ICMP - Internet Control Message Protocol + ping/traceroute

TCP/IP class
ICMP

- intro
  - encapsulation/stack position
  - basic ideas
  - header format
- message types
- redirects
- ping
- traceroute
ICMP transmitted within IP datagram so that it is routeable (unlike arp)
# Internet Protocols

<table>
<thead>
<tr>
<th>Apps</th>
<th>Transports</th>
<th>Network</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>email (smtp)</td>
<td>tcp</td>
<td>icmp</td>
<td>arp/rarp</td>
</tr>
<tr>
<td>telnet/rlogin</td>
<td></td>
<td>ip</td>
<td>ethernet II (or 802.3)</td>
</tr>
<tr>
<td>ftp/rcp</td>
<td>udp</td>
<td></td>
<td>slip or ppp</td>
</tr>
<tr>
<td>http(www)/gopher</td>
<td></td>
<td>“raw”/ip</td>
<td>phone line, ISDN</td>
</tr>
</tbody>
</table>
ICMP - ideas

- considered part of IP
- functionality includes:
  - error messages (ttl exceeded, destination unreachable, router is out of memory, can’t fragment packet)
  - network management (ping/traceroute)
  - end host configuration (router advert, netmask)
- error messages go from router/end host to original ip src, not between intermediate hops
  - don’t know route

Jim Binkley
ICMP - ideas

- error messages typically sent at IP layer, received by sending IP/UDP/TCP, latter may forward to application
- ICMP error messages never generated due to:
  - ICMP error message (loop)
  - broadcasts/multicasts
- Why? prevent broadcast storms
- error contains offending IP header + 1st 8 bytes of IP data (note tcp/udp ports)

Jim Binkley
**general icmp header**

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

depends on type/code...

checksum covers icmp header/data, not ip header
## ICMP requests types (not all)

<table>
<thead>
<tr>
<th>type</th>
<th>code</th>
<th>purpose</th>
<th>error?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>host unreachable</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>port unreachable</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>DF and must fragment</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench</td>
<td>yep</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>redirect - network</td>
<td>kinda</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
<td>no</td>
</tr>
</tbody>
</table>
## ICMP requests (cont.)

<table>
<thead>
<tr>
<th>type</th>
<th>code</th>
<th>purpose</th>
<th>error?</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/10</td>
<td>0</td>
<td>router advert/solicit</td>
<td>no</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>time exceeded, ttl = 0</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>timeout during reassembly</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>0/1</td>
<td>parameter problems</td>
<td>yes</td>
</tr>
<tr>
<td>13/14</td>
<td>0</td>
<td>timestamp request/reply</td>
<td>no</td>
</tr>
<tr>
<td>17/18</td>
<td>0</td>
<td>netmask request/reply</td>
<td>no</td>
</tr>
</tbody>
</table>
ICMP redirect

- limited dynamic routing technique
- only done on same link
- situation:
  - 1. assume dumb host with 1 default routing table entry
  - 2. two routers on same link, one is default, one is router to net X
  - 3. dumb host sends pkt to net X via default router
  - 4. default router sends ICMP redirect with correct router address to dumb host

Jim Binkley
ICMP redirect, cont.

- note: router detects redirect because it discovers that packet is being forwarded back out input i/f
- default router also forwards original packet correctly
- dumb host changes its routing table to reflect newly learned route to other net
- route added is HOST route in BSD system because we lack subnet mask knowledge
redirect picture

Rdefault

Sally to Sue

redirect to Sally

msg: next time to Sue via Rother PLEASE!

Sue

Rother

Rdefault forwards packet

Jim Binkley
ping - ICMP echo request/reply

- ping program useful diagnostic tool, uses ICMP echo request/reply packets
- BSD implementation uses “raw” sockets - i/f directly to ip layer, bypass transports
- older ping would send 1 pkt per second
- some newer pings require -s to do that and only do one ping (“joebob is alive”)
ping ping

- ping adds identifier/sequence number fields to packets
- id field, unix pid as raw socket can’t tell how to demux packets to apps, app gets all copies, must demux itself
- sequence # allows you to see if packets disappeared
- ping will also do roundtrip timing
ping ping ping

◆ so what do you learn?
  – timing info, does it take too long?
  – are packets being lost (why? didn’t tell you)
  – you can route (!!!)
  – end system’s tcp/ip stack is working at least

◆ echo reply sent by end system’s ICMP, you don’t know that you can telnet there...
ping example

$ ping cse.ogi.edu
PING cse.ogi.edu (129.95.20.2): 56 data bytes
64 bytes from 129.95.20.2 icmp_seq=0 time=8ms
64 bytes from 129.95.20.2 icmp_seq=1 time=8ms
64 bytes from 129.95.20.2 icmp_seq=2 time=20ms

---cse.ogi.edu PING statistics ---
3 packets transmitted, 3 packets received, 0% loss
round-trip (ms) min/avg/max = 8/12/20

Jim Binkley
traceroute

- `% traceroute north.pole.com`
- `traceroute` (a command) allows you to determine the routers from one end to another
- uses ICMP ttl exceeded and (UDP port unreachable OR ICMP echo reply) messages to do the job
traceroute example

% traceroute cse.ogi.edu (from sirius.cs.pdx.edu)
traceroute to cse.ogi.edu (129.95.20.2), 30 hops max ...
1. pdx-gwy (131.252.20.1) 3 ms 4 ms 3 ms
2. 198.104.197.58 (198.104.197.58) 7 ms 4 ms 8 ms
3. portland1-gw.nwnet.net (198.104.196.193) 6 ms 5 ms 5 ms
4. ogi-gw-nwnet.net (198.104.196.129) 8 ms 7 ms 7 ms
5. cse.ogi.edu (129.95.20.2) 14 ms 7 ms 9 ms

note: try from psu to intel or some other business in the Portland area. how many hops? how long?

Jim Binkley
rough traceroute algorithm

ttl = 1 (to 1st router)
while we haven’t got UDP port unreachable
    send raw/ip packet with ttl == 1
    get response
    if time exceeded note roundtrip time
else if UDP port unreachable
    quit
print output
ttl++
study questions

◆ if you can find kernel src (BSD...), when/why are the following messages sent:
  – source quench
  – routing redirect
◆ is traceroute unidirectional or bidirectional?
◆ does it really tell you the exact path?
◆ look at the ping man page and find out what a “flood” ping does. Useful?