Module 6
Internet Layer

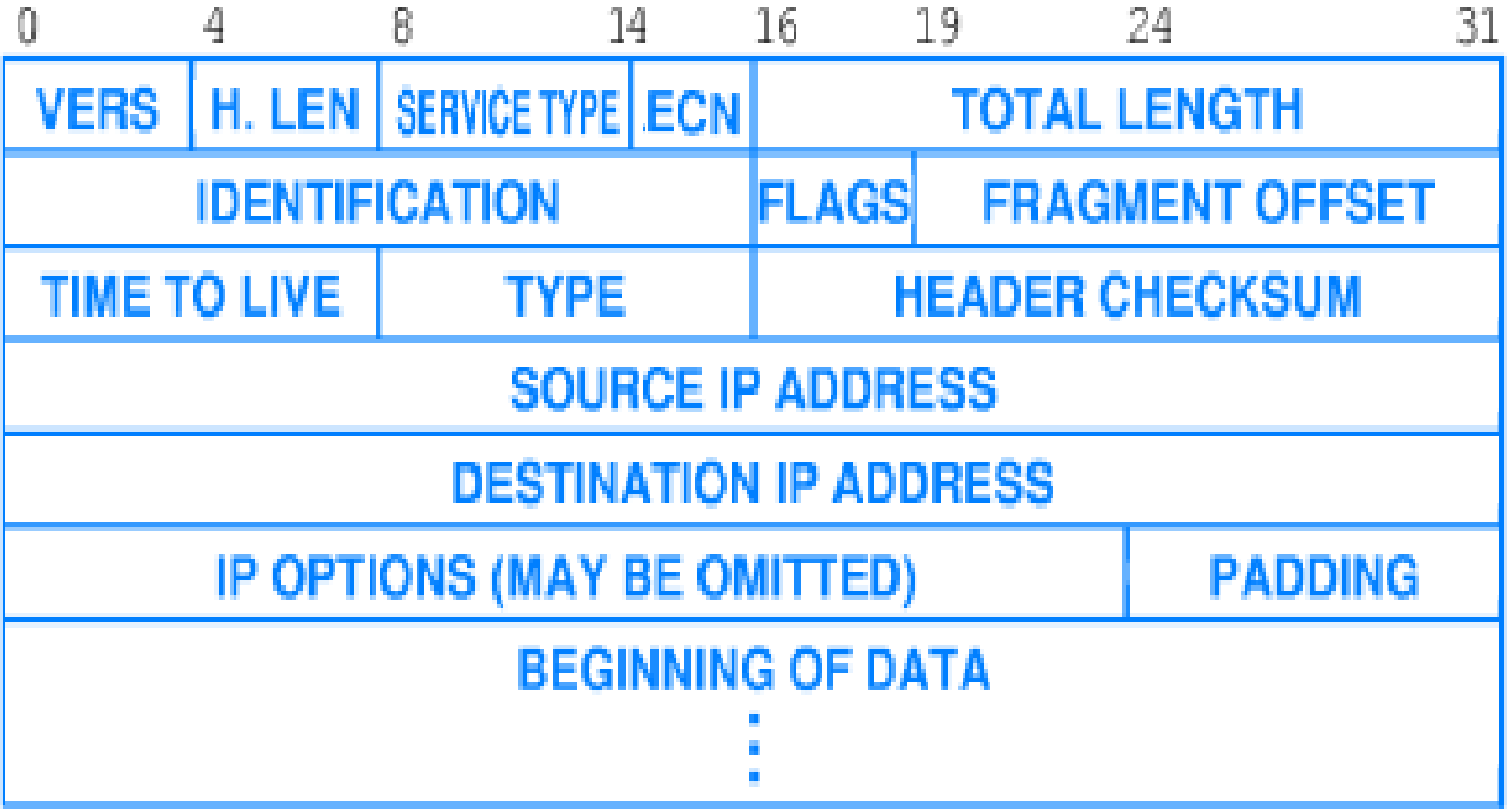
Dr. Natarajan Meghanathan
Professor of Computer Science
Jackson State University, Jackson, MS 39232
E-mail: natarajan.meghanathan@jsums.edu

Module Topics

- 6.1 IP Header
- 6.2 IP Datagram Fragmentation
- 6.3 IP Datagram Forwarding
- 6.4 IP Auxiliary Protocols and Technologies
  - ARP, DHCP, ICMP
- 6.5 IPv6

6.1 IP Header

IP Header Format (v4)



IP Header Format

- **VERS** – Each datagram begins with a 4-bit protocol version number

- **H.LEN** – 4-bit header length field specifies the number of 32-bit quantities in the header. The header size should be a multiple of 32.
- The minimum integer value for the IP header length is 5 as the top 5 32-bit rows of the IP header are mandatory for inclusion. Hence, the minimum size of the IP header is 5 * 32-bits = 5 * 4 bytes = 20 bytes.
- The maximum integer value for the IP header length (with 4 bits for the field) would be 15, necessitating the Options field to be at most 10 32-bit rows. The maximum size of the IP header is 15 * 32-bits = 15 * 4 bytes = 60 bytes.

- **SERVICE TYPE** – used to specify whether the sender prefers the datagram to be sent over a path that has minimum delay or maximum throughput, etc. A router that has multiple paths to the destination, can apply these service type preferences to select the most suitable path.

- **TOTAL LENGTH** – used to specify the total number of bytes of the data plus the header.

IP Header Format

- ECN bits (2 bits) for Explicit Congestion Notification
  - 2 bit-combinations
    - 0 0 (Non-ECT – EC not supported at transport layer)
    - 0 1 or 1 0 (ECT–EC supported at the transport layer)
    - 1 1 (CE: Congestion Experienced)
  - If the end hosts can support ECN, the source sets either 0 1 or 1 0 in the IP header of the datagrams sent.
  - A router experiencing congestion, (instead of dropping the packet right away) will overwrite the ECT bits with the CE bits, letting the destination know that the datagram was forwarded in spite of the impending congestion.
  - The destination has to now echo this EC notification in the ACK packet sent to the source (through the ECE flag in the TCP header)

IP Header Format

- IDENTIFICATION
  - 16-bit number (usually sequential) assigned to the datagram
    - used to gather all fragments for reassembly to the datagram
- FLAGS
  - 3-bit field with individual bits specifying whether the datagram is a fragment
    - If so, then whether the fragment corresponds to the rightmost piece of the original datagram
- FRAGMENT OFFSET
  - 13-bit field that specifies where in the original datagram the data in this fragment belongs
  - the value of the field is multiplied by 8 to obtain an offset
- Note: The above three fields are used mainly in Fragmentation and Reassembly.

IP Header Format

- TIME TO LIVE
  - 8-bit integer initialized by the original sender
  - it is decremented by each router that processes the datagram
  - if the value reaches zero (0)
    - the datagram is discarded and an error message is sent back to the source
- TYPE
  - 8-bit field that specifies the type of the payload
- HEADER CHECKSUM
  - 16-bit ones-complement checksum of header fields
  - The checksum bits are set to 0 during the calculation.
- SOURCE IP ADDRESS
  - 32-bit Internet address of the original sender
  - (the addresses of intermediate routers do not appear in the header)

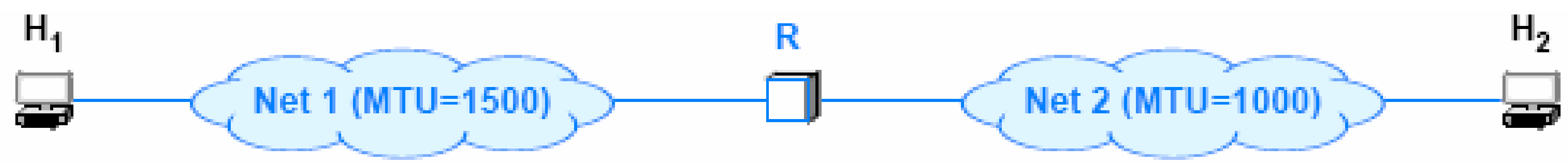
IP Header Format

- DESTINATION IP ADDRESS
  - The 32-bit Internet address of the ultimate destination
  - The addresses of intermediate routers do not appear in the header
- IP OPTIONS
  - Optional header fields used to control routing and datagram processing
  - Most datagrams do not contain any options
    - which means the IP OPTIONS field is omitted from the header
- PADDING
  - If options do not end on a 32-bit boundary
    - zero bits of padding are added to make the header a multiple of 32 bits

6.2 IP Datagram Fragmentation

IP Datagram Fragmentation

- Each hardware technology specifies the maximum amount of data that a frame can carry
  - The limit is known as a Maximum Transmission Unit (MTU) – max. size for the datagram that can be as a payload in the frame
- A router can connect networks with different MTU values
  - a datagram that a router receives over one network can be too large to send over another network

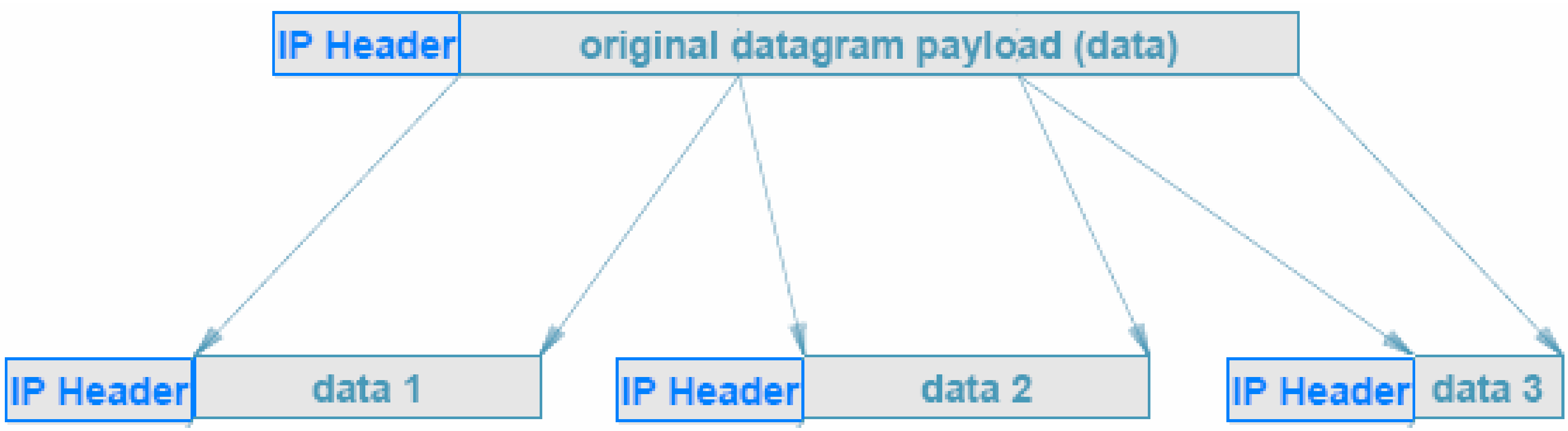


IP Datagram Fragmentation

- To solve the problem of heterogeneous MTUs
  - a router uses a technique known as fragmentation
- When a datagram is larger than the MTU of the network over which it must be sent
  - the router divides the datagram into smaller pieces called fragments
  - and sends each fragment independently
- A fragment has the same format as other datagrams
  - a bit in the FLAGS field of the header indicates whether a datagram is a fragment or a complete datagram
- Other fields in the header are assigned information for the ultimate destination to reassemble fragments
  - to reproduce the original datagram
- The FRAGMENT OFFSET specifies where in the original datagram the fragment belongs

