Solutions

1)

a. $\log_2 500 = 8.95$ Extra 1s = 9 Possible subnets: **512** Mask: **/17** (8+9)

b. $2^{32-17} = 2^{15} =$ **32,768** Addresses per subnet

c. **Subnet 1:** The first address in the this address is the beginning address of the block or **16.0.0.0**. To find the last address, we need to write 32,767 (one less than the number of addresses in each subnet) in base 256 (0.0.127.255) and add it to the first address (in base 256).

First address in subnet 1:	**16**	.	**0**	.	**0**	.	**0**
Number of addresses:	**0**	.	**0**	.	**127**	.	**255**
Last address in subnet 1:	**16**	.	**0**	.	**127**	.	**255**

d. **Subnet 500:**

Note that the subnet 500 is not the last possible subnet; it is the last subnet used by the organization. To find the first address in subnet 500, we need to add 16,351,232 (499 × 32678) in base 256 (0. 249.128.0) to the first address in subnet 1. We have 16.0.0.0 + 0.249.128.0 = **16.249.128.0**. Now we can calculate the last address in subnet 500.

First address in subnet 500:	**16**	.	**249**	.	**128**	.	**0**
Number of addresses:	**0**	.	**0**	.	**127**	.	**255**
Last address in subnet 500:	**16**	.	**249**	.	**255**	.	**255**

2)

a. The number of address in this block is $2^{32-29} = 8$. We need to add 7 (one less) addresses (0.0.0.7 in base 256) to the first address to find the last address.

From:	123	.	56	.	77	.	32
	0	.	0	.	0	.	7
To:	123	.	56	.	77	.	39

b. The number of address in this block is $2^{32-27} = 32$. We need to add 31 (one less) addresses (0.0.0.31 in base 256) to the first address to find the last address.

From:	200	.	17	.	21	.	128
	0	.	0	.	0	.	31
To:	200	.	17	.	21	.	159

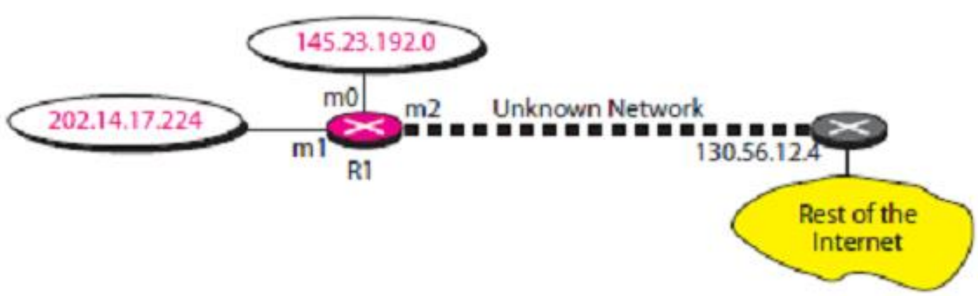
c. The number of address in this block is $2^{32-23} = 512$. We need to add 511 (one less) addresses (0.0.1.255 in base 256) to the first address to find the last address.

From:	17	.	34	.	16	.	0
	0	.	0	.	1	.	255
To:	17	.	34	.	17	.	255

d. The number of address in this block is $2^{32-30} = 4$. We need to add 3 (one less) addresses (0.0.0.3 in base 256) to the first address to find the last address.

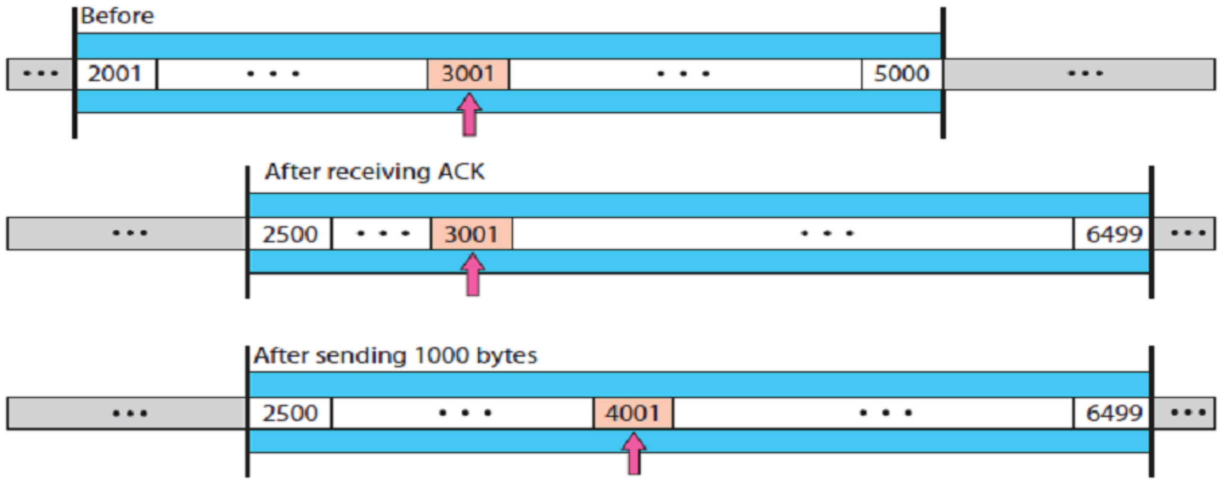
From:	180	.	34	.	64	.	64
	0	.	0	.	0	.	3
To:	180	.	34	.	64	.	67

