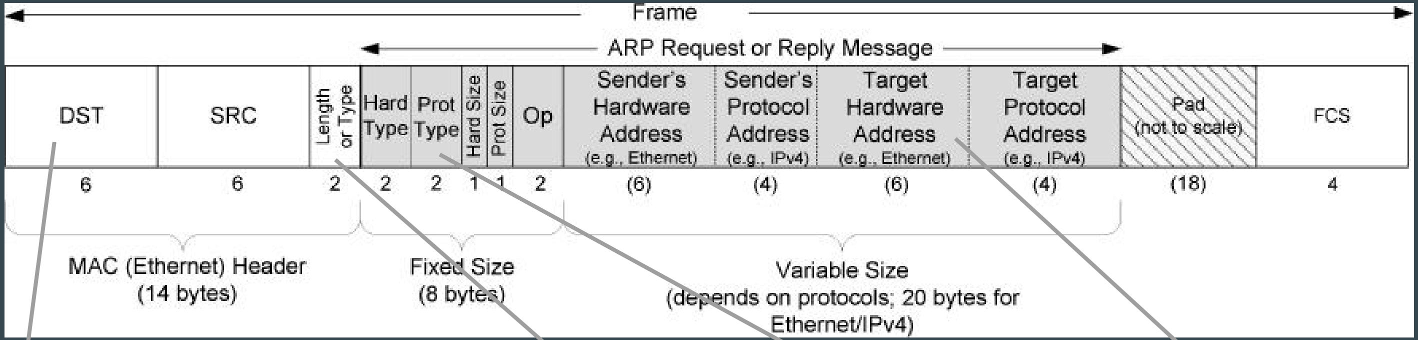
ARP Frame encapsulated in Ethernet Frame



