Lecture 11.

The Internet Layer

IP (Internet Protocol)

&

ICMP (Internet Control Message Protocol)
Internet Protocol (IP)
RFC 791 (1981)

- **Connectionless**
  - datagram delivery service

- **best-effort**

- **Unreliable**
  - no guarantees of reception & packet order
  - error-handling algorithm: throw away packet!
    - Upon buffer congestion
    - upon error check failed
IP functions

→ **in trasmission:**
  - Encapsulates data from transport layer into datagrams
  - prepare header (src & dest addresses, etc)
  - apply routing algorithm
  - send datagram to network interface

→ **in reception:**
  - check validity of incoming datagrams
  - read header
  - verify whether datagram is to be forwarded
  - if datagram has reached destination, deliver payload to higher layer protocol
### IP datagram format

20 bytes header (minimum)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Header length</td>
</tr>
<tr>
<td>Header length</td>
<td>Type of Service (TOS)</td>
</tr>
<tr>
<td>Total Length</td>
<td></td>
</tr>
<tr>
<td>16 bit identification</td>
<td>flags 3 bit</td>
</tr>
<tr>
<td>13 bit fragment offset</td>
<td></td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>Protocol</td>
</tr>
<tr>
<td>Protocol</td>
<td>Header checksum</td>
</tr>
<tr>
<td>32 bit source IP address</td>
<td></td>
</tr>
<tr>
<td>32 bit destination IP address</td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Padding (0s)</td>
</tr>
<tr>
<td>Data (if any)</td>
<td></td>
</tr>
</tbody>
</table>
**Version:** 0100 (IPv4)
- allows to use multiple IP versions simultaneously...

**Header length:** in 32bit words
- default: 0101 (5 x 32bit words = 20 bytes)
- may extend header length up to 60 bytes

**SRC and DEST addresses**
- obvious...

<table>
<thead>
<tr>
<th>Version</th>
<th>Header length</th>
<th>Type of Service TOS</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16 bit identification</td>
<td>flags 3 bit</td>
<td>13 bit fragment offset</td>
</tr>
<tr>
<td>Time to Live TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 bit source IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 bit destination IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Header length</td>
<td>Type of Service TOS</td>
<td>Total Length</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>---------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16 bit identification
flags 3 bit
13 bit fragment offset

Time to Live TTL
Protocol
Header checksum

32 bit source IP address
32 bit destination IP address

**Total length: 16 bits**

⇒ up to 65535 (including header)

⇒ Necessary, as you cannot rely on datalink for data size

⇒ example: Ethernet has minimum payload size = 46 bytes

⇒ but you may send smaller IP datagram.

⇒ How to recognize how much of the 46 bytes is IP datagram?

**Protocol: specifies to which upper layer protocol the datagram must be delivered**

⇒ 1=ICMP; 2=IGMP, 6=TCP, 17=UDP
Why the protocol field? Demultiplexing!

Typical approach
Adopted in most protocols
(see e.g. datalink protocols)

Demultiplexing was also a TCP/UDP feature (versus application layer)
done by using full socket address <src IP, src Port, dest IP, dest Port>

8 bits: not too large (in principle the Internet is not doomed to TCP/UDP)!
<table>
<thead>
<tr>
<th>Version</th>
<th>Header length</th>
<th>Type of Service (TOS)</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 bit identification</td>
<td>flags 3 bit</td>
<td>13 bit fragment offset</td>
<td></td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td>32 bit source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 bit destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TTL**: max no. of hops the datagram can remain in the network

- from 0 to 255; generally initially set to 64
- each router decrements TTL of 1 (or every 1 second latency)
- when TTL=0 (input datagram with TTL=1), packet thrown away
  - sender notified via ICMP message
- Prevents datagrams from traveling forever (e.g. captured in loops)

**Header Checksum**: header only

- Same approach of TCP/UDP
- Efficient incremental computation at routers (RFC 1141), since only TTL changes (decrements)
<table>
<thead>
<tr>
<th>Version</th>
<th>Header length</th>
<th>Type of Service TOS</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16 bit identification</th>
<th>flags 3 bit</th>
<th>13 bit fragment offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Live TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
</tr>
<tr>
<td>32 bit source IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 bit destination IP address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Precedence field:**
  - ignored today

