Computer Networks

Fall 2023/24
Exercise 4

Submission by Thursday, 29-2-2024. Submit by uploading your work to the course Moodle website.
>>> No late submissions will be accepted!
The name of the submitted file must be Exercise4_firstname_lastname1.[suffix].
For example, Exercise4_israel-israeli.pdf. The first and last name of the student must appear.

Problem 1.

Consider sending a 2800-byte datagram into a link that has an MTU of 700 bytes.
Suppose the original datagram is stamped with the identification number 530.
How many fragments are generated?
What are their characteristics (i.e., Identification numbers, sizes, offsets, fragflag)?
Justify your answers.

Size of payload data in original datagram = 2800 – 20 = 2780 bytes
Size of MTU payload data = 700 – 20 = 680 bytes
Number of fragments = ceil(datagram payload size/MTU payload size) = ceil(2780 /680) = ceil(4.08) = 5 fragments

Fragment 1: ID=530, Size=680 bytes, Offset=0, flag=1
Fragment 2: ID=530, Size=680 bytes, Offset=680, flag =1
Fragment 3: ID=530, Size=680 bytes, Offset=1360, flag =1
Fragment 4: ID=530, Size=680 bytes, Offset=2040, flag =1
Fragment 5: ID=530, Size=100 bytes, Offset=2680, flag =0

Problem 2.

Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows:

Destination Address Ranges	Link Interface
224.0.0.0 ❼ 224.255.255.255	0
225.0.0.0 ❼ 225.0.255.255	1
225.0.0.0 ❼ 225.127.255.255	2
0.0.0.0 ❼ 255.255.255.255	3

A. Rewrite the forwarding table uses the a.b.c.d/z notation:

Destination Address Ranges	Link Interface
224.0.0.0/8	0
225.0.0.0/16	1
225.0.0.0 /9	2
0.0.0.0/0	3

B. Determine the appropriate link interfaces for datagrams with the following destination addresses:

Destination Address Ranges	Link Interface
225.241.81.85	3
225.0.195.60	1
225.32.17.119	2
255.1.255.255	3

Problem 3.

A large number of consecutive IP addresses are available starting at 81.0.208.0. Suppose the following organizations are asking for the following addresses:

A – 1024
B – 4096
C – 16384
D – 4096
E – 2048
F – 1024
G – 1024

A.

Fill in the following table the first address of each organization, the last address of each organization, how many addresses each organization has, and its CIDR mask.

Important: at any time, assign the smallest IP addresses available!

Organization	First Address	Last Address	Total Hosts	Mask
A	81.0.208.0	81.0.211.255	1024	/22
B	81.0.212.0	81.0.227.255	4096	/18
C	81.0.228.0	81.1.35.255	16384	/15
D	81.1.36.0	81.1.51.255	4096	/19
E	81.1.52.0	81.1.58.255	2048	/20
F	81.1.59.0	81.1.62.255	1024	/21
G	81.1.63.0	81.1.66.255	1024	/17

B. Is there a single aggregate address that the router can publish?

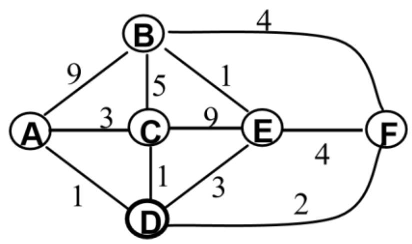
81.0.208.0/15

Problem 4.

Assuming bit time is 1 nano-sec, what is the minimum and maximum amount of time a device can wait after the 15th collision?

The options for delay is [0, 2^k – 1] where k is the number of collisions. Therefore, the options are [0, 2^15 - 1] = [0, 32768].
Maximum waiting time: 32768 x 1ns = 32768ns
Minimum waiting time: 0 x 1ns = 0ns

Problem 5.



Please complete the tables below with the final distance table for node D, as calculated by the distance vector algorithm after it has converged.

1. Assume poisoned reverse is not used.
2. Assume poisoned reverse is used

Table without using poisoned reverse.

