Lecture 11.

The Internet Layer

IP (Internet Protocol)
&
ICMP (Internet Control Message Protocol)

G.Bianchi, G.Neglia, V.Mancuso

Università degli Studi di Palermo

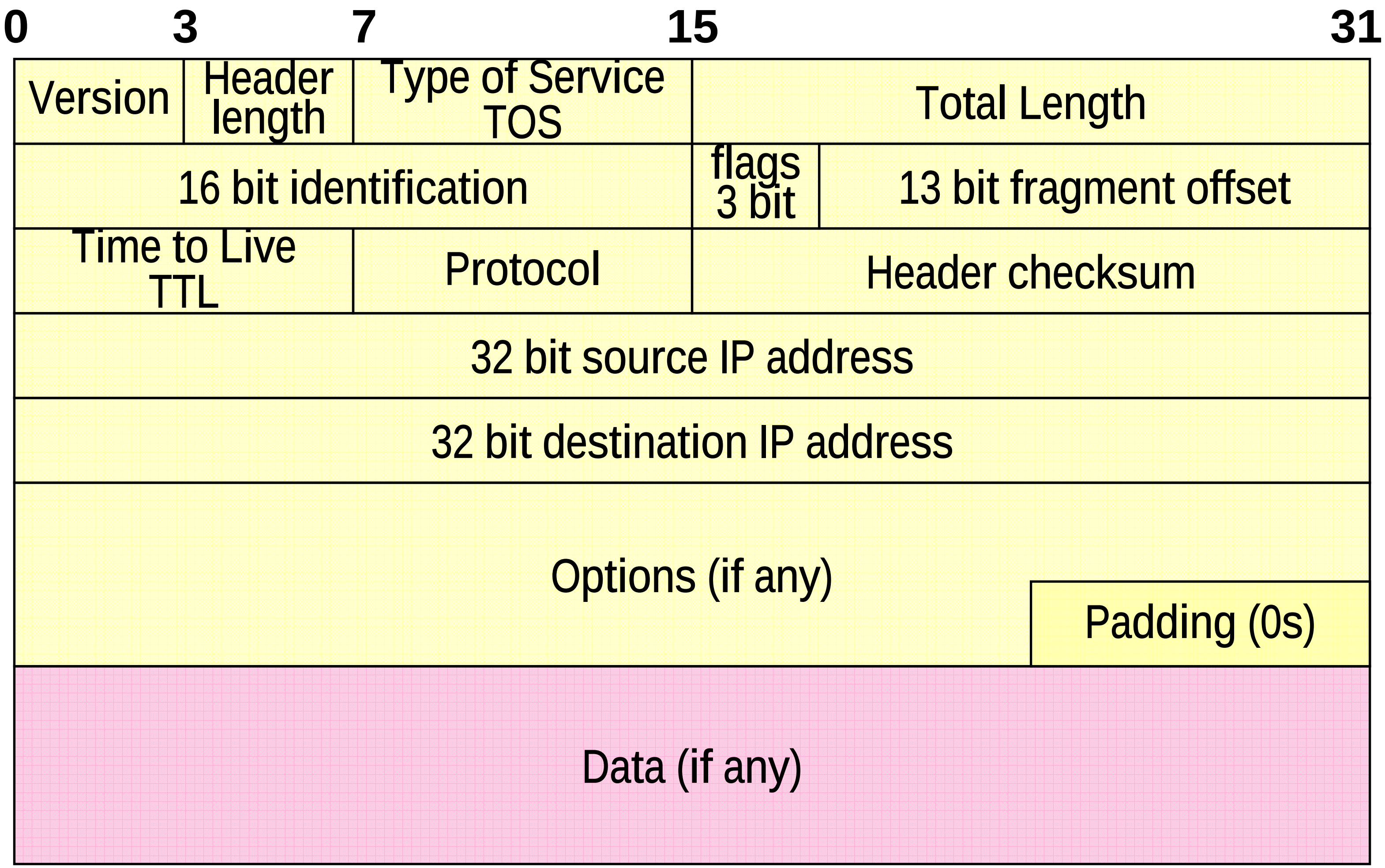
Internet Protocol (IP)
RFC 791 (1981)

- **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>
<tr><td colspan="4">Version</td><td colspan="4">Header length</td><td colspan="8">Type of Service TOS</td><td colspan="16">Total Length</td></tr>
<tr><td colspan="16">16 bit identification</td><td colspan="3">flags 3 bit</td><td colspan="13">13 bit fragment offset</td></tr>
<tr><td colspan="8">Time to Live TTL</td><td colspan="8">Protocol</td><td colspan="16">Header checksum</td></tr>
<tr><td colspan="32">32 bit source IP address</td></tr>
<tr><td colspan="32">32 bit destination IP address</td></tr>
</table>

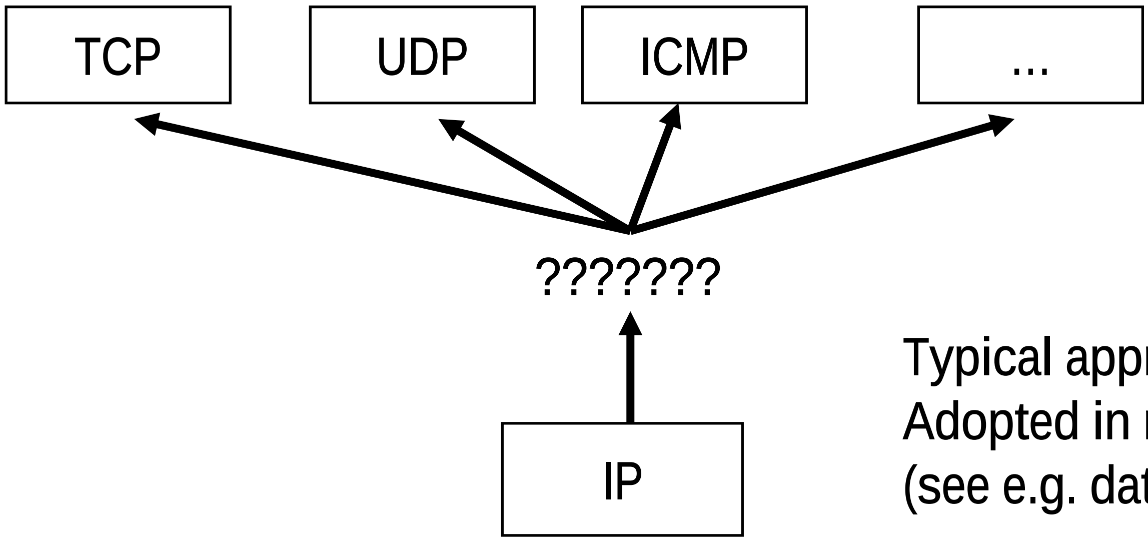
- ➔ **Version: 0100 (IPv4)**
  - ⇨ allows to use multiple IP versions simultaneously…
- ➔ **Header length: in 32bit words**
  - ⇨ default: 0101 (5 x 32bit words = 20 bytes)
  - ⇨ may extend header length up to 60 bytes
- ➔ **SRC and DEST addresses**
  - ⇨ obvious…

