HW4 - Problems From The Network and Link Layers

Class	CS 408
Type	Homework
Property	

Berna Yıldıran 26431

▼ Question #1:

1) (10 points) A certain router receives a datagram of size 7000 B (including the header). However, all of its output ports have an MTU (maximum transfer unit) of 1500 B, thus the original datagram should be fragmented into several smaller datagrams when forwarded. For each of the output fragments write the following fields of the IP header: length, ID, flag and offset! Assume 20 B for the IP header.

▼ Answer:

7000B datagram

MTU = 1500B

IP Header = 20B

1. Length= 7000 / ID= x / Flag= 0 / Offset = 0
2. Length= 1500 / ID= x / Flag= 1 / Offset = 0

((1500 - 20) + (1500 - 20)) / 8 = 185 (Header 20B)

3. Length= 1500 / ID= x / Flag= 1 / Offset = 185

((1500 - 20) + (1500 - 20) + (1500 - 20)) / 8 = 370

4. Length= 1500 / ID= x / Flag= 1 / Offset = 370

((1500 - 20) + (1500 - 20) + (1500 - 20) + (1500 - 20)) / 8 = 555

5. Length= 1500 / ID= x / Flag= 1 / Offset = 555

((1500 - 20) + (1500 - 20) + (1500 - 20) + (1500 - 20) + (1500 - 20)) / 8 = 740

(1500 - 20)*4 = 5920 (Header 20B)

6. Length= 1080 / ID= x / Flag= 0 / Offset = 740

▼ Question #2:

2) (10 points) A certain autonomous system (AS) is assigned a subnet which to the outside world (Internet) is advertised as 24.23.5.0 /24. Each router has its number of hosts attached to it, which includes the router interface on that particular side of the subnet. As a network administrator, your job is to properly and efficiently do the subnetting inside this AS!
(**Note:** for subnets with 2 interfaces we suggest to also include the network and broadcast domain)
(**Hint:** start with the largest subnets first)

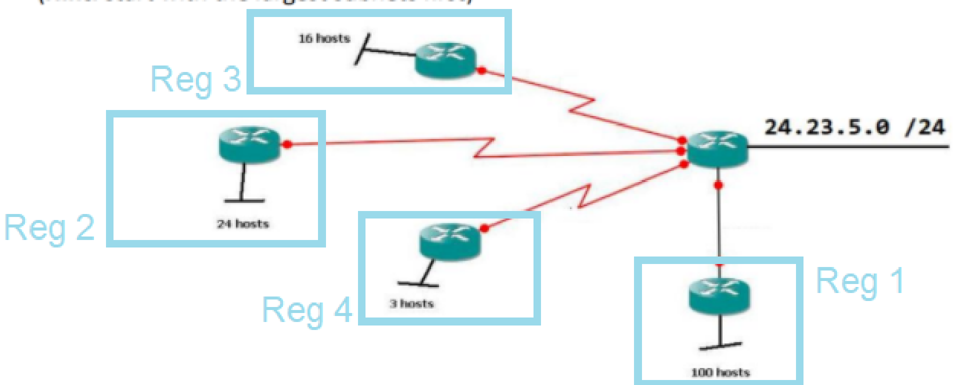


Fig.1. An autonomous system (AS) with an assigned subnet of 24.23.5.0 /24.

▼ Answer:

<u>Reg1</u>

Reg1 → Na + $R_1 i_1$ + 2 (Broadcast and Network)

Reg1 → 100 + 1 + 2 = 103

$2^6 < 2^7$ → Therefore, 103 will be round up to 2^7

Thus, Subnet mask = 32 - 7 → 25 Hosts → 7

24.23.5.0/25 (Subnet)

24.23.5.0 (Network)

24.23.5.127 (Broadcast)

<u>Reg2</u>

Reg2 → Na + $R_1 i_1$ + 2 (Broadcast and Network)

Reg2 → 24 + 1 + 2 = 27

$2^4 < 2^5$ → Therefore, 27 will be round up to 2^5

Thus, Subnet mask = 32 - 5 → 27 Hosts → 5

24.23.5.128 /27 (Subnet)

24.23.5.128 (Network)

24.23.5.159 (Broadcast)

Reg3

Reg3 → Na + $R_1 i_1$ + 2 (Broadcast and Network)

Reg3 → 16 + 1 + 2 = 19

$2^4 < 2^5$ → Therefore, 19 will be round up to 2^5

Thus, Subnet mask = 32 - 5 → 27 Hosts → 5

24.23.5.160 /27 (Subnet)