IP Datagram Fragmentation

- A router uses the network MTU and the header size to calculate (i) the maximum amount of data that can be sent in each fragment and (ii) and the number of fragments that will be needed
- The router then creates the fragments
  - It uses fields from the original header to create a fragment header
  - It copies the appropriate data from the original datagram into the fragment and transmits the fragments



Fragment FLAGS and Offset Fields

- There are three bits (FLAGS).
  - The first bit is always set to 0 (reserved for future use)
  - The second bit (MF bit) is set to 1 if there is at least one more fragment following this fragment. For the last fragment, the MF bit is set to 0.
  - The third bit (DF bit) is set to 1 if the source does not want a router to fragment a datagram if the latter's size exceeds the MTU of the downstream network. In this case, the router simply drops the datagram. If the source is fine with fragmentation, the DF bit is set to 0.

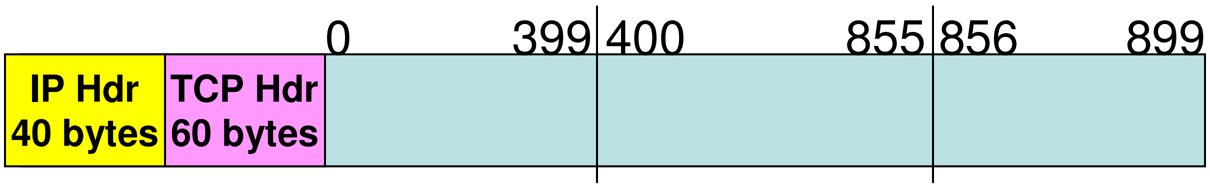
- The IP header has 16 bits for the data length (Total Length) field and only 13 bits for the Fragment Offset field. Hence, the amount of data included in all fragments (except the last fragment) should be divisible by 8.
  - Because of this, we may not be able to stuff the maximum allowable amount of data in a fragment.

- The Fragment Offset field for a fragment should record the integer obtained by dividing the starting byte number of the data in the current fragment by 8.

Sample Problem
Fragment Offset Calculations

- Let an IP datagram of the following data and header sizes be sent through a network of MTU 500 bytes. Calculate the fragment offset values that should be set for the different fragments?
- IP header = 40 bytes TCP header = 60 bytes
- Data = 900 bytes

Solution:



- MTU = 500 bytes
- Fragment 1: IP hdr = 40; TCP hdr = 60;
  Data = 400 [byte #s: 0 - 399]; Offset = 0
- Fragment 2: IP hdr = 40;
  Data = 456 [byte #s: 400 - 855]; Offset = 50
- Fragment 3: IP hdr = 40;
  Data = 44 [byte #s: 856 - 899]; Offset = 107

Datagram Reassembly from Fragments

- Recreating a copy of the original datagram from fragments is called reassembly
- Each fragment begins with a copy of the original datagram header
  – all fragments have the same destination address as the original datagram from which they were derived
- The fragment that carries the final piece of data has an additional bit set in the header
  – Thus, a host performing reassembly can tell whether all fragments have arrived successfully
- The ultimate destination should reassemble fragments

Why Reassembly at End Host?

- Fragments travel independent of each other in the intermediate networks. Sometimes, the fragments of the same datagram might take different paths.

- If an intermediate router has to take care of reassembly, there will be some state information required to maintained for the fragments of the datagram. But, IP is a stateless, connectionless protocol.

Datagram Reassembly from Fragments

- Fragments from multiple datagrams can arrive out-of-order
  - Individual fragments can be lost or arrive out-of-order
- How does it reassemble fragments that arrive out-of-order?
- A sender places a unique identification number in the IDENTIFICATION field of each outgoing datagram
- When a router fragments a datagram
  - the router copies the identification number into each fragment
- A receiver uses the identification number and IP source address in an incoming fragment
  - to determine the datagram to which the fragment belongs
- The FRAGMENT OFFSET field tells a receiver where data in the fragment belongs in the original datagram

Consequence of Fragment Loss

- A datagram cannot be reassembled until all fragments arrive
- A problem arises when one or more fragments from a datagram arrive and other fragments are delayed or lost
- The receiver must save (buffer) the fragments
  - that have arrived in case missing fragments are only delayed
- A receiver cannot hold fragments an arbitrarily long time
  - because fragments occupy space in memory
- IP specifies a maximum time to hold fragments
- When the first fragment arrives from a given datagram
  - the receiver starts a reassembly timer
- If all fragments of a datagram arrive before the timer expires
  - the receiver cancels the timer and reassembles the datagram
- If the timer expires before all fragments arrive
  - the receiver discards the fragments that have arrived

Consequence of Fragment Loss

- The result of IP's reassembly timer is all-or-nothing:
  - either all fragments arrive and IP reassembles the datagram,
  - If not then IP discards the incomplete datagram
- There is no mechanism for a receiver to tell the sender which fragments have arrived
  - The sender does not know about fragmentation
- If a sender retransmits, the datagram routes may be different
  - a retransmission would not necessarily traverse the same routers
    - also, there is no guarantee that a retransmitted datagram would be fragmented in the same way as the original

Fragmenting a Fragment

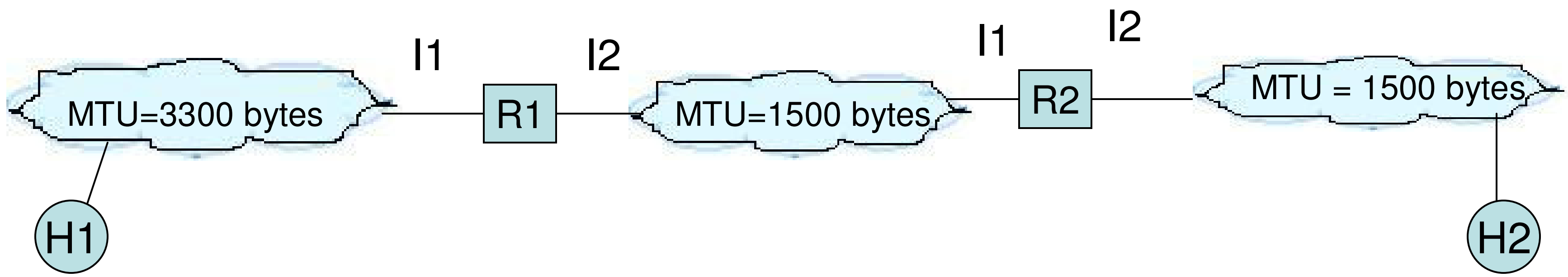
- What happens if a fragment eventually reaches a network that has a smaller MTU?
- It is possible to fragment a fragment when needed
  - A router along the path divides the fragment into smaller fragments
- If networks are arranged in a sequence of decreasing MTUs
  - each router along the path must further fragment each fragment
- Designers work carefully to ensure that such situations do not occur in the Internet

Fragmenting a Fragment

- IP does not distinguish between original fragments and subfragments
- A receiver cannot know if an incoming fragment is the result of one router fragmenting a datagram or multiple routers fragmenting fragments
- Making all fragments the same has an advantage
  - a receiver can perform reassembly of the original datagram
    - without first reassembling subfragments
    - this saves CPU time, and reduces the amount of information needed in the header of each fragment

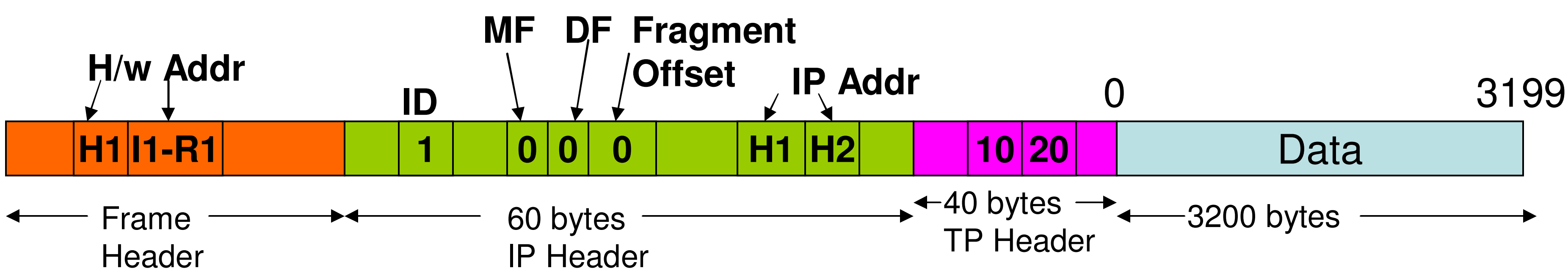
Examples for IP Datagram Fragmentation and Reassembly

