Topic #13
Today’s Lecture Objectives

• ICMP

• IPv6 (and IPv4/v6 transition mechanisms)

• DHCP
Internet Control Message Protocol (ICMP)

- Used to send information about “problems” among various network devices (including senders and receivers)
  - Example: TTL expiration
  - Example: need to fragment, but DF set

- Encapsulated in an IP packet

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Sequence Number</td>
<td>Data</td>
</tr>
</tbody>
</table>
## ICMP Type and Code Examples

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>
**Traceroute Using ICMP**

- Not quite how you think a route tracing utility would work

- Send packet with TTL=x
  - Start with 1, then 2, …

- TTL will reach 0, ICMP will be sent
  - By incrementing TTL value, responses provide a trace of the path

- Traceroute typically does each TTL value three times
IPv6

• **Initial motivation:** 32-bit address space was soon to be completely allocated.

• **Additional motivation**
  – New header format reduces processing
  – Header changes to facilitate QoS
### IPv6 Header

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>payload len</td>
</tr>
<tr>
<td></td>
<td></td>
<td>next hdr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hop limit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source address  (128 bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>destination address (128 bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data</td>
</tr>
</tbody>
</table>

- Fixed length of 40 bytes (twice as long as IPv4)
IPv6 Header (Cont)

- Version (4 bits)
  - Same as IPv4 but now value is “0110”

- Priority (8 bits)
  - Similar to priority in IPv4, but now 8 bits
  - Remains to be seen how it will be used
    • Useful within domains

- Label (20 bits)
  - Used for labeling flows
  - Allows routers to more easily identify flows (for mgt)
  - Remains to be seen how it will be used
IPv6 Header (Cont)

• Payload Length (16 bits)
  – Similar to IPv4 length field—measures

• Next Header (8 bits)
  – Used to add optional header sections, uses a linked list mechanism where the field points to next header section
  – The LAST “Next Header” serves the same role the “Protocol” field in IPv4—identifies protocol encapsulated within IPv4 packet

• Hop Limit (8 bits)
  – Servers the same role as TTL in IPv4
  – Renamed to be more accurate—measures hops instead of time
IPv6 Header (Cont)

- Source/Dest Addresses (128 bits each)
  - Highly unlikely to run out of addresses any time soon
  - $3.4 \times 10^{38}$
  - 340,000 trillion, trillion, trillion addresses
  - Ex: 111.43.78.199.220.17.49.98.170.56.204.100.145.14.97.128

- All IPv4 address range fits into a /96

- …I think there’s enough addresses now
Fields NOT Included

• No IHL (fixed header)

• Protocol field and options are combined in a link-list structure

• All fragmentation is gone
  – All DLL protocols used with IPv6 must support at least 1500 bytes

• No checksum
  – Expectation that L2 does data integrity check

• Changes around priority/flow label
**Transition From IPv4 To IPv6**

- Not all routers can be upgraded at once
  - No “flag days”
  - How will the Internet transition to IPv6?

- **Tunneling**: IPv6 carried as payload in IPv4 datagram among IPv4 routers

- **Dual-Stack**: Routers can do both, know which one it is receiving based on Version
**Tunneling**

Logical view:

```
A   B   tunnel
IPv6 IPv6 IPv6 IPv6
```

Physical view:

```
A   B   C   D   E   F
IPv6 IPv6 IPv4 IPv4 IPv6 IPv6
```

- **A-to-B:** IPv6
- **B-to-C:** IPv6 inside IPv4
- **E-to-F:** IPv6

Flow:
- **Flow: X**
- **Src: A**
- **Dest: F**
- **data**

For full details, refer to the image.
Tunneling

• Encapsulation/Tunneling are powerful tools
  – Can be used to do data hiding

• Suggest how additional headers can be inserted to provide extra functionality
  – No longer just:

<table>
<thead>
<tr>
<th>DLL/MAC</th>
<th>Network</th>
<th>Transport</th>
<th>Data</th>
</tr>
</thead>
</table>
Dynamic Host Config Protocol (DHCP)

• Used to allocate an address to a host

• Uses a client/server architecture

• Makes config easier
  – esp. if hosts move often
  – limited number of addrs (private addrs)
**DHCP Steps**

1. **DHCP Discover**
   - From client to DHCP server
   - Client uses as its src address 0.0.0.0 and locally determined port
   - Sends a one-hop IP broadcast on Port 67 using UDP
   - DHCP header has transaction ID to avoid confusion

2. **DHCP Offer**
   - From DHCP server to broadcast address
   - Includes transaction ID, IP addr, netmask, gateway, DNS, lease time

3. **DHCP Request**
   - From client to DHCP server (now has server IP to use)
   - Still uses as its address 0.0.0.0
   - Selects from among “offers” and responds (if multiple servers respond)

4. **DHCP ACK**
   - From DHCP server to broadcast address
   - ACKs client’s selection
Private Addresses and NAT

• DHCP can be used to allocate public addresses, but often it is associated with private addresses
  – But we’ll talk about private addressing later

• For now, the private address ranges are:
  – 10/8 (10.0.0.0 to 10.255.255.255)
  – 172.16.0/12 (172.16.0.0 to 172.31.255.255)
  – 192.168/16 (192.168.0.0 to 192.168.255.255)