24.23.5.160 (Network)

24.23.5.191 (Broadcast)

Reg4

Reg4 → Na + $R_1 i_1$ + 2 (Broadcast and Network)

Reg4 → 3 + 1 + 2 = 103

$2^2 < 2^3$ → Therefore, 6 will be round up to 2^3

Thus, Subnet mask = 32 - 3 → 29 Hosts → 3

24.23.5.192 /29 (Subnet)

24.23.5.192 (Network)

24.23.5.199 (Broadcast)

Router Interface Reg1 - R → 24.23.5.200/30

24.23.5.200 (Network)

24.23.5.203 (Broadcast)

Router Interface Reg2 - R → 24.23.5.204/30

24.23.5.204 (Network)

24.23.5.207 (Broadcast)

Router Interface Reg3 - R → 24.23.5.208/30

24.23.5.208 (Network)

24.23.5.211 (Broadcast)

Router Interface Reg4 - R → 24.23.5.212/30

24.23.5.212 (Network)

24.23.5.215 (Broadcast)

▼ Question #3:

3) (10 points) Inside your AS you have the 6 subnets given bellow. Summarize them so when advertised to the outside world you'll have as least entries (subnets) to advertise as possible. Of course, avoid over-summarization.

243.157.50.0 /22; 243.157.54.0 /22; 243.157.58.0 /22; 243.157.62.0 /22; 243.157.10.0 /22 243.157.42.0 /22

▼ Answer:

Subnet is 22 → Therefore, 32 - 22 = 10, last 10 indexes represents host, because of that they are all zero.

1. 243.157.62.0 /22

   243.157.001111**10.000000**

   This will become:

   243.157.60.0 /22

   243.157.001111**00.000000**

3. 243.157.58.0 /22

   243.157.001100**00.000000**

   This will become:

243.157.56.0 /22

243.157.001110**00.000000**

4. 243.157.54.0 /22

   243.157.001100**00.000000**

   This will become:

   243.157.52.0 /22

   243.157.001101**00.000000**

4. 243.157.50.0 /22

   243.157.001100**10.000000**

   This will become:

   243.157.48.0 /22

   243.157.001100**00.000000**

6. 243.157.42.0 /22

   243.157.000010**10.000000**

   This will become:

   243.157.40.0 /22

   243.157.001010**00.000000**

5. 243.157.10.0 /22

   243.157.000010**10.000000**

   This will become:

   243.157.8.0 /22

   243.157.000010**00.000000**

- 1 and 2 merged → 243.157.8.0/22 (Advertised)

- (A) 3 and 4 merged → 243.157.48.0/22
- (B) 5 and 6 merged → 243.157.56.0/22

- (A) and (B) merged → 243.157.50.0/22 (Advertised)

Final Summarization:

243.157.0.0/22 → Find longest match for 243.157.8.0/22 and 243.157.48.0/22

▼ Question #4:

4) (10 points) In Fig.2 you have the topology of a certain network. Construct the forwarding table for router *u* towards all the other routers using the Dijkstra's algorithm. Show all of the steps!

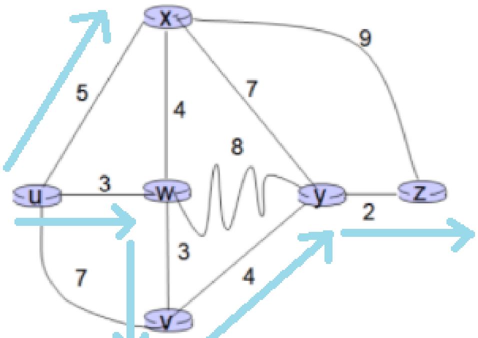


Fig.2. Topology of a certain network

▼ Answer:

Step	N'	D(v)p(v)	D(w)p(w)	D(x)p(x)	D(y)p(y)	D(z)p(z)
0	u	7, u	3, u	5, u	Inf.	Inf.
1	uw	6, w		5, u	11, w	Inf.
2	uwx	6, w			11, w	14, x
3	uwxv				10, v	14, x
4	uwxvy					12, y
5	uwxvyz					

Step #0

u → v = 7

u → w = 3

u → x = 5

There is no direct path from u to y and z, therefore infinity.
Choose shortest path node = w

Step #1

u → w → v = 3 + 3 = 6 < 7

Therefore updated as 6.

u → w → w = There is no path to itself

u → w → x = 3 + 4 = 7 > 5

There is no change.
u → w → y = 3+ 8 = 11 < infinity

Therefore, updated as 11.