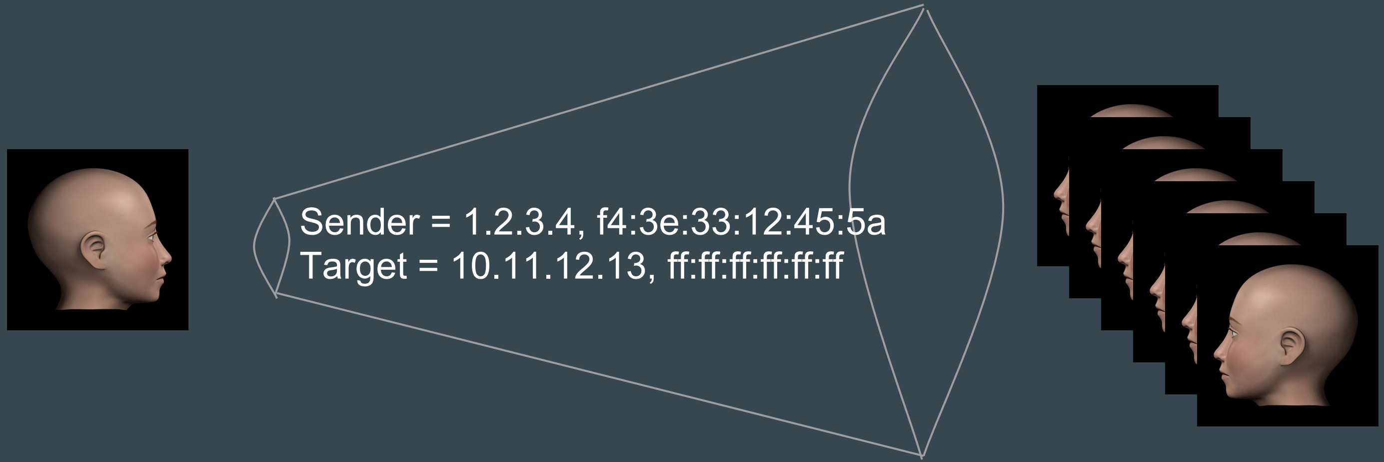
ARP = 0x0806

IP = 0x0800

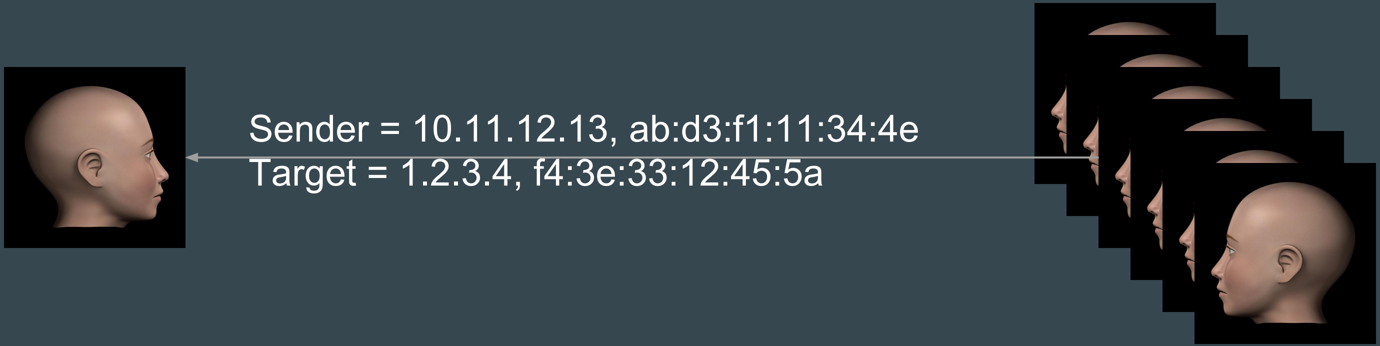
0's in request

DST = link-layer broadcast address in a request. Unicast in a response.

ARP



ARP



Sender address in Ethernet header does not have to be the same as the “sender’s HW address” in ARP response.

Possible way to detect an eavesdropper

Destination IP address = eavesdropper
Destination Ethernet address ≠ eavesdropper

IP forwarding

Destination	Mask	Next hop	Interface
0.0.0.0	0.0.0.0	10.0.0.1	10.0.0.100
10.0.0.0	255.255.255.128	10.0.0.100	10.0.0.100

IP forwarding - algorithm

Let D be destination IP from packet header.
Let d_j be destination in j^{th} entry of forwarding table.
Let m_j be mask in j^{th} entry of forwarding table.

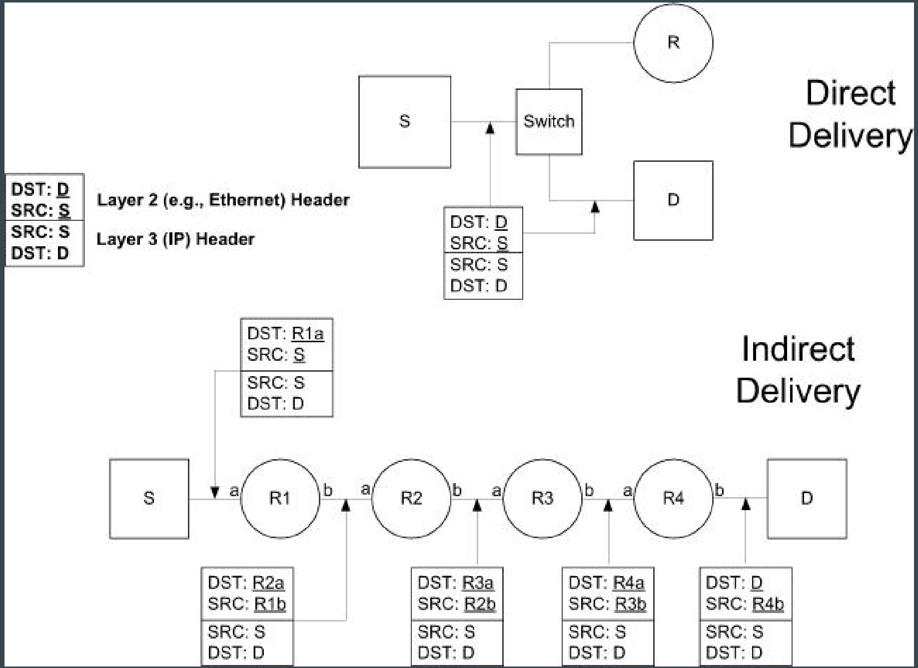
- Find all entries with: $(D \,\&\, m_j) = d_j$
- From amongst those, pick entry with most 1's in m_j
- > 1 best match $\Rightarrow$ use some tie-breaking rule
- 0 matches $\Rightarrow$ "host unreachable."