- **TOS bits (3,4,5,6):**
  - bit 3: minimize delay
  - bit 4: maximize throughput
  - bit 5: maximize reliability
  - bit 6: minimize monetary cost

- **last bit:** unused

- **TOS:**
  - 0 1 2 3 4 5 6 7

- **Precedence field**
  - TOS bits
  - Only 1 TOS bit set at a time

- **all bits to 0 = normal service**

---

G.Bianchi, G.Neglia, V.Mancuso
TOS bits

- RFC 1340 & 1349 specify how these bits should be set by standard apps. Examples:
  - FTP data = max_thr
  - telnet = min_del
  - SNMP (simple network management protocol) = max_reliability
  - NNTP (usenet news) = min_cost

- Routers may ignore TOS
  - TOS is just a suggestion
  - In practice, TOS field not set by hosts and ignored by routers until 1992-1993

Today (from 1998), TOS field renascence: Differentiated Services Code Point (DSCP)
Options

Up to 40 extra bytes (10 x 32bit words) available for options.
Common options:

► Record Route Option (RRO)
  ➞ 60 bytes header set with remaining options field empty
  ➞ each crossed router adds its IP address
    ➞ maximum of 9 hops recordable - not practical today

► Timestamp Option
  ➞ like RRO, but routers also stamp crossing time instant

► Source Route Option (Loose, Strict)
  ➞ allows sender to specify which routers must be crossed by
    the datagram (i.e. bypasses network routing tables)
    ➞ loose: cross the routers specified, in the order, plus
      others along the path (interconnecting specified ones)
    ➞ strict: ALL routers specified, and no others! (may fail if
      routers not directly connected)
Record Route Option details

- **Code (1 byte):** specifies option
  - code for RRO = 7

- **len (1 byte):** specifies bytes reserved for option
  - max = 39 bytes as extra header is at most 40 bytes, generally 39

- **ptr (1 byte):** tells where next address must be stored
  - minimum ptr value = 4, others multiple (8, 12, 16, 20, 24, 28, 32, 36)
  - ptr = 40 indicates that list is full

Which router IP address recorded (there are two!)?? RFC791 says outgoing interface!

G.Bianchi, G.Neglia, V.Mancuso
Traceroute

👉 Originally a debugging software program written by Van Jacobson
  ➞ Test TTL field
  ➞ Makes smart use of TTL

👉 Allows to trace the route from source to destination host
  ➞ Not limited to 9 hops as when RR option is used
  ➞ Does not require ANY specific router capability
**Traceroute idea (1)**

- Send subsequent sets of 3 UDP packets to destination
  - Start using TTL=1
  - After each set, increments TTL of 1 unit
  - Listen for the response…
Traceroute idea (2)

When router decrements TTL to 0:
- throws away packet
- returns ICMP “time exceeded” message
  - clearly containing router IP address

Transmitting host:
- records router (pretty print with reverse name lookup)
- computes RTT to router
ICMP “Time exceeded” error

- **ICMP header**: 8 bytes
- **ICMP data part**:
  - **IP header**: carrying src IP = router IP
  - **IP data**: IP Header (including options) + first 8 bytes of original IP data

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (11)</td>
<td>Code (0 or 1)</td>
<td>Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unused (must be all 0s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Header (including options) + first 8 bytes of original IP data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Code = 0**: traceroute case (router detected a TTL decrement to 0)

**Code = 1**: timed out while reassembling
Traceroute idea (3)

- Send UPD packet to destination, using VERY HIGH PORT NUMBERS
- since no UDP socket will listen, destination host will return ICMP message “port unreachable”
- Hence sender knows that packet has reached destination!
Traceroute from 131.175.21.3:
131.175.21.8
131.175.12.42
131.175.15.2

Traceroute from 131.175.15.2:
131.175.15.1
131.175.12.32
131.175.21.3
### Traceroute (Cisco, from uniroma1)

