (routing table, forwarding information base)
  - Over medium term, one path from host A to host B

- Global, hierarchical addresses, not flat addresses
  - 32 bits in IPv4  (128 bits in IPv6)
  - ARP (Address Resolution Protocol) maps IP to MAC addresses for final delivery

The IP Narrow Waist

<table>
<thead>
<tr>
<th>Model</th>
<th>Protocols</th>
<th>The “narrow waist”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>Many (Ethernet, …)</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>TCP / UDP</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>Many (HTTP, SMTP)</td>
<td></td>
</tr>
</tbody>
</table>
IPv4 Packet Format

- Version is 4
- Header length is number of 32 bit words
- Limits size of options

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>19</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>HLen</td>
<td>TOS</td>
<td>Length</td>
<td>Identifier for Fragments</td>
<td>Flags</td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (variable)</td>
<td>Pad (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IPv4 Header Fields …

- Type of Service
- Abstract notion, never really worked out
  - Routers ignored
- But now being redefined for Diffserv

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>19</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>HLen</td>
<td>TOS</td>
<td>Length</td>
<td>Identifier for Fragments</td>
<td>Flags</td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (variable)</td>
<td>Pad (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IPv4 Header Fields …

- Length of packet
- Min 20 bytes, max 65K bytes (limit to packet size)

IPv4 Header Fields …

- Fragment fields
- More on this in a minute
### IPv4 Header Fields …

- **Time To Live**
  - Decremented by router and packet discarded if = 0
  - Prevents immortal packets
  - ping / traceroute

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0</td>
</tr>
<tr>
<td>HLen</td>
<td>4</td>
</tr>
<tr>
<td>TOS</td>
<td>8</td>
</tr>
<tr>
<td>Length</td>
<td>16</td>
</tr>
<tr>
<td>Identifier for Fragments</td>
<td>19</td>
</tr>
<tr>
<td>Flags</td>
<td>20</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>21</td>
</tr>
<tr>
<td>TTL</td>
<td>24</td>
</tr>
<tr>
<td>Protocol</td>
<td>25</td>
</tr>
<tr>
<td>Checksum</td>
<td>31</td>
</tr>
<tr>
<td>Source Address</td>
<td>32-47</td>
</tr>
<tr>
<td>Destination Address</td>
<td>48-63</td>
</tr>
<tr>
<td>Options (variable)</td>
<td>64-255</td>
</tr>
<tr>
<td>Pad (variable)</td>
<td>256-319</td>
</tr>
<tr>
<td>Data</td>
<td>320-1535</td>
</tr>
</tbody>
</table>

- **Identifies higher layer protocol**
  - E.g., TCP, UDP

- **De-mux’ing key at destination host**
### IPv4 Header Fields ...

- **Header checksum**
  - Doesn’t cover data

- **Recalculated by routers (TTL drops)**

- **Disappears for IPv6**


<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Bit Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0</td>
<td>0-3</td>
</tr>
<tr>
<td>HLen</td>
<td>4</td>
<td>4-7</td>
</tr>
<tr>
<td>TOS</td>
<td>8</td>
<td>8-11</td>
</tr>
<tr>
<td>Length</td>
<td>16</td>
<td>16-31</td>
</tr>
<tr>
<td>Identifier for Fragments</td>
<td>32</td>
<td>32-39</td>
</tr>
<tr>
<td>Flags</td>
<td>33</td>
<td>33-34</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>35,36</td>
<td>35-36</td>
</tr>
<tr>
<td>TTL</td>
<td>37,38</td>
<td>37-38</td>
</tr>
<tr>
<td>Protocol</td>
<td>39</td>
<td>39-39</td>
</tr>
<tr>
<td>Checksum</td>
<td>40</td>
<td>40-40</td>
</tr>
<tr>
<td>Source Address</td>
<td>41-43</td>
<td>41-43</td>
</tr>
<tr>
<td>Destination Address</td>
<td>44-47</td>
<td>44-47</td>
</tr>
<tr>
<td>Options (variable)</td>
<td>48-51</td>
<td>48-51</td>
</tr>
<tr>
<td>Pad (variable)</td>
<td>52-63</td>
<td>52-63</td>
</tr>
<tr>
<td>Data</td>
<td>64</td>
<td>64-63</td>
</tr>
</tbody>
</table>

- **Source/destination IP addresses**
  - Not Ethernet

- **Unchanged by routers**

- **Not authenticated by default**
IP Addresses and Datagram Forwarding

- IP addresses have hierarchy
  - MAC addresses are basically random

- How the source gets the packet to the destination:
  - if source is on same network (LAN) as destination, source sends packet directly to destination host, using MAC address
  - else source sends data to a router on the same network as the source (using router’s MAC address)
  - router will forward packet to a router on the next network over (by sending out through a different one of its interfaces, and MAC address on that network for next router)
  - and so on...
  - until packet arrives at router on same network as destination; then, router sends packet directly to destination host (MAC address)

- Requirements
  - every host needs to know address of a router on its LAN
  - every router needs a routing table to tell it which neighboring network to forward a given packet on
  - Need some kind of support for mapping IP address → MAC address

IP vs. MAC addresses

- All 192.168 addresses are this way
- ...

