Dashboard / My courses / COSC264 / Mid-term test 2019 / Mid-term test 2019

Started on	Saturday, 29 August 2020, 7:59 AM
State	Finished
Completed on	Saturday, 29 August 2020, 8:21 AM
Time taken	22 mins 9 secs
Grade	**49.20** out of 85.00 (**58**%)

Information

This is the COSC 264 mid-term test for 2019. A few important hints:

- The programming language in the Coderunner problems is Python3.
- The socket questions make reference to the C API discussed in the lecture (though Python uses similar names for many calls).
- Please read text and instructions carefully, be careful with units.
- This test consists of two parts, one "electronic" part and one part (physical layer) that you need to submit on paper. Plan your time budget accordingly.
- Some test cases are of the form 'print(abs(your_function(..) - somenumber)<somesmallnumber)'. In test cases like this, the number 'somenumber' is the expected result, but to accommodate small floating point errors, a difference between the result of your function and the expected result is allowed, provided the difference is smaller in magnitude than 'somesmallnumber'.

Question **1**
Correct
Mark 2.00 out of 2.00

Please calculate the propagation delay for a signal traveling a distance of 15,000 km, assuming a speed of light of C=200,000 km/s. Please give your answer in seconds.

Answer: 0.075 ✔

$$\text{propagation delay} = \frac{d}{v} = \frac{15000 \text{ Km}}{200{,}000 \text{ Km/s}}$$

Correct
Marks for this submission: 2.00/2.00.

Question **2**
Correct
Mark 2.00 out of 2.00

Please calculate the transmission delay for a packet of length *L*=1,500 bytes over a link with a data rate of *R*=5 Mbps. Please give it in seconds.

Answer: 0.0024 ✔

$L = 1500 \text{ bytes} \times 8 \text{ bit/byte}$
$\Rightarrow L = 12000 \text{ bits}$

Correct
Marks for this submission: 2.00/2.00.

$$\text{Transmission delay} = \frac{12\,000 \text{ bits}}{5 \times 10^6 \text{ bps}}$$

Question **3**

Correct

Mark 5.00 out of 5.00

Consider a fixed output link of a router. To cope with transient overload situations, the output link has a buffer or queue, so that packets arriving to the output while another packet is being transmitted over the same output do not need to be dropped. The queue is organised as a FIFO (first-in-first-out) queue.

When a new packet arrives, it is checked whether the output link is currently transmitting a packet:

- If not, the transmission of the new packet is started immediately and the waiting time of the packet is zero.
- Otherwise, the packet is sent to the end of the queue. Once the output has finished transmitting the previous packet, it will take the head-of-queue packet out of the queue and start its transmission. The waiting time of the newly arriving packet is the sum of the time required to finish the transmission of packet currently in service when the new packet arrives, plus the transmission times of all packets ahead in the queue.

All arriving packets have the same size of *L* bits, and the data transmission rate on the outgoing link is *R* bits per second. Write a Python function which calculates the waiting time of a newly arriving packet. This function takes as input:

- The parameters *L* and *R*.
- A flag which tells whether there is another packet currently being transmitted (True) or not (False)
- The number *N* of other packets already stored in the queue (valid only if the flag is True)

You can assume that the arrival time of the new packet is random and that on average about half of the currently transmitted packet has already finished transmission.

For example:

Test	Result
`print(abs(queueingDelay(1000,1000000,True,0)-0.0005)<0.00001)`	`True`
`print(abs(queueingDelay(1000,1000000,False,0)-0.0000)<0.00001)`	`True`

Answer: (penalty regime: 0, 10, 20, ... %)

Reset answer

```
def queueingDelay (packetSize_bits, dataRate_bps, flagCurrentTransmission, numberInQueue):
    L    =  packetSize_bits
    R    =  dataRate_bps
    flag =  flagCurrentTransmission
    N    =  numberInQueue
    if flag == True:
        return N*(L/R) + (L/2)/R
    else:
        return 0
```

assume that on average half of current transmitted packet has already finished transmission.

time to transmit N packets

	Test	Expected	Got	
✔	`print(abs(queueingDelay(1000,1000000,True,0)-0.0005)<0.00001)`	`True`	`True`	✔
✔	`print(abs(queueingDelay(1000,1000000,False,0)-0.0000)<0.00001)`	`True`	`True`	✔

Passed all tests! ✔

Correct

Marks for this submission: 5.00/5.00.

Question **4**

Correct

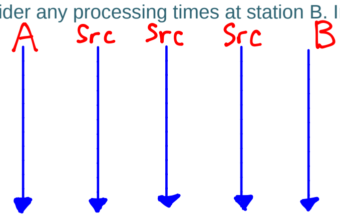
Mark 6.00 out of 6.00

We consider **packet switching**.

We are given a system with a number of N+2 stations A, R_1, R_2, ..., R_N, B such that end host A is connected to the first router R_1, the first router R_1 is connected to router R_2, and so on, and the last router R_N is connected to the other end host B (i.e. all the stations form a chain). Hence, there are N routers.