- D:\users>tracert www.cisco.com
- Rilevazione route a www.cisco.com [198.133.219.25]
- su un massimo di 30 punti di passaggio:

| 1 | <10 ms | <10 ms | <10 ms | 151.100.37.1 |
| 2 | 30 ms  | 40 ms  | 30 ms  | 151.100.238.1|
| 3 | 30 ms  | 40 ms  | 120 ms | rc-uniroma1.rm.garr.net [193.206.131.49] |
| 4 | 30 ms  | 40 ms  | 40 ms  | rt-rc-1.rm.garr.net [193.206.134.161] |
| 5 | 60 ms  | 40 ms  | 50 ms  | na-rm-2.garr.net [193.206.134.45] |
| 6 | 150 ms | *      | 150 ms | garr.ny4.ny.dante.net [212.1.200.145] |
| 7 | 170 ms | 190 ms | 161 ms | 500.POS2-0.GW6.NYC9.ALTER.NET [157.130.254.245] |
| 8 | 160 ms | *      | 171 ms | 527.at-5-0-0.XR2.NYC9.ALTER.NET [152.63.24.70] |
| 9 | 151 ms | 200 ms | 210 ms | 0.so-3-0-0.TR2.NYC9.ALTER.NET [152.63.22.94] |
| 10| 250 ms | *      | 250 ms | 125.at-6-2-0.TR2.SAC1.ALTER.NET [152.63.9.249] |
| 11| 280 ms | 260 ms | *      | 196.ATM7-0.XR2.SFO4.ALTER.NET [152.63.51.17] |
| 12| 281 ms | 250 ms | 261 ms | 190.ATM6-0.GW8.SJC2.ALTER.NET [152.63.52.181] |
| 13| 250 ms | 341 ms | 370 ms | cisco.customer.alter.net [157.130.200.30] |
| 14| 300 ms | 281 ms | 390 ms | 192.150.47.2 |
| 15| *      | 261 ms | 260 ms | www.cisco.com [198.133.219.25] |

- Rilevazione completata.
**Traceroute (UCLA, from uniroma1)**

- D:\users>tracert www.ucla.edu
- Rilevazione route a www.info.ucla.edu [164.67.80.80]
- su un massimo di 30 punti di passaggio:

<table>
<thead>
<tr>
<th>Numero</th>
<th>Latenza (ms)</th>
<th>Tempo (ms)</th>
<th>Latenza (ms)</th>
<th>Indirizzo IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;10</td>
<td>10</td>
<td>&lt;10</td>
<td>151.100.37.1</td>
</tr>
<tr>
<td>2</td>
<td>160</td>
<td>40</td>
<td>161</td>
<td>151.100.238.1</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>41</td>
<td>40</td>
<td>rc-uniroma1.rm.garr.net [193.206.131.49]</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>30</td>
<td>*</td>
<td>rt-rc-2.rm.garr.net [193.206.134.165]</td>
</tr>
<tr>
<td>5</td>
<td>240</td>
<td>40</td>
<td>70</td>
<td>na-rm-1.garr.net [193.206.134.41]</td>
</tr>
<tr>
<td>6</td>
<td>150</td>
<td>*</td>
<td>150</td>
<td>garr.ny4.ny.dante.net [212.1.200.145]</td>
</tr>
<tr>
<td>7</td>
<td>241</td>
<td>280</td>
<td>150</td>
<td>212.1.201.35</td>
</tr>
<tr>
<td>8</td>
<td>310</td>
<td>401</td>
<td>*</td>
<td>Abilene-DANTE.abilene.ucaid.edu [212.1.200.222]</td>
</tr>
<tr>
<td>9</td>
<td>*</td>
<td>300</td>
<td>*</td>
<td>clev-nycm.abilene.ucaid.edu [198.32.8.29]</td>
</tr>
<tr>
<td>10</td>
<td>*</td>
<td>481</td>
<td>330</td>
<td>ipls-clev.abilene.ucaid.edu [198.32.8.25]</td>
</tr>
<tr>
<td>11</td>
<td>661</td>
<td>*</td>
<td>261</td>
<td>kscy-ipls.abilene.ucaid.edu [198.32.8.5]</td>
</tr>
<tr>
<td>12</td>
<td>*</td>
<td>400</td>
<td>431</td>
<td>dnvr-kscy.abilene.ucaid.edu [198.32.8.13]</td>
</tr>
<tr>
<td>13</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Richiesta scaduta</td>
</tr>
<tr>
<td>14</td>
<td>350</td>
<td>*</td>
<td>320</td>
<td>losa-scrm.abilene.ucaid.edu [198.32.8.18]</td>
</tr>
<tr>
<td>15</td>
<td>340</td>
<td>*</td>
<td>330</td>
<td>USC--abilene.ATM.calren2.net [198.32.248.85]</td>
</tr>
<tr>
<td>16</td>
<td>320</td>
<td>321</td>
<td>340</td>
<td>ISI--USC.POS.calren2.net [198.32.248.26]</td>
</tr>
<tr>
<td>17</td>
<td>*</td>
<td>311</td>
<td>330</td>
<td>UCLA--ISI.POS.calren2.net [198.32.248.30]</td>
</tr>
<tr>
<td>18</td>
<td>341</td>
<td>340</td>
<td>551</td>
<td>cbn6-gsr.calren2.ucla.edu [169.232.1.22]</td>
</tr>
<tr>
<td>19</td>
<td>531</td>
<td>581</td>
<td>*</td>
<td>ci7200-2msa-cbn6.cbn.ucla.edu [169.232.3.18]</td>
</tr>
<tr>
<td>20</td>
<td>341</td>
<td>340</td>
<td>541</td>
<td>ikura.library.ucla.edu [164.67.80.80]</td>
</tr>
</tbody>
</table>