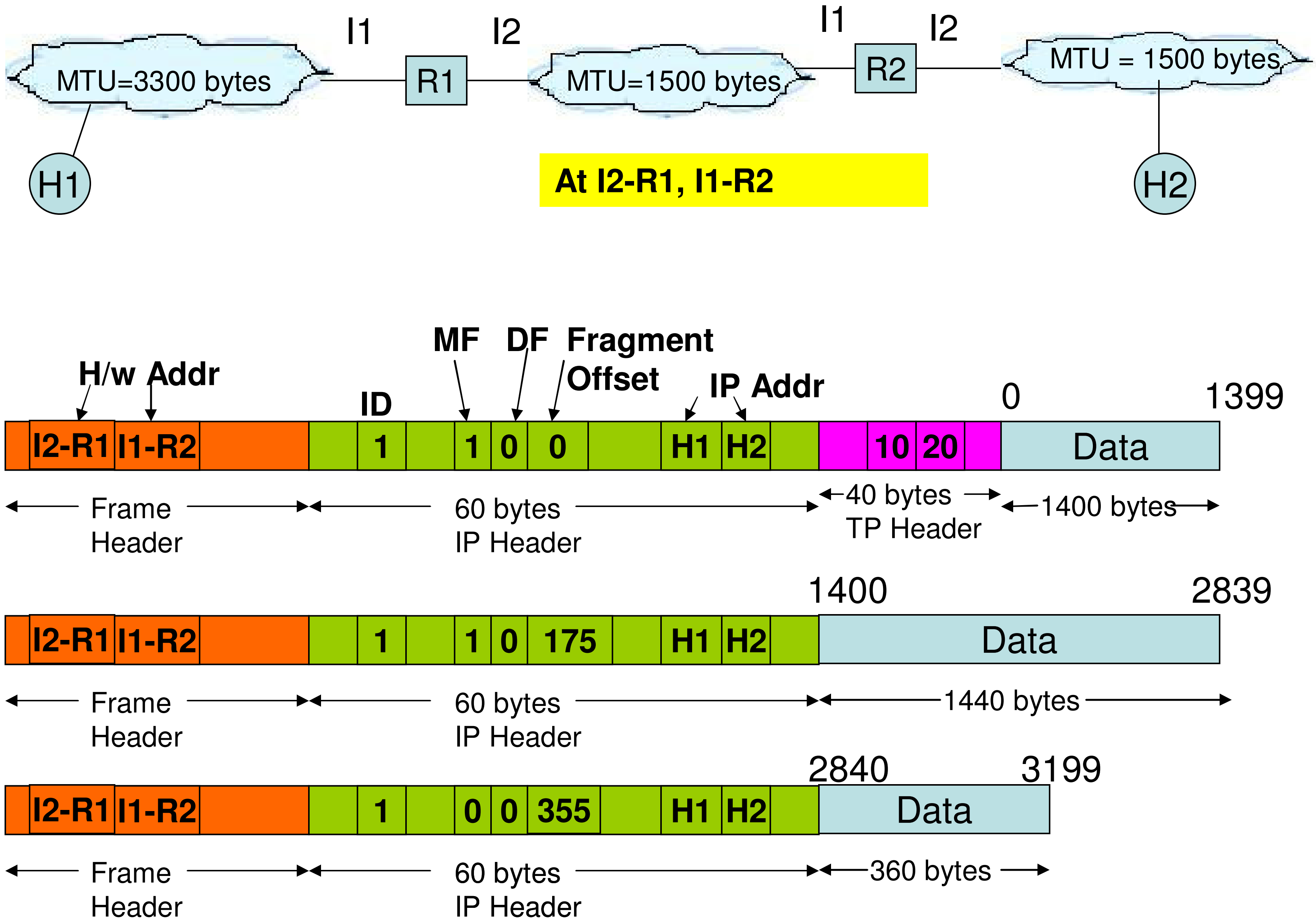
Dr. Natarajan Meghanathan

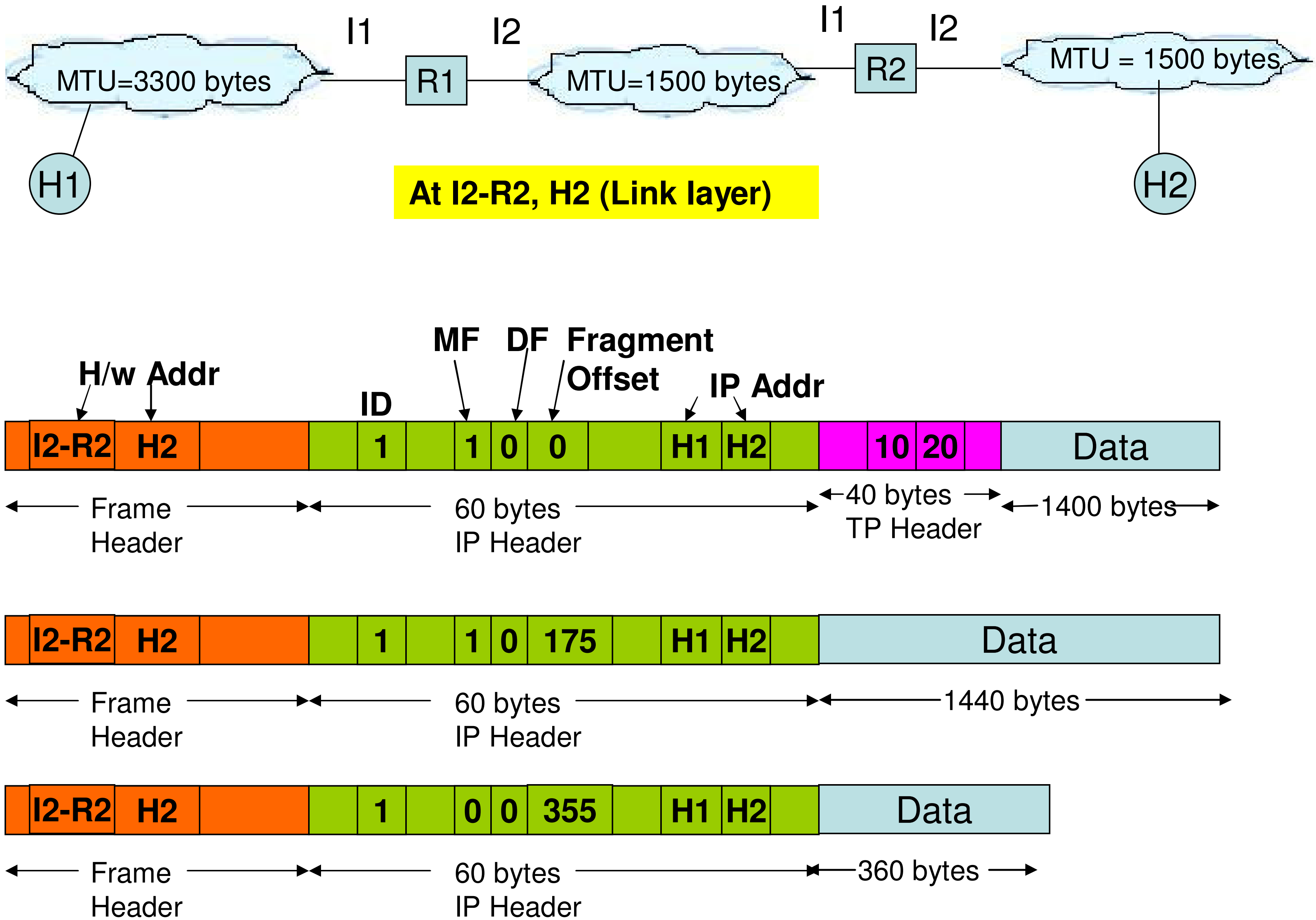


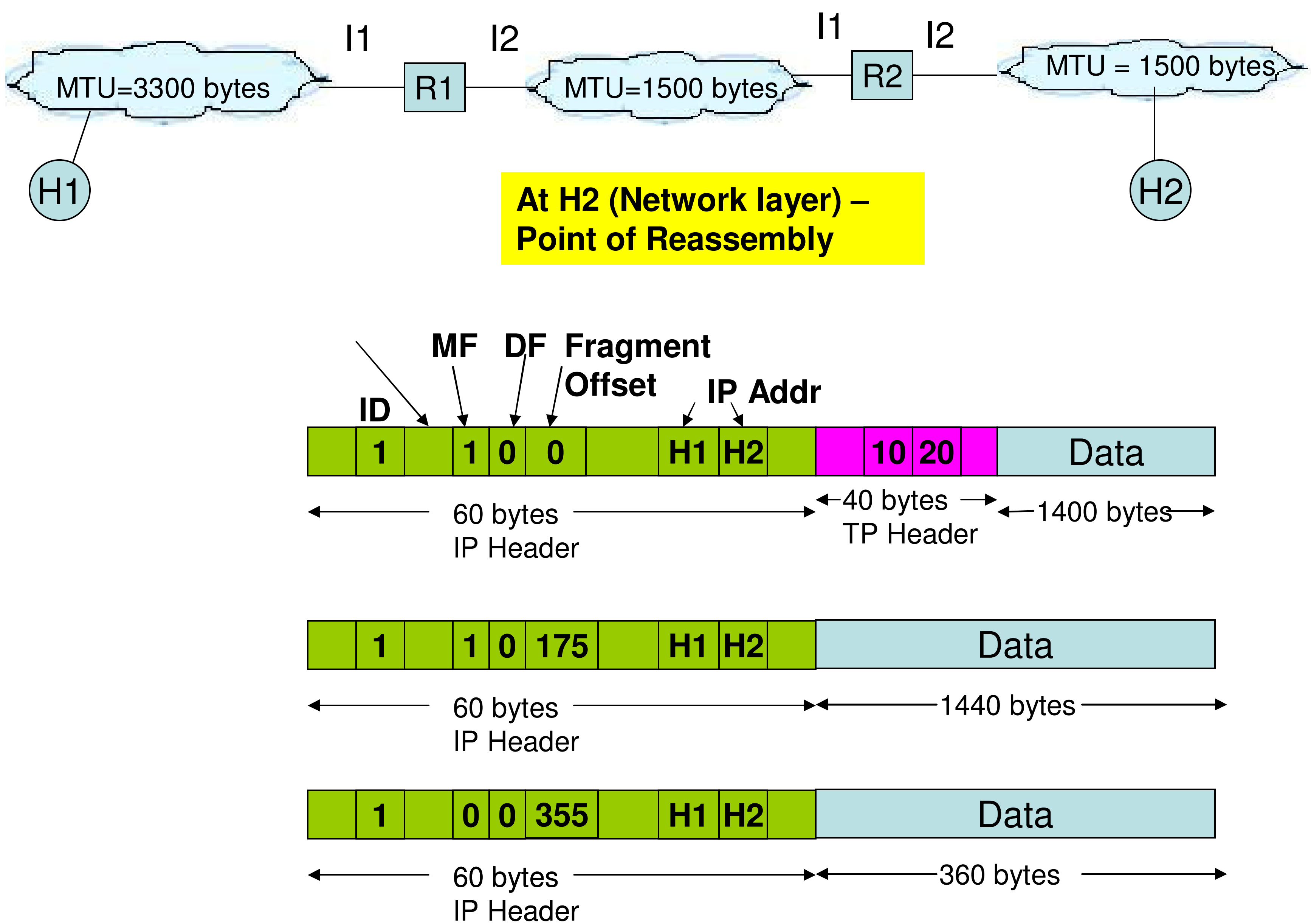
Data Size = 3200 bytes; IP header size = 60 bytes, TP header size = 40 bytes

