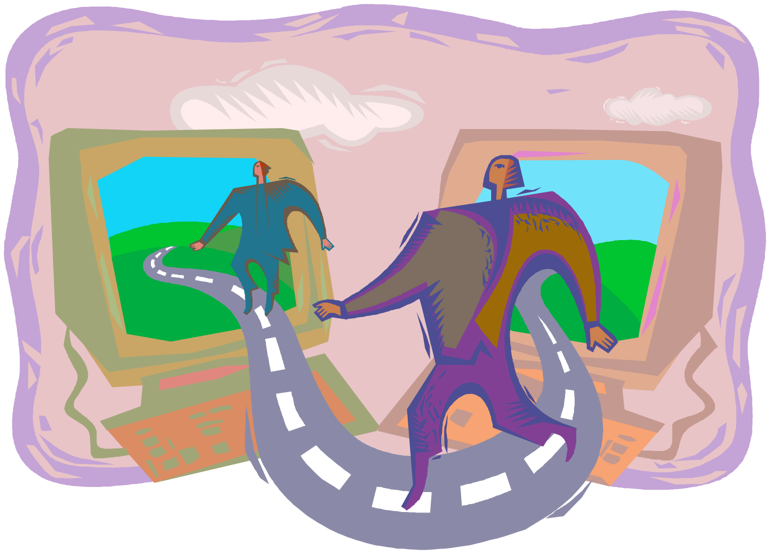
Discussion #11
EE450



-IP Fragmentation

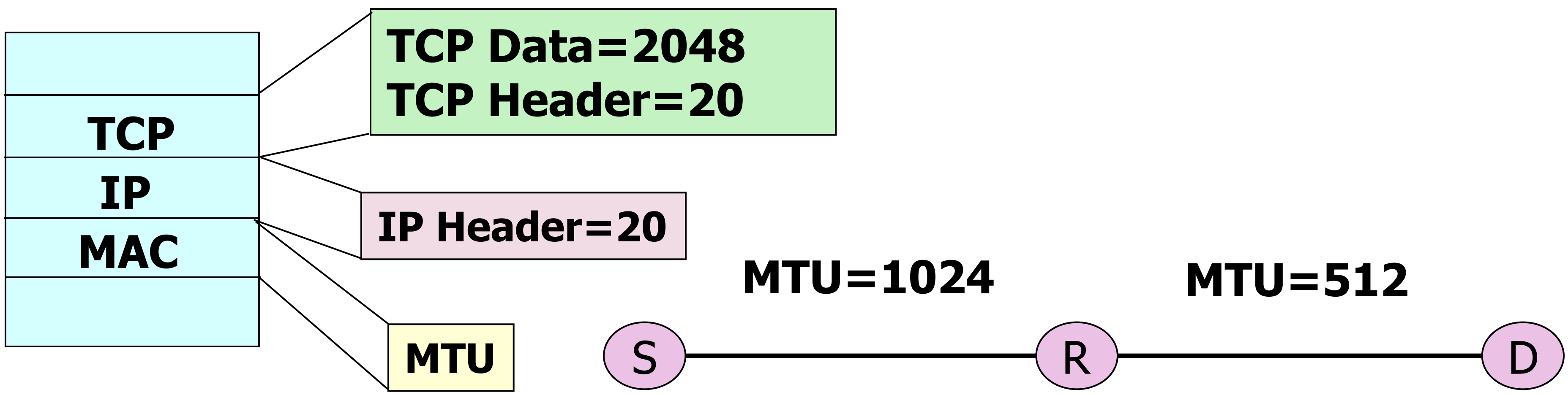
- Addressing and Subnetting

Problem#1: Description

- Suppose that a TCP message that contains 2048 bytes of data and 20 bytes of TCP header is passed to IP for delivery across two networks of the Internet (i.e. from the source host to a router to the destination host).
- The first network has an MTU of 1024 bytes and the second network has an MTU of 512 bytes.
- Each network's MTU gives the total packet size that may be sent including the network header.

Problem#1: Question

- Give the sizes of the fragments delivered to the network layer at the destination host, assuming all IP headers are 20 bytes.