<table>
<tr><td>Version</td><td>Header length</td><td>Type of Service TOS</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">16 bit identification</td><td>flags 3 bit</td><td>13 bit fragment offset</td></tr>
<tr><td colspan="2">Time to Live TTL</td><td>Protocol</td><td colspan="2">Header checksum</td></tr>
<tr><td colspan="5">32 bit source IP address</td></tr>
<tr><td colspan="5">32 bit destination IP address</td></tr>
</table>

- **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>
<tr><td>Version</td><td>Header length</td><td>Type of Service TOS</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">16 bit identification</td><td>flags 3 bit</td><td>13 bit fragment offset</td></tr>
<tr><td colspan="2">Time to Live TTL</td><td>Protocol</td><td colspan="2">Header checksum</td></tr>
<tr><td colspan="5">32 bit source IP address</td></tr>
<tr><td colspan="5">32 bit destination IP address</td></tr>
</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 1second 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)

Version	Header length	Type of Service TOS	Total Length	
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				

- ➔ 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				0

➔ *Only 1 TOS bit set at a time*

➔ *all bits to 0 = normal service*

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)

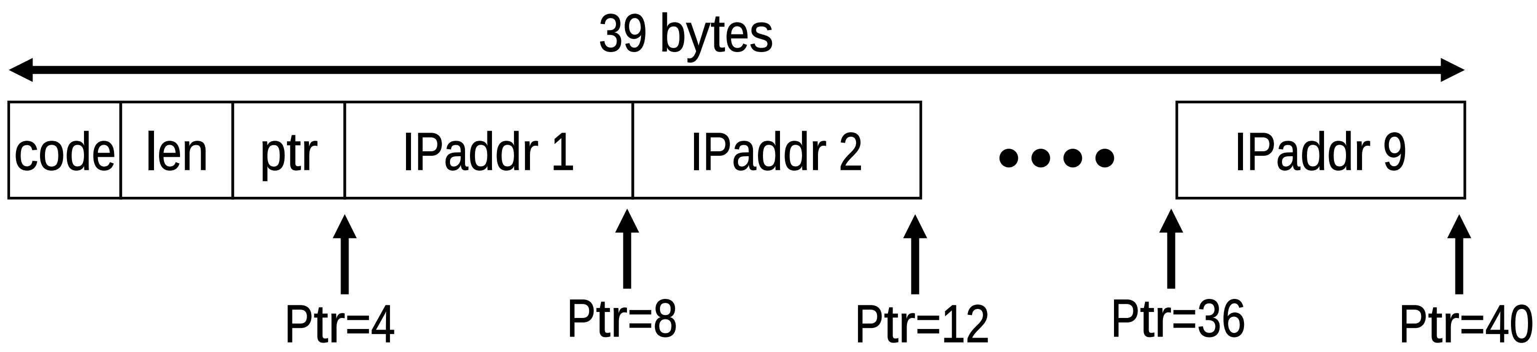
Options

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=39bytes 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

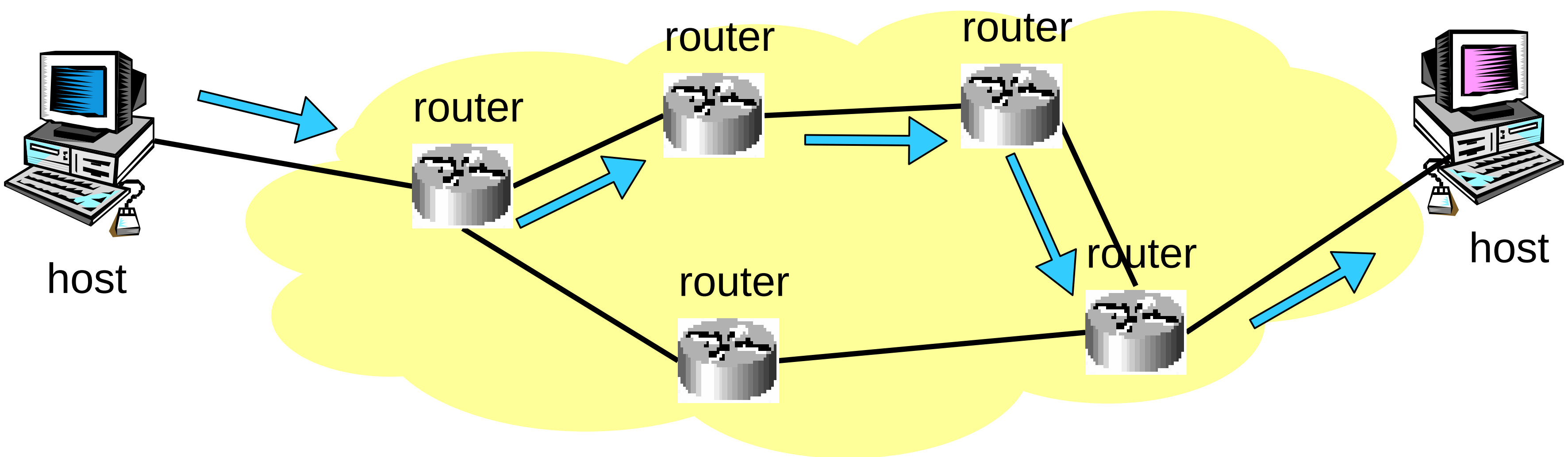
Which router IP address recorded (there are two!)??? RFC791 says outgoing interface!