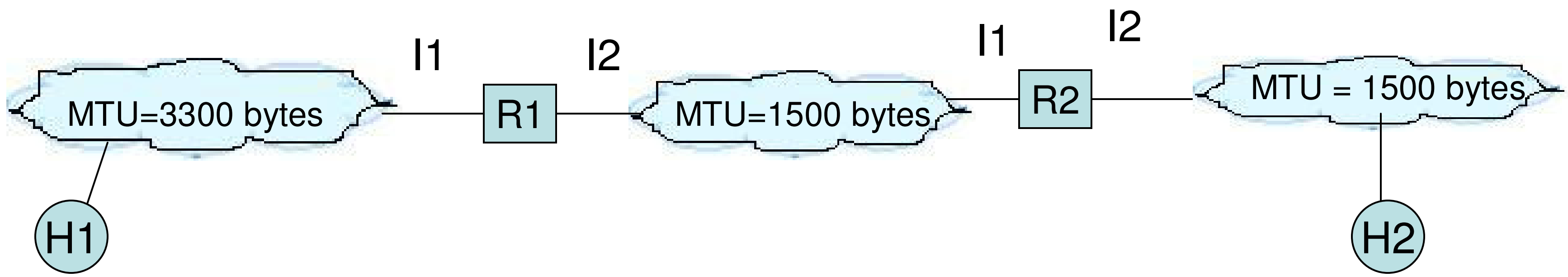
At H1, I1-R1



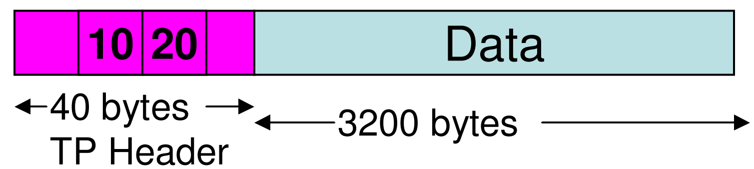

I1
I2
I1
I2
MTU=3300 bytes
R1
MTU=1500 bytes
R2
MTU = 1500 bytes
H1
At I2-R1, I1-R2
H2
H/w Addr
ID
MF
DF
Fragment Offset
IP Addr
0
1399
I2-R1 I1-R2 1 1 0 0 H1 H2 10 20 Data
Frame Header
60 bytes IP Header
40 bytes TP Header
1400 bytes
1400
2839
I2-R1 I1-R2 1 1 0 175 H1 H2 Data
Frame Header
60 bytes IP Header
1440 bytes
2840
3199
I2-R1 I1-R2 1 0 0 355 H1 H2 Data
Frame Header
60 bytes IP Header
360 bytes


MTU=3300 bytes
I1
R1
I2
MTU=1500 bytes
I1
R2
I2
MTU = 1500 bytes
H1
H2
At I2-R2, H2 (Link layer)
H/w Addr
ID
MF
DF
Fragment Offset
IP Addr
I2-R2 H2 1 1 0 0 H1 H2 10 20 Data
Frame Header
60 bytes IP Header
40 bytes TP Header
1400 bytes
I2-R2 H2 1 1 0 175 H1 H2 Data
Frame Header
60 bytes IP Header
1440 bytes
I2-R2 H2 1 0 0 355 H1 H2 Data
Frame Header
60 bytes IP Header
360 bytes


I1
I2
I1
I2
MTU=3300 bytes
R1
MTU=1500 bytes
R2
MTU = 1500 bytes
H1
H2
At H2 (Network layer) –
Point of Reassembly
MF
DF
Fragment
Offset
IP Addr
ID
1
1
0
0
H1
H2
10
20
Data
60 bytes
IP Header
40 bytes
TP Header
1400 bytes
1
1
0
175
H1
H2
Data
60 bytes
IP Header
1440 bytes
1
0
0
355
H1
H2
Data
60 bytes
IP Header
360 bytes

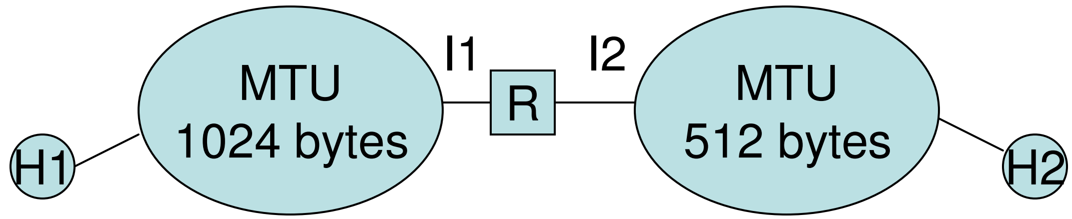


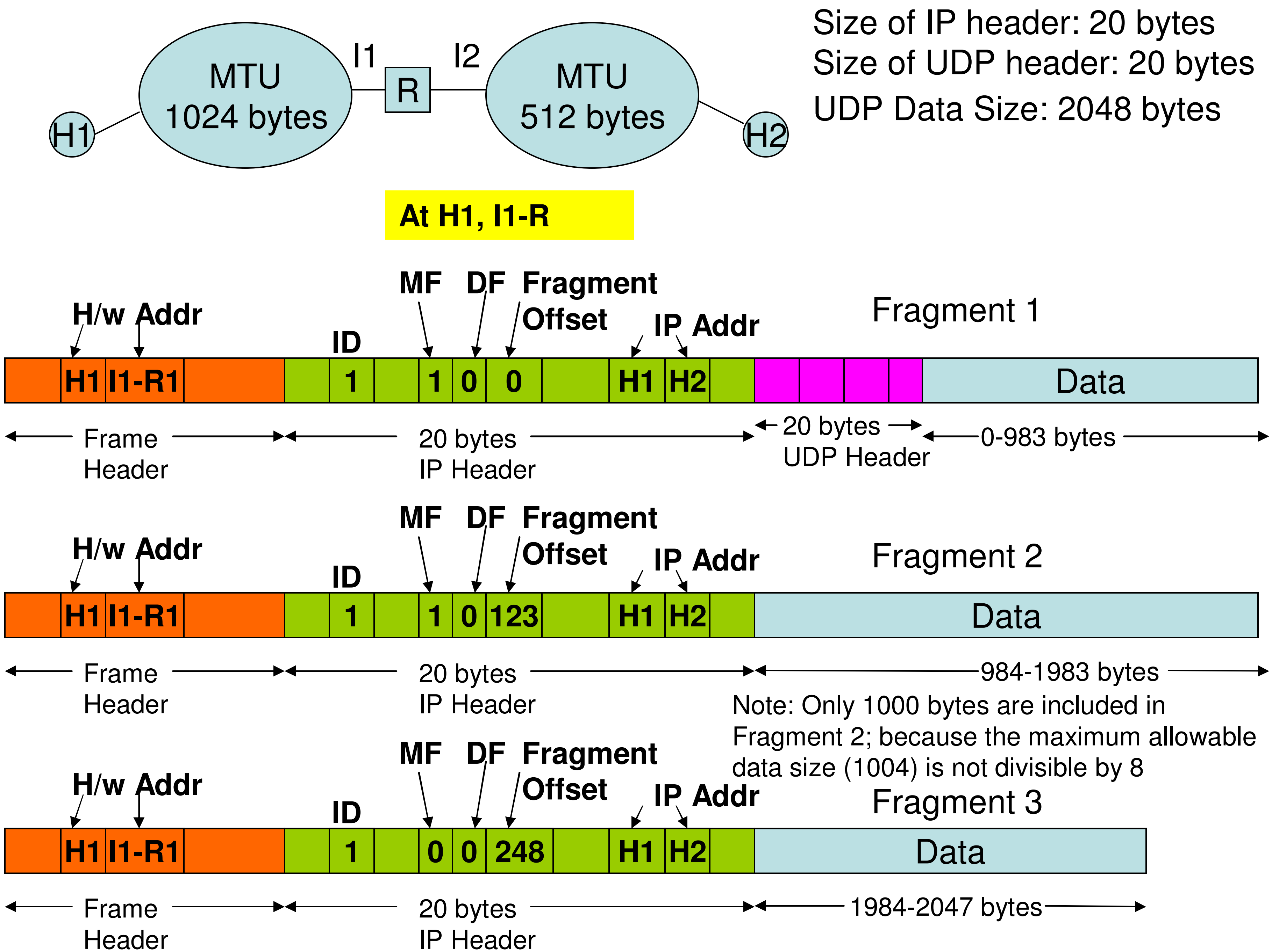
At H2 – Transport Layer



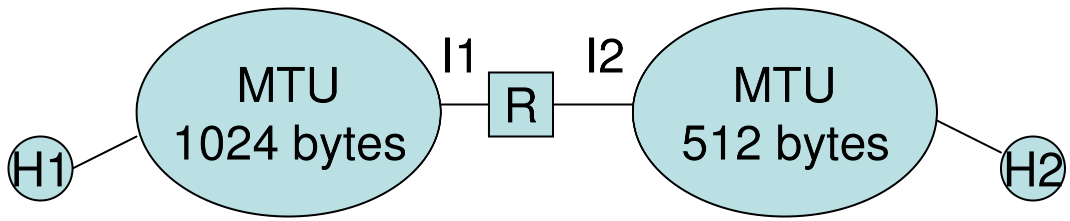
Example on Fragmenting a Fragment

- Suppose a UDP message that contains 2048 bytes of data and 20 bytes of UDP header is passed to IP for delivery across two networks of the Internet (as shown below: i.e., from the source host to a router to the destination host).
- The MTU of the source and destination networks are 1024 bytes and 512 bytes respectively.
- Show the structure of all the fragments of this datagram along with the values for the MF bit, DF bit, Offset field, the starting and ending byte number for the portion of actual data in each fragment.
- Assume the IP header used is of the minimum size.



