Module Code: COMP222101

Module Title: Networks

School of Computing

Semester 2 2023/24

Calculator instructions:

- You are allowed to use a non-programmable calculator.

Dictionary instructions:

- You are **not** allowed to use your own dictionary in this examination. A basic English dictionary is available to use: raise your hand and ask an invigilator, if you need it.

Examination Information

- There are **2 hours** to complete the examination.
- There are **14** pages to this examination.
- Answer **all 3** questions.
- The number in brackets **[ ]** indicates the marks available for each question or part question.
- You are reminded of the need for clear presentation in your answers.
- The total number of marks for this examination paper is **60**.
- You are allowed to use annotated materials.

with solutions

Question 1

NOTE ALL MCQ QUESTIONS WILL BE REFORMATTED SO AS TO BE SUITABLE FOR SCANNING ONCE A USABLE TEMPLATE BECOMES AVAILABLE

(a) Consider the IPv4 address range given in CIDR notation by `84.221.102.0/23`. If the full range was listed in increasing numerical order, what would be the final address? Give your answer in standard IPv4 notation with no additional text or punctuation.

Solution:

`84.221.103.255` [1].

[1 mark]

(b) How many addresses are there in the this range?

Solution:

$2^9 = 512$; [1] for either.

[1 mark]

(c) What is the full, 16-byte hexadecimal form of the contracted address `2112::ff0:3e0:0:0:2a`? Give your answer in standard IPv6 format with no additional text or punctuation.

Solution:

`2112:0000:0000:0ff0:03e0:0000:0000:002a` [2]. Drop one mark for any missed leading zeros, one for wrong number of byte pairs, one for any other mistake to a minimum of zero.

[2 marks]

(d) What is the contracted form of `2f20:0000:0000:0000:0fff:0000:0000:000d`? Give your answer in standard IPv6 format with no additional text or punctuation.

Solution:

`2f20::fff:0:0:d` [2]. Drop 1 mark for using two sets of double colons. Drop 1 mark for using double colons for the rightmost run of zeroes rather than the leftmost. Drop one mark for not removing the leading zero before `d` or `ffff`. Drop 1 mark for any other mistakes, to a minimum of zero.

[2 marks]

(e) For a normal DNS request initiated by a host, the local DNS server (that the host is configured to use) sends messages iteratively, first to the root server, then to the TLD server, and finally to the authoritative server. Consider instead a fully recursive scenario in which the local DNS server contacts the root server, which then communicates directly with the TLD server, which in turn communicates directly with the authoritative server, to get the final hostname. How many messages would be sent in total, starting from the host? You should ignore caching.

Solution:

8 [1].

[1 mark]

(f) Why do you think this communication pattern is not used in practice?

Solution:

Increases the number of messages sent/received by each of the root, TLD, and authoritative servers [1]. More precisely, increases by a factor of 2 / doubles the number of such messages [1]. Give both marks if they say this is less tolerant to a system fault.

[2 marks]

(g) Suppose the DNS cache was placed on the host that initiated the DNS query, rather than on the local DNS server. State one advantage of moving the cache to the host. You do not need to explain your answer.

Solution:

One of: faster retrieval for the user; would free up resources on the local DNS server [1]. Give a half mark for less congestion.

[1 mark]

(h) Now state one disadvantage of moving the cache to the host. As before, you do not need to explain your answer.

Solution:

One of: benefits of caching would only be applied to one user; would require some resources in the host [1].

[1 mark]

(i) Suppose two applications have been written in Java for a client-server architecture that uses TCP. Which of the following defines the server application? Write down the numbers from the list below for all that apply. You do not need to explain your answer.

1 Before communication can begin, the client needs to know the hostname of the server, but the server does not need to know the hostname of the client.
2 The server listens to a port number that is known to the client.
3 The server initiates the connection handshaking.
4 The client has one or more instances of the `Socket` class, but no instances of `ServerSocket`.
5 The server has one instance of `ServerSocket` but no instances of `Socket`.
6 The server application is larger than the client application because of the greater complexity.