Routing table

- 192.168.93.10
- 140.142.13.107
IPv4 Header Fields …

- IP options indicate special handling
  - Timestamps
  - “Source” routes
- Rarely used …

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>19</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>HLen</td>
<td>TOS</td>
<td>Length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier for Fragments</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source Address

Destination Address

Options (variable) | Pad (variable)

Data

Problems / Strengths of IPv4

- TOS becomes traffic class / flow
- Length includes just the data
- No fragmentation info
- TTL still there
- Protocol field encoded through NextHdr
- No checksum
- Source / dest still there (but more bits)

The IPv6 header

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>4</th>
<th>12</th>
<th>16</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Traffic Class</td>
<td>FlowLabel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload Length</td>
<td>NextHdr</td>
<td>HopLimit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source address

Destination address

Next header / data
Fragmentation: What, Why, and Why Not

- Different networks may have different frame limits (MTUs)
  - Ethernet 1.5KB, FDDI 4.5KB
- Don’t know if packet will be too big for path beforehand
  - Could fragment on demand inside the network
    - IPv4
  - Could return an error to sending host
    - IPv6

Fragmentation and Reassembly

- Strategy
  - fragment only when necessary (MTU < Datagram size)
    - try to avoid fragmentation at source host
  - this implies that refragmentation must be possible
    - fragments are self-contained IP datagrams
  - delay reassembly until destination host
  - do not recover from lost fragments
Fragment Fields

- Fragments of one packet identified by (source, dest, frag id) triple
  - Make unique
- Offset gives start, length changed
- Flags are:
  - More Fragments (MF)
  - Don’t Fragment (DF)
  - Unused

Fragment Fields Diagram:

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| 0               | 4               | 8               | 16              | 19              | 31              |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Version         | HLen            | TOS             | Identifier for Fragments | Flags | Fragment Offset |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| TTL             | Protocol        | Checksum        | Source Address   | Destination Address |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Options (variable) | Pad (variable) | Data            |                 |                 |
```

Packet Format

Fragmenting a Packet Diagram:

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| 0               | 4               | 8               | 16              | 19              | 31              |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Start of header | Data bytes      | Data bytes      | Data bytes      | Data bytes      | Data bytes      |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Length          | Identifier for Fragments | Flags | Fragment Offset | TTL             | Protocol        |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Checksum        | Source Address  | Destination Address | Options (variable) | Pad (variable) | Data            |
```

10/9/2006
Fragment Considerations

- Making fragments be datagrams provides:
  - Tolerance of loss, reordering and duplication
  - Ability to fragment fragments

- Reassembly done at the endpoint
  - Puts pressure on the receiver, not network interior

- **Consequences of fragmentation:**
  - Loss of any fragments causes loss of entire packet
  - Need to time-out reassembly when any fragments lost

Avoiding Fragmentation

- Always send small datagrams
  - Might be too small
    - Why does that matter?

- “Guess” MTU of path
  - Use DF flag. May have large startup time

- **Discover actual MTU of path**
  - One RT delay w/help, much more w/o
    - Hosts send packets, routers return error if too large
Why Not?

• Why not implement fragmentation / reassembly in the network service?

• Not often used, but
  – Header overhead in every packet
  – Processing overhead on every packet
    • "Fast path" processing requires additional checks
  – Processing overhead when fragmentation needed
    • Have to create new IP headers, so…
    • Have to compute new checksums

ICMP

• What happens when things go wrong?
  – Need a way to test/debug a large, widely distributed system

• ICMP = Internet Control Message Protocol (RFC792)
  – Companion to IP – required functionality

• Used for error and information reporting:
  – Errors that occur during IP forwarding
  – Queries about the status of the network
ICMP Generation

Error during forwarding!

source

ICMP packet

dest

IP packet

Type | Code | Checksum

Common ICMP Messages

• Destination unreachable
  – “Destination” can be host, network, port or protocol
• Packet needs fragmenting but DF (don’t fragment) flag is set
• Redirect
  – To shortcut circuitous routing
• TTL Expired
  – Used by the “traceroute” program
• Echo request/reply
  – Used by the “ping” program
• Cannot Fragment
• Busted Checksum

• ICMP messages include portion of IP packet that triggered the error (if applicable) in their payload
ICMP Restrictions

- The generation of error messages is limited to avoid cascades … error causes error that causes error!

- Don’t generate ICMP error in response to:
  - An ICMP error
  - Broadcast/multicast messages (link or IP level)
  - IP header that is corrupt or has bogus source address
  - Fragments, except the first

- ICMP messages are often rate-limited too.

Key Concepts

- Network layer provides end-to-end data delivery across an internetwork, not just a LAN
  - Datagram and virtual circuit service models
  - IP/ICMP is the network layer protocol of the Internet

- Next: More detailed look at routing and addressing