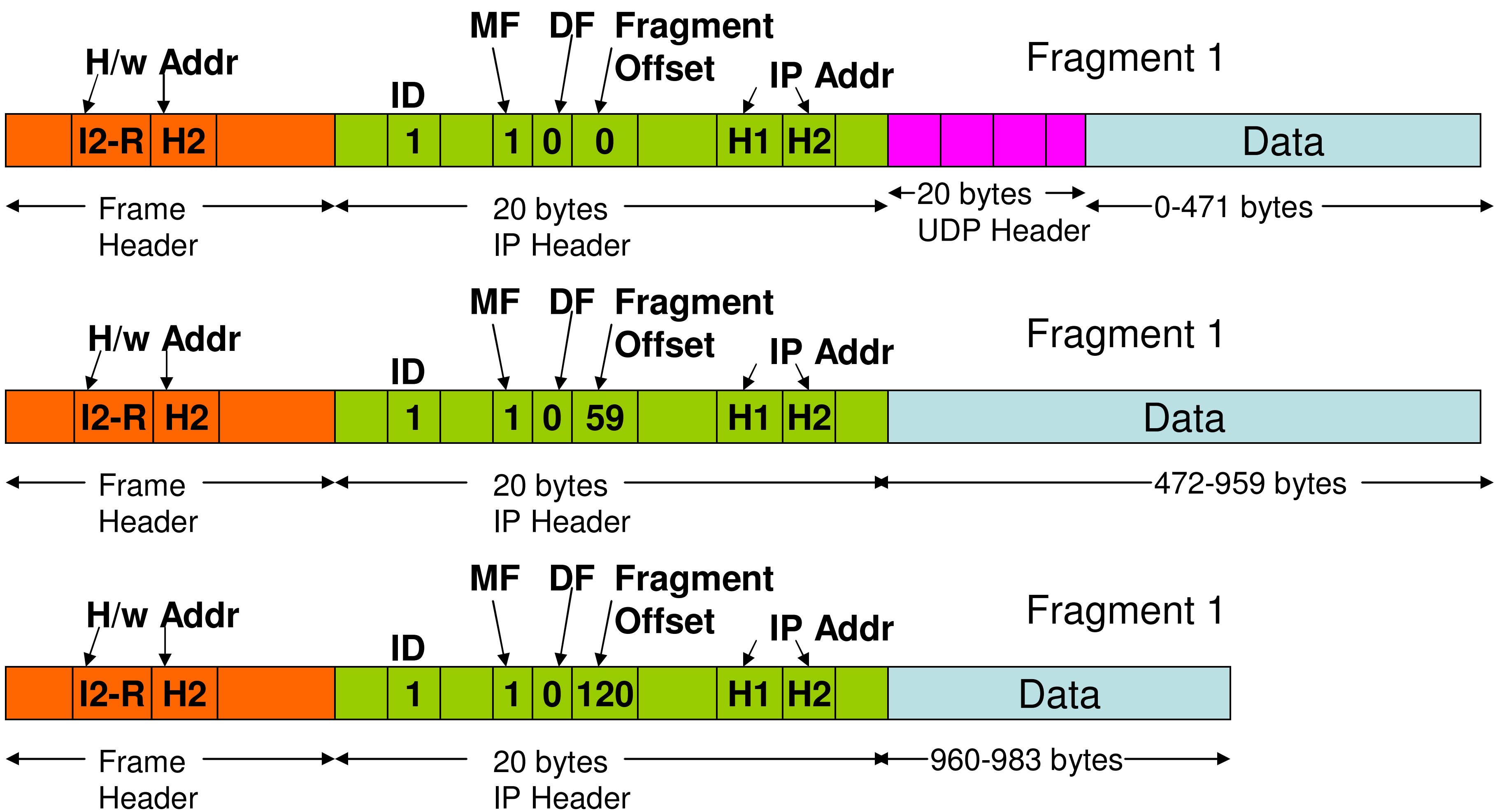


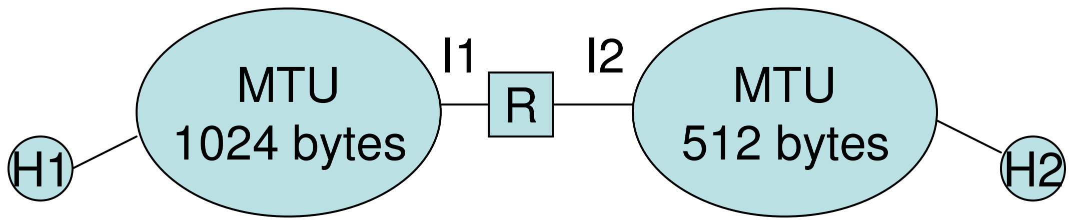
MTU 1024 bytes
I1
R
I2
MTU 512 bytes
H1
H2
Size of IP header: 20 bytes
Size of UDP header: 20 bytes
UDP Data Size: 2048 bytes
At H1, I1-R
Fragment 1
H/w Addr
ID
MF
DF
Fragment Offset
IP Addr
H1 I1-R1
1
1 0 0
H1 H2
Data
Frame Header
20 bytes IP Header
20 bytes UDP Header
0-983 bytes
Fragment 2
H/w Addr
ID
MF
DF
Fragment Offset
IP Addr
H1 I1-R1
1
1 0 123
H1 H2
Data
Frame Header
20 bytes IP Header
984-1983 bytes
Note: Only 1000 bytes are included in Fragment 2; because the maximum allowable data size (1004) is not divisible by 8
Fragment 3
H/w Addr
ID
MF
DF
Fragment Offset
IP Addr
H1 I1-R1
1
0 0 248
H1 H2
Data
Frame Header
20 bytes IP Header
1984-2047 bytes



Size of IP header: 20 bytes
Size of TCP header: 20 bytes
TCP Data Size: 2048 bytes

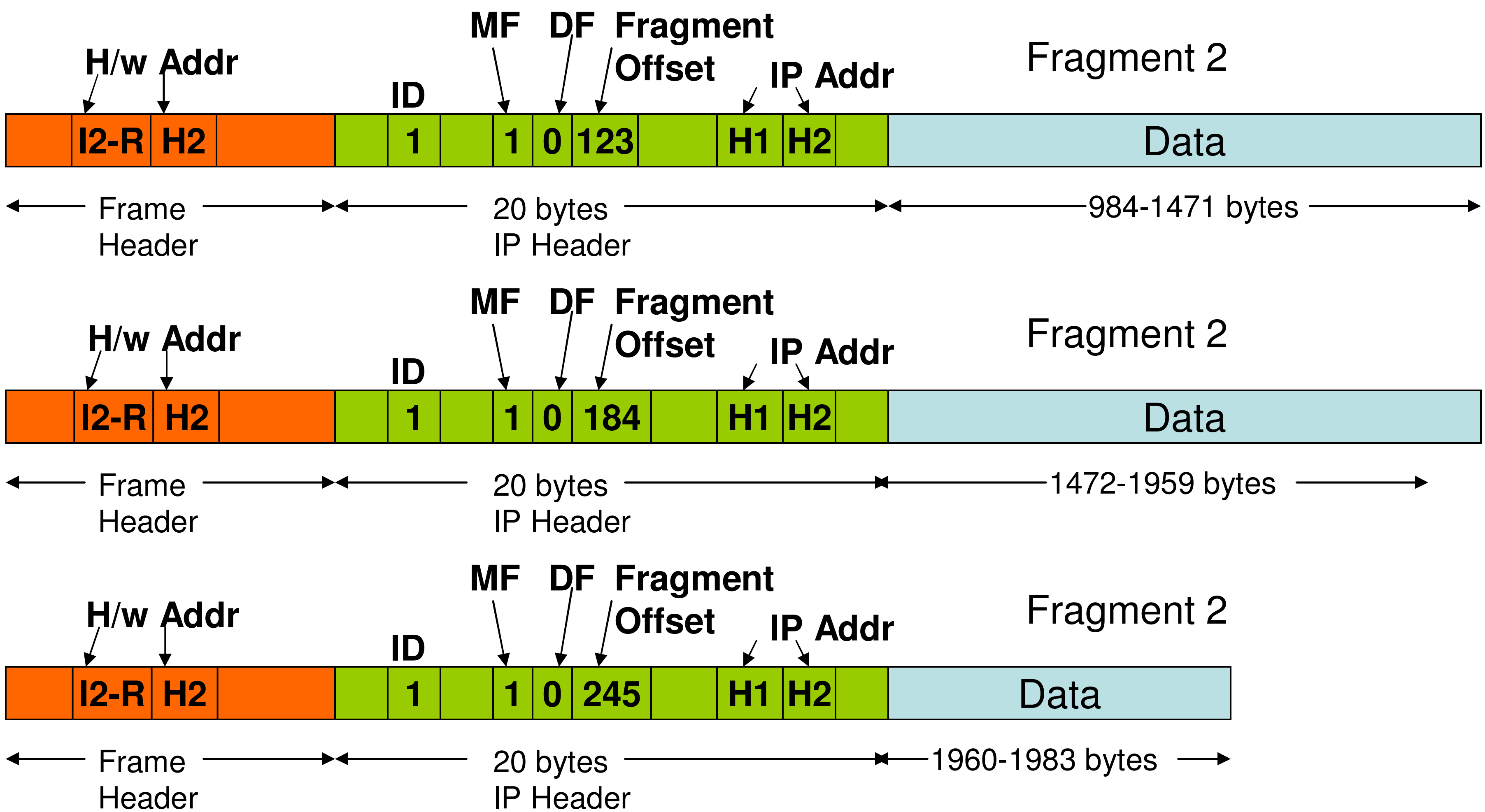
At I2-R

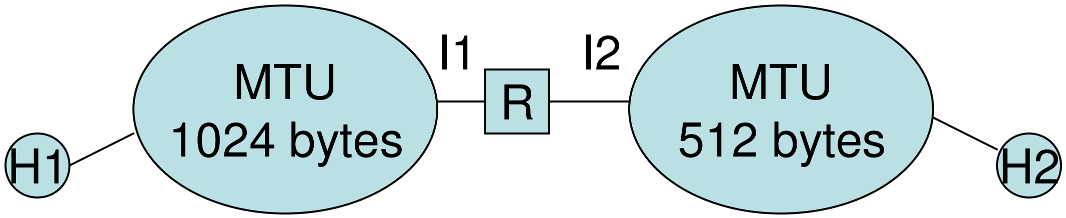




Size of IP header: 20 bytes
Size of TCP header: 20 bytes
TCP Data Size: 2048 bytes

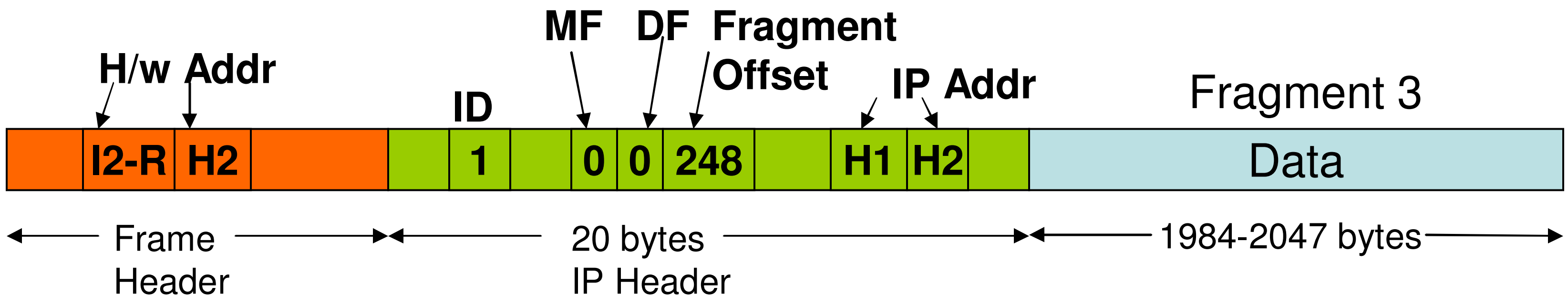
At I2-R





Size of IP header: 20 bytes
Size of TCP header: 20 bytes
TCP Data Size: 2048 bytes

At I2-R



Example on Fragment Loss

- Suppose an IP packet is fragmented into 10 fragments, each with a 1% (independent probability) of loss. With one transmission of all the 10 fragments, what is the probability of losing the whole packet due to the loss of a fragment? Also, what is the probability of loss of the whole packet if the packet is transmitted twice,
  - assuming all fragments received must have been part of the same transmission?
  - assuming any given fragment may have been part of either transmission?