Solution:

1, 2 and 4 [2]. Start from 2 marks and subtract 1 for each of 1, 2 and 4 not given, and subtract half a mark for any other letters that *are* given, to a minimum of zero.

[2 marks]

(j) The remainder of Question 1 concerns an alternative to the client-server architecture known as peer-to-peer, in which two hosts communicate with each other, but neither is a client or server – each can initiate the connection. This means that both hosts run the same application, rather than requiring two distinct applications.

Someone starts developing some Java code for a TCP peer-to-peer application in which one host sends short messages, entered by the user on the keyboard, to the other host, which displays it. The connection only lasts as long as required to send or receive each message. A problem is immediately encountered: It is not possible to simultaneously wait for user input from the keyboard, *and* wait for a connection from the other host. Therefore a multi-threaded solution is developed in which one thread performs the client-like operations, and a second thread performs the server-like operations.

Inspect the following code for the server-like thread implemented as a nested class within the peer-to-peer application.

```
1     public class ServerOperation implements Runnable {
2       public void run() {
3         ServerSocket serverSock = new ServerSocket( HOSTNAME, PORT );
4         while( true ) {
5           Socket socket = serverSock.accept();
6           BufferedReader inStream = new BufferedReader(
7                                       new InputStreamReader(
8                                          socket.getInputStream() ) );
9           System.out.println( inStream.readLine() );
10          socket.close();
11        }
12      }
13    }
```

Identify two mistakes in this code that will cause compilation to fail, giving line numbers, briefly justifying your choices. You may assume all of the necessary libraries have been imported. You should not attempt to explain how you would correct these mistakes.

Solution:

- All network I/O, from line 3 to line 10 $\left[\frac{1}{2}\right]$, not included in a `try...catch` clause $\left[\frac{1}{2}\right]$; *or*, no exception in line 1.
- Constructor for `ServerSocket` on line 3 $\left[\frac{1}{2}\right]$ should not have an argument for the hostname $\left[\frac{1}{2}\right]$.

Drop 1 mark for more than 2 attempts. Note the inner class can access `HOSTNAME` and `PORT` as part of the outer class, so this is not a valid answer.

[2 marks]

(k) Suppose this thread was initialised as follows.

`ServerOperation serverOp = new ServerOperation();`

What line or lines of code would you use to start the thread? Your answer should consist only of Java code. You should not attempt to justify your answer.

Solution:

`Thread t = new Thread(serverOp);` [1]

`t.start();` [1]

Except for the choice of thread name `t` (which must be the same on both lines), must match the above code exactly for each mark. Drop half mark per line for minor mistakes such as spelling / capitalisation errors.

[2 marks]

(l) How many port numbers need to be specified by the programmer for this peer-to-peer application to perform correctly? You should assume each peer-to-peer application runs on separate hosts.

Solution:

1 [1].

[1 mark]

(m) It is now decided to switch to UDP. Why do you think this might be an advantage over TCP? Your answer must be specific to this peer-to-peer problem; no marks will be given for general statements about UDP.

Solution:

No need to be multi-threaded anymore $\left[\frac{1}{2}\right]$ as neither host needs to initiate the connection in (connectionless) UDP $\left[\frac{1}{2}\right]$.

[1 mark]

(n) A restriction with the peer-to-peer application described so far is that it needs to know the hostname (or IP address) of the other host. How would you circumvent this problem? Your answer should be brief; you may lose marks for extensive text that does not specifically address this question, or for attempting multiple answers.

Solution:

Have a centralised server that each peer-to-peer application, now acting as a client, uses to discover other peers; [1] for this or any reasonable answer.

[1 mark]

[Question 1 Total: 20 marks]

Question 2

(a) Which of the following best describes the use of public key encryption when A wants to send an encrypted message to B?

1 A encrypts the message using A's private key.
2 A encrypts the message using a combination of A's private and public keys.
3 A encrypts the message using a combination of B's private and public keys.
4 A encrypts the message using B's public key.
5 A encrypts the message using a combination of A's private key and B's public key.