Now suppose that A has a message of size M bits which is an integer multiple of the maximum packet user data size of S bits. Station A prepares M/S packets and sends them back-to-back, without any gap. An individual packet has total size $S+O$ bits, where O is the number of overhead bits per packet. All the routers can process incoming packets without a gap: if a router has finished processing one packet (which takes P seconds processing time) and the next packet has been completely received at this time, processing of this next packet can start immediately and we have a kind of "pipelining effect". Note that all routers can process incoming packets and transmit outgoing packets at the same time. All the links in the system can transmit at a data rate of R bits per second, and the propagation delay on each link is T seconds.

Suppose station A starts transmission of the first packet at time 0. Find a general expression for the time by which station B will have received all M/S packets (use the simplification that M/S is an integer and there are hence no slack packets). You do not need to consider any processing times at station B. Implement your expression as a Python function.



For example:

Test	Result
`print(abs(packetSwitching(3, 10000, 1000, 100, 0.001, 1000000, 0.02)-0.0973)<0.0001)`	`True`

Answer: (penalty regime: 0, 10, 20, ... %)

Reset answer

```python
def packetSwitching (numberRouters, messageSize_b, userDataSize_b, overheadSize_b, processingT
    N  =  numberRouters
    M  =  messageSize_b
    S  =  userDataSize_b
    O  =  overheadSize_b
    P  =  processingTime_s
    R  =  dataRate_bps
    T  =  propagationDelay_s
    TR = ((S+O)/R) #trasmission delay
    Q = (M/S-1)*(S+O)/R #Queuing delay
    total_time = (T+TR)*(N+1) + P*N + Q #not consider processing time at station B
    return total_time
```

	Test	Expected	Got	
✔	`print(abs(packetSwitching(3, 10000, 1000, 100, 0.001, 1000000, 0.02)-0.0973)<0.0001)`	`True`	`True`	✔

Passed all tests! ✔

Correct

Marks for this submission: 6.00/6.00.

Question **5**
Correct
Mark 2.00 out of 2.00

Which conversion function will you have to use to convert a 32 bit integer value from network representation to host representation?

Select one:
- a. htons()
- b. inet_aton()
- c. ntohs()
- d. htonl()
- e. ntohl() ✔

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **6**
Correct
Mark 2.00 out of 2.00

We are given a specific port number *p*. How many processes within a server can bind a socket to this port number *p* at most?

Select one:
- a. two
- b. arbitrarily many
- c. one ✔
- d. five
- e. zero

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **7**
Correct
Mark 2.00 out of 2.00

Which socket type will you need to use for reliable and in-sequence data transfer of a large block of bytes?

Select one:
- a. Stream socket (SOCK_STREAM) ✔
- b. Datagram socket (SOCK_DGRAM)
- c. Raw socket (SOCK_RAW)
- d. Sequenced-packet socket (SOCK_SEQPACKET)

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **8**
Correct
Mark 2.00 out of 2.00

Suppose that a sender and receiver communicate over datagram sockets. The sender calls write() three times: the first time with 10 bytes of data, the second and third time with 20 bytes of data each. All the data is successfully transferred to the receiver. The receiver will call recvfrom() on this socket and attempts to read 1024 bytes into a buffer. How many bytes will the recvfrom() call return?

Select one:

- a. 1023 Bytes
- b. 5 Bytes
- c. 30 Bytes
- d. 1024 Bytes
- e. 10 Bytes ✔ receiver gets the first message of 10 bytes

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **9**
Correct
Mark 2.00 out of 2.00

Suppose a station has opened a datagram socket using the socket() function and as its next step wants to send some data over this socket. Which function will the station have to use?

Select one:

- a. sendto() ✔
- b. write()
- c. send()

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **10**
Correct
Mark 2.00 out of 2.00

Which layer of the OSI reference model carries out modulation and demodulation?

Select one:

- a. Application layer
- b. Representation layer
- c. Link layer
- d. Session layer
- e. Network layer
- f. Transport layer
- g. Physical layer ✔

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **11**
Correct
Mark 2.00 out of 2.00

Which layer of the OSI reference model is responsible for translating between different representations of a data type?

Select one:

- a. Session layer
- b. Representation layer ✔
- c. Transport layer
- d. Application layer
- e. Physical layer
- f. Network layer
- g. Link layer

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **12**
Not answered
Marked out of 5.00

In the OSI reference model error control is carried out on both the link layer (with a per-hop scope) and on the transport layer (end-to-end scope). Suppose we have a "perfect" transport-layer error control, i.e. all errors are detected and repaired reliably. Please argue why it may still be useful to additionally have link-layer error control. What can go wrong if no link layer transmission is protected by error control?

Question **13**
Correct
Mark 6.00 out of 6.00

We consider a fragmentation and reassembly mechanism between a transmitter and receiver. Suppose that the maximum frame size is F bytes in total (for example, for Ethernet we would have $F=1,500$ bytes), out of which there are O bytes overhead. The protocol entity at the transmitter gets a message of M bytes length from the higher layers and wants to break these down into fragments. You can assume that the maximum message size is $M_{max} = 65,535$ bytes.

