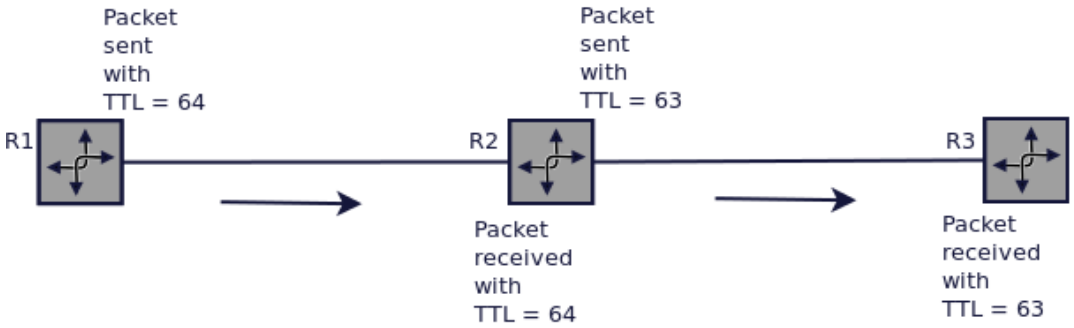
IP Datagram Fields (cont.)

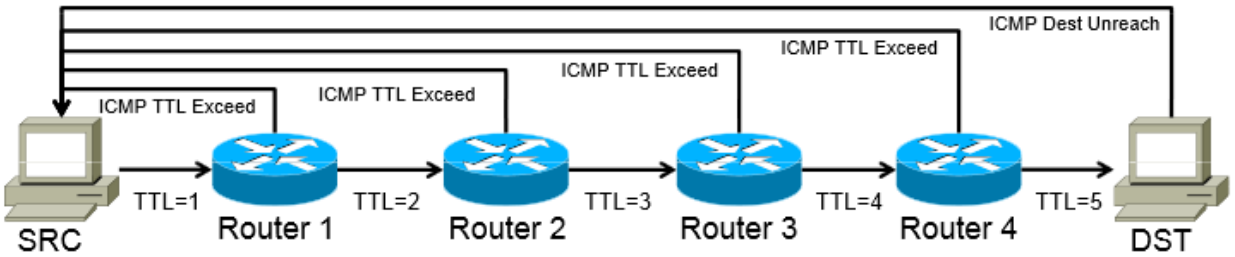
Time-To-Live (TTL) – **8-bit field – controls maximum number of hops visited by datagram and/or time spend in the network**

mutable field

- **field is decremented by one each time datagram is processed by a router – when TTL reaches 0, datagram must be dropped**
- **ensures that**
  1) **datagram does not circulate/loop forever, or**
  2) **to limit its journey** (e.g. LAN only: TTL = 1)



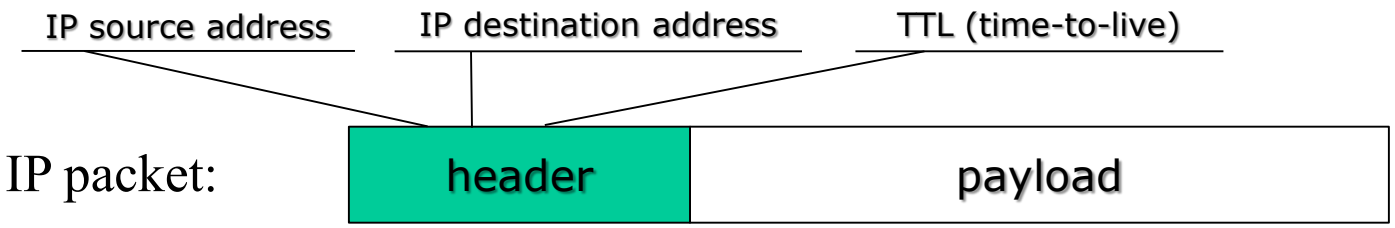
TTL in Traceroute:

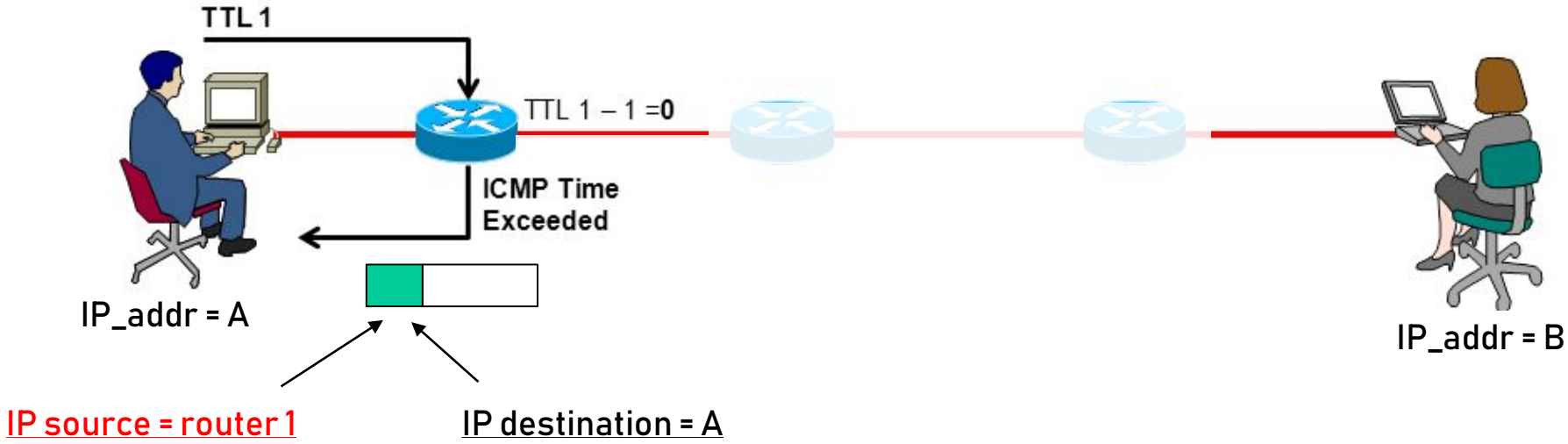


Network Basics (cont.)

How Does Traceroute Work?

TTL starts at 64, 128
every router decreases it by 1
when TTL=0, router drops the packet

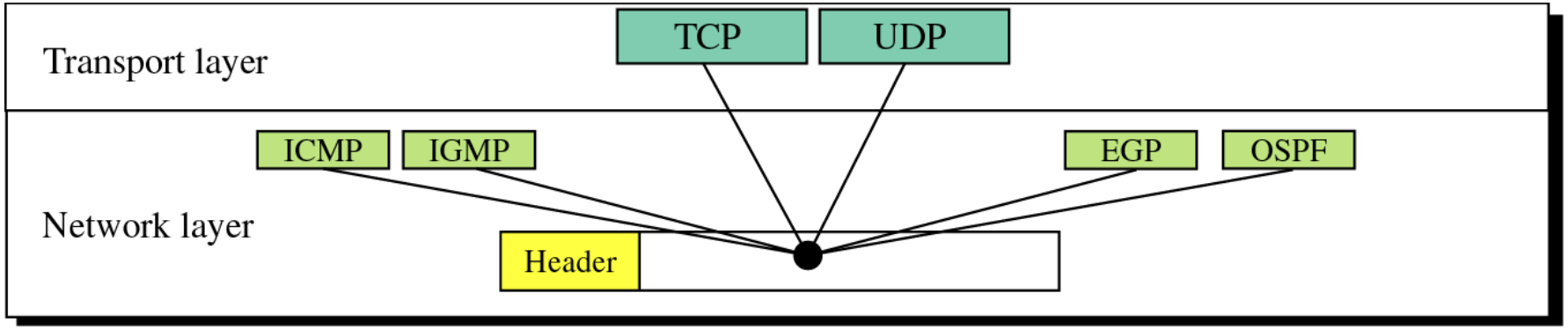




IP Datagram Fields (cont.)

Protocol – **8-bit field – indicates specific higher-level protocol that uses the services of IP layer –** IP datagram can encapsulate data from a number of higher-layer protocols

- **used only at final destination to facilitate demultiplexing process**
- protocol number is glue that binds network and transport layer, while port number is glue that binds transport and application layer together
- **values: 1 – ICMP, 2 – IGMP, 6 – TCP, 17 – UDP, 89 – OSPF**



IP Datagram Fields (cont.)