Solution:

4 [1].

[1 mark]

(b) It is suggested that encryption can be used to test if the message was corrupted or altered during transmission. Do you agree with this claim? Briefly explain your answer.

Solution:

Yes [1], because any errors/alterations in the message *en route* will cause the decryption to fail [1]. Drop one mark for unnecessarily long answers such as those giving examples.

[2 marks]

(c) If you knew that only trusted users had your public key, could asymmetric encryption be used as a form of authentication? Give a brief justification of your answer.

Solution:

Yes [1], because if decryption failed then the message was not sent by one of the trusted users [1]. Man-in-the-middle is not valid since, as described, you know only trusted users have your public key, and no-one needs to transmit your key for it to be potentially intercepted (how we got into this situation in the first place is a separate issue).

[2 marks]

(d) The remaining subquestions in Question 2 refer to the following problem.

Someone is developing an online chess application in which one server plays against multiple clients. When the client makes a move, they send this to the server using some agreed message format. In turn, the server sends its move back to the same client. Given many clients are expected, with infrequent communication with the server for each client, it is decided that non-blocking I/O is the best choice of server architecture.

The code segment for the main loop in which the `Selector selector` is polled for new events is given below.

```
1     while( true ) {
2       selector.select();
3       Iterator keys = selector.selectedKeys().iterator();
4       while( keys.hasNext() ) {
5         SelectionKey key = (SelectionKey) keys.next();
6         keys.remove();
7          if( !key.isValid() ) continue;
8          if( key.isAcceptable() ) accept( key );
9          if( key.isReadable() ) clientMove( key );
10         if( key.isWritable() ) serverMove( key );
11       }
12    }
```

First, what would happen if line 6 was deleted? Your answer should be specific to this chess application - no marks will be awarded for generic statements about non-blocking I/O.

Solution:

The same move (either way) will be repeated indefinitely $\left[\frac{1}{2}\right]$, or a new game with the same client will be started multiple times $\left[\frac{1}{2}\right]$. No mark for ‘not duplicating the event,’ as this is too generic.

[1 mark]

(e) Suppose the server can calculate its moves quickly. To take advantage of this, someone suggests modifying the code by merging lines 9 and 10 into a single line, so as to send the server move back the client as soon as possible. The new line 9 becomes

```
9     if( key.isReadable() || key.isWritable() ) returnMove();
```

where `returnMove()` records the client move and returns the server move. Given the client always moves first, does this make sense? Justify your answer.

Solution:

No [1], as there is no guarantee that a readable channel is writable [1]; for instance, the client may make a move and then disconnect while the server is still in `returnMove()`. Give mark for any reasonable justification.

[2 marks]

(f) Returning to the original version of the code, it is now decided to allow the server to calculate moves independently of sending them, to allow for a more balanced use of CPU time. A new method `calculateMove()` is implemented that calculates move(s) for the most recent client move(s) received, and stores them in preparation to be sent to the client.

Select from the following list all the changes you would make to the code to implement this feature. Your answer should consist solely of the numbers given below, separated by spaces; do not add any other text or punctuation.

1 Before line 1, register the selection key for no active events (`OP_READY`) with the selector.
2 Place line 2 in a `try...except` clause to catch a timeout, in which event `calculateMove()` is called.
3 Replace line 7 with: `if( !key.isValid() ) calculateMove();`
4 Change line 2 to use the version of `select` that accepts a timeout, with some short timeout.
5 Add a call to `calculateMove()` immediately after line 2, line 3, or line 11.
6 Somewhere between lines 8 and 10, add a conditional that checks for no events (`key.isReady()`) and calls `calculateMove()`.

Solution:

2 [1] and 4 [1]. Drop half marks for each incorrect number given.