Solution: Fragment Loss Example

- Probability that a fragment will be lost = 0.01
- Probability that a fragment will be received = 0.99

- With one transmission of all the 10 fragments, the packet loss happens if one or of the fragments get lost. The probability of loss of one or more fragments is 1 – Probability (loss of no fragments)

  = 1 – [Probability that each fragment is received]

  $= 1 - (0.99)^{10}$

  = 1 – 0.9044 = 0.0956
- Probability that there will be loss of packet = Probability of loss of one or more fragments

  = 0.0956 (9.56%).

Solution: Fragment Loss Example

With two transmissions

(a) All the fragments must be received from the same transmission

A packet is lost if the packet loss occurs in the 1st transmission and the packet loss occurs in the 2nd transmission.

Prob[packet lost]

= Prob[packet lost in 1st transmission] * Prob[packet lost in 2nd transmission]

= 0.0956 * 0.0956

= 0.009139 (0.91%)

Solution: Fragment Loss Example

<u>With two transmissions</u>

(b) A fragment can arrive in either of the two transmissions

A packet is lost if one or more fragments fail to arrive in both transmissions.

Prob[a fragment lost in both transmissions] = Prob[a fragment lost in 1st transmission]*Prob[a fragment lost in 2nd transmission]

= 0.01 * 0.01 = 0.0001

Prob[a fragment not lost in at least one transmission]

= 1 – Prob[a fragment lost in both transmissions] = 1 – 0.0001 = 0.9999

Prob[packet lost] = 1 – Prob[packet not lost]

= 1 – Prob[all the 10 fragments not lost in at least one transmission]

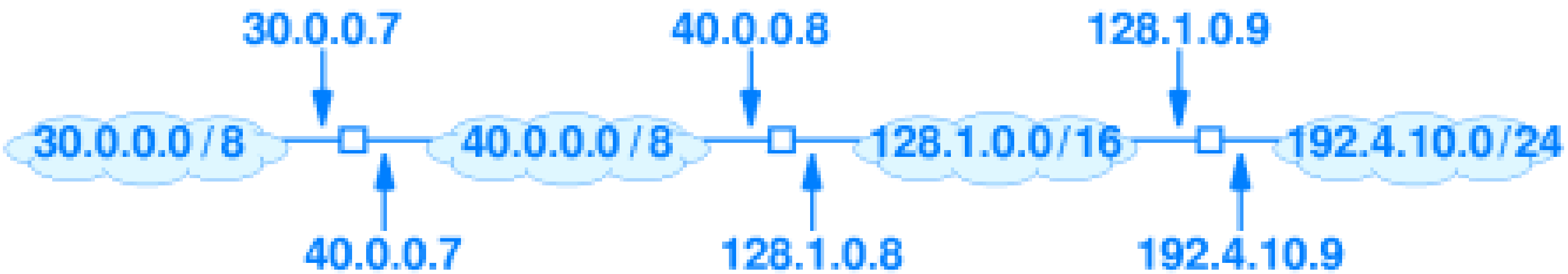
= $1 - (0.9999)^{10}$

= 1 – 0.999

= 0.001 (0.1%)

6.3 IP Datagram Forwarding

IP Datagram Forwarding



Routing Table at the Middle Router

Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

IP Datagram Forwarding

- The mask field is used to extract the network prefix of the IP address during lookup.
- Given a destination IP address D, for each entry in the routing table, the routing software computes the Boolean AND of the mask in an entry and D to extract the network prefix of D. If the extracted network prefix of D matches with that of the Destination network prefix in the entry, then an address match has occurred; the packet can be forwarded through the next-hop listed in the entry.

- Example: Consider the network topology in the previous slide. Consider a datagram destined for address 192.4.10.3 arrives at the middle router.
- The routing software computes a Boolean AND of 192.4.10.3 with each of 255.0.0.0, 255.255.0.0, 255.255.255.0 to find the matching destination network prefix.
- The routing software succeeds in the fourth entry as 192.4.10.3 & 255.255.255.0 = 192.4.10.0, which is the "Destination" network prefix in that entry. Hence, the datagram is forwarded to the next-hop router whose IP address is 128.1.0.9.

IP Datagram Forwarding Example

- Suppose a router has built up the routing table shown in the following table. The router can deliver packets directly over interfaces 0 and 1, or it can forward packets to routers R1, R2 or R3.
- Determine what the router does with a packet addressed to each of the following destinations:
  - (a) 148.110.17.131
  - (b) 148.110.16.18
  - (c) 212.40.163.90
  - (d) 212.40.163.17

Network Prefix/ Destination Network	Subnet Mask	Next Hop
148.110.16.0	255.255.255.128	Interface 0
148.110.16.128	255.255.255.128	Interface 1
148.110.17.0	255.255.255.128	R1
212.40.163.0	255.255.255.192	R2
<default>		R3

IP Datagram Forwarding Solution

- (a) Consider the destination IP address 148.110.17.131
- Taking a bit-wise AND with the subnet mask 255.255.255.128:

  148. 110. 17.   1 0 0 0 0 0 1 1

  255. 255. 255. 1 0 0 0 0 0 0 0

  ---

  148. 110. 17. 1 0 0 0 0 0 0 0

  148. 110. 17. 128 - it does not match with any of the corresponding network prefixes (148.110.16.0, 148.110.16.128 or 148.110.17.0) for the subnet mask 255.255.255.128 in the routing table.
- Taking a bit-wise AND with the subnet mask 255.255.255.192:

  148. 110. 17.   1 1 0 0 0 0 1 1

  255. 255. 255. 1 1 0 0 0 0 0 0

  --

  148. 110. 17.   1 1 0 0 0 0 0 0

  148. 110. 17. 192 - it does not match with the network prefix 212.40.163.0.
- Since, there is no match with any of the routing table entries with their masks, we have to forward the packet to the default router, R3.

IP Datagram Forwarding Solution

- (b) Consider the destination IP address 148.110.16.18
- Taking a bit-wise AND with the subnet mask 255.255.255.128:

  148. 110. 16. 0 0 0 1 0 0 1 0

  255. 255. 255. 1 0 0 0 0 0 0 0

  ----

  148. 110. 16. 0 0 0 0 0 0 0 0

  148. 110. 16. 0 - it matches with the entry for which the next hop is interface 0.

- (c) Consider the destination IP address 212.40.163.90
- Taking a bit-wise AND with the subnet mask 255.255.255.192:

  212. 40. 163. 0 1 0 1 1 0 1 0

  255. 255. 255. 1 1 0 0 0 0 0 0

  ----

  212. 40. 163. 0 1 0 0 0 0 0 0

  212. 40. 163. 64 - it does not match with the entry 212.40.163.0. Hence, send the packet to the default next hop router, R3.

IP Datagram Forwarding Solution

- (d) Consider the destination IP address 212.40.163.17
- Taking a bit-wise AND with the subnet mask 255.255.255.192:

212. 40. 163. 0 0 0 1 0 0 0 1

255. 255. 255. 1 1 0 0 0 0 0 0

---

212. 40. 163. 0 0 0 0 0 0 0 0

212. 40. 163. 0 - it matches with the entry for which the next hop is R2.

IP Datagram Forwarding (Alternate Approach)

Network Prefix/ Destination Network	Subnet Mask	Next Hop
148.110.16.0	255.255.255.128	Interface 0
148.110.16.128	255.255.255.128	Interface 1
148.110.17.0	255.255.255.128	R1
212.40.163.0	255.255.255.192	R2
<default>		R3

- For each of the above combinations of network prefix and subnet mask, determine the range of valid IP addresses.
- If the *test* IP addresses fall within the range, then the packet is sent through the appropriate next hop.

148.110.16.0 255.255.255.128

We have 25 bits for the network + subnet part and 7 bits for the host part

148.110.16.0 0 0 0 0 0 0 0

The range of unicast IP addresses are: **148.110.16.1 to 148.110.16.126**

148.110.16.128   255.255.255.128
We have 25 bits for the network + subnet part and 7 bits for the host part
148.110.16.1 0 0 0 0 0 0 0
The range of unicast IP addresses are: **148.110.16.129 to 148.110.16.254**

148.110.17.0   255.255.255.128
We have 25 bits for the network + subnet part and 7 bits for the host part
148.110.17.0 0 0 0 0 0 0 0
The range of unicast IP addresses are: **148.110.17.1 to 148.110.17.126**

212.40.163.0   255.255.255.192
We have 26 bits for the network + subnet part and 6 bits for the host part
212.40.163.0 0 0 0 0 0 0 0
The range of unicast IP addresses are: **212.40.163.1 to 212.40.163.62**

- (a) 148.110.17.131 – Default R3
- (b) 148.110.16.18 – Interface 0
- (c) 212.40.163.90 - Default R3
- (d) 212.40.163.17 – R2

6.4 IP Auxiliary Protocols and Technologies

Address Resolution Protocol (ARP)

- <u>Motivation for Address Translation</u>
- IP datagrams contain IP addresses, but the physical network interface hardware cannot understand it. So, if the datagram has to be sent to a host or next hop router in a given physical network, it has to be encapsulated in a frame and addressed to the hardware address (link-level address) of the host/router interface on that network.

- ARP cache/table: The protocol enables each host on a network to build a table of mappings between IP addresses and link-level addresses.
- The entries in the table are discarded if not refreshed.