Header Checksum – **16-bit field – aids in detecting errors in header only!**

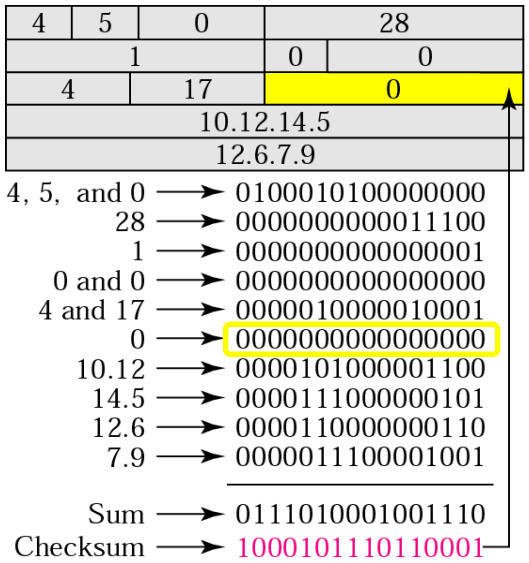
mutable field

- **checksum must be recomputed and stored again at each router as TTL and some options fields may change**
- **routers discard datagrams for which an error is detected**
- **checksum calculation:**
  1) **divide header into 16-bit (2-byte) sections – checksum field itself is set to 0**
  2) **sum all sections using 1s complement arithmetic**

Each intermediate router must:

1) verify / recompute checksum on every incoming packet

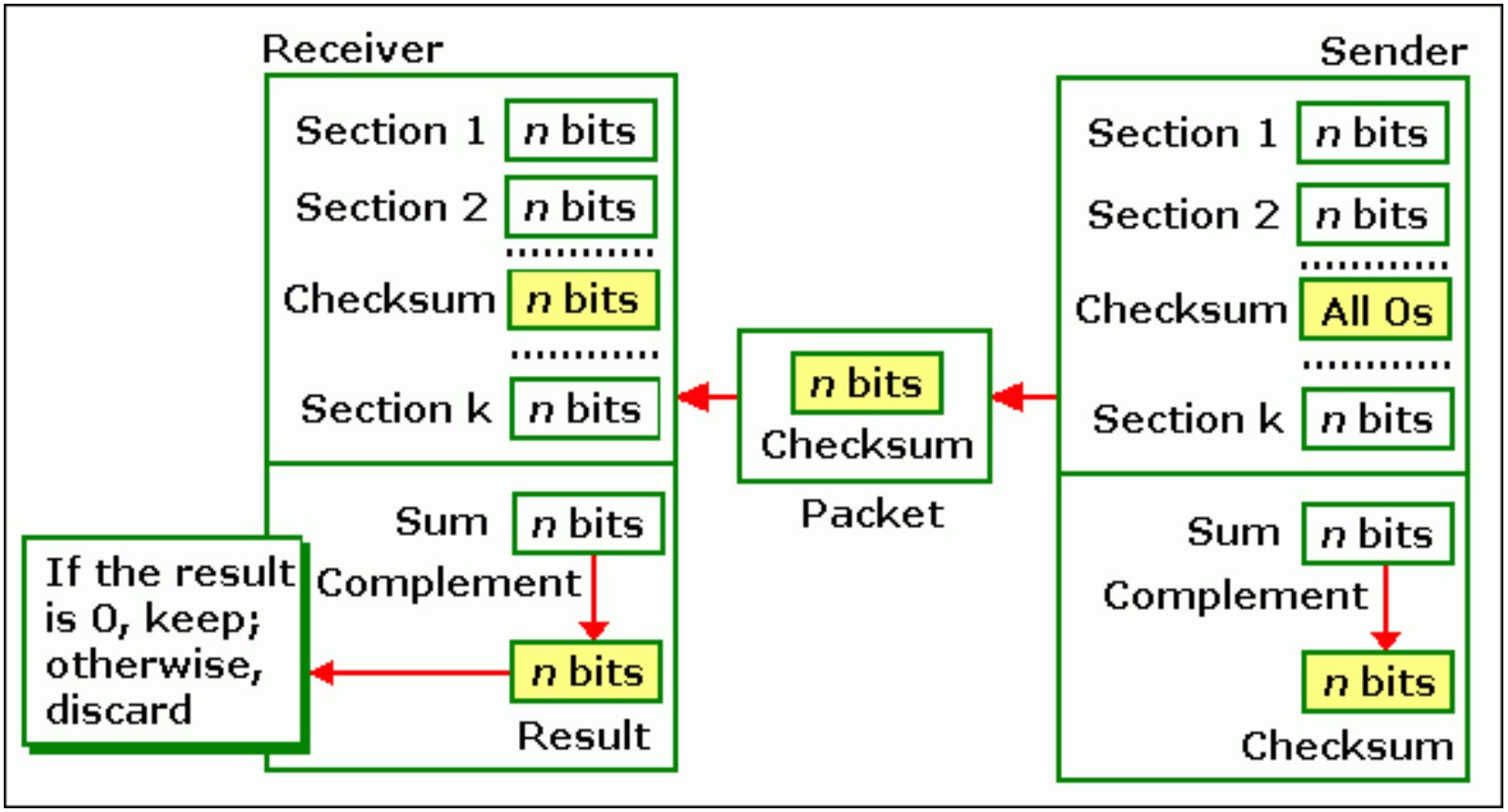
2) (re)compute checksum for every outgoing packet !!!



Error detection / correction is not the responsibility of network-layer.
Why is, then, IP willing to perform error detection on IP headers?!

IP Datagram Fields (cont.)

Example **[Checksum validation at Receiver …]**

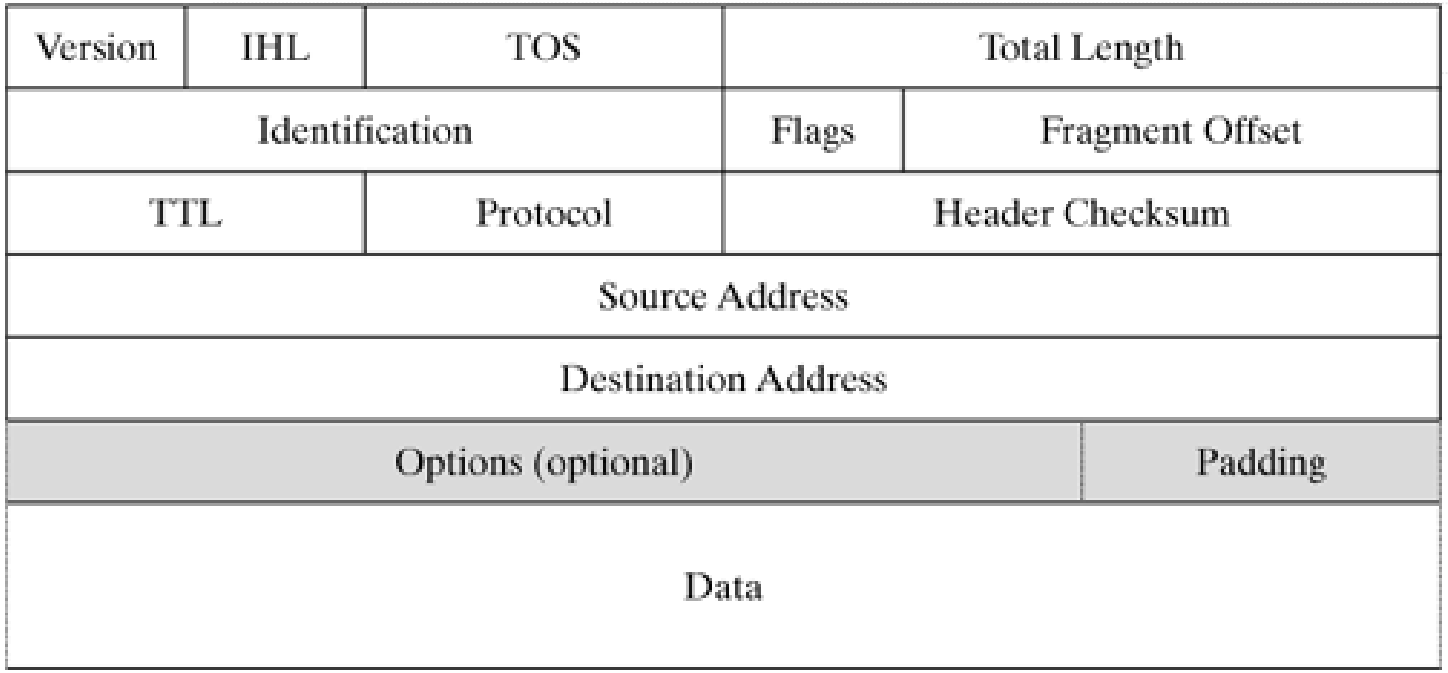


http://t4planet.tripod.com/checksum.html

IP Datagram Fields (cont.)