IP forwarding - examples

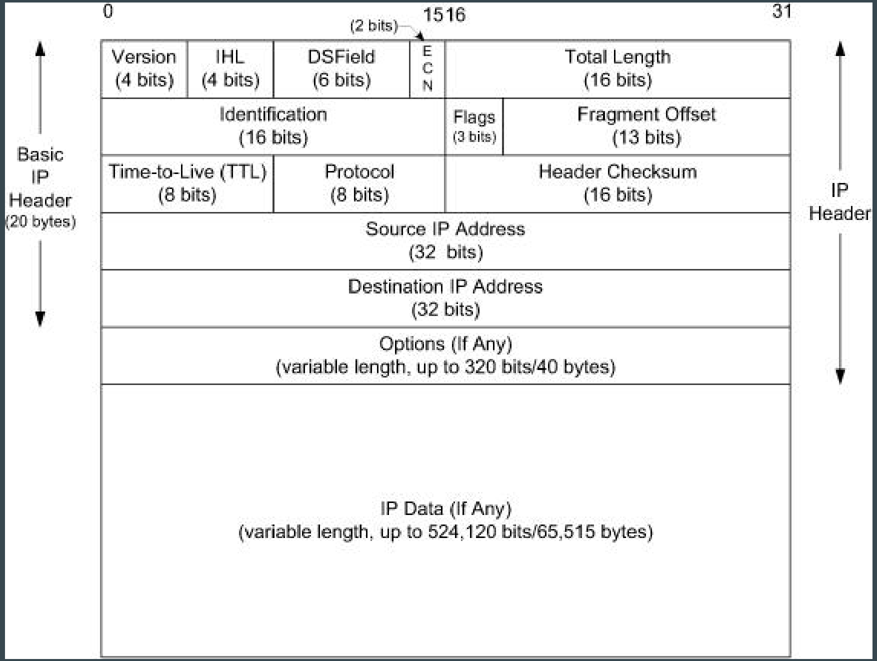
Destination	Mask	Next hop	Interface
0.0.0.0	0.0.0.0	10.0.0.1	10.0.0.100
10.0.0.0	255.255.255.128	10.0.0.100	10.0.0.100

- 10.0.0.19 matches both entries. 2nd entry is best-match.
- 178.162.3.4 matches 1st entry, not 2nd
- 10.0.0.131 matches 1st entry, not 2nd

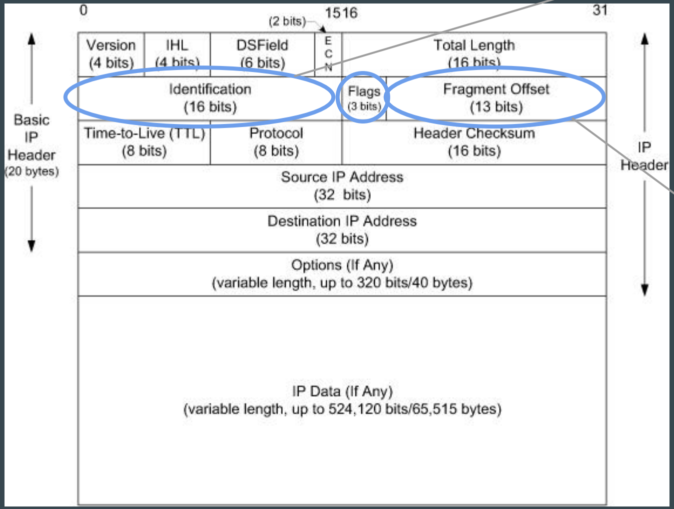
IP forwarding - “direct” vs. “indirect” delivery



IP packet format



Fragmentation - the IP Header



All fragments of a datagram have the Same ID.