Problem#1: Solution

Across the first network:

- Packets have room for 1024-20=1004 bytes of IP-level data.
- Since 1004 is not multiple of 8, each fragment can contain at most $8 \times \lfloor 1004/8 \rfloor = 1000$ bytes.
- We need to transfer 2048 +20 bytes of such data.
- This would be fragmented into fragments of size 1000,1000 and 68 bytes.
- **For Packet sizes across the first network , add 20 bytes to each fragment.**

Problem#1: Solution

Unfragmented and fragmented datagrams

Start of Header				
Ident=x			0	Offset=0
Rest of Header				
2068 data bytes				

The M bit and the offset field in the header are used for IP fragmentation as illustrated.

Start of Header				
Ident=x			**1**	***Offset=0***
Rest of Header				
1000 data bytes				

Start of Header				
Ident=x			**1**	***Offset=125***
Rest of Header				
1000 data bytes				

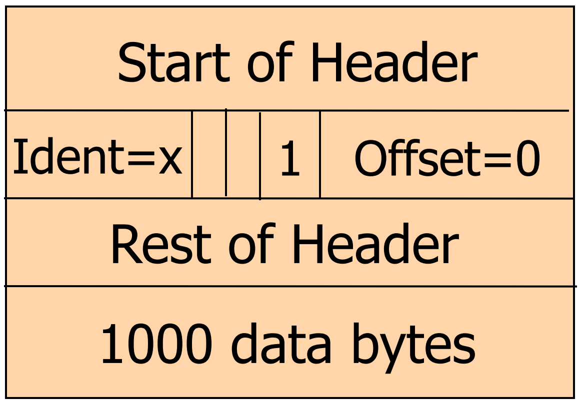
Start of Header				
Ident=x			**0**	***Offset=250***
Rest of Header				
68 data bytes				

Problem#1: Solution

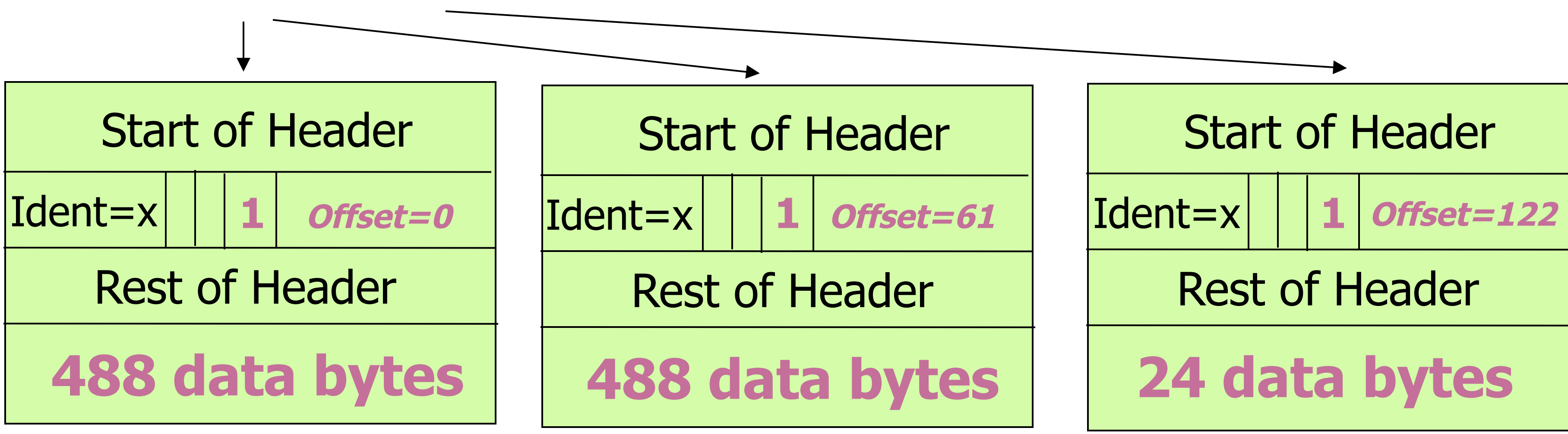
Across the second network:

- Packets have room for 512-20=492 bytes of IP-level data.
- So the 68-byte packet will not be fragmented but the other two 1000-byte packets will be fragmented.
- Since 492 is not multiple of 8, each fragment can contain at most $8 \times \lfloor 492/8 \rfloor = 488$ bytes.
- Each 1000-byte fragment would be fragmented into fragments of size 488,488 and 24 bytes.
- **For Packet sizes across the second network, add 20 bytes to each fragment.**

Problem#1: Solution
Unfragmented and fragmented datagrams



The M bit and the offset field in the header are used for IP fragmentation as illustrated.