[2 marks]

(g) It is now decided to re-write the chess server to use UDP rather than non-blocking I/O. Key parts of the code are given below.

```
1  public class ChessServer {
2    public static DatagramPacket serverMove( DatagramPacket clientMove )
3      { ...  /* Not shown */ };
4    public static void main(String[] args) throws UnknownHostException {
5      DatagramSocket socket = new DatagramSocket();
6      while( true ) {
7        DatagramPacket pack = new DatagramPacket(new byte[100],100,
8                                  InetAddress.getLocalHost(),8787);
9        socket.receive(pack);
10       DatagramPacket move = serverMove( pack );
11       socket.send(move);
12     }
13   }
14 }
```

There are two mistakes in this code. Identify these mistakes, and for each, give the line number above where you think the error occurs, and the change to the code you would make to correct the error. You do not need to explain your answers.

Solution:

First mistake: *Either* line 4 [1] as `UnknownHostException` needs to be (at least) `IOException` [1]; *or*, Lines 5-11 [1] need to be in a `try...catch` clause to catch a possible `IOException` [1].

Second mistake: Line 5 [1] needs the port number in the `DatagramSocket` constructor [1]. Note the extra arguments on lines 7/8 are redundant but will not stop it from working, nor do they correct for omitting the port number in the constructor.

[4 marks]

(h) What does the number `100` on line 7 refer to?

Solution:

Largest message size in bytes that can be received [1]. Do not give mark if receiving not explicitly stated (given code tells us nothing about the length that can be sent).

[1 mark]

(i) Given there are no persistent connections in UDP, what do you think is the simplest mechanism by which the server can identify which client it is playing with?

Solution:

Use the client IP address [1] extracted/gettable from the `DatagramPacket` [1]. Give one mark for a workable answer that is more complex than this, and therefore not the "simplest" mechanism.

[2 marks]

(j) Since UDP is unreliable, some messages are inevitably lost during transmission. How would you modify the server code to display an error message if this happens? Your answer should use the line numbers given above. You may lose marks for changes that do not directly address this feature.

Solution:

Immediately after line 5 $\left[\frac{1}{2}\right]$, set the socket timeout $\left[\frac{1}{2}\right]$ to something of the order of a few seconds $\left[\frac{1}{2}\right]$.

Place the `socket.receive()` on line 5 $\left[\frac{1}{2}\right]$ in a `try...catch` clause for the timeout exception $\left[\frac{1}{2}\right]$, which emits the error message $\left[\frac{1}{2}\right]$.

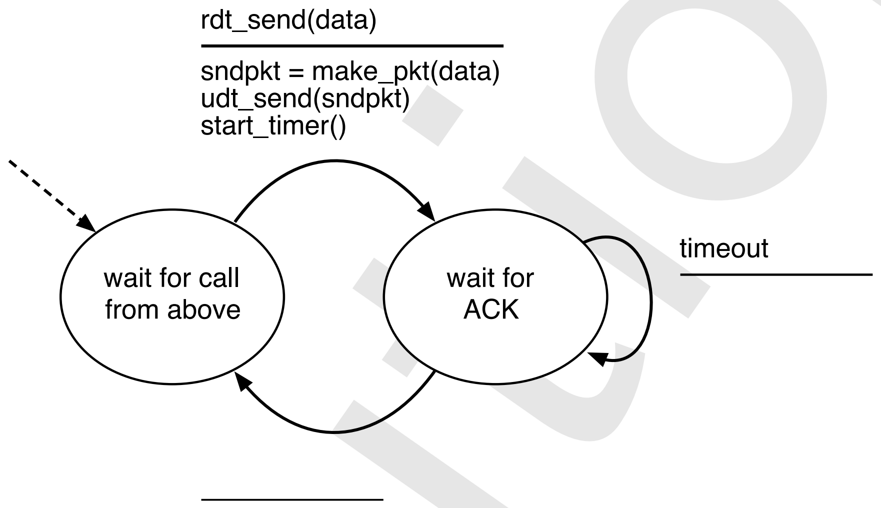
[3 marks]

[Question 2 Total: 20 marks]

Question 3

(a) Suppose you have an unreliable data channel which is 'lossy' - that is, packets may be lost during transmission. The packets themselves cannot become corrupted. You are asked to devise a protocol that reliably sends data over this channel that uses a countdown timer: The sender is to send the data and start a timer by calling `start_timer()`, then wait for the receiver to respond with a positive acknowledgement ACK. If the ACK is not received before a timeout occurs, the packet is re-sent. For the first part of this question, you may assume that ACKs are never lost.