3)

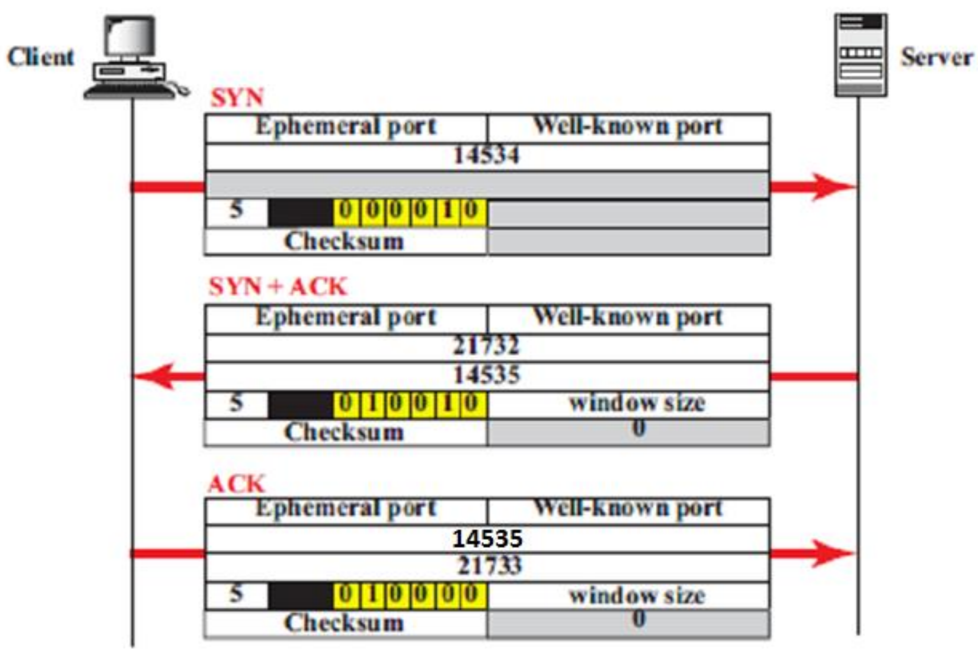


4)

—



5)



Note: The randomly generated port numbers for the client side are referred to as the ephemeral port numbers. ***Ephemeral*** is defined as short-lived or transitory. Ephemeral port numbers are only used for the duration of a single communication between client and server, so they are indeed short-lived.

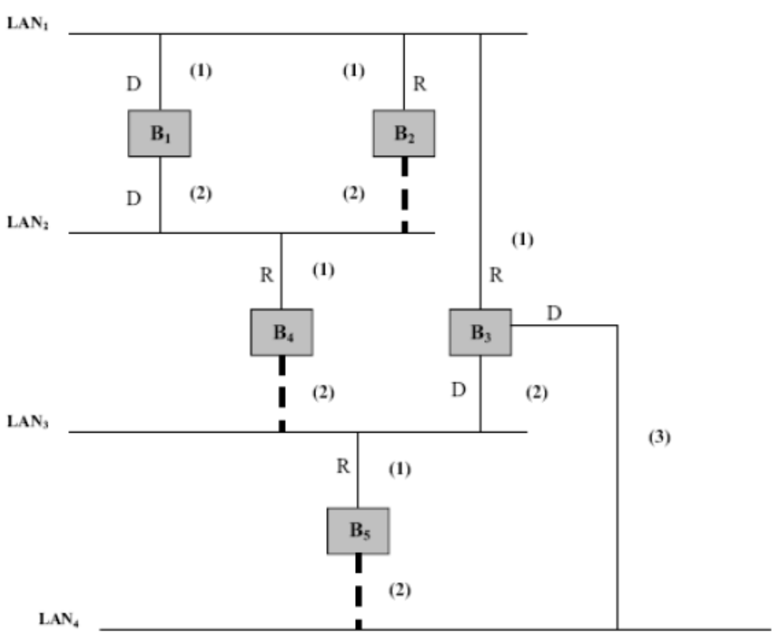
6)