- Rilevazione completata.
Traceroute (Tor Vergata)

D:\users>tracert www.uniroma2.it

Rilevazione route a list.uniroma2.it [160.80.2.16]
su un massimo di 30 punti di passaggio:

1  <10 ms  <10 ms  <10 ms  151.100.37.1
2   30 ms   40 ms   30 ms  151.100.238.1
3   30 ms   80 ms   80 ms  rc.uniroma1.rm.garr.net [193.206.131.49]
4   40 ms   40 ms   61 ms  uniroma2-rc.uniroma2.rm.garr.net [193.206.131.150]
5   60 ms   70 ms  171 ms  list.uniroma2.it [160.80.2.16]

Rilevazione completata.
Traceroute (New Zealand, 1999)

C:	rcert www.auckland.ac.nz

Rilevazione route a www.auckland.ac.nz [130.216.1.7] su un massimo di 30 punti di passaggio:

1  <10 ms 10 ms <10 ms 151.100.37.1
2  591 ms 230 ms 991 ms 151.100.238.1
3  1553 ms 1682 ms 1873 ms rc-uniromaI.rm.garr.net [193.206.131.49]
4  1993 ms 2213 ms 1573 ms rt-rc-old.rm.garr.net [193.206.134.213]
5  1873 ms * 1241 ms na-rm-2.garr.net [193.206.134.45]
6  1352 ms 1211 ms 1042 ms garr-neaples.ny.dante.net [212.1.200.105]
7  1222 ms 1442 ms 1702 ms 500.POS2-2.GW9.NYC4.ALTER.NET [157.130.19.21]
8  1462 ms 1031 ms 1052 ms 110.ATM2-0.XR2.NYC4.ALTER.NET [152.63.21.206]
9  *   2123 ms 2113 ms 188.ATM3-0.TR2.NYC1.ALTER.NET [146.188.179.38]
10  2073 ms * 1572 ms 104.ATM5-0.TR2.CHI4.ALTER.NET [146.188.136.153]
11  1192 ms * * 198.ATM7-0.XR2.CHI4.ALTER.NET [146.188.208.229]
12  1873 ms 1602 ms 991 ms 194.ATM9-0-0.BR1.CHI1.ALTER.NET [146.188.208.13]
13  591 ms 1161 ms 972 ms us-il-chi-core2-rtr-a10-0-0.px.concentric.net [137.39.23.70]
14  1132 ms 991 ms * us-ca-la-core1-a1-0-0d17.rtr.concentric.net [207.88.0.101]
15  1352 ms 1172 ms 991 ms b2-fa-0-0-0.losangeles.clix.net.nz [206.111.43.34]
16  *   872 ms * 203.167.249.222
17  941 ms 1162 ms * ba2-fe0-1-0-acld.Auckland.clix.net.nz [203.97.2.244]
18  *   1192 ms 1372 ms clix-uofauckland-nz-2.cpe.clix.net.nz [203.167.226.46]
19  1181 ms 1382 ms 1232 ms www.auckland.ac.nz [130.216.1.7]

Rilevazione completata.
Traceroute (New Zealand, 2000)

D:\users>tracert www.auckland.ac.nz
Rilevazione route a www.auckland.ac.nz [130.216.1.7]
su un massimo di 30 punti di passaggio:

1 <10 ms <10 ms <10 ms 151.100.37.1
2 30 ms 40 ms 40 ms 151.100.238.1
3 30 ms 40 ms 40 ms rc-uniroma1.rm.garr.net [193.206.131.49]
4 70 ms 40 ms 110 ms rt-rc-1.rm.garr.net [193.206.134.161]
5 231 ms 40 ms 70 ms na-rm-2.garr.net [193.206.134.45]
6 150 ms 351 ms 160 ms garr.ny4.ny.dante.net [212.1.200.145]
7 * 190 ms 210 ms 500.POS3-0.GW5.NYC9.ALTER.NET [157.130.254.241]
8 170 ms 180 ms 321 ms 520.at-6-0-0.XR1.NYC9.ALTER.NET [152.63.24.26]
9 250 ms 201 ms 170 ms 181.ATM5-0.BR3.NYC9.ALTER.NET [152.63.23.145]
10 171 ms 170 ms * 137.39.52.2
11 301 ms 230 ms 240 ms a0-0d752.edge1.lax-ca.us.xo.com [207.88.0.185]
12 281 ms 410 ms 361 ms us-ca-la-core1-g4-0-0.rtr.concentric.net [206.111.0.4]
13 581 ms 641 ms 561 ms b2-fa-0-0-0.losangeles.clix.net.nz [206.111.43.34]
14 440 ms 561 ms 441 ms core2-atm1-0-0-2-acld.auckland.clix.net.nz [203.167.249.222]
15 * 430 ms 411 ms ba2-fe0-1-0-acld.auckland.clix.net.nz [203.97.2.244]
16 380 ms 381 ms 431 ms clix-uofauckland-nz-2.cpe.clix.net.nz [203.167.226.46]
17 390 ms 431 ms 390 ms www.auckland.ac.nz [130.216.1.7]
Rilevazione completata.
IP datagram fragmentation
Why fragmentation
physical networks have different
Maximum Transmission Units (MTU)

Routing decision

MTU=1500

191.133.5.7
IF1 (FDDI)

191.133.22.1
IF2 (Ethernet)

112.42.32.1
IF3 (ATM)

Fragmentation

IP Datagram
3000 bytes

MTU=4352

Fragmentation performed at
IP level (physical network
remains unaware)
# MTU examples

**RFC 1191**

<table>
<thead>
<tr>
<th>Network</th>
<th>MTU (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM token ring 16Mbps</td>
<td>17914</td>
</tr>
<tr>
<td>4 Mbps Token Ring (IEEE 802.5)</td>
<td>4464</td>
</tr>
<tr>
<td>FDDI</td>
<td>4352</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1500</td>
</tr>
<tr>
<td>IEEE 802.2 802.3</td>
<td>1492</td>
</tr>
<tr>
<td>X.25</td>
<td>576</td>
</tr>
<tr>
<td>point to point (PPP, SLIP)</td>
<td>May be set to 296 for interactive use</td>
</tr>
</tbody>
</table>
Fragmentation & reassembly

Fragmentation eventually performed by intermediate router
Fragments are independent packets, and may take independent routes
Reassembly always at destination (typical of IP)

G.Bianchi, G.Neglia, V.Mancuso
Picky notation

→ **IP datagram**
  - unit of end to end transmission at the IP layer
  - before fragmentation and after reassembly

→ **IP packet**
  - unit of data passed between the IP layer and the link layer
  - a packet can be either a complete datagram or a fragment
Multiple fragmentation
is possible!

Source host: 3000

Destination host: 3000

G.Bianchi, G.Neglia, V.Mancuso
### IP header - fragmentation fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td>4 bits, identifies the IP version number (currently 4)</td>
</tr>
<tr>
<td><strong>Header length</strong></td>
<td>16 bits, specifies the length of the header in 32-bit units</td>
</tr>
<tr>
<td><strong>Type of Service (TOS)</strong></td>
<td>8 bits, provides service class information for networks</td>
</tr>
<tr>
<td><strong>Total Length</strong></td>
<td>16 bits, specifies the length of the datagram in 32-bit units</td>
</tr>
<tr>
<td><strong>Identification</strong></td>
<td>16 bits, unique value per datagram: allows to understand to which datagram fragments belong</td>
</tr>
<tr>
<td><strong>Flags</strong></td>
<td>3 bits: [X, DF, MF]</td>
</tr>
<tr>
<td><strong>Fragment offset</strong></td>
<td>13 bits, specifies the position of the fragment in the original datagram payload</td>
</tr>
<tr>
<td><strong>Time to Live (TTL)</strong></td>
<td>8 bits, specifies how long the packet can live before it is discarded</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>8 bits, identifies the protocol used in the datagram</td>
</tr>
<tr>
<td><strong>Header checksum</strong></td>
<td>16 bits, provides a checksum to detect if the header has been damaged</td>
</tr>
<tr>
<td><strong>Source IP address</strong></td>
<td>32 bits, specifies the source IP address</td>
</tr>
<tr>
<td><strong>Destination IP address</strong></td>
<td>32 bits, specifies the destination IP address</td>
</tr>
</tbody>
</table>