The beginnings of a finite state machine for the sender is given below, however it is currently incomplete. Inspect the diagram and then answer the questions that follow.



Consider the following actions, and select from the list below those that you would put below the line for the `timeout` event. Just give the numbers in your answer, with no additional punctuation. The names should be self-explanatory and most are the same as used in lectures. Partial marks will be given for partially correct answers.

1 `sndpkt = make_pkt(data)`
2 `udt_send(sndpkt)`
3 `start_timer()`
4 `stop_timer()`
5 Λ.

Solution:

2 and 3 [2]. One mark for each correct response, drop half mark for each incorrect response.

[2 marks]

(b) What is the minimum action or set of actions from the list below that should go above the line alongside the arrow from 'wait for ACK' to 'wait for call from above'? Just give the numbers in your answer, with no additional punctutation.

1 rdt_send(sndpkt)
2 rdt_rcv(rcvpkt)
3 rdt_rcv(rcvpkt) && isACK(rcvpkt)
4 rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
5 timeout

Solution:

2 [1]. Note that a check for an ACK is not "minimum" since, as described, it can be nothing else. **[1 mark]**

(c) Which of the following should go below this same line?

1 sndpkt = make_pkt(data)
2 udt_send(sndpkt)
3 start_timer()
4 stop_timer()
5 Λ.

Solution:

4 [1]. **[1 mark]**

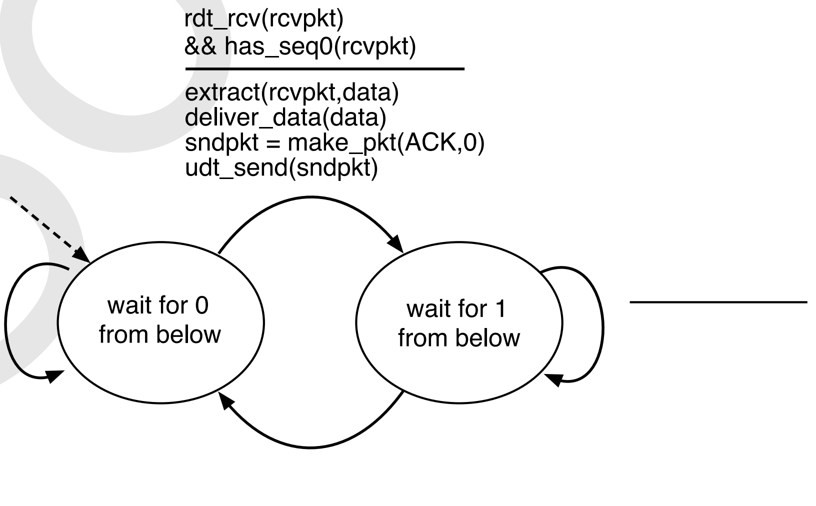
(d) It is still necessary to select a suitable time before the `timeout` event is triggered, which requires estimating how long packets may be delayed for. State one way in which a packet may be delayed, but not lost, on a wide area network. You do not need to justify your answer.

Solution:

Any one of: inefficient routing; slow forwarding at a router; congestion; [1] for any of these, or any other reasonable answer. **[1 mark]**

(e) In reality, of course, it is also possible that the ACK will be lost. The obvious solution to this is the same as for packet corruption; that is, to assign sequence numbers to both the packet and the ACK.

Assume that the sender has now been modified to alternate between two sequence numbers 0 and 1, and now consider the receiver finite state machine. As before, inspect the following incomplete finite state machine and answer the questions that follow.