Source and Destination IP Addresses – **32-bit fields – must remain unchanged until IP datagram reaches its final destination**

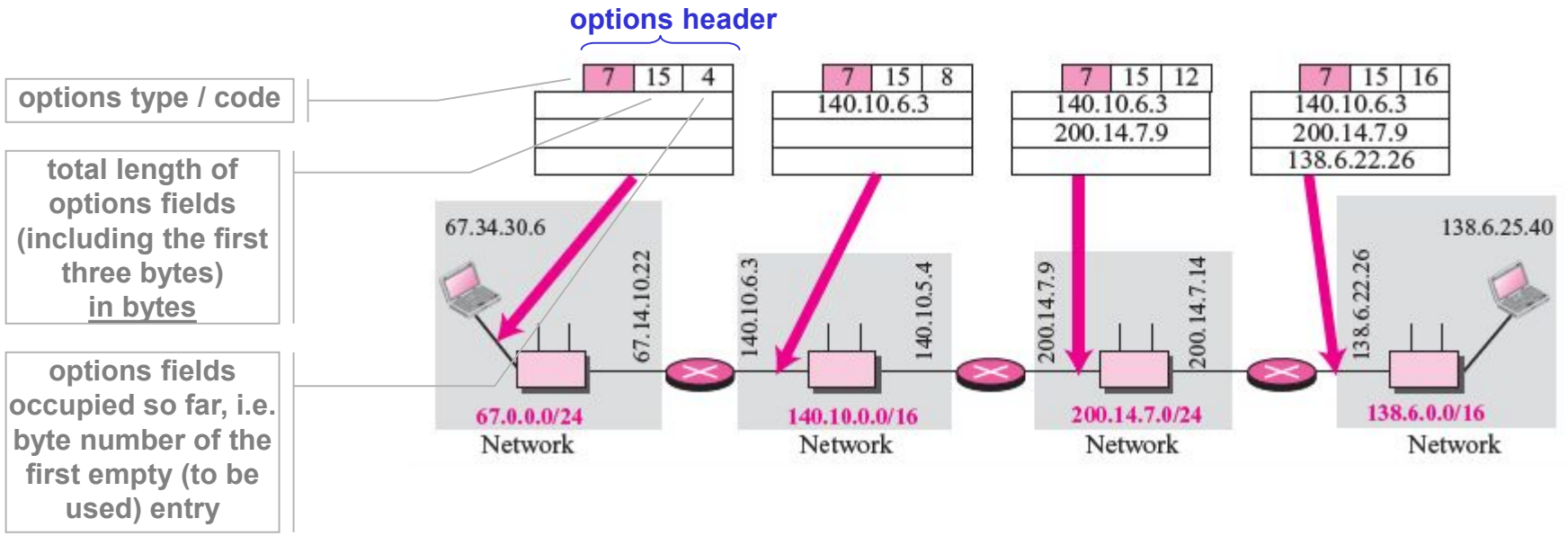
Options – **32-bit field(s) – not required for every datagram! – allows expansion of IP header for special purposes**



IP Datagram Fields (cont.)

(a) **Record Route option** – used to trace route that a datagram takes

- source creates empty fields for IP addresses – up to 9!!!
  (40 bytes for options – 4 bytes for option header) / 4 bytes for IP address
- each router that processes datagram inserts its outgoing IP address



From "TCP/IP Protocol Suite" by B. Forouzan, 4/e, pp. 200

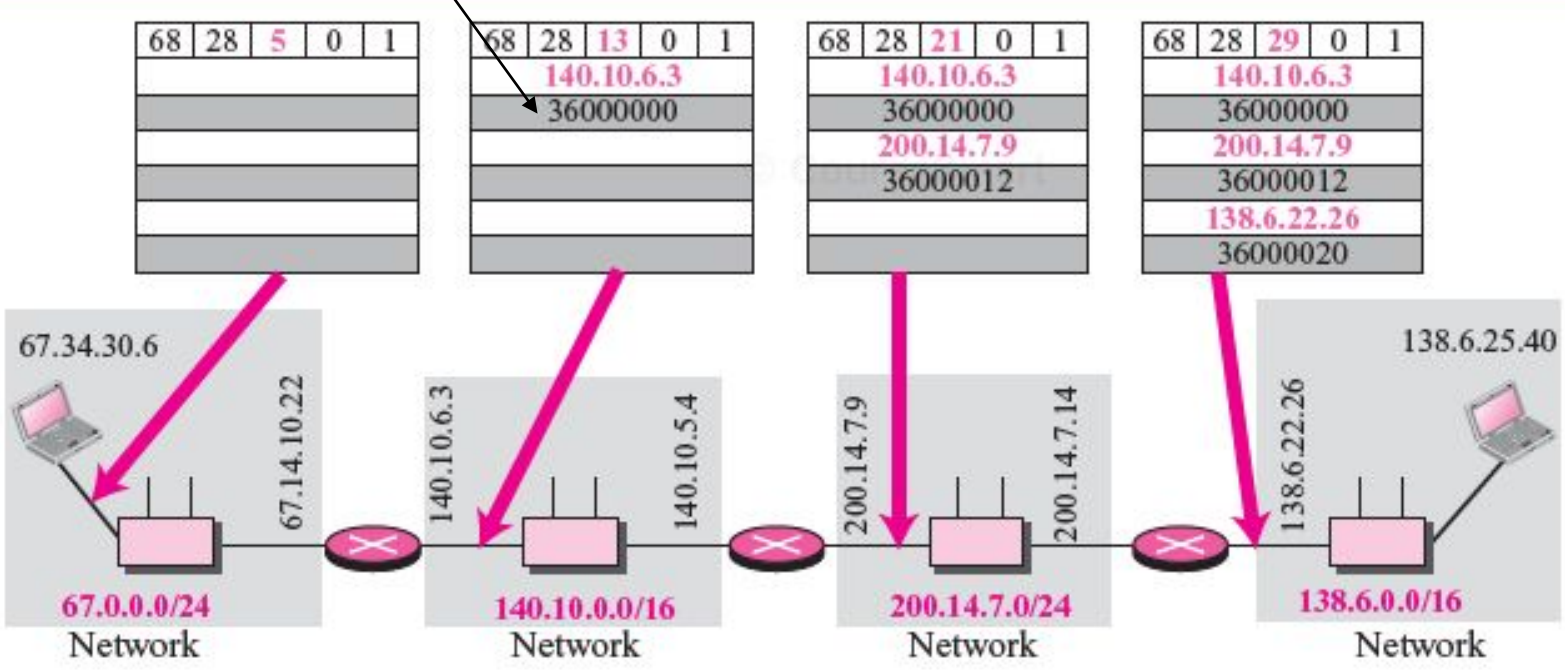
IP Datagram Fields (cont.)

Options (cont.)

(b) **Timestamp option** – similar to (a), plus records datagram end-processing time by each router, in milliseconds, universal time

- can help managers and users track the behavior of the routers in the Internet
- allows us to estimate the time it takes for a datagram to go from one router to another

Time in milliseconds since midnight UT!



From "TCP/IP Protocol Suite" by B. Forouzan, 4/e, pp. 203

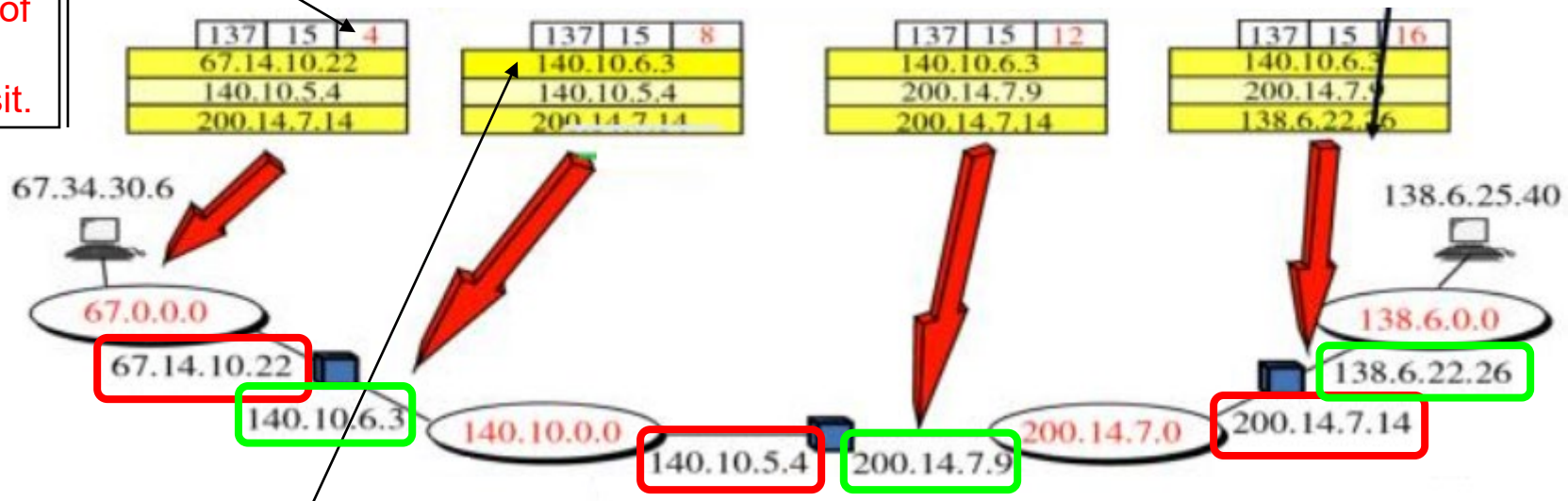
IP Datagram Fields (cont.)