Table for D	Neighbors (via…)			
destinations	A	C	E	F
A	1	4	7	5
B	8	6	4	6
C	3	1	7	5
E	5	5	3	6
F	4	4	7	2

Table using poisoned reverse.

Table for D	Neighbors (via…)			
destinations	A	C	E	F

A	1	∞	∞	∞
B	∞	6	4	6
C	∞	1	∞	∞
E	∞	∞	3	6
F	∞	∞	7	2

Problem 6.

Suppose you are implementing a NAT router using the PAT (Port Address Translation) scheme. However, you forget to check the port number of the incoming packets and use the same port number for all the mappings.

A. Suppose you have one device in the LAN and you're trying to connect to a TCP server, would you be able to connect successfully? Justify your answer.
   Yes you would be able to connect successfully. Since there is only one device on the LAN, then the port number that is being mapped to for all mappings must be that devices port number. Therefore all incoming packets will be sent to this device.

B. A friend comes over and connects to your router and you both connect different TCP servers, would both of you be able to connect successfully? Justify your answer.

   No. If there are 2 devices connected to the router, but all of the routes are mapped to same port number then the router would not be able to distinguish between the 2 devices and wouldn't know which device to send an incoming packet to. Therefore one or both of the devices might have connection errors, data might be sent to the wrong device, or the connections might not be established at all.

C. You both try to connect to the same TCP server, would both of you be able to connect successfully? Would one of you? Justify your answer.

No, for the same reason as question B. The router would not be able to distinguish between the 2 devices and would not know which device to send the incoming packets to.

Problem 7.

Suppose a TCP sender uses Reno TCP to control its congestion window. The initial congestion window size is 1 MSS, and the initial threshold is 32 MSS. The sender experiences one timeout event at time t=10 RTT, and one 3ACK event at time t=15 RTT, after which it resumes its normal operation. What is the size of the congestion window at time t = 20 RTT?

Time (RTT)	**Congestion Window Size (MSS)**	**Threshold (MSS)**	**Event**
0	1	32	-
1	2	32	-
2	4	32	-
3	8	32	-
4	16	32	-
5	32	32	-
6	33	32	-
7	34	32	-
8	35	32	-
9	36	32	-
10	1	18	Timeout
11	2	18	-
12	4	18	-
13	8	18	-
14	16	18	-
15	8	8	3ACK
16	9	8	-
17	10	8	-
18	11	8	-
19	12	8	-
20	13	8	-

Problem 7.

30 points

Assume the following simple TCP sender code snippet. The code does not support congestion control and flow control.
But it seems there are three bugs in the code – can you spot them?
For each bug:

1. What is the buggy code? and explain why it is incorrect.
2. What is the correct code? Explain why it fixes the problem.

```
window_size = 10 seq_num
= 1
buffer = [] # 1MB of data
 while buffer not
empty:
  window_to_send = window_size

send_window(buffer[0❼window_to_send])
seq_num += 1
    event =
wait_for_event()
   if event ==
ACK:
    buffer = buffer[(event.ack_num-seq_num)❼end of buffer]
seq_num -= event.ack_num
   else if event ==
TIMEOUT:
    resend_segments(buffer[0❼window_to_send])
```

Error 1: send_window(buffer[0 -> window_to_send]) can cause an error if the size of the buffer is smaller than the window size and therefore “buffer[0 -> window_to_send]” can cause an index out of bounds error.
Fix: When defining window_to_send, ensure that it cannot be bigger than the buffer size, which can be done like so “window_to_send = min(window_size, buffer.length)”

Error 2: “buffer = buffer[(event.ack_num – seq_num) -> end of buffer]”. While this error might not cause a compilation or runtime error, it is a logic error that will not execute what we intended it to. The event.ack_num will contain the absolute index of the next byte the receiver is expecting in the buffer, therefore subtracting seq_num will result in the wrong index.
Fix: “buffer = buffer[event.ack_num -> end of buffer]”.

Error 3: “seq_num -= event.ack_num” is incorrect. The event.ack_num holds the index of then next byte the receiver is expecting, which is the next expected sequence number. Therefore it mustn’t decrease seq_num by event.ack_num, but must set it equal to event.ack_num.
Fix: “seq_num = event.ack_num”