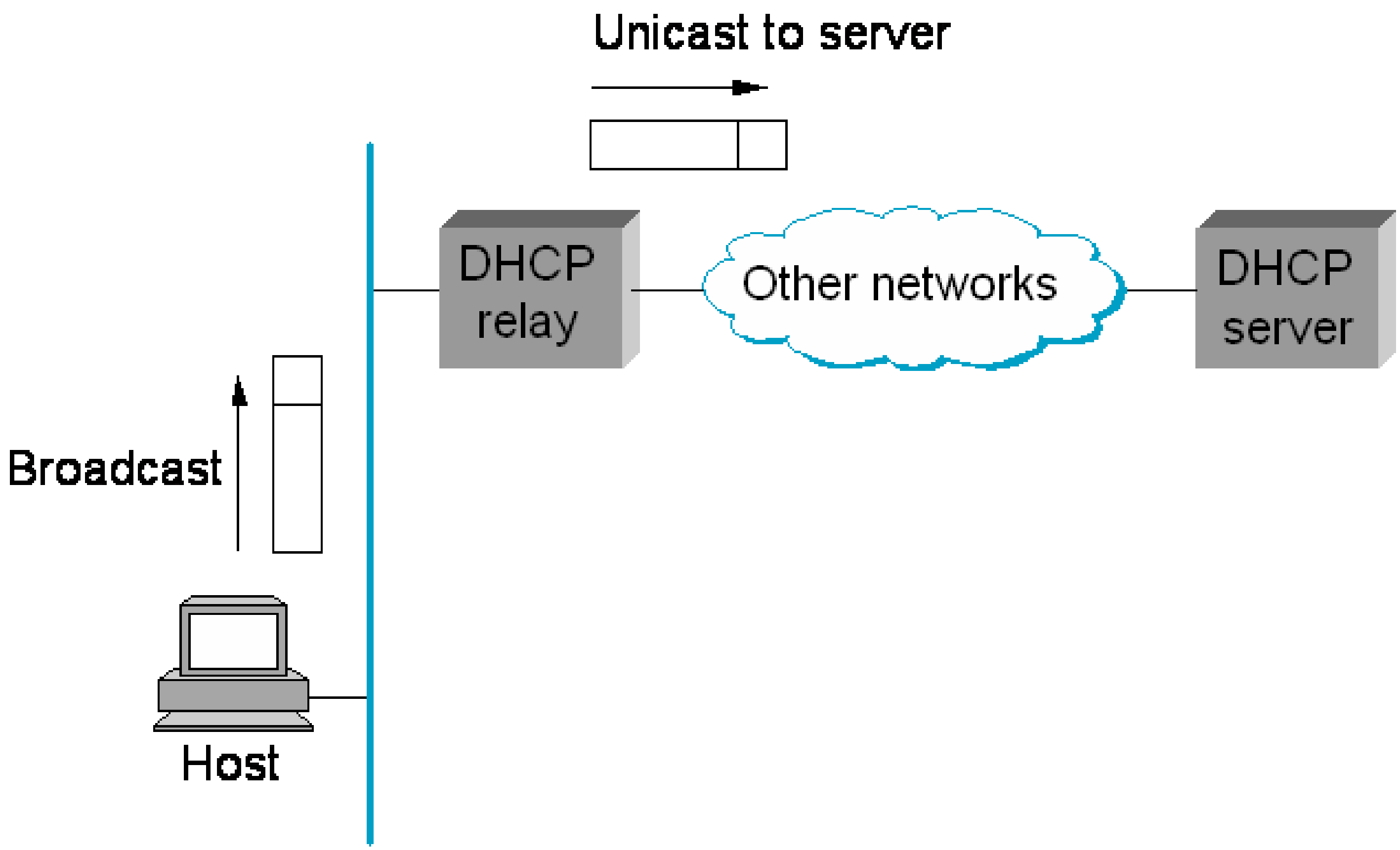
Address Resolution Protocol (ARP)

- If a host wants to send an IP datagram to a host (or router) that is known to be on the same network (i.e., the sending and receiving host have the same IP network number), the host first checks its ARP table for a mapping of the receiver's IP address to link-level address.
- If no mapping is found, the host broadcasts a ARP query (that contains the IP address in question) onto the network.
- Each host receives the query and checks to see if it matches its IP address. If it does match, the host sends a response message that contains its link-level address back to the originator of the query.
- The target host of the ARP query also creates an entry in its ARP table and stores the mapping between the IP address and link-level address of the query's sender. All the other hosts discard the ARP query.
- The ARP query also includes the IP address and link-level address of its sender. This information is read by all the hosts receiving the broadcast. Hosts add an entry in the ARP table to create a mapping between the IP address and link-level address of the sender (or refresh the mapping using the query, if an entry already exists).

Dynamic Host Configuration Protocol (DHCP)

- IP addresses need to be reconfigurable as the network part of the IP address of a host has to be at least changed depending on the network to which the host is attached.
- When a host connects to a network, it should be able to get assigned the IP address corresponding to that particular network. The protocol that handles this problem is called the Dynamic Host Configuration Protocol (DHCP)
- DHCP relies on the existence of a DHCP server that provides the configuration information to hosts. The DHCP server would maintain a pool of available IP addresses from which it will handover one address to a requesting host. The network administrator has to only allocate a range of IP addresses (all with the same IP network number) to each network.
- A host that boots up or connects to a network needs to be able to contact the DHCP server.

Dynamic Host Configuration Protocol (DHCP)



Dynamic Host Configuration Protocol (DHCP)

- The host broadcasts a DHCPDISCOVER message (with target IP address 255.255.255.255) to all the hosts in the network.
- Only the DHCP server or the DHCP relay agent pick up the DHCP message. If the message is picked up by a DHCP server, it assigns an IP address to the requesting host and notifies it through a DHCPREPLY message.
- If the DHCPDISCOVER message is picked up by a DHCP relay agent, it unicasts the message to the DHCP server, which assigns the address and notifies the requesting host through the relay agent.
- An IP address is assigned on a “lease basis” and the requesting client has to periodically refresh it. If not refreshed, the IP address is returned to the available pool of addresses and is free to be assigned to another host.

Internet Control Message Protocol (ICMP)

- IP is configured with a companion protocol called ICMP that defines a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully.
- ICMP messages do not have special priority
  - They are forwarded like any other datagram and not retransmitted, if lost.

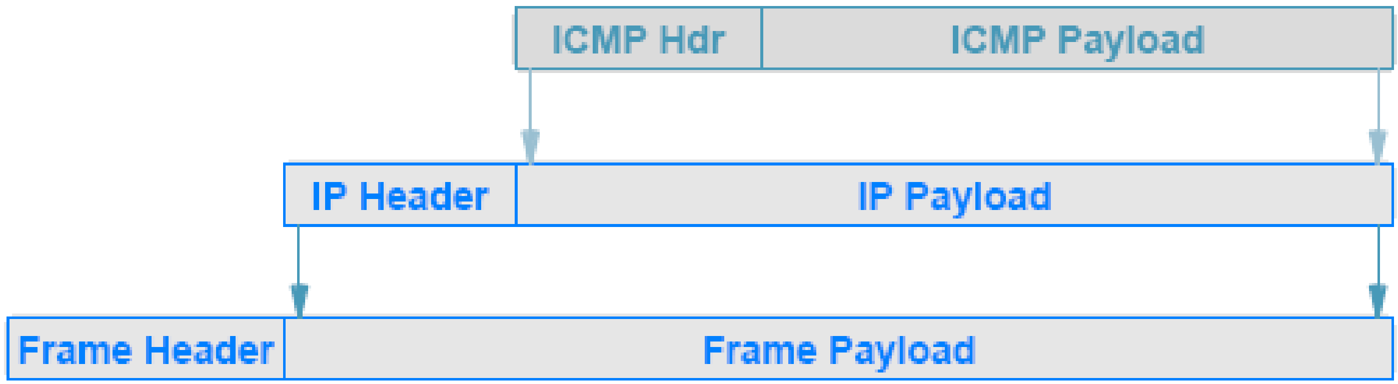
- Some of the ICMP error messages are for the following reasons:

- Destination host unreachable
- TTL reached 0
- Checksum failed
- Cannot fragment

- Besides, ICMP also includes some control messages that help to improve network performance. These are:

- ICMP redirect
- ECHO Request/Reply

ICMP Encapsulation



Number	Type	Purpose
0	Echo Reply	Used by the ping program
3	Dest. Unreachable	Datagram could not be delivered
5	Redirect	Host must change a route
8	Echo	Used by the ping program
11	Time Exceeded	TTL expired or fragments timed out
12	Parameter Problem	IP header is incorrect
30	Traceroute	Used by the traceroute program

6.5 IPv6

IPv6

Address Space Allocation

- 128 bits
- No classes. But the leading bits specify different uses of the IPv6 addresses.

Address Notation

- The standard representation is x:x:x:x:x:x:x:x, where each “x” is a hexa-decimal representation of the 16-bit piece of the address.
- Example: 47CD:1234:4422:AC02: 0022:1234:A456:0124

Aggregatable global unicast addresses: Has a 001 prefix.

Link local use addresses: The “link local use” address enables a host to construct an address that will work on the network to which it is connected without being concerned about the global uniqueness of the address.

Has a 1111 1110 10 as prefix

IPv4-compatible-IPv6 address: Obtained by zero-extending a 32-bit IPv4 address to 128 bits.

Multicast addresses: Have 1111 1111 as prefix

Example: IPv4-compatible IPv6 Address

Compute the IPv4-compatible IPv6 address for 143.132.10.45

Represented as : : 143.132.10.45 (prefixed with 128-32 = 96 0s)

Writing each of the 8-bit decimal value in binary,

: : 10001111:10000100:00001010:00101101

➔ : : 8F84: 0A2D

IPv6

Address Notation

- An address with a large number of contiguous 0s can be written more compactly by omitting all the 0 fields. For example,