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)

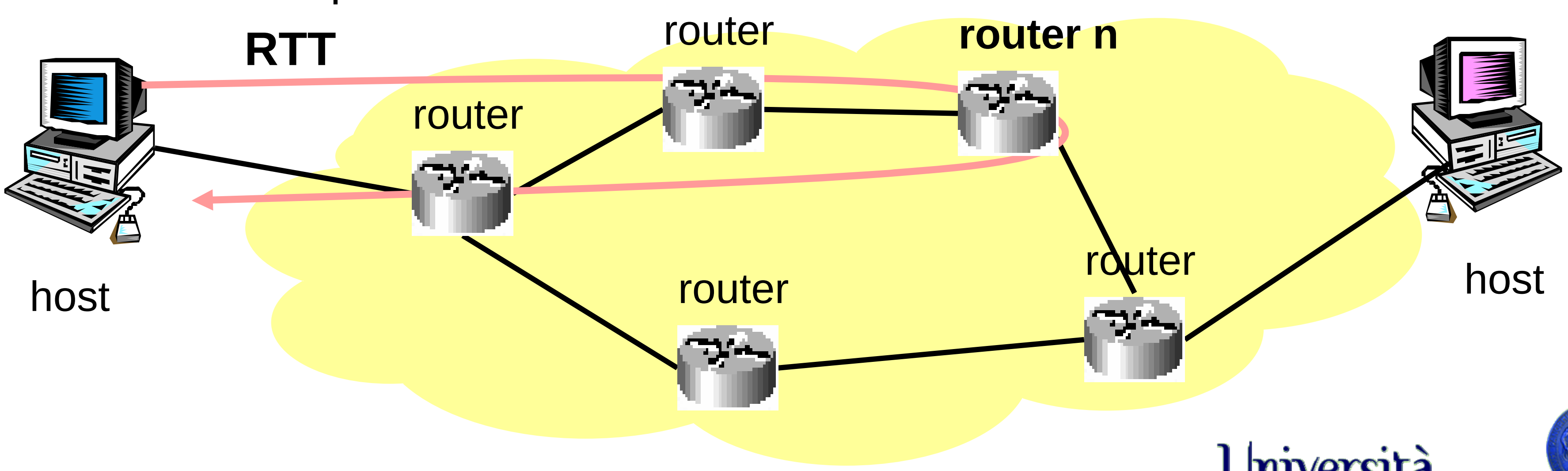
- **Send subsequent sets of 3 UDP packets to destination**
  - Start using TTL=1
  - after each set, increments TTL of 1 unit
  - Listen for the response…