Options (cont.)

(c) **Strict Source Route option** – used by source to predetermine route for a datagram (e.g., enforce the min-distance or min-cost)

- source provides a list of IP addresses, i.e. sequence of routers, that datagram must (is allowed) to visit on its way to destination

Indicates the byte position that contains IP address of the next router to visit.



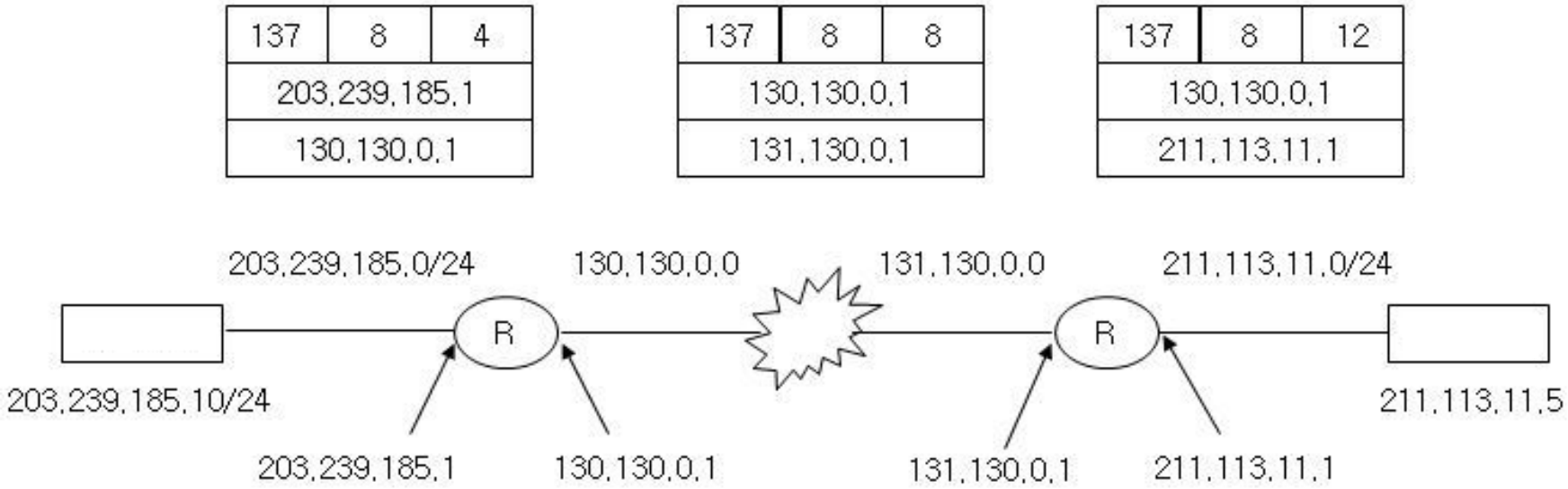
From "TCP/IP Protocol Suite" by B. Forouzan, 4/e, pp. 201

Router replaces the IP of the next router to visit with its own IP. Location to 'write over' indicated by the byte pointer. Result: receiving station obtains 'Record Router'.

(d) **Loose Source Route option** – similar to (c), but it is more relaxed – each router in the list must be visited, but datagram can visit other routers as well

IP Datagram Fields (cont.)

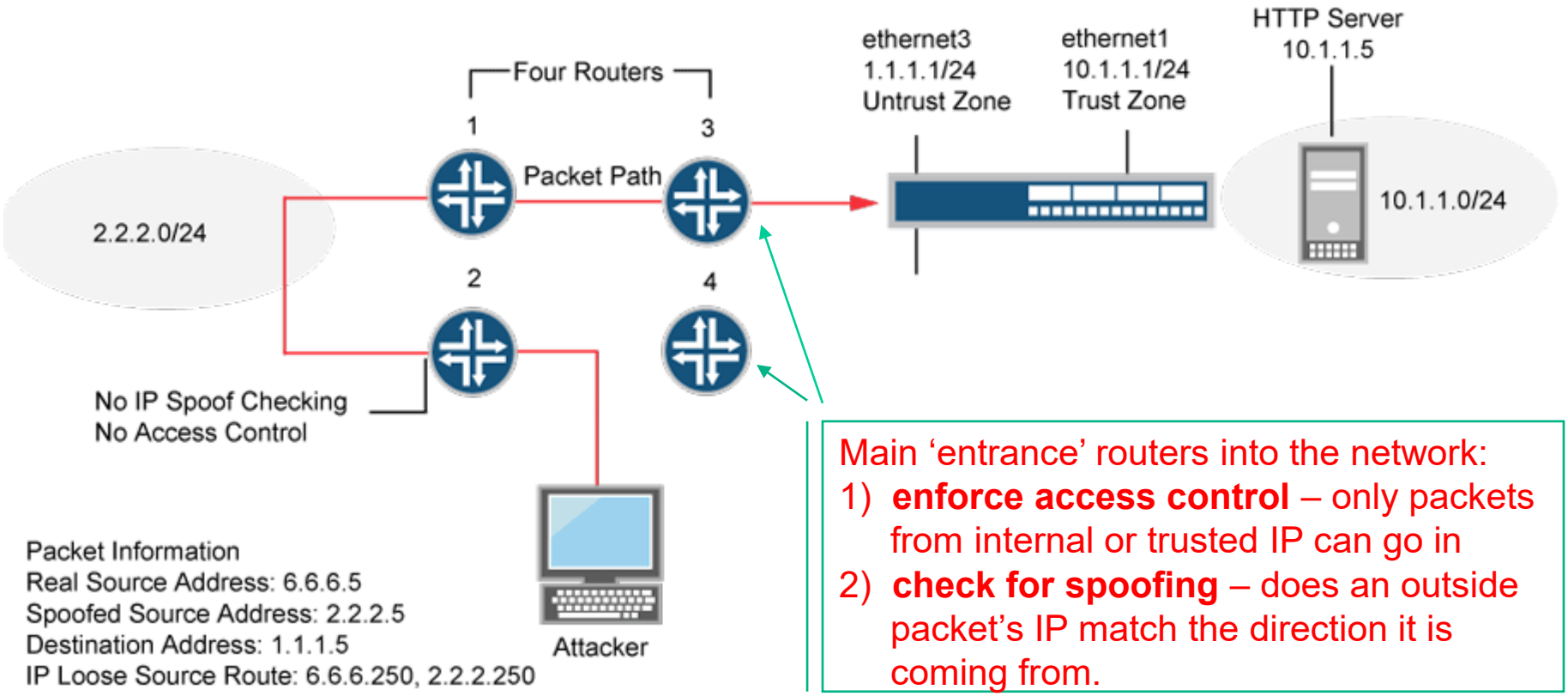
Example **[Strict Source Routing]**



IP Datagram Fields (cont.)

Example **[Security Issues with Strict Source Routing]**

Although the uses of IP source route options were originally benign, attackers have learned to put them to more devious uses. They can use IP source route options to hide their true address and access restricted areas of a network by specifying a different path.



https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-attacker-evasion-technique.html

IP Datagram Fields (cont.)

Example [IP Datagram fields]

An IP packet has arrived with the first 8 bits as shown: 01000010
The receiver discards the packet. Why?

Solution:

There is an error in this packet. The 4 left-most bits (**0100**) show the version, which is correct. The next 4 bits (**0010**) show the header length, which means ($2 \times 4 = 8$), which is wrong. The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

Example [IP Datagram fields]

In an IP packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

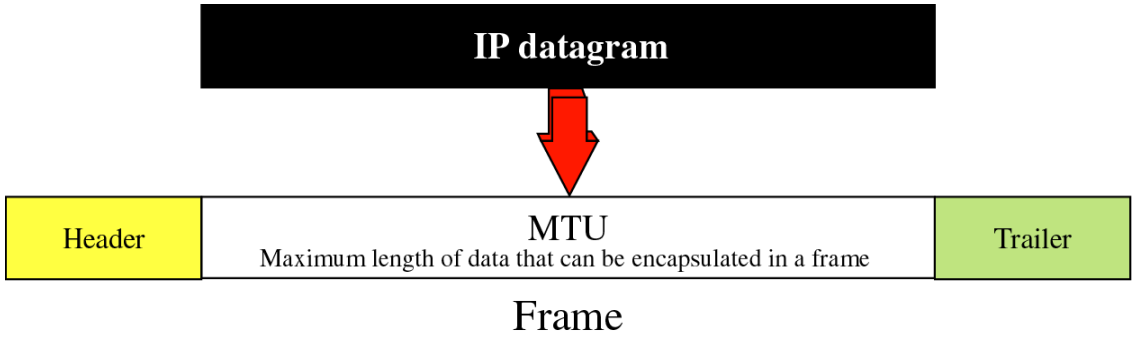
Solution:

The HLEN value is 8, which means the total number of bytes in the header is 8x4 or 32 bytes. The first 20 bytes are the main header, the next 12 bytes are the options.

