Internet Control Message Protocol

ICMP

• Way for source to learn why datagrams not delivered

• Researchers found alternate uses
  ○ ping
  ○ traceroute
  ○ path MTU discovery
**IP Error Detection**

- Checksum
  - if checksum is invalid, receiver discards datagram
  - why can’t the receiver alert the sender of the corrupted datagram?
- Who computes the checksum?
  - final destination or each router?

**ICMP and IP**

- IP uses ICMP to report errors
- ICMP is transported over IP
  - ICMP messages are *encapsulated* in IP datagrams
  - just like TCP and UDP are transported over IP
**ICMP Messages**

- ICMP message
  - type
  - code
  - first 8 bytes of IP datagram

  triggering the ICMP message

<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>network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>datagram too big</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</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>

(more listed on pg. 357)

**ICMP Messages Examples**

- echo request/reply
- destination unreachable
- TTL expired
- datagram too big

<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>network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>datagram too big</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</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>
**Uses of ICMP**

**Ping**

- Uses the echo request and echo reply messages
- Records round-trip time between source and destination

```
bayou% ping www.cs.unc.edu
PING dove.cs.unc.edu (152.2.131.244): 56 data bytes
64 bytes from 152.2.131.244: icmp_seq=0 ttl=53 time=41.827 ms
64 bytes from 152.2.131.244: icmp_seq=1 ttl=53 time=41.765 ms
64 bytes from 152.2.131.244: icmp_seq=2 ttl=53 time=108.391 ms
64 bytes from 152.2.131.244: icmp_seq=3 ttl=53 time=69.423 ms
64 bytes from 152.2.131.244: icmp_seq=4 ttl=53 time=41.279 ms
^C
--- dove.cs.unc.edu ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 41.279/60.537/108.391 ms
```

**Uses of ICMP**

**Traceroute**

- Uses the TTL field
  - TTL decremented at each router before forwarding
  - if router decrements to 0, packet discarded, router sends ICMP message (type 11)
- Sends packets with increasing TTL fields (starting at 1)
  - 3 packets for each TTL value
- Records RTT (like ping)
% traceroute access.cs.clemson.edu

Tracing route to yoda.cs.clemson.edu [130.127.48.4] over a maximum of 30 hops:

<table>
<thead>
<tr>
<th>Hop</th>
<th>Time (ms)</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>192.168.2.127</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>68.152.247.190</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>68.152.247.205</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>205.152.122.25</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>axr00asm-7-0-0-0.bellsouth.net [65.83.237.20]</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>pxr00asm-2-0-0.bellsouth.net [65.83.236.2]</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>so-7-0-0-0.garl1.Atlantal.Level3.net [67.72.8.41]</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>so-6-0-0.gar2.Atlantal.Level3.net [209.247.9.161]</td>
</tr>
<tr>
<td>9</td>
<td>19</td>
<td>so-3-0.bbr2.Atlantal.Level3.net [209.247.11.225]</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
<td>ae-0-0.bbr2.Washington1.Level3.net [64.159.0.230]</td>
</tr>
<tr>
<td>11</td>
<td>31</td>
<td>so-7-0-0.edge1.Washington1.Level3.net [209.244.11.14]</td>
</tr>
<tr>
<td>13</td>
<td>32</td>
<td>205.171.251.33</td>
</tr>
<tr>
<td>14</td>
<td>33</td>
<td>dca-core-02.inet.qwest.net [205.171.8.221]</td>
</tr>
<tr>
<td>15</td>
<td>34</td>
<td>atl-core-02.inet.qwest.net [205.171.8.153]</td>
</tr>
<tr>
<td>16</td>
<td>45</td>
<td>atl-edge-19.inet.qwest.net [205.171.21.122]</td>
</tr>
<tr>
<td>17</td>
<td>49</td>
<td>216.207.16.42</td>
</tr>
<tr>
<td>18</td>
<td>52</td>
<td>130.127.3.25</td>
</tr>
<tr>
<td>19</td>
<td>50</td>
<td>130.127.12.4</td>
</tr>
<tr>
<td>20</td>
<td>78</td>
<td>yoda.cs.clemson.edu [130.127.48.4]</td>
</tr>
</tbody>
</table>