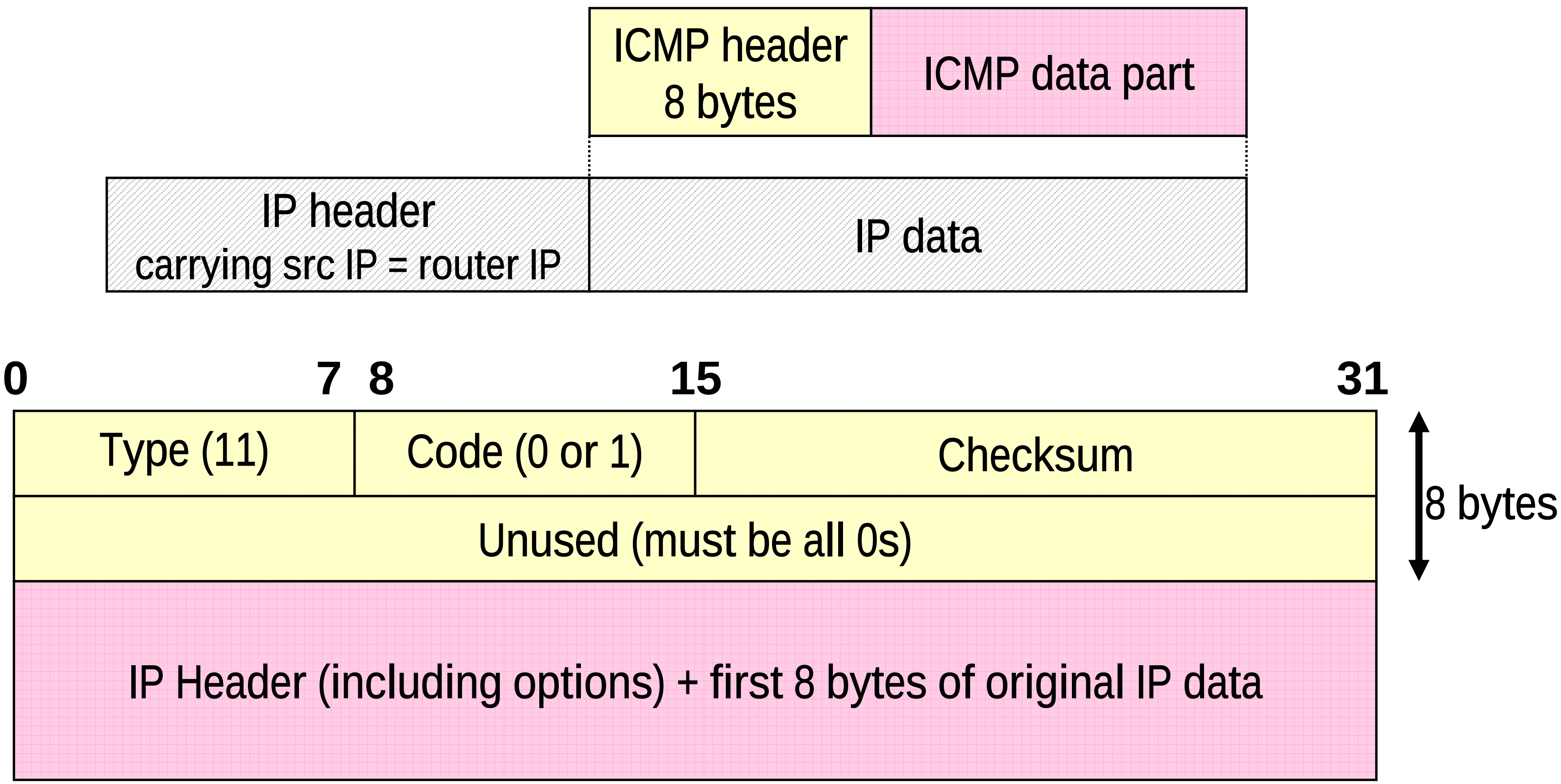
Università degli Studi di Palermo

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

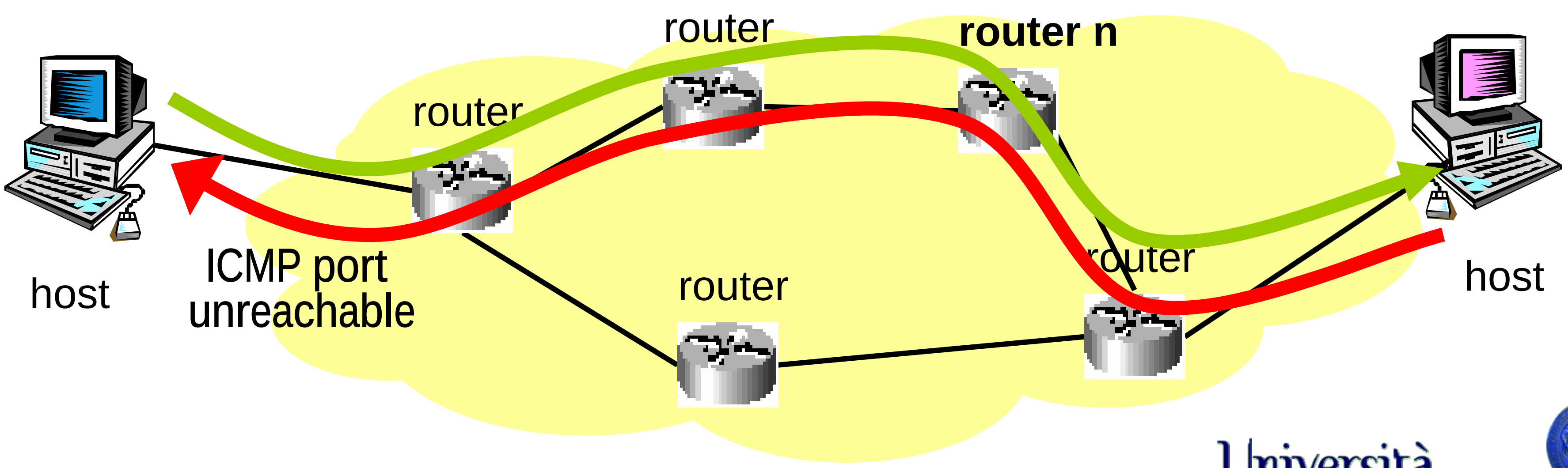


Code = 0: traceroute case (router detected a TTL decrement to 0)

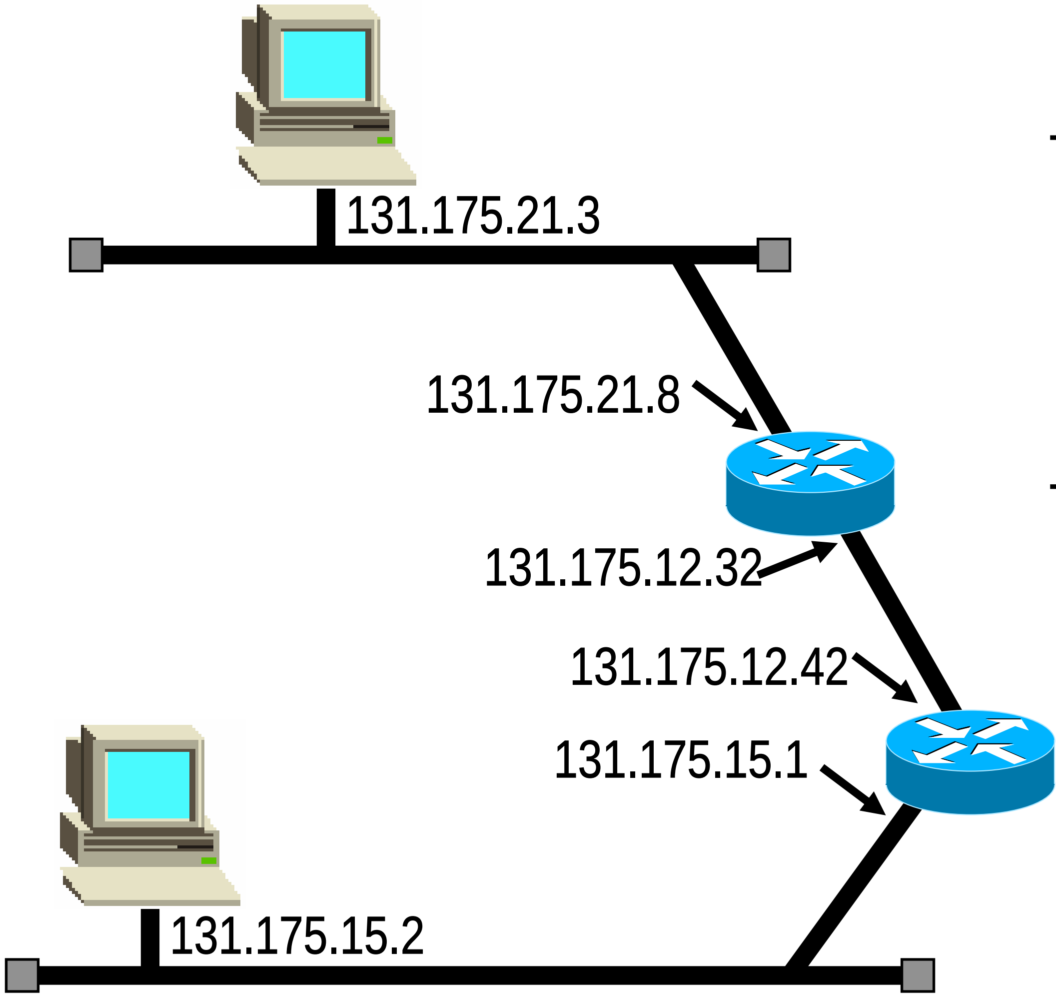
Code = 1: timed out while reassembling

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!



Router: which IP returned?