IP Datagram Fragmentation

Maximum Transfer Unit (MTU) – **maximum amount of data that link-layer frame can carry = hard limit on IP datagram length**

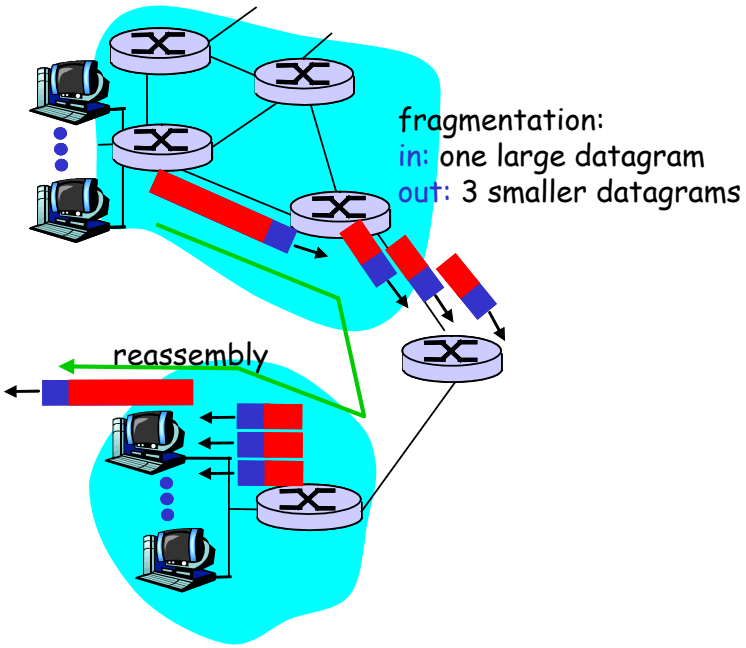
- **MTU differs from one data-link layer protocol to another**
  - **(a) Token Ring (4 Mbps): MTU = 4,464 bytes**
  - **(b) Ethernet: MTU = 1,500 bytes**
  - **(c) PPP: MTU = 296 bytes**



Hard limit on IP datagram size is not a problem.
What is a problem is that each of the links along the route between sender and receiver can use different link-layer protocols,
and each of these protocols can have different MTUs.

IP Datagram Fragmentation (cont.)

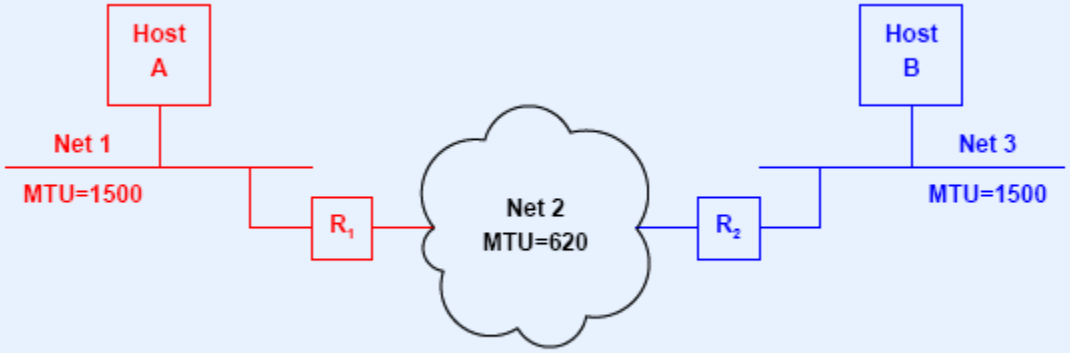
IP Datagram Fragmentation – process of dividing datagram into smaller fragments that meet MTU requirements of underlying data-link layer protocol



- datagram can be fragmented by source-host or any other router in the path; however **reassembly of datagram is done only by destination host!** – parts of a fragmented datagram may take different routes
- once fragmented datagram may be further fragmented if it encounters network with even smaller MTU
- **when a datagram is fragmented, each fragment gets its own header with most fields repeated, but some changed**
  - host or router that fragments datagram must change values of three fields: ***flags***, ***fragmentation offset*** and ***total length***

IP Datagram Fragmentation (cont.)

Example **[Example, from the book by D. E. Comer]**



- Hosts A and B send datagrams of up to 1500 octets
- Router R_1 fragments large datagrams from Host A before sending over Net 2
- Router R_2 fragments large datagrams from Host B before sending over Net 2

IP Datagram Fragmentation (cont.)

Identification – **16-bit field – uniquely identifies datagram originating from source host**

- **to guarantee uniqueness, IP uses counter to label each datagram**
- **when IP sends a datagram, it copies current counter value to identification field, and increments counter by one**
- **when datagram is fragmented, identification field is copied into all fragments**
- **identification number helps destination in reassembling datagram – all fragments with same identification value should be assembled into one datagram**

Flags – **3-bit field**

- **1st bit is reserved**
- **2nd bit is called "do not fragment" bit**
  - **if its value is 1, machine must NOT fragment datagram**
  - **if fragment cannot pass through physical network router discards packet and sends ICMP error message back to source host**
- **3rd bit is called "more fragment" bit**
  - **if its value is 1, datagram is not last fragment – there are more fragments after this one**
  - **if its value is 0, this is last or only fragment**

D: Do not fragment
M: More fragments

IP Datagram Fragmentation (cont.)

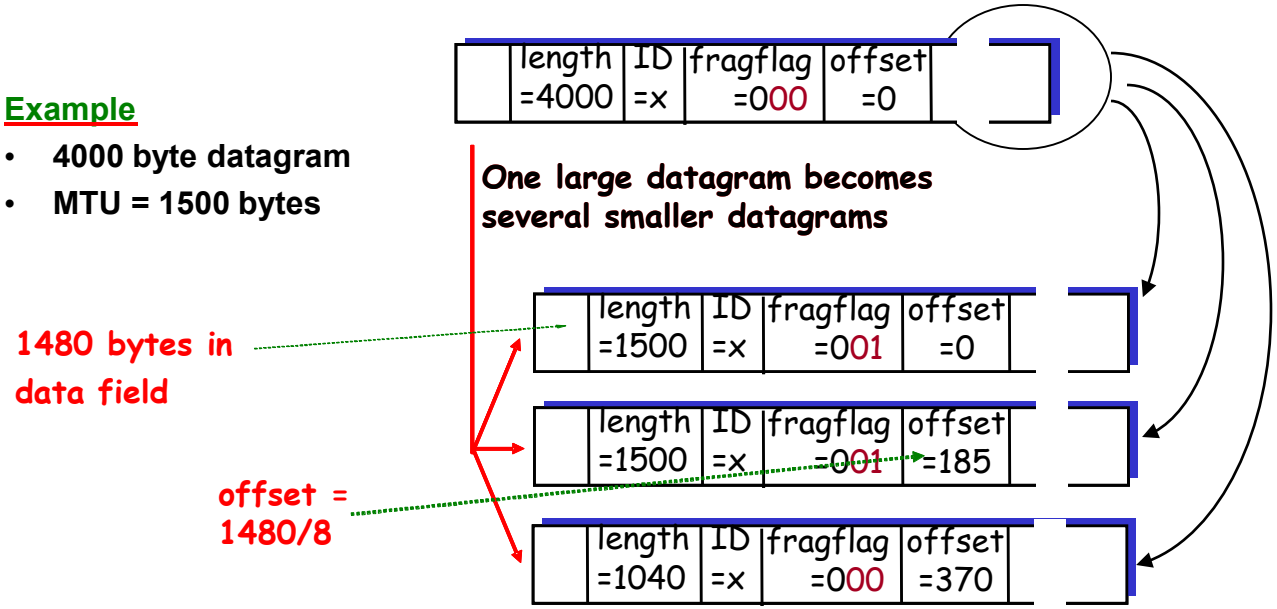
Fragmentation Offset – **13-bit field – shows relative position of fragment's data with respect to whole datagram**

- **the offset is measured in units of 8 bytes – this is done because offset field is only 13 bits long and otherwise could not represent sequences greater than 8191**
- **this forces hosts and routers to choose fragment sizes divisible by 8**

Example [fragmentation]