To manage the fragmentation-and-reassembly process, each fragment includes an 'offset' field in its header, which indicates the byte index of the first byte of the fragment payload within the overall message -- the first fragment would contain the value 0 here.

Please write a Python function which takes as parameters the values for the maximum frame size F, the per-frame overhead O, and the overall message size M, and which returns a list containing the (value of the) offset field of the first fragment, the offset field of the second fragment, and so on.

For example:

Test	Result
`print(fragmentOffsets(1500,40,3000)==[0,1460,2920])`	`True`

Answer: (penalty regime: 0, 10, 20, ... %)

Reset answer

```python
import math
def fragmentOffsets (fragmentSize_bytes, overheadSize_bytes, messageSize_bytes):
    F  =  fragmentSize_bytes
    O  =  overheadSize_bytes
    M  =  messageSize_bytes
    num_fragments = math.floor(M/F) + 1
    result = []
    for i in range(num_fragments):
        result += [(F-O)*i]
    return result
```

	Test	Expected	Got	
✔	`print(fragmentOffsets(1500,40,3000)==[0,1460,2920])`	`True`	`True`	✔

Passed all tests! ✔

Correct
Marks for this submission: 6.00/6.00.

Question **14**
Not answered
Marked out of 6.00

Please explain the operation of TDMA (Time Division Multiple Access).

Question **15**
Not answered
Marked out of 6.00

Please explain the basic operation of a bridge and how bridges learn which station is in which of the coupled local area networks.

Question **16**
Not answered
Marked out of 6.00

Please explain the operation of the nonpersistent-CSMA protocol. Also discuss the advantages and disadvantages of choosing the probability distribution for the random backoff time such that the expected backoff time is comparatively small related to the size of a packet.

Question **17**
Correct
Mark 4.00 out of 4.00

Suppose we have a system with three stations wishing to transmit data. There are three time slots available, each time slot large enough for a packet transmission. Each of the three stations picks one of the three time slots randomly with uniform distribution.

What is the probability that all three stations pick distinct time slots and hence enjoy a successful transmission without any collision? Please give the result to three digits after the decimal point without any rounding.

Answer: 0.222

number of stations = n
number of time slots = x

$$p(\text{distinct}) = \frac{x!}{n^{x}} = \frac{3!}{3^3}$$

Correct
Marks for this submission: 4.00/4.00.

Question **18**
Correct
Mark 5.00 out of 5.00

Write a Python function which takes a 32-bit IPv4 address and converts this into its dotted-decimal string representation.

For example:

Test	Result
`print(IPToString(0x20304050))`	32.48.64.80

Answer: (penalty regime: 0, 10, 20, ... %)

Reset answer

```
def IPToString (addr):
    return '{}.{}.{}.{}'.format((addr & 0xFF000000) >> 24,
    (addr & 0x00FF0000) >> 16,
    (addr & 0x0000FF00) >> 8,
    (addr & 0x000000FF) )
```

	Test	Expected	Got	
✔	`print(IPToString(0x20304050))`	32.48.64.80	32.48.64.80	✔

Passed all tests! ✔

Correct
Marks for this submission: 5.00/5.00.

Question **19**
Correct
Mark 2.00 out of 2.00

The IPv4 protocol:

Guarantees that datagrams are received in the same order as they are sent.	False ✔
Uses 64-bit wide addresses	False ✔
Does not provide reliable service	True ✔
Requires an initial connection setup before any data can be sent.	False ✔
Uses acknowledgements	False ✔

Your answer is correct.

Correct
Marks for this submission: 2.00/2.00.

Question **20**
Correct
Mark 3.20 out of 4.00

Below a forwarding table of an IP router is shown. Note that in some cases several entries match the same destination address. In such a case, the most specific entry is chosen, i.e. the entry with the most one-bits in the network mask. Please work out the forwarding decisions made by a router for different destination addresses.

Destination Network / Netmask	Outgoing interface
130.2.0.0 / 16	eth0
130.2.10.0 / 24	eth1
177.4.12.0 / 24	eth2
145.6.0.0 / 16	eth0
132.16.12.0 / 24	directly attached
0.0.0.0/0	eth3

132.16.12.2	Deliver to directly attached station ✔
130.2.11.10	Forward to eth0 ✔
145.6.7.8	Forward to eth0 ✔
130.3.4.5	Forward to eth3 ✔ 0.0.0.0/0 require no byte to match
130.2.10.11	Forward to eth1 ✔

Your answer is correct.

Correct
Marks for this submission: 4.00/4.00. Accounting for previous tries, this gives **3.20/4.00**.

Question **21**
Not answered
Marked out of 6.00

Please explain the operation of the ARP protocol. What is it good for and how does it operate?

Question **22**

Not answered

Marked out of 6.00

The IPv4 protocol supports a fragmentation-and-reassembly mechanism. One of the rules is that reassembly is **only** carried out in the final destination and not in an intermediate router. Please justify this rule. Why does it make sense?

◄ Quiz: Local area networks and medium access control (Practice copy)

Jump to...

Quiz: IPv4 Networking ►