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-uniromaI.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:

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

- 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-uniromal.rm.garr.net [193.206.131.49]
4	40 ms	40 ms	61 ms	uniromall2-rc.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:\>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	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:

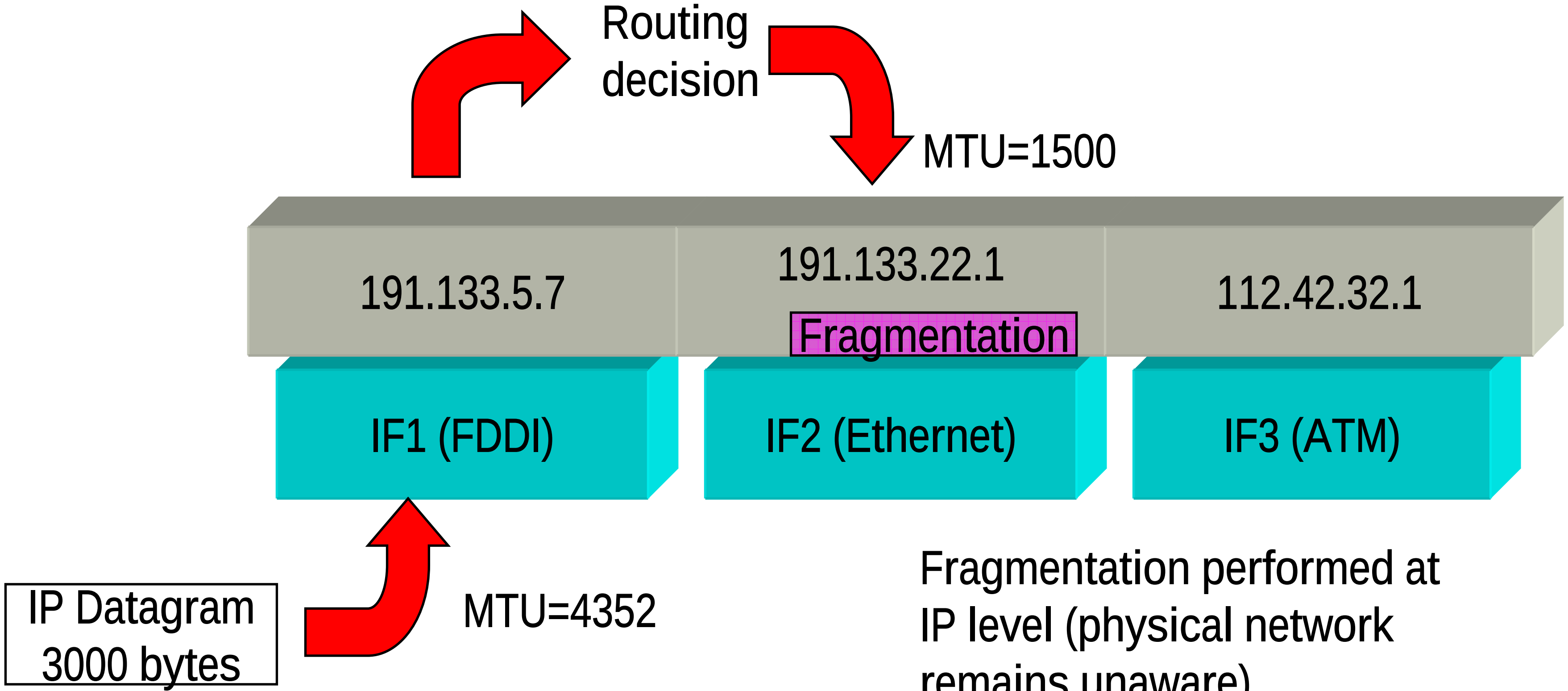
1	<10 ms	<10 ms	<10 ms	151.100.37.1
2	30 ms	40 ms	40 ms	151.100.238.1
3	30 ms	40 ms	40 ms	rc-uniromaI.rm.garr.net [193.206.131.49]
4	70 ms	40 ms	110 ms	rt-rc-1.rm.garr.net [193.206.134.161]
5	231 ms	40 ms	70 ms	na-rm-2.garr.net [193.206.134.45]
6	150 ms	351 ms	160 ms	garr.ny4.ny.dante.net [212.1.200.145]
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)



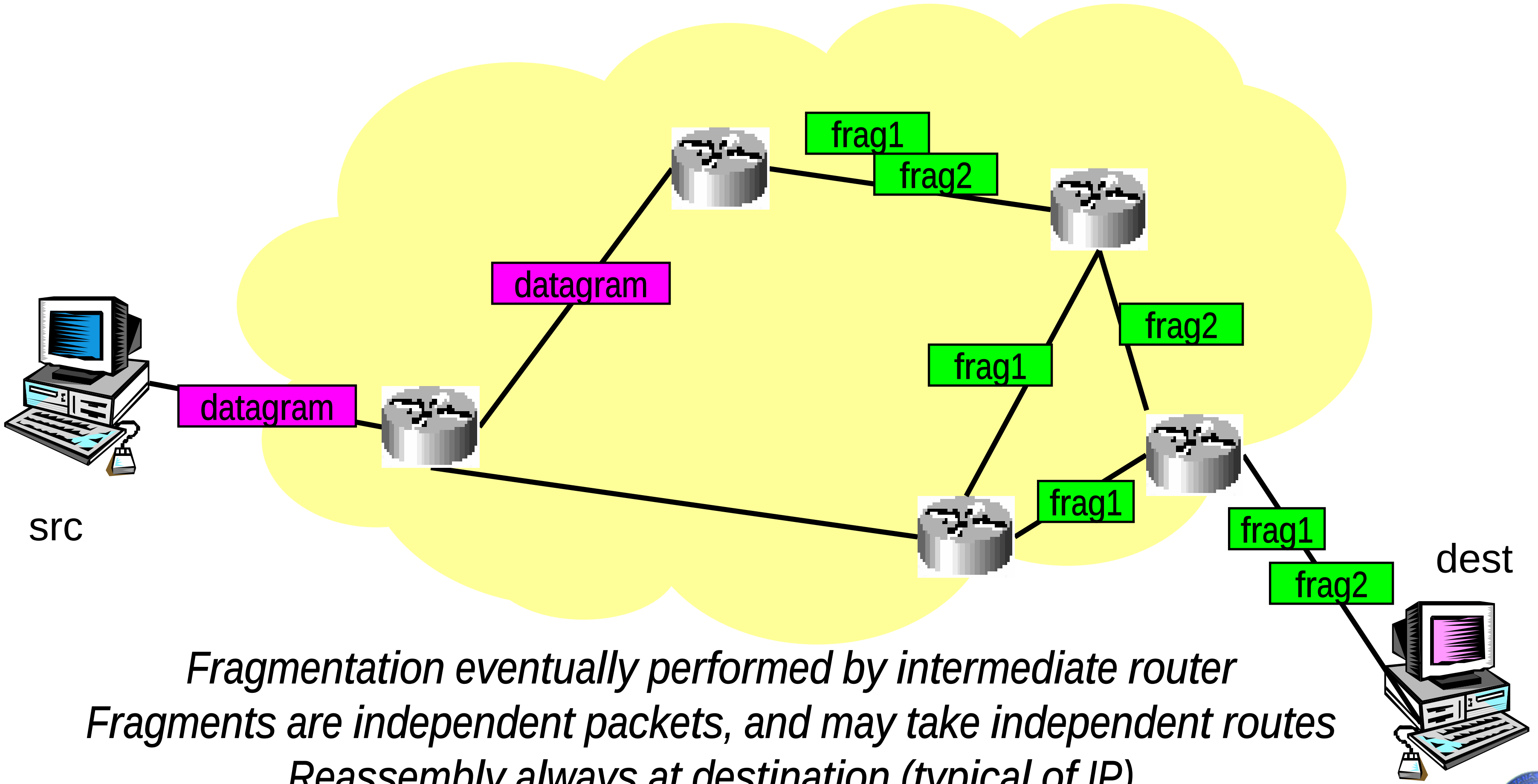
Fragmentation performed at IP level (physical network remains unaware)