M	offset	bytes data	source
1	0	360	1st original fragment
1	45	152	1st original fragment
1	64	360	2nd original fragment
1	109	152	2nd original fragment
1	128	360	3rd original fragment
0	173	16	3rd original fragment

If fragmentation had been done originally for this MTU, there would be four fragments. The first three would have 360 bytes each; the last would have 320 bytes.

7)

The spanning tree is as follows:



8) Any IP address in range 101.101.101.65 to 101.101.101.126

Four equal size subnets:

101.101.128/19
101.101.160/19
101.101.192/19
101.101.224/19

9)

Step	N'	D(s),p(s)	D(t),p(t)	D(u),p(u)	D(v),p(v)	D(w),p(w)	D(y),p(y)	D(z),p(z
0	X	∞	∞	∞	3,x	6,x	6,x	∞
1	Xv	∞	7,v	6,v	3,x	6,x	4,v	∞
2	Xvy	∞	7,v	6,v	3,x	6,x	4,v	18,y
3	Xvyu	10,u	7,v	6,v	3,x	6,x	4,y	18,y
4	Xvyuw	10,u	7,v	6,v	3,x	6,x	4,y	18,y
5	Xvyuwt	8,t	7,v	6,v	3,x	6,x	4,y	12,t
6	xvyuwts	8,t	7,v	6,v	3,x	6,x	4,y	12,t
7	xvyuwtsz	8,t	7,v	6,v	3,x	6,x	4,y	12,t

10) 230m / (2.3×108m/sec) = 1 µs; which means at 16Mbps the propagation delay is 16 bit delay. The five stations add another five bit delay, therefore the total ring latency is 16+5=21 bit delay. Token is 3 bytes, which results in 24 bit delay, therefore for ring latency to match the transmission delay, monitor needs to add an additional 24-21= 3 bit delay. Similary at 4Mbps the monitor needs to add 24 − (4 + 5) = 15 additional bit delay.

11)

(a)

Information Stored at Node	Distance to Reach Node					
	A	B	C	D	E	F
A	0	∞	3	8	∞	∞
B	∞	0	∞	∞	2	∞
C	3	∞	0	∞	1	6
D	8	∞	∞	0	2	∞
E	∞	2	1	2	0	∞
F	∞	∞	6	∞	∞	0

(b)

Information Stored at Node	Distance to Reach Node					
	A	B	C	D	E	F
A	0	∞	3	8	4	9
B	∞	0	3	4	2	∞
C	3	3	0	3	1	6
D	8	4	3	0	2	∞
E	4	2	1	2	0	7
F	9	∞	6	∞	7	0

(c)

Information Stored at Node	Distance to Reach Node					
	A	B	C	D	E	F
A	0	6	3	6	4	9
B	6	0	3	4	2	9
C	3	3	0	3	1	6
D	6	4	3	0	2	9
E	4	2	1	2	0	7
F	9	9	6	9	7	0

12)

D	Confirmed	Tentative
1.	(D,0,-)	
2.	(D,0,-)	(A,8,A)
		(E,2,E)
3.	(D,0,-)	(A,8,A)
	(E,2,E)	(B,4,E)
		(C,3,E)
4.	(D,0,-)	(A,6,E)
	(E,2,E)	(B,4,E)
	(C,3,E)	(F,9,E)
5.	(D,0,-)	(A,6,E)
	(E,2,E)	(F,9,E)
	(C,3,E)	
	(B,4,E)	
6.	previous + (A,6,E)	
7.	previous + (F,9,E)	

13)

Destination Address	Range Link Interface
10000000 through (64 addresses) 10111111	0
11000000 through(32 addresses) 11011111	1
11100000 through (32 addresses) 11111111	2
00000000 through (128 addresses) 01111111	3

14) Subnet 2: 223.1.17.0/25
Subnet 1:223.1.17.128/26
Subnet 3: 223.1.17.192/28

15) **NOT in Midterm II** Wait for 51,200 bit times. For 10 Mbps, this wait is $51.2*10^3$ bits /10 * 10^6bps = 5.12msec
For 100 Mbps, the wait is 512 µ sec.