R	DF	MF

Reserved

Do not Fragment

More Fragments

Offset from start of data.
8-byte units.

Fragmentation - how it works via example

ID = i
DF = 0

IP Header 20 bytes	IP Data 3980 bytes

IP Header 20 bytes	IP Data 1480 bytes

ID = i
MF = 1
Offset = 0

IP Header 20 bytes	IP Data 680 bytes

ID = i
MF = 1
Offset = 185

IP Header 20 bytes	IP Data 1820 bytes

ID = i
MF = 0
Offset = 270

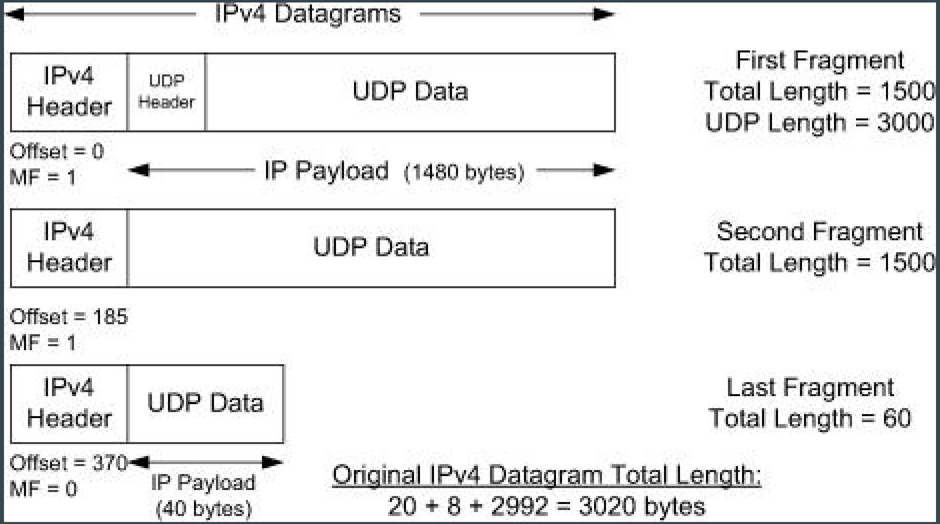
Fragmentation - issues (1)

- Complexity of algorithms ⇒ lots of bugs

  See RFC 791

Fragmentation - issues (2)

- Higher layer protocol header is separated from data.



Fragmentation - issues (3)

- Identification field may not be long enough at 16 bits.
    - ID does not repeat for 120 seconds @ 1500 byte packets $\Rightarrow$ 6.5 Mbps throughput
    - Throughput 1 Gbps @ 1500 byte packets $\Rightarrow$ ID space exhausted in < 1 second
    - Proposed solution RFC 6864: receiver relies on ID field for fragments only

Header Checksum

- “1’s complement of 1’s complement 16-bit sum.”

Checksum ← 00 00
Perceive header as chunks of 16 bits = 2 bytes
Checksum ← Checksum + (each of those 2 bytes)
Add carry back into Checksum
Checksum ← bit-complement of Checksum

Header Checksum, example

- E3 4F 23 96 44 27 99 F3

00 00	E3 4F	1 06 E5	1 4B 0C	E4 FF
+ E3 4F	+ 23 96	+ 44 27	+ 99 F3	+ 00 01
= E3 4F	= 1 06 E5	= 1 4B 0C	= 1 E4 FF	= E5 00

Checksum = 1A FF

Checking the checksum

- Compute checksum of entire header (including checksum)
- Result should be 00 00
- E.g.,
  - (E3 4F) + (23 96) + (44 27) + (99 F3) + (1A FF) = 1 FF FE
  - 1's complement: (FF FE) + (00 01) = FF FF
  - Complement = 00 00