There is no path from w → z, therefore infinity.

Follow the shortest path node = u

Step #2

u → w → x → x = There is no path to itself

For w and y, no better path exist.

u → x → z = 5 + 9 = 14 < infinity

Therefore, updated as 14.

Follow the shortest path node = v

Step #3

u → w → v → y = 3 + 3 + 4 = 10 < 11

Therefore, updated as 10.

Follow the smallest path node = y

Step #4

u → w → v → y → z = 3 + 3 + 4 + 2 = 12 < 14

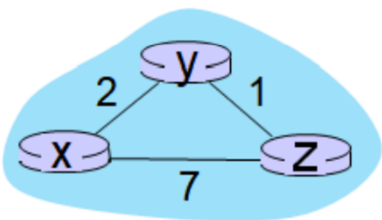
Therefore, updated as 12.

FORWARDING TABLE

Destination	Interface
x	(u, x)
w	(u, w)
v	(u, w)
y	(u, w)
z	(u, w)

▼ Question #5:

5) (15 points) In Fig.3 you have the topology of a certain network. Construct the forwarding table **for all of the routers** towards all the other routers in the network using the Bellman-Ford's algorithm. You can assume that all the routers send their newly computed/changed distance vectors (DV's) values to their neighbor at the same time, i.e. for t=0 all of them have only the DVs for their neighbors, at t=1 they do the first exchange of DVs, at t=2 they do the second exchange of DVs, etc. **Show all of the steps until the algorithm converges for all of the routers** (i.e. there are no more changes in DVs)!
(Note: if everything as done correctly, you shouldn't have more to do more than 3 iterations (including the initialization) for any of the routers.)
(**Hint:** fill-up the all of the tables below corresponding to new DV values after an exchange with the neighbors. The initial tables for all of the routers are given.)



▼ Answer:

Fig.3. Topology of a certain network

	Dx()	X	Y	Z	Dy()	X	Y	Z	Dz()	X	Y	Z
t=0	X	0	2	7	X	∞	∞	∞	X	∞	∞	∞
	Y	∞	∞	∞	Y	2	0	1	Y	∞	∞	∞
	Z	∞	∞	∞	Z	∞	∞	∞	Z	7	1	0
	DV table of X				DV table of Y				DV table of Z			
t=1	Dx()	X	Y	Z	Dy()	X	Y	Z	Dz()	X	Y	Z
	X	0	2	3	X	0	2	7	X	0	2	7
	Y	2	0	1	Y	2	0	1	Y	2	0	1
	Z	7	1	0	Z	7	1	0	Z	3	1	0
	DV table of X				DV table of Y				DV table of Z			
t=2	Dx()	X	Y	Z	Dy()	X	Y	Z	Dz()	X	Y	Z
	X	0	2	3	X	0	2	3	X	0	2	3
	Y	2	0	1	Y	2	0	1	Y	2	0	1
	Z	3	1	0	Z	3	1	0	Z	3	1	0
	DV table of X				DV table of Y				DV table of Z			

▼ Question #6:

6) (10 points) A sender has this data payload to send D=10011010. If both the sender and the receiver have agreed to use $1 + x^3$ as their generator (i.e. G=1001), then compute the CRC (Cyclic Redundancy Check) for D! Show all the details!

▼ Answer:

D = 10011010

$$1.x^0 + 0.x^1 + 0.x^2 + 1.x^3 \rightarrow 1001$$

r+1 = 4 → Therefore, r = 3 bits

D **(append 3 zero)** : G

1	0	0	1	1	0	1	0	0	0	0	:	1	0	0	1	
1	0	0	1													XOR
0	0	0	0	1												DELETE most significant digit
	0	0	0	0												XOR
	0	0	0	1	0											DELETE most significant digit
		0	0	0	0											XOR
		0	0	1	0	1										DELETE most significant digit
			0	0	0	0										XOR
			0	1	0	1	0									DELETE most significant digit
				1	0	0	1									XOR
				0	0	1	1	0								DELETE most significant digit
					0	0	0	0								XOR
					0	1	1	0	0							DELETE most significant digit
						1	0	0	1							XOR
						0	1	0	1	0						DELETE most significant digit
							1	0	0	1						XOR
							0	0	1	1						REMAINDER