Problem#1: Solution

Unfragmented and fragmented datagrams

Start of Header				
Ident=x			1	Offset=125
Rest of Header				
1000 data bytes				

The M bit and the offset field in the header are used for IP fragmentation as illustrated.

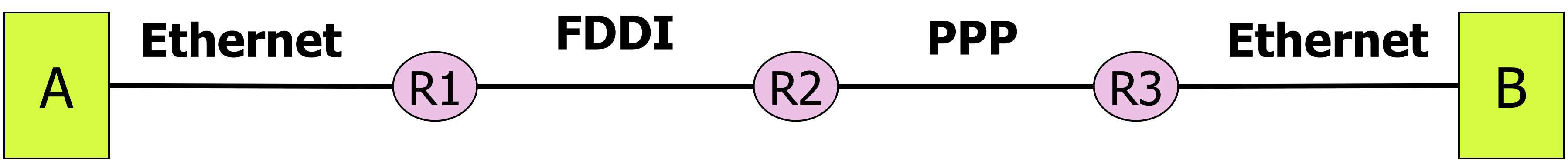
Start of Header				
Ident=x			**1**	***Offset=125***
Rest of Header				
488 data bytes				

Start of Header				
Ident=x			**1**	***Offset=186***
Rest of Header				
488 data bytes				

Start of Header				
Ident=x			**1**	***Offset=247***
Rest of Header				
24 data bytes				

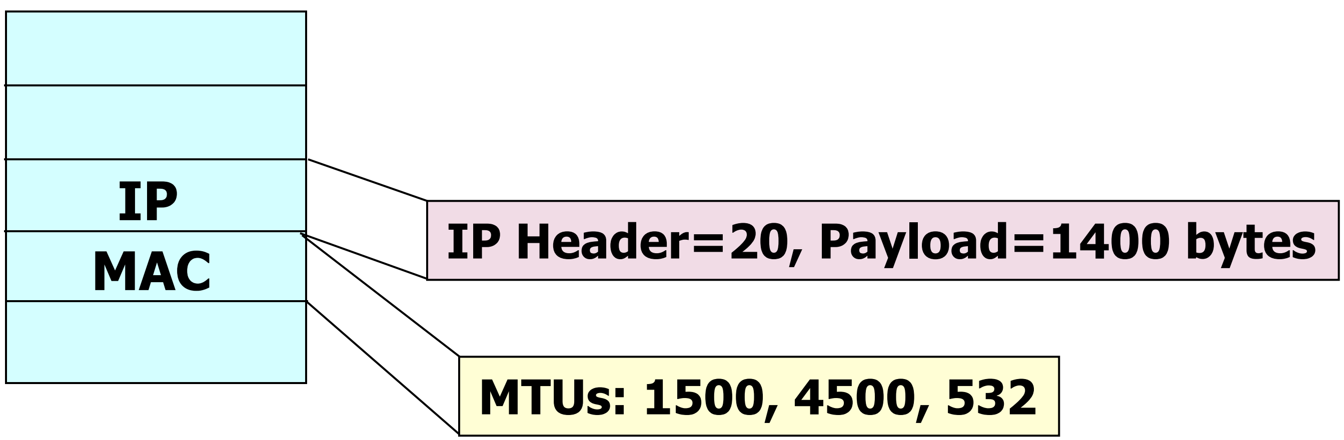
Problem#2: Description

- We have an IP datagram of size 1420 bytes (including the IP header) traversing the sequence of physical networks with different MTUs as follow:
  - Ethernet : MTU = 1500 bytes
  - FDDI : MTU = 4500 bytes
  - PPP : MTU = 532 bytes