**Identification:**
- unique value per datagram: allows to understand to which datagram fragments belong

**Flags:** 3 bits: [X, DF, MF]
- X = unused (0), DF = Don't Fragment, MF = More Fragments

**Fragment offset = payload fragment position in the original datagram payload**
fragmentation

Identification = xxx, DF = 0, MF=0, Fragment Offset =0

Identification = xxx, DF = 0, MF=1, Fragment Offset =0

Identification = xxx, DF = 0, MF=1, Fragment Offset = m₁

Identification = xxx, DF = 0, MF=0, Fragment Offset = m₂
Fragment offset

Must be number for 0 to 65515 \(2^{16}-1-20\)
only 13 bits available

idea: fragment = offset measured in 8 bytes units

example: fragment offset=322 means fragment payload portion starts from original datagram payload byte=2576

consequence: fragment size must be a multiple of 8 bytes

consequence: max 8192 fragments...

out of order fragments

initial order is reconstructed by fragment offset specification
More Fragment flag

- Necessary to understand that fragmentation is finished
  - Fragment headers carry length of FRAGMENT, not of DATAGRAM!
  - Hence, there is no way to deduce initial datagram size
  - Trick: use the MF bit
- MF=1: other fragments follow
- MF=0: this is last fragment
  - For non-fragmented datagram, MF=0
The problem of fragmentation

→ If one fragment lost, the entire datagram must be retransmitted!
  ➔ Retransmission is task of higher layers: IP does not deal with it

Fragmentation is seen as a problem and overhead, and avoided whenever possible
Dont Fragment (DF) flag

- Impedes routers to perform fragmentation on datagram

- what happens when router receives datagram with DF on, and cannot deliver in because of too little MTU?
  - Throws away datagram
  - Returns ICMP message “fragmentation needed but dont fragment bit set” (type 3, code 4) to source
Path MTU discovery

- TCP MSS exchange discovers lower MTU (2800) between source and destination network
- some internal MTU may be lower (1500)
- this may be different on uplink & downlink paths
  - uplink = 1500; downlink = dest = 2800
Path MTU discovery with TCP
RFC 1191

→ Establish connection and determine MSS
→ Send first segment with MSS and DF=1 in IP datagram
→ if ack received, MSS is OK (no fragmentation)
  ⇒ proceed sending datagrams with DF=1
  (route changes may occur)

→ if ICMP message "fragmentation needed but dont fragment bit set" received:
  ⇒ reduce segment size (more later) until datagram received

→ when some time passes (and routes may change) try with higher MTU
  ⇒ generally try with original MSS
  ⇒ RFC1191 recommends 10 minutes; solaris2.2 uses 30s
ICMP “Fragmentation needed but DF bit set” error

A specific case (code 4) of the ICMP “Unreachable” error (type 3)

ICMP message format:

<table>
<thead>
<tr>
<th></th>
<th>Type (3)</th>
<th>Code (4)</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused (all 0s)</td>
<td></td>
<td>MTU of next hop network (bytes)</td>
<td></td>
</tr>
<tr>
<td>IP Header (including options) + first 8 bytes of original IP data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MTU specification only in newer ICMP implementations

in New Routers Requirements RFC (1993), new form requested

old implementations: unused (all 0 field)
MSS determination

-With newer ICMP message form:
  ➔ TCP Host sets MSS= discovered MTU
    ➔ (minus TCP and IP header, of course)
  ➔ and iterates procedure
    ➔ a following network may have lower MTU!