R = 011

D' = D | R = 10011010 | 011 = 10011010011 (sent by the sender)

▼ Question #7:

a) (5 points) A sender has this data payload to send D=01011010001011. It uses a 2D parity check with 5 rows and 4 columns and **odd parity** (i.e. in each row and column, including the parity bits, the number of 1s should be an odd number). Construct the message that will be send after 2D parity is added and show it in matrix form.
(Note: If the number of bits in the payload D is not enough to fill-up the matrix with 5 rows and 4 columns, at the end of D you can append the necessary number of zeros so as to fill-up the matrix)

▼ Answer (a):

<u>2D Odd Parity</u>

- Total number of 1's in the data, including the parity bit, should be odd

					Odd Row Parity
	0	1	0	1	1
	1	0	1	0	1
	0	0	1	0	0
	1	1	0	0	1
	0	0	0	0	1
Odd Column Parity	1	1	1	0	1

Message that will be sent → 01011 | 10101 | 00100 | 11001 | 00001 | 11101

b) (10 points) In the table below you have the message that a certain sender which uses 2D **even parity** check for error detection and/or correction has sent. Are there any transmission errors? If so, where? If so, can you correct them? Why?

101101	0
001010	0
101100	1
001010	0
100101	

▼ **Answer (b):**

2D Odd Parity

- Total number of 1's in the data, including the parity bit, should be even.

							Even Row Parity
	1	0	1	1	0	1	0 (CORRECT)
	0	0	1	0	1	0	0 (CORRECT)
	1	0	1	1	0	0	1 (CORRECT)
	0	0	1	0	1	0	0 (CORRECT)
Even Column Parity	1 (FALSE)	0 (CORRECT)	0 (CORRECT)	1 (FALSE)	0 (CORRECT)	1 (CORRECT)	

There are two errors in the table. The corrected version of the table is at the below:

CORRECT TABLE

							Even Row Parity
	1	0	1	1	0	1	0
	0	0	1	0	1	0	0
	1	0	1	1	0	0	1
	0	0	1	0	1	0	0
Even Column Parity	0	0	0	0	0	1	

Question #8:

8) (15 points) A sender sends an *n* bit message through an unreliable channel which has a bit-error probability *p*. What is the probability of:

a) (5 points) exactly one bit is flipped from the transmitted message

b) (5 points) at least one bit is flipped from the transmitted message

c) (5 points) exactly *b* bits are flipped from the transmitted message

Answer:

1. $P(E) = n * p * (1-p)^{n-1}$
2. $P(E) = 1 - (1-p)^n$
3. $P(E) = \binom{n}{b} * p^b * (1-p)^{n-b}$

Question #9:

9) (25 points) Consider the figure below (Fig.4), which shows the arrival of 6 messages for transmission at different multiple access wireless nodes at times t = <0.8, 1.2, 2.9, 3.1, 4.3, 4.6> and each transmission requires exactly one-time unit.

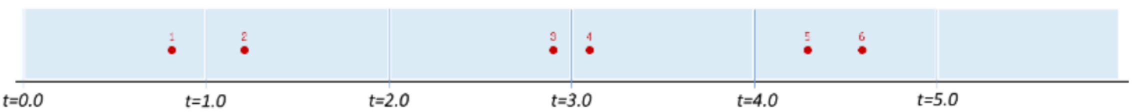


Fig.4. Transmission times of different multiple access nodes sharing a transmission medium

▼ Answer (a): Aloha

▼ a.1

a.1) Suppose all nodes are implementing the Aloha protocol. For each message, indicate the time at which each transmission begins. Separate each value with a comma and no spaces.

<0.8, 1.2, 2.9, 3.1, 4.3, 4.6>

▼ a.2

a.2) Which messages transmit successfully? Write your answer as a comma separated list with no spaces using the messages' numbers

All messages will be destroyed and none of them will be able to reach to the destination. All of these messages needs to be transmitted by using the back-off algorithm.

<>

▼ Answer (b): Slotted Aloha

▼ b.1

b.1) Suppose all nodes are implementing the Slotted Aloha protocol. For each message, indicate the time at which each transmission begins. Separate each value with a comma and no spaces.

<1,2,3,4,5,5>

▼ b.2

b.2) Which messages transmit successfully? Write your answer as a comma separated list with no spaces using the messages' numbers

Messages #1, Message #2, Message #3, and Message #4 are successfully transmitted because their transmission started before the collision.