16) **Not in Midterm II**

At t = 0 A transmits. At t = 576 , A would finish transmitting. In the worst case, B begins transmitting at time t=324, which is the time right before the first bit of A's frame arrives at B. At time t=324+325=649 B 's first bit arrives at A . Because 649> 576, A finishes transmitting before it detects that B has transmitted. So A incorrectly thinks that its frame was successfully transmitted without a collision.

17)

a) 172.20.159.254
b) 1024 subnets and 62 hosts
c) 16 subnets and 4094 hosts
d) 172.26.52.254
e) 192.168.189.191
f) 128 subnets and 510 hosts
g) 172.26.156.0
h) 172.16.112.0
i) 172.24.5.255
j) 4096 subnets and 4094 hosts
k) 172.22.114.1
l) 192.168.113.0
m) 172.29.215.1
n) 172.25.42.255
o) 172.25.174.1 through to 172.25.175.254
p) 64 subnets and 1022 hosts
q) 16 subnets and 4094 hosts
r) 256 subnets and 254 hosts
s) 16 subnets and 14 hosts
t) 10.200.80.0
u) 172.27.120.1 through to 172.27.121.254
v) 172.20.11.255
w) 172.30.88.0
x) 192.168.27.128
y) 172.23.105.255

18) a)

Group 1:
1st customer: 170.128.0.0/24
2nd customer: 170.128.1.0/24
…
200th customer: 170.128.199.0/24

Group 2:
1st customer: 170.128.200.0/25
2nd customer: 170.128.200.128/25
…
112th customer: 170.128.255.128/25

Group 3:
1st customer: 170.129.0.0/26
2nd customer: 170.129.0.64/26
…
100th customer: 170.129.24.192/26

b) Total Addresses: $2^{32-12} = 2^{20}$.
Addresses Used: 200*256+112*128+64*100 = 51200+14336+6400 = 71936.
Remaining: $2^{20} – 71936 = 976640$.

19)

Subnets No.	Network Address	Direct Broadcast Address	Range of Host IP
0	183.120.0.0	183.120.0.255	183.120.0.1 to 183.120.0.254
15	183.120.15.0	183.120.15.255	183.120.15.1 to 183.120.15.254
31	183.120.31.0	183.120.31.255	183.120.31.1 to 183.120.31.254

20)

Bi	*Interface*	*dst*	*Interface*	*dst*	*Interface*	*dst*
B1	A-interface	A	B2-interface	C		
B2	B1-interface	A	B3-interface	C	B4-interface	D
B3	B2-interface	A,D	C-interface	C		
B4	B2-interface	A	D-interface	D		

21)

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.**

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.**

22) With longest prefix match, going through the routing table from top to down, the first subnet number which matches the given subnet and its corresponding next hop is the chosen one.

(a) 128.96.171.92. Applying the subnet mask 255.255.254.0.

128. 96. **10101011** . **01011100**

AND

255.255. **11111110** . **00000000**

---

128. 96. **10101010** . **00000000** = 128.96.170.0 Next hop: **interface0**

(b) 128.96.167.151. Applying the subnet masks 255.255.254.0

128. 96. **10100111** . **10010111**

AND

255.255. **11111110** . **00000000**

---

128. 96. **10100110** . **00000000**

= 128.96.166.0 Next hop: **R2**

(c) 128.96.163.151

None of the subnet entries match, hence use default router **R4.**

(d) 128.96.169.192. Applying the subnet mask 255.255.254.0.

128. 96. **10101001** . **11000000**

AND

255.255. **11111110** . **00000000**

128. 96. **10101000** . **00000000** = 128.96.168.0 Next hop: **interface1**

(e) 128.96.165.121. Apply the subnet mask 255.255.254.0, no one within the first three rows matches. Then applying the subnet mask 255.255.252.0,