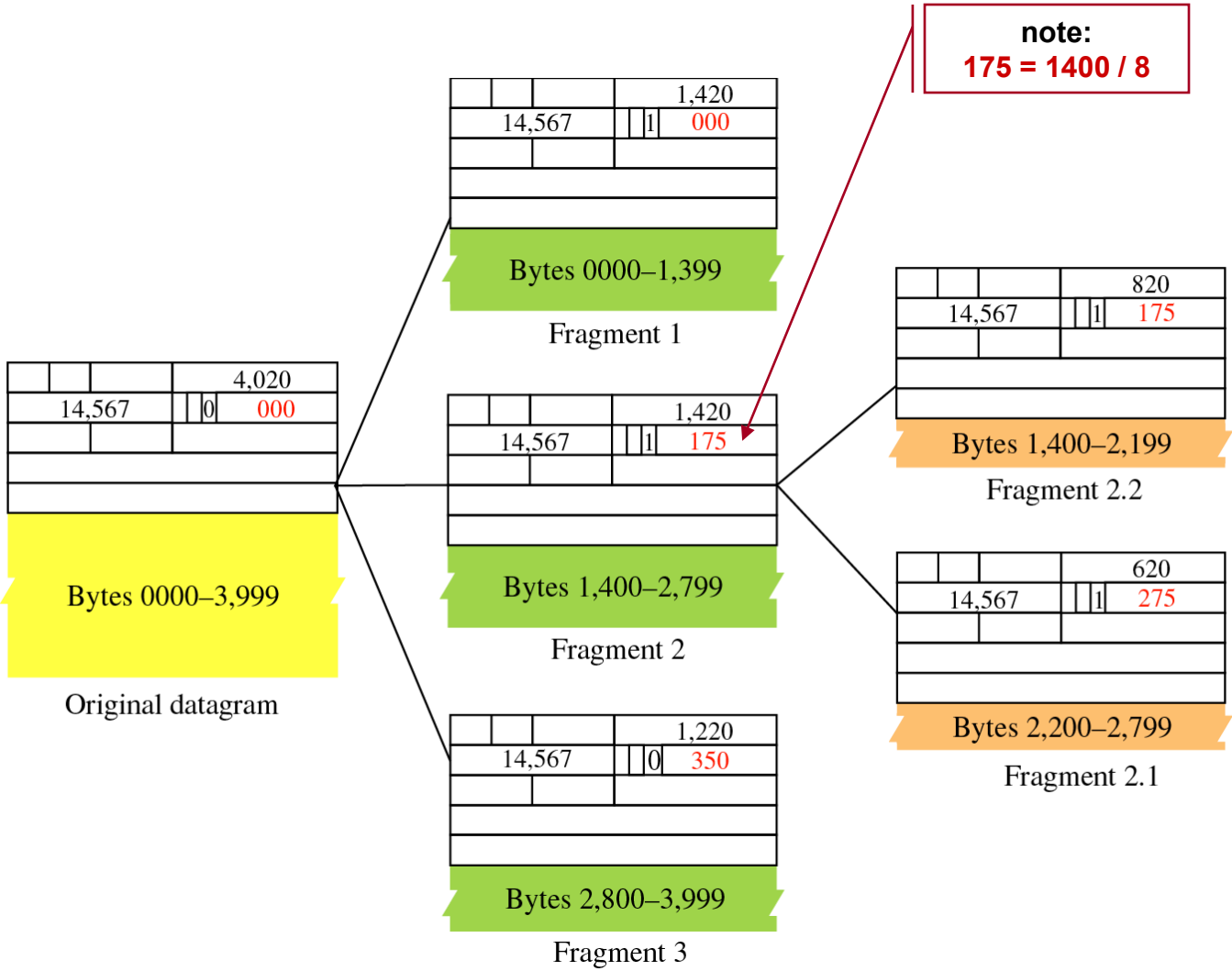
Example

- **4000 byte datagram**
- **MTU = 1500 bytes**



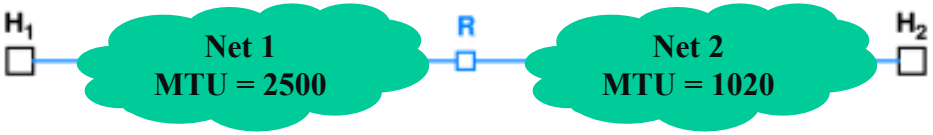
IP Datagram Fragmentation (cont.)

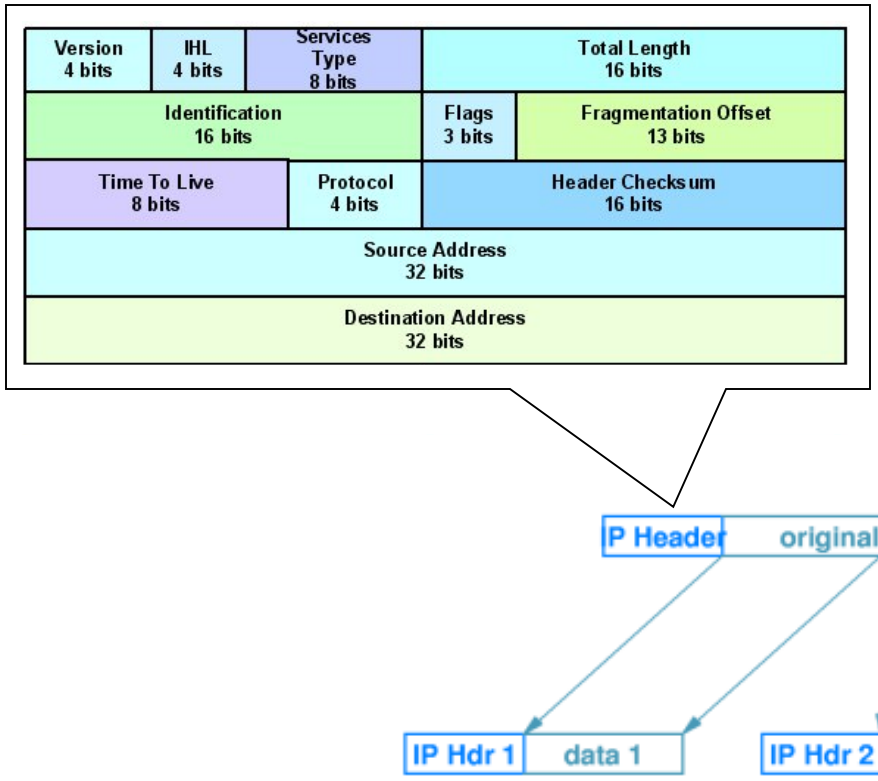
Example [fragmentation of a fragment]

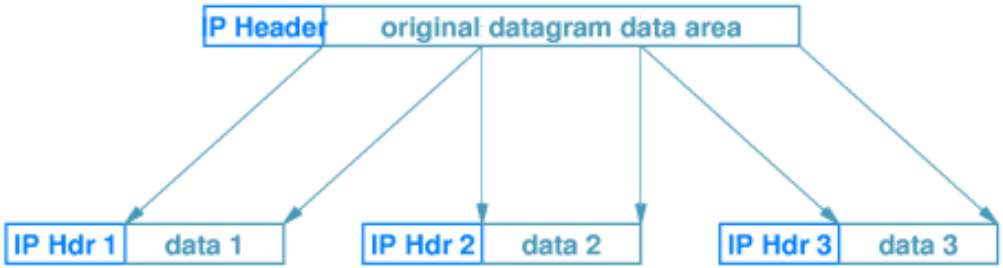


IP Datagram Fragmentation (cont.)

Example [fragmentation at router R]