Università degli Studi di Palermo

MTU examples
RFC 1191

Network	MTU (bytes)
IBM token ring 16Mbps	17914
4 Mbps Token Ring (IEEE 802.5)	4464
FDDI	4352
Ethernet	1500
IEEE 802.2 802.3	1492
X.25	576
point to point (PPP, SLIP)	May be set to 296 for interactive use

Fragmentation & reassembly



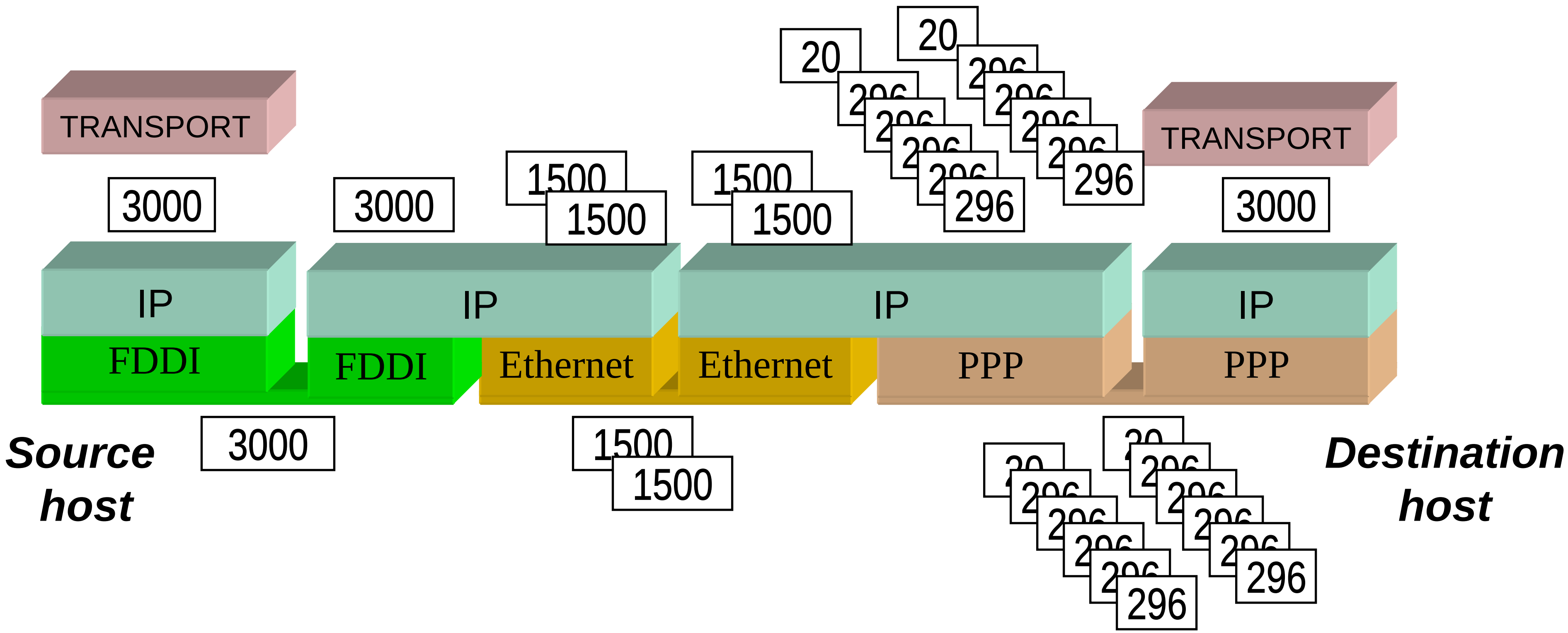
Fragmentation eventually performed by intermediate router
Fragments are independent packets, and may take independent routes
Reassembly always at destination (typical of IP)

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!