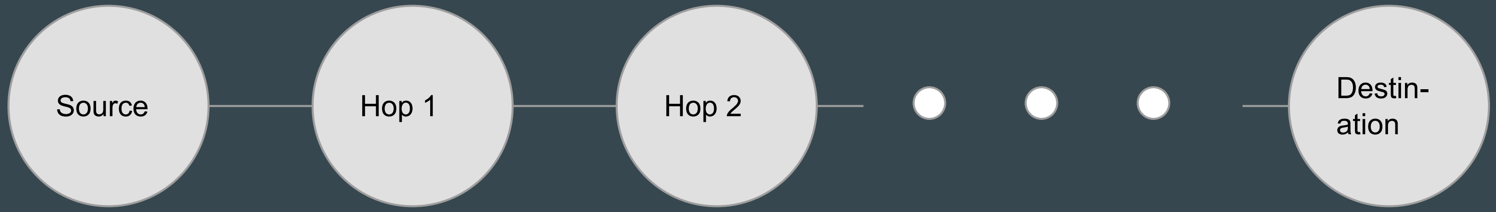
TTL - Time To Live

- Use to limit datagram lifetime in the network
- Decremented by 1 @ each hop
- $\Rightarrow$ header checksum must be recomputed @ every hop

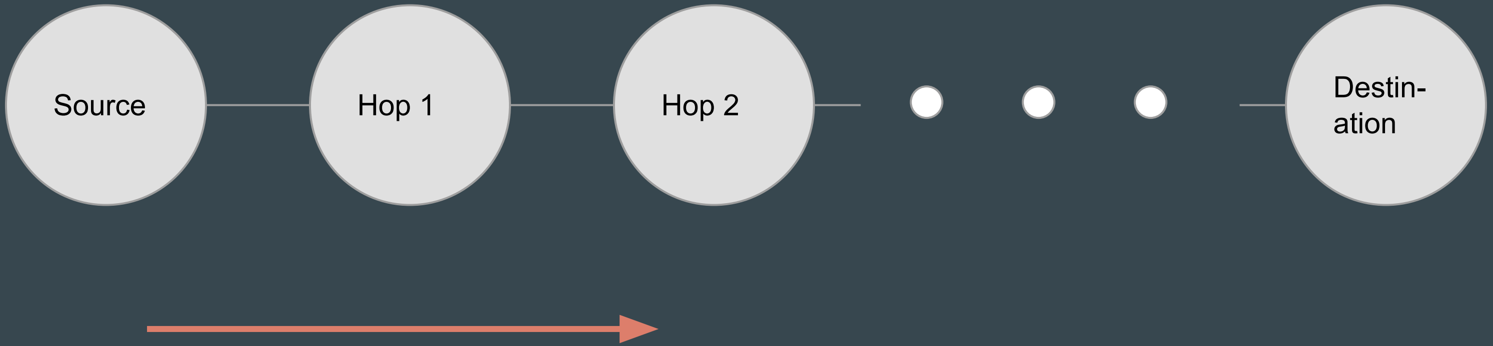
Traceroute - a nifty use of TTL

- Objective - want to find path from here (source) to destination
- Method - leverage TTL field