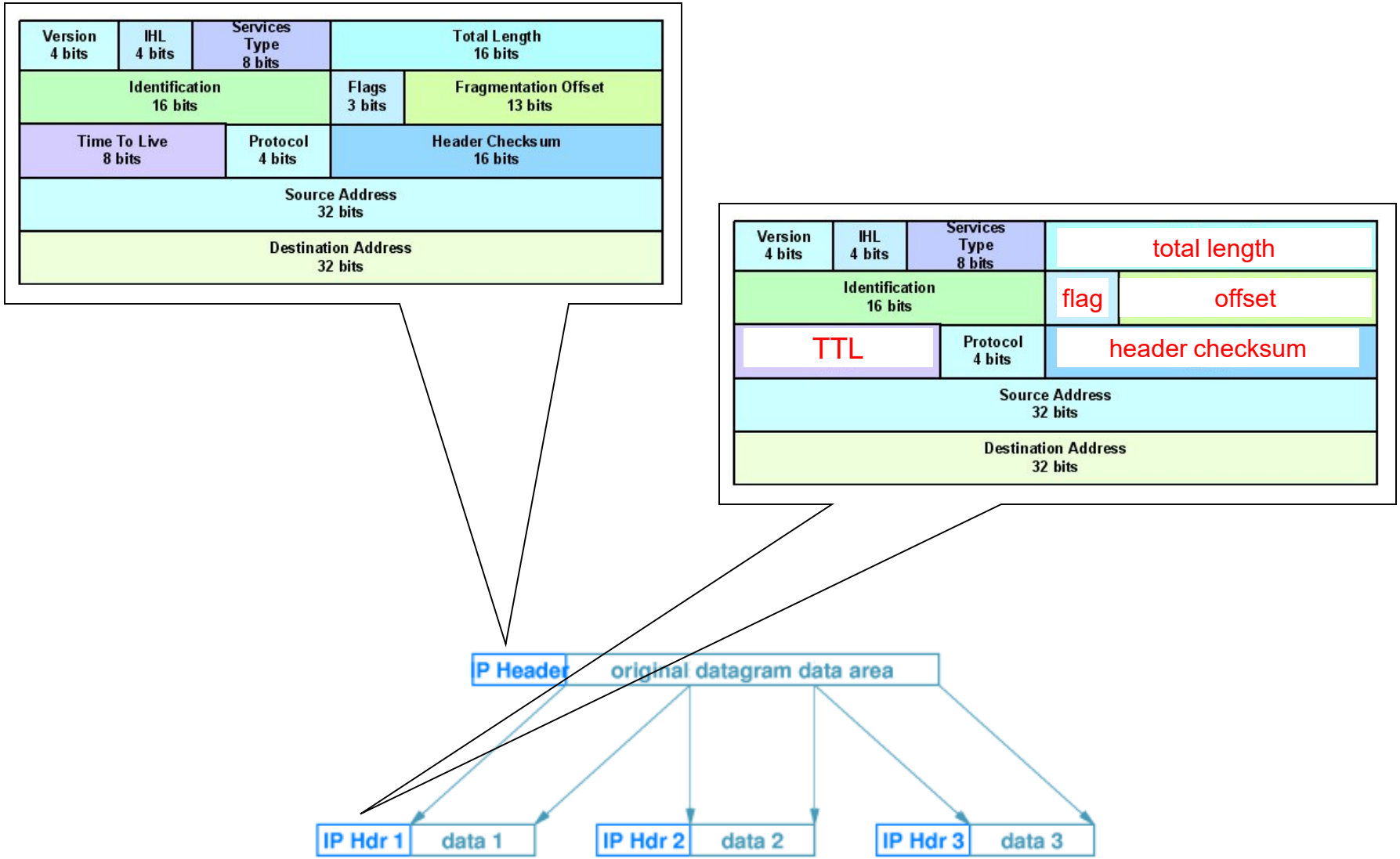




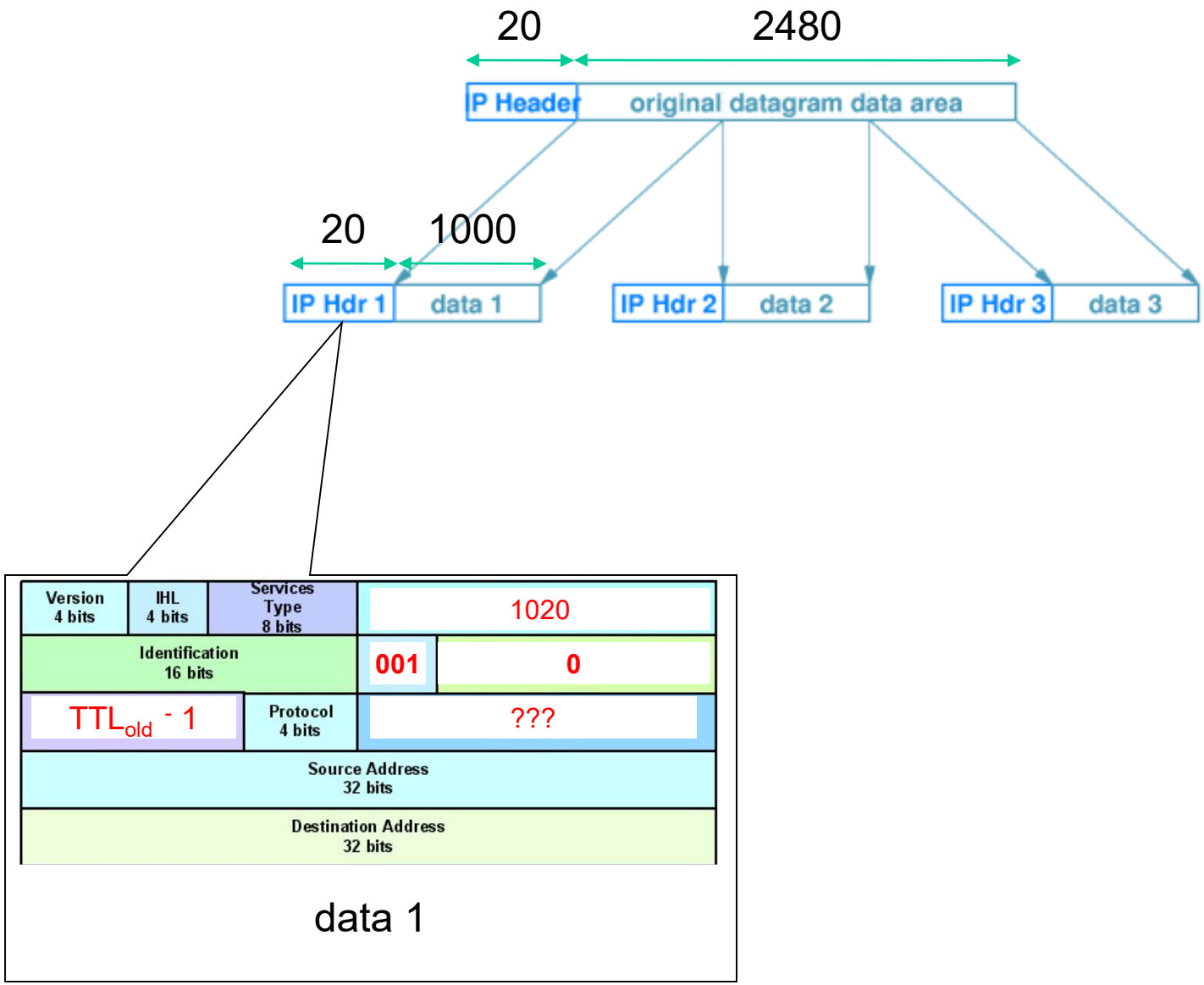


http://penguin.dcs.bbk.ac.uk/academic/networks/network-layer/fragmentation/index.php

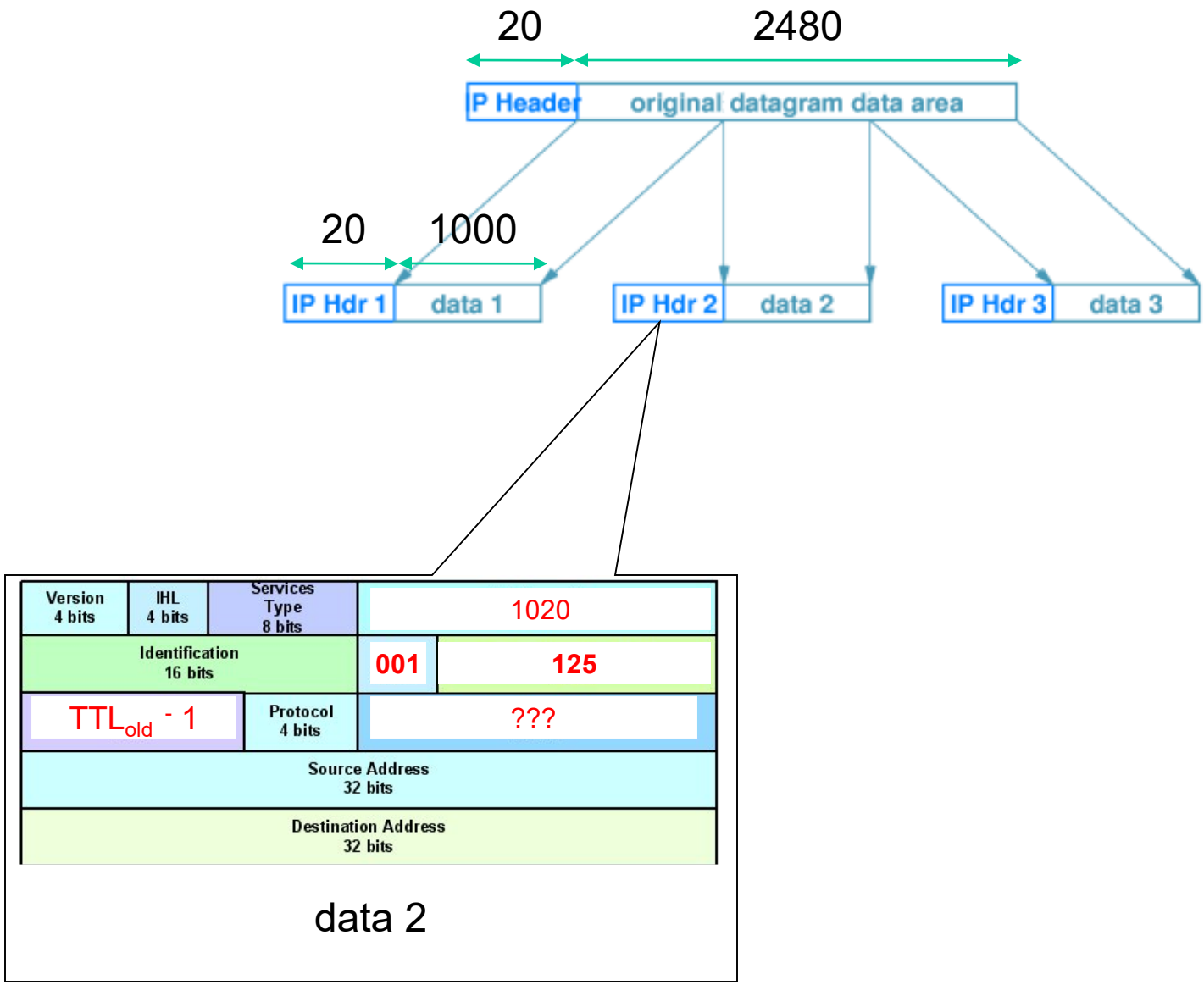
Example [fragmentation at router R: old vs. new headers – what changes?]