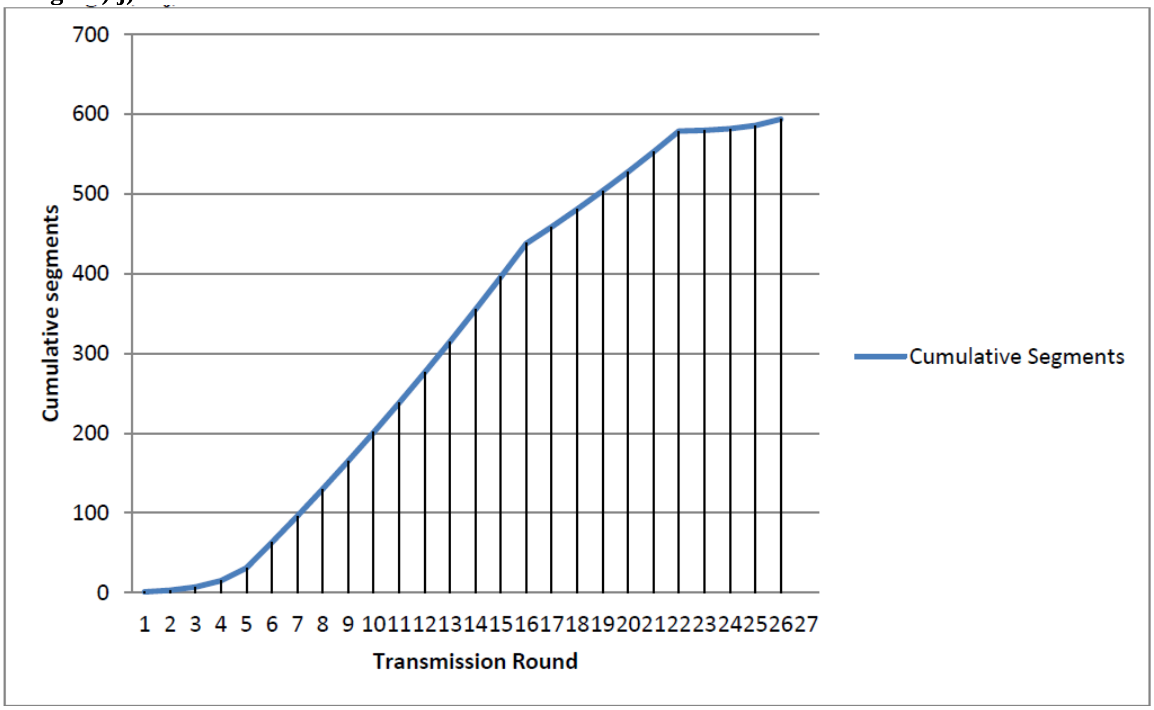
128. 96. **10100101** . **01111001**

AND

255.255. **11111100** . **00000000**

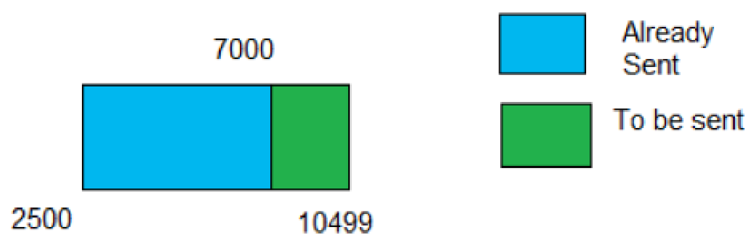
128. 96. **10100100** . **00000000** = 128.96.164.0 Next hop: **R3**

23) **a) RTT number cannot be 0, b) 1 to 6 and 23 to 26, c) 6 to 16 and 17 to 22, d) Triple Duplicate Ack e) Timeout, f) 32, g) 21, h) 13, i) ssthresh = 6 and cwnd = 6 (The cwnd size at round 27 before the triple duplicate ACK was 13. So after the triple duplicate ACK the new threshold and cwnd will be 13/2 i.e. 6 after rounding off) j)**

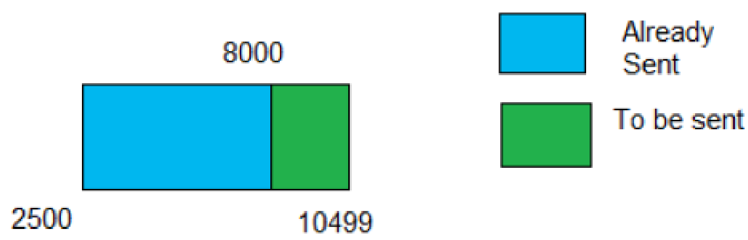


k) 7th l) Same as (j) just replace the segments with bytes after multiplying number of segments by 2KB.

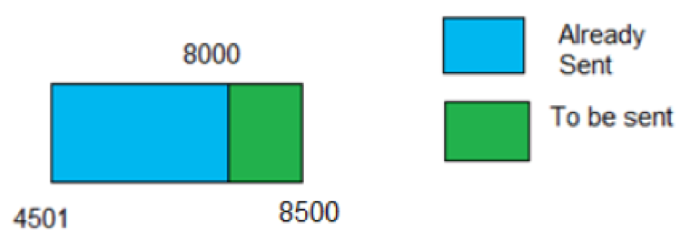
24) i)


7000
2500
10499
Already Sent
To be sent

ii)


8000
2500
10499
Already Sent
To be sent

iii)


8000
4501
8500
Already Sent
To be sent