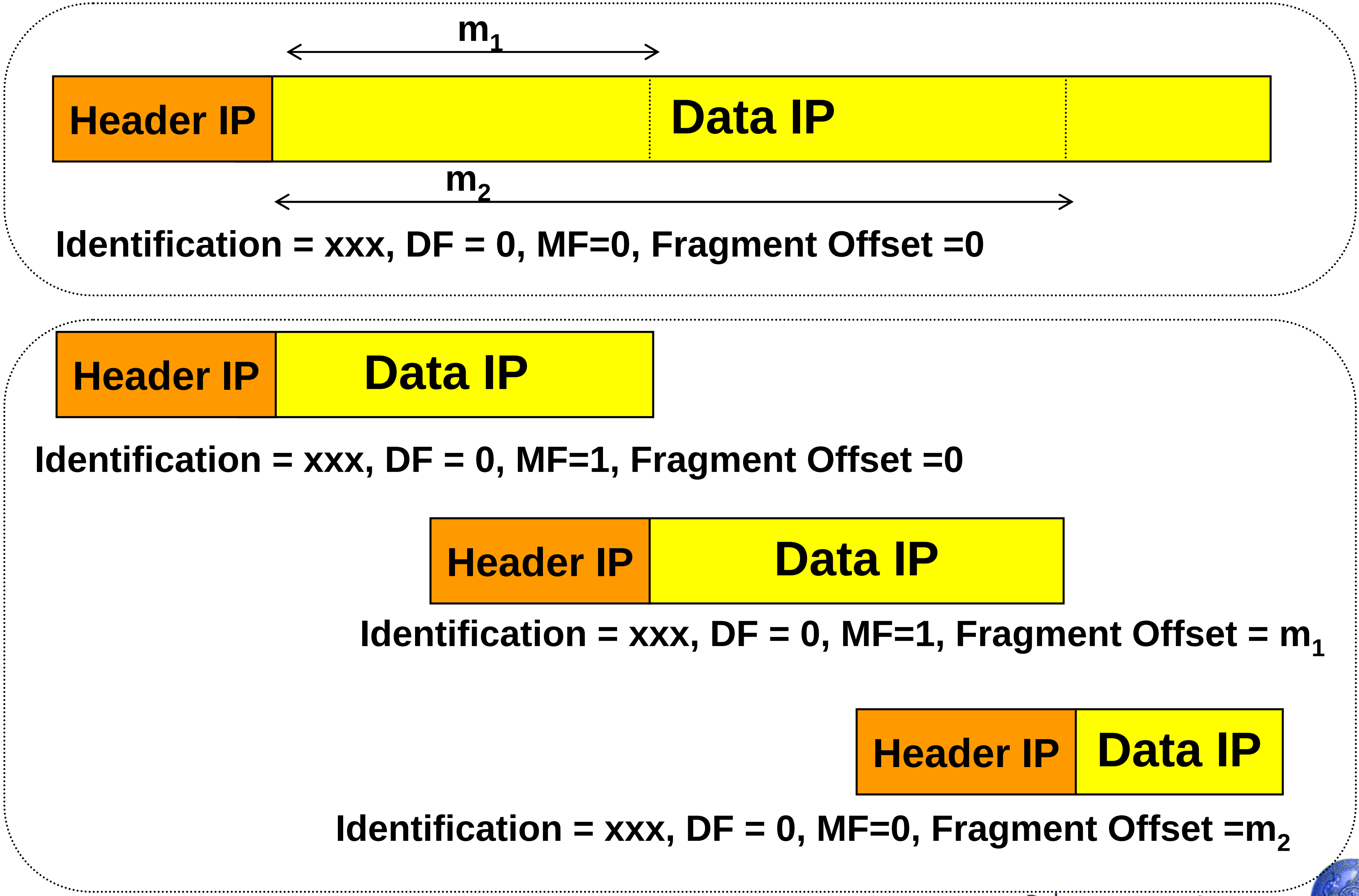
G.Bianchi, G.Neglia, V.Mancuso

IP header - fragmentation fields

<table>
<tr><td>Version</td><td>Header length</td><td>Type of Service TOS</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">16 bit identification</td><td>flags 3 bit</td><td>13 bit fragment offset</td></tr>
<tr><td colspan="2">Time to Live TTL</td><td>Protocol</td><td colspan="2">Header checksum</td></tr>
<tr><td colspan="5">32 bit source IP address</td></tr>
<tr><td colspan="5">32 bit destination IP address</td></tr>
</table>

- ➔ **Identification:**
  - ⇨ unique value per datagram: allows to understand to which datagram fragments belong
- ➔ **flags: 3 bits: [X,DF,MF]**
  - ⇨ X=unused (0), DF=Dont_Fragment , MF=More_Fragments
- ➔ **fragment offset = payload fragment position in the original datagram payload**

fragmentation



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!**
  - ⇨ Restransmission is task of higher layers: IP does not deal with it

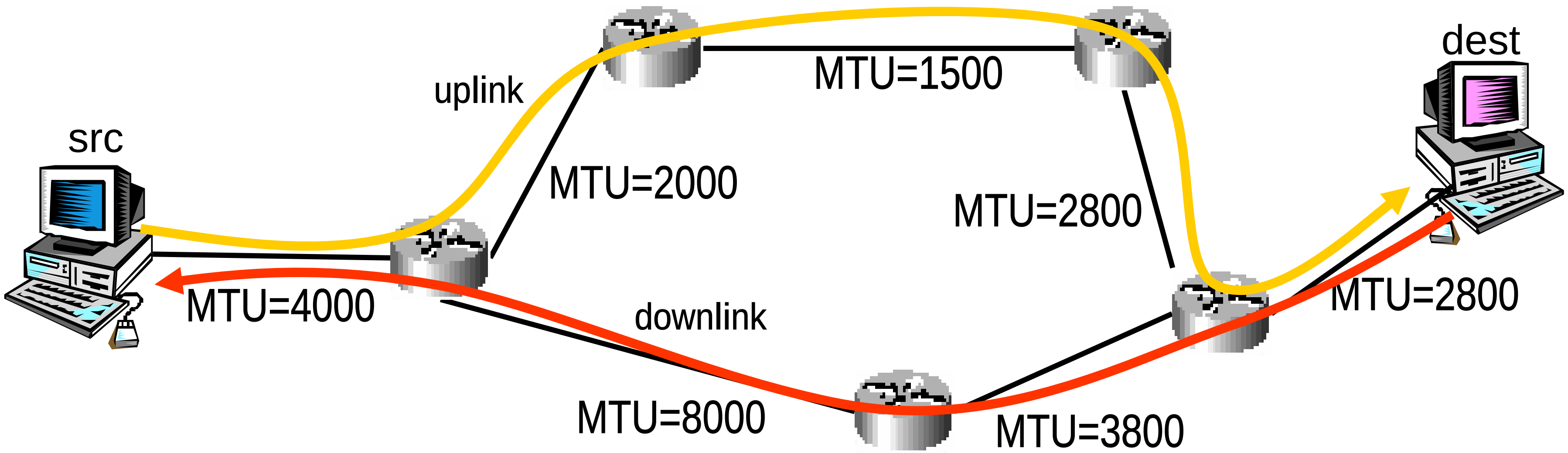
Fragmentation is seen as a problem and overhead, and avoided whenever possible