**Uses of ICMP**

**Path MTU Discovery**

- Minimum MTU along a path
  - *path MTU*
  - avoid fragmentation

- Probe for path MTU
  - send datagram with NO_FRAGMENT bit set
  - ICMP type 3, code 4 if larger than MTU
  - send increasingly smaller datagrams until no ICMP errors received
The Transport Layer
Transport services and protocols

- Transport protocols:
  - Provide logical communication between application processes running on different hosts
  - Execute on the end systems (and not in the network)

- Transport vs. network layer services:
  - Network layer: data transfer between end systems
  - Transport layer: data transfer between processes
    - Relies on, and enhances, network layer services

Transport Layer Protocols
Internet Transport Services

- TCP: Reliable, in-order, unicast delivery
  - Congestion control
  - Flow control
  - Connection setup

- UDP: Unreliable, unordered ("best-effort"), unicast or multicast delivery
  - (Minimal) error detection

- Services not available:
  - Performance guarantees
    - No guarantees of available bandwidth
    - No guarantees of end-to-end delay
**Transport Layer**

**Protocols & Services**

- Fundamental transport layer services
  - Multiplexing/Demultiplexing
  - Error detection
  - Reliable data delivery
  - Pipelining
  - Flow control
  - Congestion control

- Service implementation in Internet transport protocols
  - UDP
  - TCP

**Transport Layer Services**

**Multiplexing/Demultiplexing**

- Each end-system has a single protocol “stack”
  - The stack is shared between all applications using the network

- **Multiplexing** is the process of allowing multiple applications to use the network simultaneously
  - (To send data into the network concurrently)

- **Demultiplexing** is the process of delivering received data to the appropriate application
Multiplexing/Demultiplexing

**Review: Protocol layering in the Internet**

At the sender, each layer takes data from above
- May subdivide into multiple data units at sending layer
- Adds header information to create new data unit
- Passes new data unit to layer below
- The process is reversed at the receiver

**Multiplexing/Demultiplexing**

**Demultiplexing**

Demultiplexing is the process of delivering received segments to the correct application-layer process
- IP address (in network-layer datagram header) identifies the receiving machine
- Port number (in transport-layer segment header) identifies the receiving process
Multiplexing/Demultiplexing

Transport protocol specific demultiplexing

- Demultiplexing actions depend on whether the transport layer is connectionless (UDP) or connection-oriented (TCP)

- UDP demultiplexes segments to the socket
  - UDP uses 2-tuple to identify the socket
    - <destination IP address, destination port number>
  - Socket is “owned” by some process (allocated by OS).

- TCP demultiplexes segments to the connection
  - TCP uses 4-tuple to identify connection
    - <source IP addr, source port nbr, destination IP addr, destination port nbr>
  - Connection (and its socket) is owned by some process

### Multiplexing/Demultiplexing Examples

**Examples**

![Diagram showing multiplexing/demultiplexing examples]
**Internet Transport Protocols**

**User Datagram Protocol (UDP) [RFC 768]**

- No frills, “bare bones” Internet transport protocol
- Best effort service — UDP segments may be:
  - Lost
  - Delivered out of order
  - Delivered multiple times
- “Connectionless”
  - No handshaking between UDP sender, receiver
  - Each UDP segment handled independently of others
- Error Detection
  - Based on checksum
  - Make sure received packets haven’t been corrupted

**UDP Datagram Format**

- IP source and destination address not included
  - assumed to be in IP header
- Checksum is optional
- Checksum includes
  - UDP header
  - IP source
  - IP destination
  - IP type

Length field is length in bytes, of UDP segment (including header)
UDP And Encapsulation

- UDP is carried inside an IP datagram

UDP Is it useful?

- Who uses UDP?
  - Often used for streaming multimedia applications
  - Loss tolerant
  - Rate sensitive

- Reliable transfer over UDP still possible
  - Reliability can always be added at the application layer
  - (Application-specific error recovery)

Why use UDP?

- No connection establishment (which can add delay)
- Simple: no connection state at sender, receiver
- Small segment header
- No congestion control: UDP can blast away as fast as desired