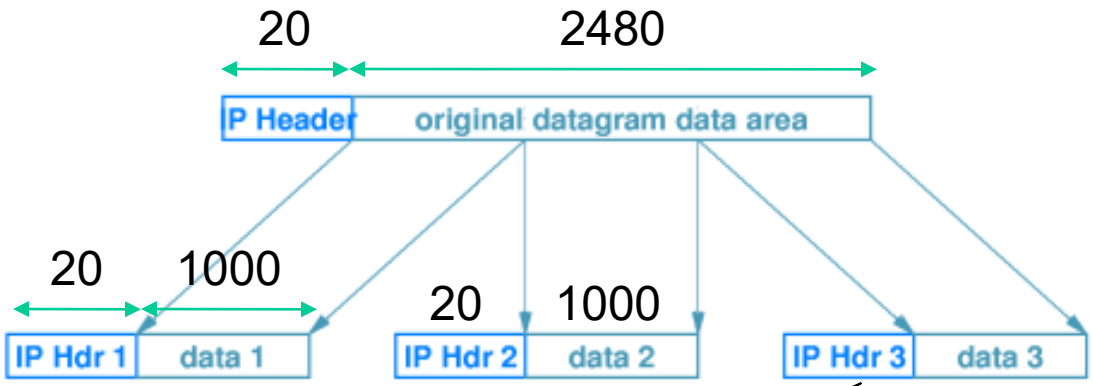
Example [fragmentation at router R: fragment 1]

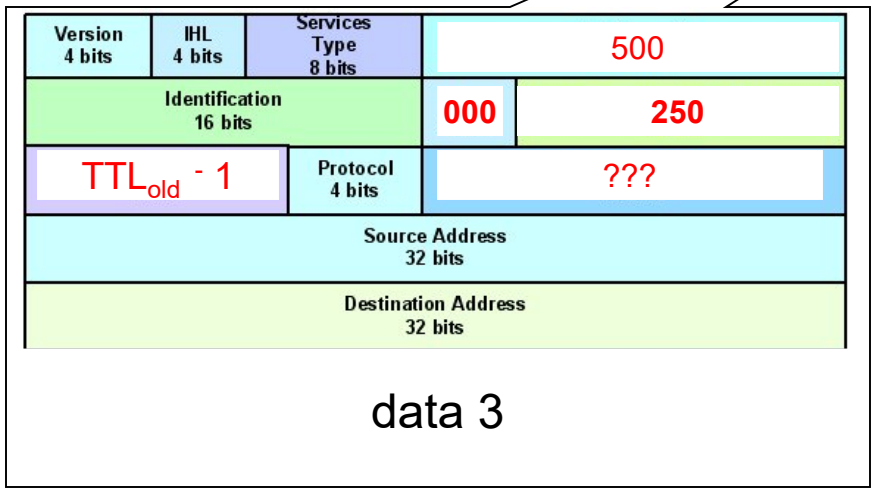


Example [fragmentation at router R: fragment 2]



Example [fragmentation at router R: fragment 3]



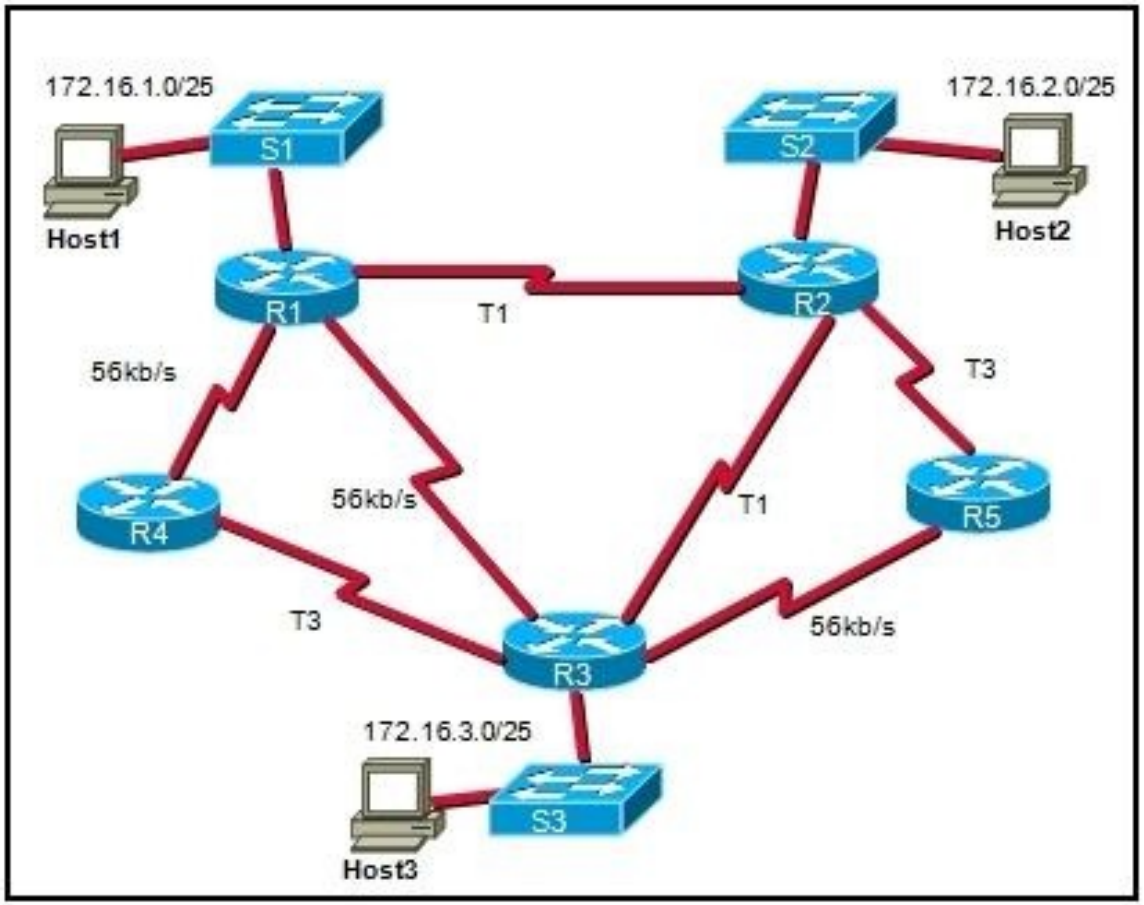


IP Datagram Fragmentation (cont.)

Example **[fragmentation at routers ?!]**

Assume an intermediate router (e.g., **R4**) has volunteered to do packet de-fragmentation on behalf of end hosts.

Would that be a good idea?! If yes, why? If not, why?



Answer:

No, it would not be a good idea:

1) Fragments may not all arrive at the same router.
2) Reassembled packet may be fragmented again.
3) Would take up too much time. Router are supposed to be fast.

IP Datagram Fragmentation (cont.)

A point to remember from class discussion …

big original IP packets → fragmentation may be required → higher processing load on routers

Wouldn't, then, creating small IP packets at the source remove the need for (i.e., complexities related to) fragmentation altogether??

(many) small original IP packets → each packet has an IP header that has to be checked and processed by routers → higher processing load on routers

Exercise

1. **A packet has arrived with Flag's M bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?**

2. **A packet has arrived with an *M* bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?**

3. **A packet has arrived with an *M* bit value of 1 and a fragmentation offset value of zero. Is this the first fragment, the last fragment, or a middle fragment?**

4. **A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?**

5. **A packet has arrived in which the offset value is 100, the value of HLEN is 5 and the value of the total length field is 100. What is the number of the first byte and the last byte?**

1. If the *M* bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A nonfragmented packet is considered the last fragment.

2. If the *M* bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). However, we can definitely say the original packet has been fragmented because the *M* bit value is 1.

3. Because the *M* bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

4. To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

5. The first byte number is $100 \times 8 = 800$. The total length is 100 bytes and the header length is 20 bytes (5×4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must 879.