  47CD:0000:0000:0000:0000:0000:A456:0124

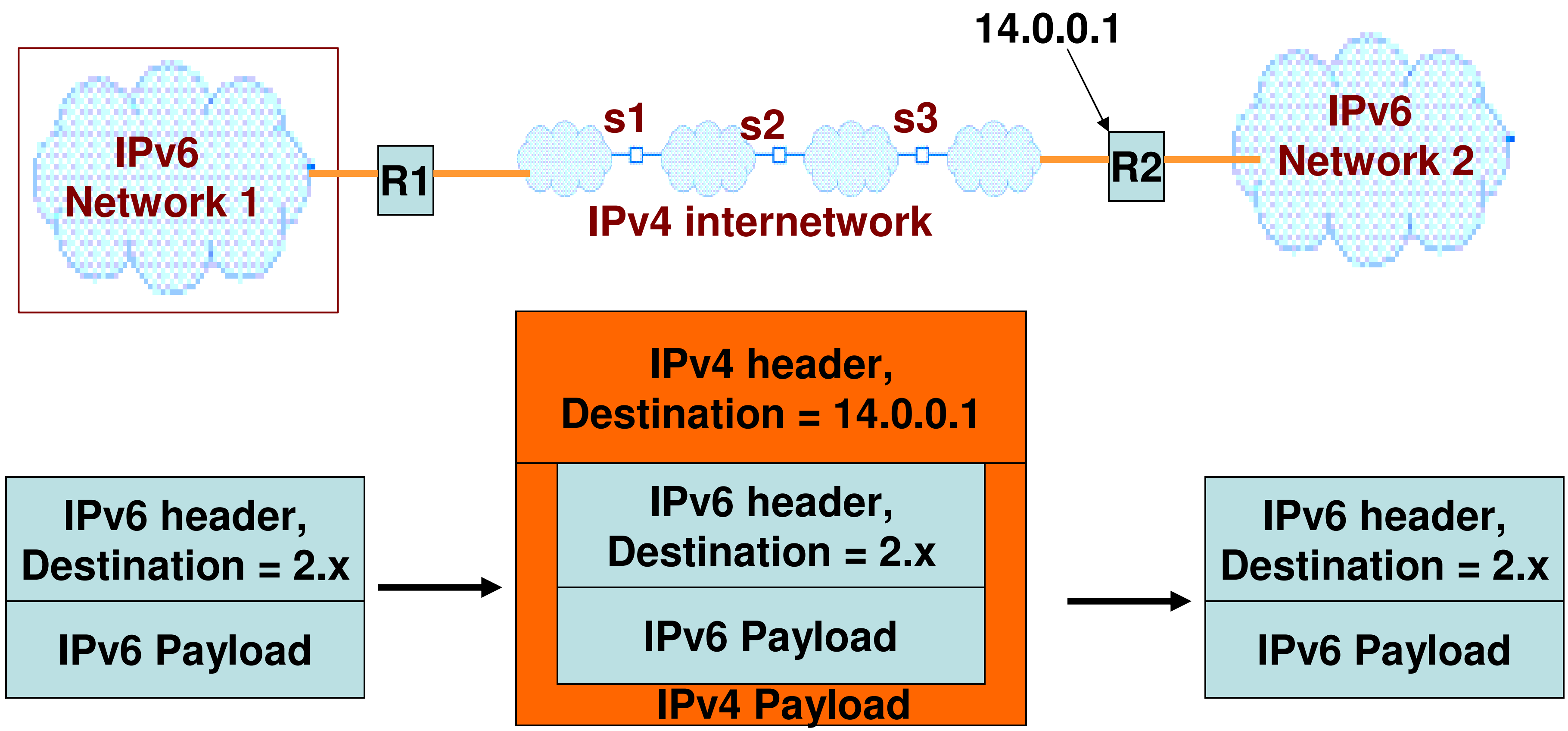
  can be written as 47CD::A456:0124

Transition from IPv4 to IPv6:

- Two IPv6 nodes should be able to communicate through an IPV4 network.
- An IPv4 to IPv6 communication (or vice-versa) goes through a NAT-like gateway at the IPv6 side → The IPv6 address space network is treated like a private network and the IPv6 addresses are replaced with one public IPv4 address.

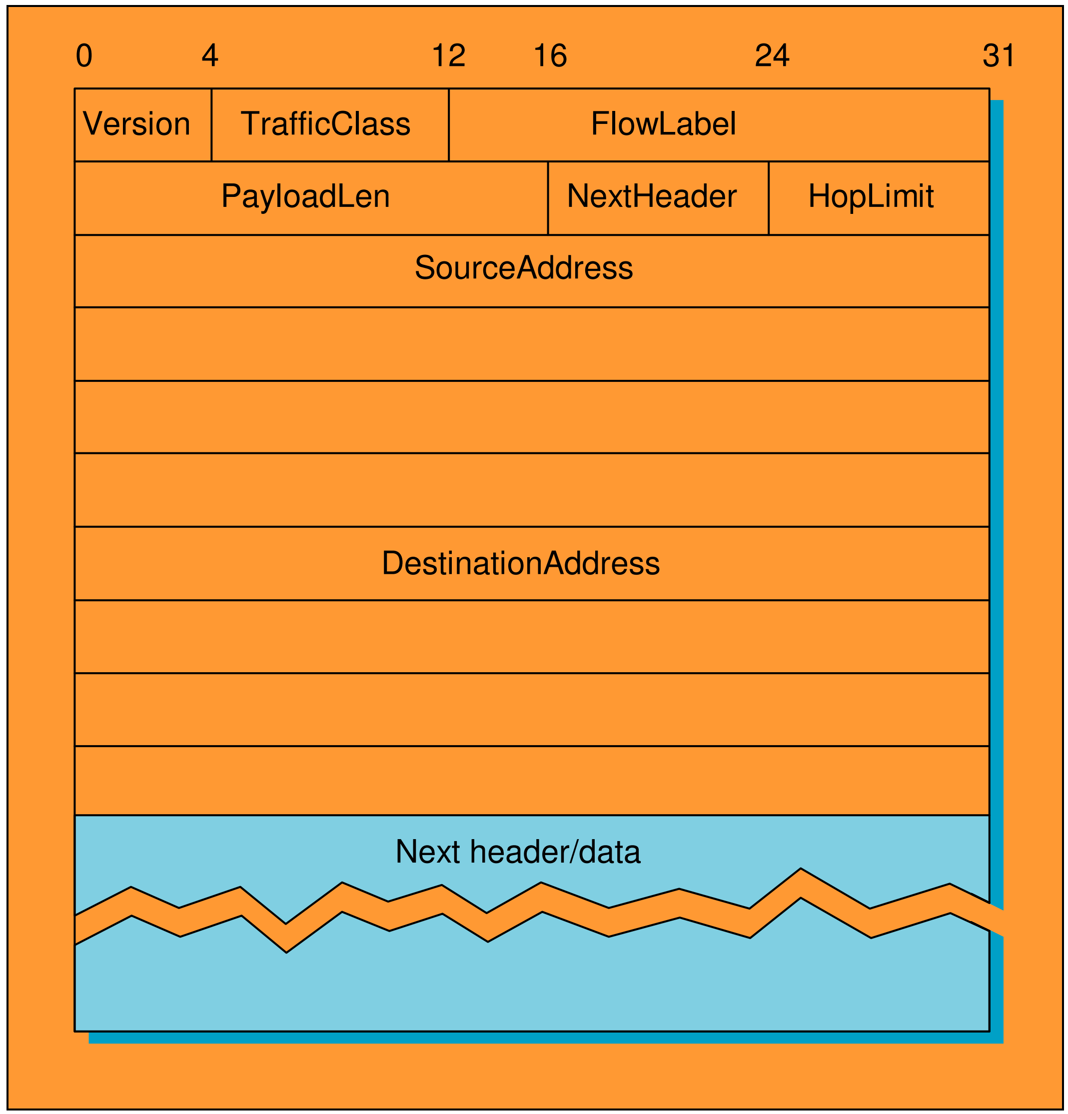
IPv6 in IPv4 Tunneling

- The entire IPv6 datagram is encapsulated inside the IPv4 payload.
- For the IPv6 interface of router R1, the number of hops to the IPv6 network 2 is ONE; while for the IPv4 interface of router R1, the number of hops to the IPv4 interface of router R2 is THREE.

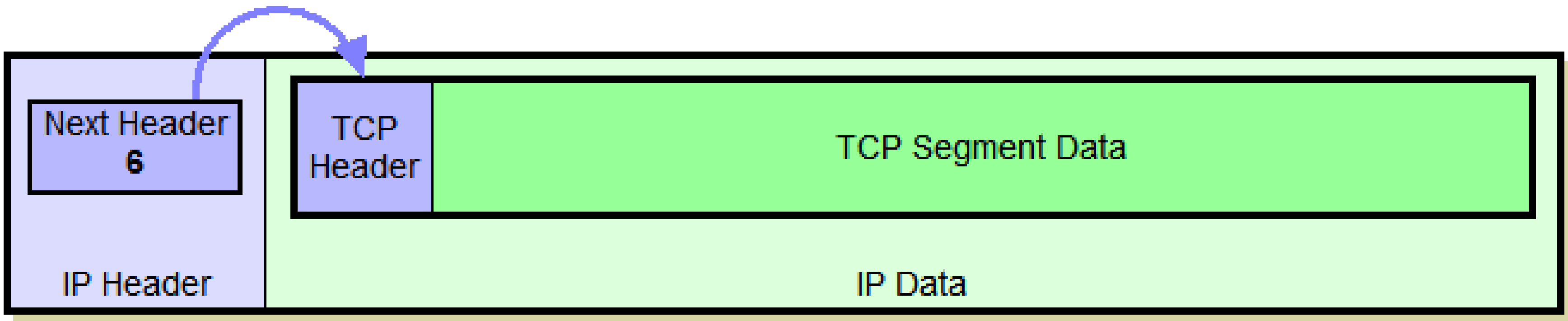


IPv6 Packet Header

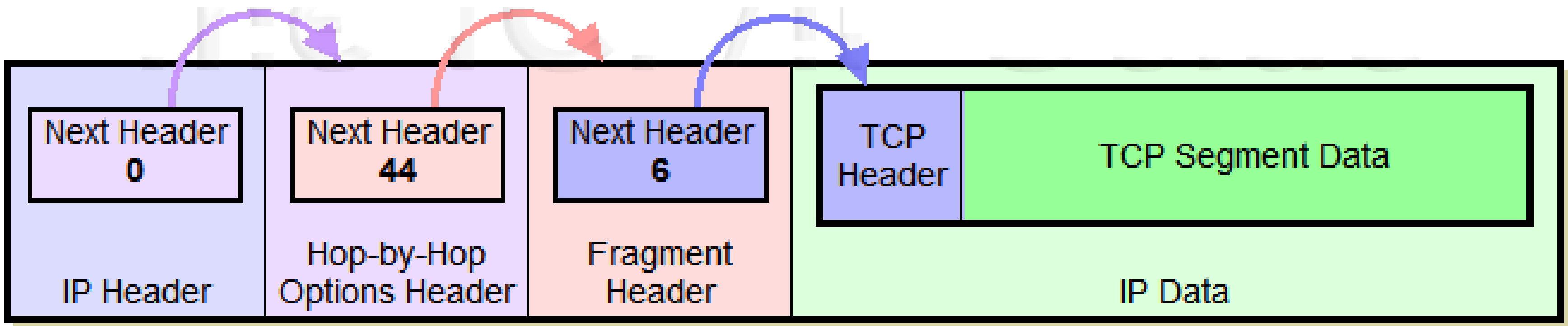
IPv6 Base Header



Use of Extension Headers in IPv6



IPv6 Datagram With No Extension Headers Carrying TCP Segment



IPv6 Datagram With Two Extension Headers Carrying TCP Segment

Picture taken from http://www.tcpipguide.com