What event or combination of events triggers the arrow from 'wait for 1 from below' to itself; that is, what should go above the line? Select all from the list below that apply. If multiple events are selected, this means you think they must all occur for the arrow to be triggered.

```
1 rdt_rcv(rcvpkt)
2 corrupt(rcvpkt)
3 notcorrupt(rcvpkt)
4 has_seq0(rcvpkt)
5 has_seq1(rcvpkt)
6 isACK(rcvpkt)
7 isNAK(rcvpkt)
```

Solution:

1 and 4 [2]. One mark for each of 1 and 4, drop a half mark for each option given that is incorrect. **[2 marks]**

(f) What are the corresponding action or actions - that is, what should go below the same line?

```
1 sndpkt = make_pkt(ACK,0)
2 sndpkt = make_pkt(ACK,0,checksum)
3 sndpkt = make_pkt(ACK,1)
4 sndpkt = make_pkt(ACK,1,checksum)
5 udt_send(sndpkt)
6 udt_send(rcvpkt)
```

Solution:

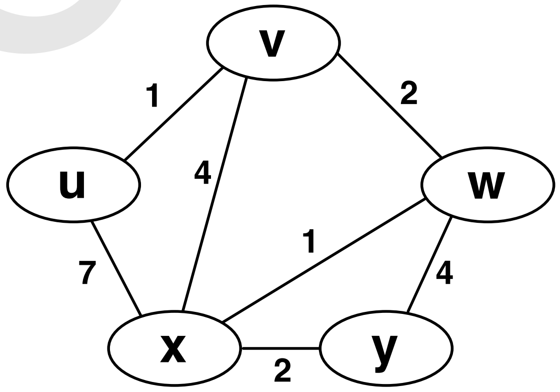
1 and 5 [2]. One mark for each correct answer, minus a half mark for each incorrect option. **[2 marks]**

(g) Note that no events or actions have been provided for the bottom and leftmost arrows. **Very briefly** describe how these events and actions relate to those for the top and rightmost arrows.

Solution:

Same but with $0 \leftrightarrow 1$ [1]. **[1 mark]**

(h) Determining an efficient routing within a small subnetwork is typically performed using link state algorithms, the most common being Dijkstra's algorithm. To demonstrate this algorithm, consider the network below with 5 nodes u, v, w, x and y.



The numbers alongside the connections between nodes are the link costs. What might these represent?

Solution:

Any one of: time taken to travel between nodes; inversely related to congestion; financial cost; [1] for any reasonable answer, half mark for simply saying 'bandwidth' if this is not backed up by saying that higher bandwidths mean lower costs.

[1 mark]

(i) Suppose all of the link costs were equal. What would Dijkstra's algorithm then actually determine?

Solution:

Routes from a given node to all other nodes [1] that have the lowest number of hops/nodes/connections [1]. One mark maximum for 'shortest routes.'

[2 marks]

(j) Now perform Dijkstra's algorithm on the given network starting from the node u. The initial state is given in tabular form as follows.

Step	N'	$D(v),p(v)$	$D(w),p(w)$	$D(x),p(x)$	$D(y),p(y)$
0	u	1,u	∞	7,u	∞

As normal, N' is the set of nodes for which the optimal path from u is known, $D(A)$ is the current cost from u to any node A, and $p(A)$ is the node immediately prior to A on the current best path from u to A.

Complete the table, then answer the remaining questions.

Firstly, what is the final step number - that is, the step number for which $N' = uvwxy$ for the first time?

Solution:

Note for all of these subquestions that the full table is:

Step	N'	$D(v),p(v)$	$D(w),p(w)$	$D(x),p(x)$	$D(y),p(y)$
0	u	1,v	∞	7,x	∞
1	uv		3,v	5,v	∞
2	uvw			4,w	7,w
3	$uvwx$				6,x
4	$uvwxy$				

For the first question, the answer is 4 [1].

[1 mark]

(k) At the end of the algorithm, what is your final $D(w), p(w)$?

Solution:

3, v; [1] for each.

[2 marks]

(l) Similarly, what is your final $D(x), p(x)$?

Solution:

4, w; [1] for each.

[2 marks]

(m) Last of all, what is your final $D(y), p(y)$?

Solution:

6, x; [1] for each.

[2 marks]

[Question 3 Total: 20 marks]