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>
<tr><th>0   7</th><th>8   15</th><th>31</th><th></th></tr>
<tr><td>Type (3)</td><td>Code (4)</td><td>Checksum</td><td rowspan="2">8 bytes</td></tr>
<tr><td colspan="2">Unused (all 0s)</td><td>MTU of next hop network (bytes)</td></tr>
<tr><td colspan="3">IP Header (including options) + first 8 bytes of original IP data</td><td></td></tr>
</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)

G.Bianchi, G.Neglia, V.Mancuso

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

IP header	ICMP message

0 — 7	8 — 15	31
Type	Code	Checksum
Further header specification – depends on type & code		
Content – depends on Type and Code		

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

TYPE	DESCRIPTION	QUERY or ERROR	CODES
0	Echo reply	QUERY	0
3	Destination Unreachable	ERROR	Many: 0-15
4	Source Quench	ERROR	0
5	Redirect	ERROR	Many: 0-3
8	Echo Request	QUERY	0
9	Router Advertisement	QUERY	0
10	Router Solicitation	QUERY	0
11	Time Exceeded	ERROR	0=transit, 1=reassembly
12	Parameter problem	ERROR	0=bad header, 1=missing opt
13	Timestamp request	QUERY	0
14	Timestamp reply	QUERY	0
15	*Information Request (obsolete)*	*QUERY*	*0*
16	*Information Reply (obsolete)*	*QUERY*	*0*
17	Address Mask Request	QUERY	0
18	Address Mask Reply	QUERY	0

ICMP Error Message format

always contain infos about IP packet that generated error

0 — 7	8 — 15	31	
Type	Code	Checksum	8 bytes
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

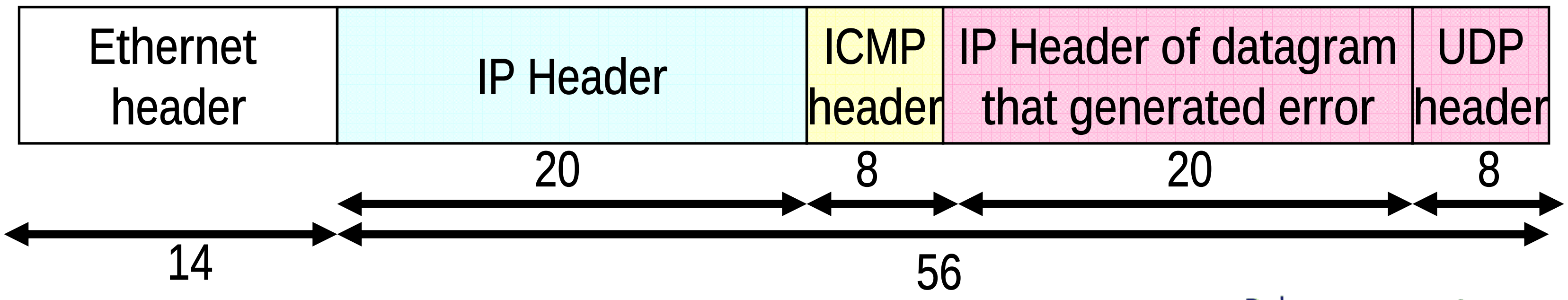
Destination Unreachable (type 3) codes

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

Destination Unreachable header

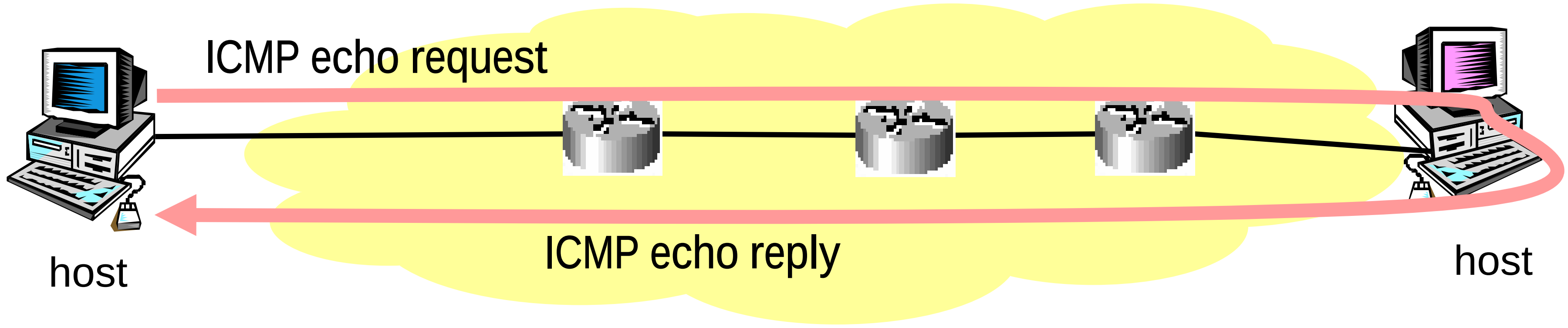
0 … 7	8 … 15	16 … 31	
Type (11)	Code (0-15)	Checksum	
00000000	00000000	00000000	00000000
IP Header + first 8 bytes IP data (UDP header)			

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)
- ➔Meaningwhile, 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

0 – 7	8 – 15	16 – 31
Type (0 or 8)	Code (0)	Checksum
Identifier (16 bits)		Sequence number
Optional data		

- ➔ 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

Sintassi: ping [-t] [-a] [-n numero] [-l lunghezza] [-f] [-i TTL] [-v TOS]
[-r numero] [-s numero] [[-j host-list] | [-k host-list]]
[-w timeout] elenco-destinatione

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 Restringe 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

ICMP Source Quench error

0 7 8 15 31

Type (4)	Code (0)	Checksum	
00000000	00000000	00000000	00000000
IP Header + first 8 bytes IP data (UDP header)			

- ➔ **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

0 … 7	8 … 15	31
Type (17 or 18)	Code (0)	Checksum
Identifier (16 bits)		Sequence number
32 bit subnet mask		

12 bytes

Timestamp request/reply

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

0 … 7	8 … 15	31
Type (17 or 18)	Code (0)	Checksum
Identifier (16 bits)		Sequence number
Originate timestamp *(filled by requestor)*		
Receive timestamp *(filled by replying when receives)*		
Transmit timestamp *(filled by replying when transmitting)*		

20 bytes