-With older ICMP message form
  ➔ no MTU specification: must guess!
    ➔ Ordered list of all possible network MTUs specifed
       into details in RFC1191
       » ..., 1500, 1492, 1006, 576, 552, 544, 512, 508, 296, ...
ICMP
Internet Control Message Protocol
RFC 792 (1981)
Which layer ICMP is?

- Allows routers and network entities to exchange control (query & error) messages
  - network layer protocol?

- **ICMP messages encapsulated into IP datagrams, and mux-demuxed by IP as a transport protocol**
  - transport layer protocol?

- **ICMP messages exchanged among the sw processes running IP (at routers and hosts)**
  - destination is the IP layer.... Is thus ICMP below IP layer???

But, ultimately, who cares which layer ICMP is?!!
For Internet people, ICMP is part of the Internet Layer...
ICMP packets

- Travel in the network as normal IP packets
  - contribute to increase network traffic & congestion
- Are subject to dropping & error
- Intermediate routers do not recognize them as ICMP
  - unless error occurs
### ICMP encapsulation

<table>
<thead>
<tr>
<th>IP header</th>
<th>ICMP message</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Further header specification – depends on type &amp; code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content – depends on Type and Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TYPE:** identifies ICMP message (15 types standardized)

**CODE:** further specifies the message (available in some type cases)

**CHECKSUM:** covers entire ICMP message (usual algo)
# ICMP message Types

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION</th>
<th>QUERY or ERROR</th>
<th>CODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Echo reply</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Destination Unreachable</td>
<td>ERROR</td>
<td>Many: 0-15</td>
</tr>
<tr>
<td>4</td>
<td>Source Quench</td>
<td>ERROR</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Redirect</td>
<td>ERROR</td>
<td>Many: 0-3</td>
</tr>
<tr>
<td>8</td>
<td>Echo Request</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Router Advertisement</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Router Solicitation</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Time Exceeded</td>
<td>ERROR</td>
<td>0=transit, 1=reassembly</td>
</tr>
<tr>
<td>12</td>
<td>Parameter problem</td>
<td>ERROR</td>
<td>0=bad header, 1=missing opt</td>
</tr>
<tr>
<td>13</td>
<td>Timestamp request</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Timestamp reply</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>Information Request (obsolete)</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Information Reply (obsolete)</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Address Mask Request</td>
<td>QUERY</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Address Mask Reply</td>
<td>QUERY</td>
<td>0</td>
</tr>
</tbody>
</table>
### ICMP Error Message format

always contain infos about IP packet that generated error

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Further header specification

- IP Header (including options)
- + first 8 bytes of original IP data

(All UDP header - include source & destination port numbers for TCP header!!)

- **ICMP error NEVER generated in response to ICMP error messages**
  - To avoid possibility of infinite loop
  - but errors may be generated when caused by ICMP query

- **ICMP errors also non generated when “broadcast storm” possible**
  - Datagram destined to IP broadcast or multicast address
  - Datagram sent as link-layer broadcast
  - Fragment other than the first
  - Datagram whose source does not describe single host
# Destination Unreachable (type 3) codes

<table>
<thead>
<tr>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Network unreachable</td>
</tr>
<tr>
<td>1</td>
<td>Host unreachable</td>
</tr>
<tr>
<td>2</td>
<td>Protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>Port unreachable</td>
</tr>
<tr>
<td>4</td>
<td>Fragmentation needed but DF bit set</td>
</tr>
<tr>
<td>5</td>
<td>Source route failed</td>
</tr>
<tr>
<td>6</td>
<td>Destination network unknown</td>
</tr>
<tr>
<td>7</td>
<td>Destination host unknown</td>
</tr>
<tr>
<td>8</td>
<td>Source host isolated (obsolete)</td>
</tr>
<tr>
<td>9</td>
<td>Destination network administratively prohibited</td>
</tr>
<tr>
<td>10</td>
<td>Destination host administratively prohibited</td>
</tr>
<tr>
<td>11</td>
<td>Network Unreachable for ToS</td>
</tr>
<tr>
<td>12</td>
<td>Host Unreachable for ToS</td>
</tr>
<tr>
<td>13</td>
<td>Communication administratively prohibited by filtering</td>
</tr>
<tr>
<td>14</td>
<td>Host precedence violation</td>
</tr>
<tr>
<td>15</td>
<td>Precedence cutoff in effect</td>
</tr>
</tbody>
</table>
**Destination Unreachable header**