Traceroute - how it works, via example



Traceroute - how it works, via example, Step 1



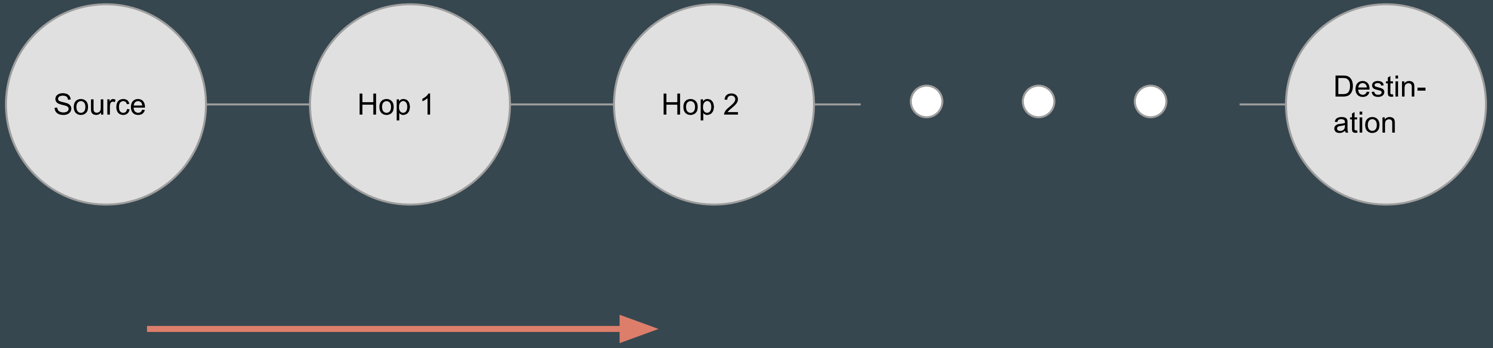
Source IP = Source
Destination IP = Destination
TTL = 1

Traceroute - how it works, via example, Step 1

Source IP = Hop 1

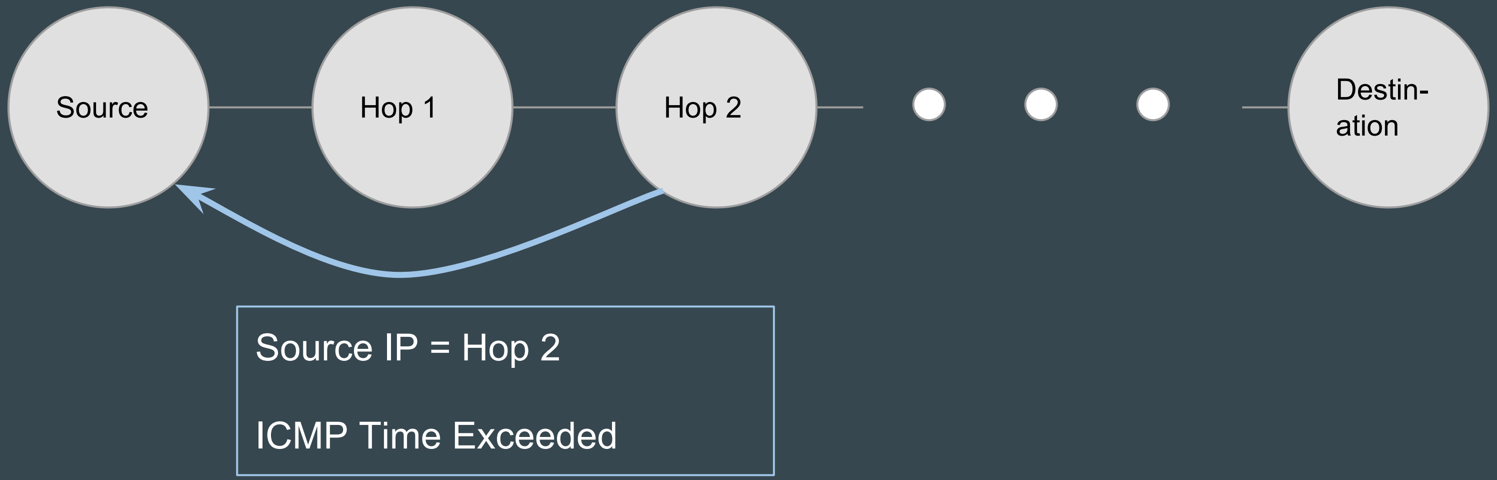
ICMP Time Exceeded
IP header + 8 bytes of data
from original datagram

Traceroute - how it works, via example, Step 1



Source IP = Source
Destination IP = Destination
TTL = 2

Traceroute - how it works, via example, Step 1



Question

It so happens that an IP packet sent by S to D incurs no errors in transit.

- True or false:

  The value of the header checksum that D sees is the value that S put in the checksum field.