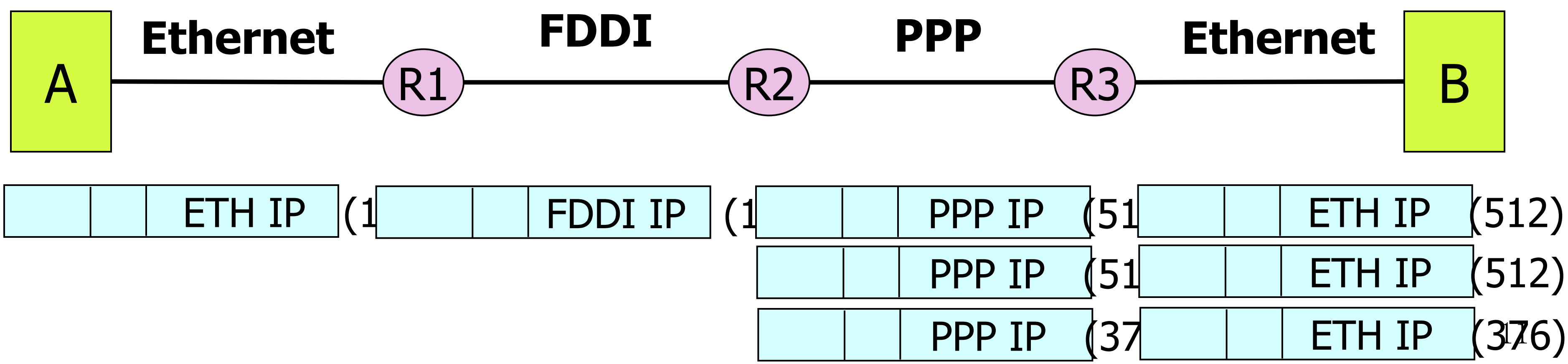
Problem#2: Question

- Show the unfragmented datagram as well as the fragmented datagrams and clearly identify the Header fields used in IP fragmentation.



Problem#2: Solution

- The datagram will not be fragmented across the first two networks.
- Since the MTU of the PPP network is less than the size of the datagram (without the IP header), it will be fragmented to 3 datagrams.
- The fragments of the original datagram travel across the 4^{th} network with no further fragmentation and they will reassembled at the endpoint.



Problem#2: Solution

Unfragmented and fragmented datagrams

The M bit and the offset field in the header are used for IP fragmentation as illustrated.

Start of Header				
Ident=x			0	Offset=0
Rest of Header				
1400 data bytes				

Start of Header				
Ident=x			**1**	***Offset=0***
Rest of Header				
512 data bytes				

Start of Header				
Ident=x			**1**	***Offset=64***
Rest of Header				
512 data bytes				

Start of Header				
Ident=x			**0**	***Offset=128***
Rest of Header				
376 data bytes				

Discussion 11

Chapter 7 (7^{th} edition), # 6, 7, 8

Problem 6

In step 4 of the CSMA/CA protocol, a station that successfully transmits a frame begins the CSMA/CA protocol for a second frame at step 2, rather than at step 1. What rationale might the designers of CSMA/CA have had in mind by having such a station not transmit the second frame immediately (if the channel is sensed idle)?

Suppose that wireless station H1 has 1000 long frames to transmit. (H1 may be an AP that is forwarding an MP3 to some other wireless station.) Suppose initially H1 is the only station that wants to transmit, but that while half-way through transmitting its first frame, H2 wants to transmit a frame. For simplicity, also suppose every station can hear every other station's signal (that is, no hidden terminals). Before transmitting, H2 will sense that the channel is busy, and therefore choose a random back off value.

Now suppose that after sending its first frame, H1 returns to step 1; that is, it waits a short period of times (DIFS) and then starts to transmit the second frame. H1's second frame will then be transmitted while H2 is stuck in back off, waiting for an idle channel. Thus, H1 should get to transmit all of its 1000 frames before H2 has a chance to access the channel. On the other hand, if H1 goes to step 2 after transmitting a frame, then it too chooses a random back off value, thereby giving a fair chance to H2. Thus, fairness was the rationale behind this design choice.

Problem 7

Suppose an 802.11b station is configured to always reserve the channel with the RTS/CTS sequence. Suppose this station suddenly wants to transmit 1,000 bytes of data, and all other stations are idle at this time. As a function of SIFS and DIFS, and ignoring propagation delay and assuming no bit errors, calculate the time required to transmit the frame and receive the acknowledgment.

A frame without data is 32 bytes long. Assuming a transmission rate of 11 Mbps, the time to transmit a control frame (such as an RTS frame, a CTS frame, or an ACK frame) is (256 bits)/(11 Mbps) = 23 usec. The time required to transmit the data frame is (8256 bits)/(11 Mbps) = 751

DIFS + RTS + SIFS + CTS + SIFS + FRAME + SIFS + ACK

= DIFS + 3SIFS + (3*23 + 751) usec = DIFS + 3SIFS + 820 usec