<table>
<thead>
<tr>
<th>Type (11)</th>
<th>Code (0-15)</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

IP Header + first 8 bytes IP data (UDP header)

Unused header: exception: code 4 (no fragmentation because DF set)
Port unreachable: port known from 8 bytes IP data carried
ICMP message size: 56 bytes (+ physical network header)
ICMP Echo request/reply: PING program

- Name “ping” = sonar operation to locate objects
- Client sends ICMP echo request (type 8) and waits for ICMP echo reply (type 0)
- Meanwhile, measures the RTT
- done over multiple packets, measures loss %

Is the *first* diagnostic tool used when doubts on a computer connectivity arise
Ping diagnostic results

⇒ No response
  ⇒ no other connection is possible!
  ⇒ When IP software up, ICMP sw must!

⇒ Significant Lost packets ( > 2 or 3%) 
  ⇒ transmission errors on LAN/WAN
  ⇒ severe congestion and dropping at routers

⇒ high RTT ( > 100 or 200 ms)
  ⇒ congestion
  ⇒ interactive user may suffer

⇒ no loss, constant low delay
  ⇒ network perfect: the problem stays in the application...
Echo message format

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (0 or 8)</td>
<td>Code (0)</td>
<td>Checksum</td>
<td>Identifier (16 bits)</td>
<td>Sequence number</td>
</tr>
<tr>
<td>Optional data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Sequence number starts from 0
- generally, 1 ping every 1s (option -s unix, -t dos)
- Identifier: set by client (unix: id of sending process)
- Echo reply: integrally copies packet (changes only type)
- RTT computation: time of sending stored in data portion
Ping: examples

D\users>ping
[-v TOS]
   [-r numero] [-s numero] [[-j host-list] | [-k host-list]]
   [-w timeout] elenco-destinazione
Opzioni:
   -t             Ping eseguito sull'host specificato finché non viene interrotto.
   -a             Risolve gli indirizzi in nomi host.
   -n numero     Invia numero di richieste di eco.
   -l lunghezza  Invia dimensione buffer.
   -f             Imposta il flag Non frammentare nel pacchetto.
   -i TTL        Vita pacchetto.
   -v TOS        Tipo di servizio.
   -r count      Registra route per il conteggio dei punti di passaggio.
   -s count      Marca orario per il conteggio dei punti di passaggio.
   -j host-list  Libera route di origine lungo l'elenco host.
   -k host-list  Restringi route di origine lungo l'elenco host.
   -w timeout    Intervallo attesa (in millisecondi) per ogni risposta.

D\users>ping -n 3 net.infocom.uniroma1.it
Esecuzione di Ping net.infocom.uniroma1.it
   [151.100.37.12] con 32 byte di dati:
   Risposta da 151.100.37.12: byte=32 durata=10ms
   Risposta da 151.100.37.12: byte=32 durata<10ms
   Risposta da 151.100.37.12: byte=32 durata<10ms

D\users>ping -n 3 www.cisco.com
Esecuzione di Ping www.cisco.com
   [198.133.219.25] con 32 byte di dati:
   Risposta da 198.133.219.25: byte=32 durata=240ms
   Risposta da 198.133.219.25: byte=32 durata=231ms
   Risposta da 198.133.219.25: byte=32 durata=230ms

G.Bianchi, G.Neglia, V.Mancuso
**ICMP Source Quench error**

<table>
<thead>
<tr>
<th>Type (4)</th>
<th>Code (0)</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

- Error that *MAY* be generated by host when receives datagram at a rate too fast to be processed
- represents a form of flow control for UDP

- use of source quench required by RFC 1009...
- ... but deprecated by new router requirements RFC!
  - Consumes network bandwidth and is ineffective & unfair!
ICMP query examples

Address mask request/reply

- e.g. during boot of diskless computer
- Example: 07 8 1 5 3 1

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (17 or 18)</td>
<td>Code (0)</td>
<td>Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier (16 bits)</td>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 bit subnet mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 12 bytes

Timestamp request/reply

- timestamp = number of ms past midnight UTC (0 - 86,400,000)
- Example: 12 bytes

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (17 or 18)</td>
<td>Code (0)</td>
<td>Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier (16 bits)</td>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Originate timestamp (filled by requestor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive timestamp (filled by replying when receives)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit timestamp (filled by replying when transmitting)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 20 bytes