<1,2,3,4>

▼ Answer (c): CSMA

▼ c.1

c.1) Suppose all nodes are implementing Carrier Sense Multiple Access (CSMA), but without collision detection. Suppose that the time from when a message transmission begins until it is beginning to be received at other nodes is 0.4 time units. (Thus if a node begins transmitting a message at t=2.0 and transmits that message until t=3.0, then any node performing carrier sensing in the interval [2.4, 3.4] will sense the channel busy.) For each message, indicate the time at which each message transmission begins, or indicate that message transmission does not begin due to a channel that is sensed busy when that message arrives. Separate each value with a comma and no spaces, and if the channel is sensed busy, substitute it with 's'

- Message #1

  <u>0.8</u> + 0.4 = 1.2

- Message #2 → There is another package → Message #2 wont be transmitted → **<u>‘s’</u>**

- Message #1

  1.2 + 1.0 = 2.2 → [1.2,2.2]

- Message #3

  <u>2.9</u> + 0.4 = 3.3

  3.3 + 1.0 = 4.1 → [3.3,4.3]

- Message #4

  <u>3.1</u> + 0.4 = 3.5

  3.5 + 1.0 = 4.5 → [4.5,5.5]

- Message #5 won’t be sent because channel is busy. → **<u>‘s’</u>**

- Message #6

  <u>4.6</u> + 0.4 = 5.0

  5.0 + 1.0 = 6.0 → [5.0,6.0]

<0.8,s,2.9,3.1,s,4.6>

▼ c.2

c.2) Which messages transmitted successfully? Write your answer as a comma separated list with no spaces using the messages' numbers

The working principle of CSMA is, it listens the channel before transmission happens. If the transmission channel is busy (another message is transmitting), it does not stops them. However, if there is a collision, it aborts the colliding transmissions. Therefore, only Message #1, and Message #6 will be transmitted successfully.

<1,6>

▼ Answer (d): CSMA/CD

▼ d.1

d.1) Suppose all nodes are implementing Carrier Sense Multiple Access (CSMA), with collision detection (CSMA/CD). Suppose that the time from when a message transmission begins until it is beginning to be received at other nodes is 0.4 time units, and assume that a node can stop transmission instantaneously when a message collision is detected. (Thus if a node begins transmitting a message at t=2.0 and transmits that message until t=3.0, then any node performing carrier sensing in the interval [2.4, 3.4] will sense the channel busy.) For each message, indicate the time at which each message transmission begins, or indicate that message transmission does not begin due to a channel that is sensed busy when that message arrives. Separate each value with a comma and no spaces, and if the channel is sensed busy, substitute it with 's

When see the collision stop the transmission → x

CSMA/CD is effective after a collision.

nodes at times t = <0.8, 1.2, 2.9, 3.1, 4.3, 4.6>

- Message #1

  0.8 + 0.4 = 1.2 (start transmission)

  0.8 + 1 = **1.8** (receive the message)

  1.8 + 0.4 = 2.2 (complete transmission)

  Sensing: [1.2,2.2] busy

- Message #2 → won't be transmitted because there is another package.

  1.2+1.0 = 2.2 → [1.2,2.2] channel is busy because of Message #1 → 's'

- Message #3

  2.9 + 0.4 = 3.3

  2.9 + 1.0 = **3.9** → [3.3,4.3] channel is busy.

  3.9 + 0.4 = 4.3

  Sensing: [3.3,4.3] busy

- Message #4 (Start time 3.1) → since channel is busy until 4.3, won't be transmitted. → 's'

- Message #5

  4.3 + 0.4 = 4.7

  4.3 + 1.0 = **5.3**

  5.3 + 0.4 = 5.7

  Sensing: [4.7,5.7] busy

- Message #6 (Start time 4.6)→ Since the channel is busy until 4.7, message won't be transmitted. → 's'

<0.8,s,2.9,s,4.3,s>

▼ d.2

d.2) Which messages transmitted successfully? Write your answer as a comma separated list with no spaces using the messages' numbers

CSMA/CD is a model which resends the data frame whenever a conflict occurs. However, for this question since we do not consider the resend operation, the transmitted messages will: Message #1, Message #3, Message #5.

<1,3,5>

▼ d.3

d.3) At what time did each message stop transmitting due to a collision. Write your answer as a comma separated list with no spaces using the messages' numbers in order, and if a message didn't stop, write 'x' for that message

<0.8,x,2.9,x,4.3, x>