



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Examination Answer Books

12 Pages

Instructions to Candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
3. The answers must be legibly written. The Examiners can either ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.

Complete, fold and seal
(see note 2 below)

SURNAME: Jordan

FORENAME(S): Oliver

1	Student Number	P144558
2	Title of Examination Paper	Network Multiplayer Games
3	Module Code	M6X01105
4	Date and Time of Examination	13/5/19 12:00 pm
5	Examination Venue	1Q310
6	Seat Number	G050
7	Number of Answer Books handed in	

List the numbers of the questions you have attempted	Office Use
1 a)	13 ✓
1 b)	
2 a)	15 ✓
2 b)	
3 a)	16 ✓
3 b)	
4 a)	16 ✓
4 b)	
4 c)	
4 d)	
4 e)	
Total	60% ✓ GT 24/5/19

1

- a) Additional features for this transport protocol could include an acknowledgement system. Each time data is sent, it is sent with a value of x . When the data is received, it sends back an acknowledgement with value $x+1$, signalling the next data to send. If an acknowledgement is not sent back, the data x has not been received and can be retransmitted.

Expanding on this system, data can be sent in sets of ~~the~~ a determined size, and acknowledgements are only sent back after ~~the~~ all this data is received. While it takes longer for the server to discover if data has not been received, it spends less time waiting for acknowledgements. Retransmits in this system take longer to initiate, and longer to send since the whole set of data is retransmit due to not knowing the specific piece that was lost.

These systems ensure data will reach the machine reliably.

As data may take different amounts of time to reach the machine, data may not be received in the correct order. In the two systems above, the former solves this by acknowledging each piece of data as it arrives, ensuring its ordered. The latter system would need to make use of the numbers sent with data in order to keep data in the correct order.

- 1 b) ~~Non~~-preemptive kernels will interrupt tasks with low priority in order to run tasks with high priority. Certain tasks may fail in this kernel if they are not re-entrant functions. Consider the following:

```
func()
{
    temp = x + y ←
    y = x
    x = temp
    return()
}
```

If an interrupt occurs in this code where the arrow is, the value of temp may change and the task produces an incorrect outcome.

However, in this system high priority tasks will never wait for low priority tasks to finish. If functions are re-entrant, then this kernel will always provide responsive high priority tasks.

This ~~sys~~ kernel lends itself well to round robin scheduling, as long as functions are re-entrant. Since functions can be interrupted easily at any point, the round robin system can fairly allocate time.

~~Non~~-Pre-emptive kernels allow for non re-entrant functions, but high priority tasks would need to wait for low priority ones to complete before being run. This kernel ensures functions and tasks are completed correctly, but may produce an unresponsive system as a result. This may be suited to a round robin approach.

2a) An example of a non-deterministic MAC protocol is ethernet IEEE 803.3.

Benefits include NICs being able to determine if a data's destination is ~~for the~~ ~~the~~ the NICs without needing to copy the data to the machine.

Drawbacks include the existence of collision domains, and the solutions to this result in increased latency. If data traffic is high, this can also slow data transfer here.

For networked games, collisions can cause latency in gameplay, resulting in ~~choppy~~ choppy entity movement for example. Splitting collision domains can also add latency.

b) Since a finite amount of IP addresses exist, subnetting creates additional networks from one particular address.

Subnetting 199.7.10.0

Class C IP... 8 host bits...

Need to borrow bits for ~~subnets~~ subnets...

$$2^n - 2 \geq N$$

Where n is borrowed bits & N is subnets req.

So $2^n - 2 = 9$

Try $2^2 - 2 \geq 9$

Try $2^3 - 2 \geq 9$

Try $2^4 - 2 \geq 9$

$4 - 2 = 9$ NOT TRUE

$8 - 2 \geq 9$ NOT TRUE

$16 - 2 \geq 9$ TRUE

So we need to borrow 4 bits from host...

Checking host number...

$$NH = 2^h - 2$$

Where NH is available hosts, h is ~~borrowed~~ ^{remaining host} bits

PTO

$$NH = 2^4 - 2, \text{ as } 8 \text{ host bits} - 4 \text{ borrowed} = 4$$

$$NH = 14$$

Available host covers 12 max host requirement.

So subnetting 199.7.10.0 into 128 or 32 16 8 & 2 1

$$11001000.00001000.$$

Mask

$$11111111.11111111.11111111.11110000$$

10

10

10

3

a) Deadlock is when a task ~~it~~ will never have the resources it needs to finish. It occurs if the following 4 conditions hold true:

1) Hold and wait: task is holding one resource while waiting for another. Held resource is required for another task to release the resource ~~the~~ first requires.

2) ?

3) ?

4) ?

Starvation is when multiple tasks require a resource, but ~~it~~ end up waiting indefinitely as that resource is always used, leaving those tasks ~~a~~ uncompleted.

To prevent these, tasks that wait too long with a resource need to release the resource. Multiple resources could also be provided. Queues could also be used to allow tasks to finish.

b) A mutex (from 'mutually exclusive') is a way to ensure a single use resource (eg a printer), is never used by more than ~~one~~ task at a time.

Semaphores are different in that they can control a resource of a finite count.

These can be applied to games in various ways. Perhaps a server can run a specific number of game lobbies, so a task ~~that~~ creates these lobbies can only complete if the semaphore's count is above 0.

Perhaps a mutex is used to protect a 1v1 fighting game ~~connection~~ - when no opponent is found, mutex is open and closes when an opponent connects.

- 4 a) Consistency ensures that all players see the same representation of a world. If these worlds are not correctly synched, or synched close enough, some players may have an unfair experience.

Responsiveness ensures that actions the player makes are ~~correct~~ relayed to a server in a timely manner, so that the player's interaction with a game world feels real, and so that player actions occur at the right time on other machines.

Consistency can only be achieved by sending world state data from the server more frequently, resulting in a higher bandwidth required and a drop in responsiveness. 6

With less updates like this, responsiveness can increase, but the world the player sees is less accurate to the actual world state.

- b) Sending only a player's actions to a server, for example, only sending that a gun is fired, instead of also what it hit, ammo count etc, can help reduce the amount of data sent. Sending data only when something needs to update also reduces data sent. 2

- c) Entity movement can be calculated by a player's machine using the values such as velocity, direction etc, received at set intervals from the server. Player actions are more responsive the less data sent by the server, along with the less frequent updates. This frees up bandwidth for sending player actions. 3.5

Player machines can also estimate the result of an action before the server confirms it. ✓

- d) Interpolation uses two values at certain intervals and calculates values in between these depending on how close to either value's actual time we are. This can be used to fill in player movement between two positions at two times.

Extrapolation uses the current value to estimate a value at a future interval, and can be used to predict ~~extra~~ positions before receiving them.

- e) Smaller intervals between server updates can reduce this problem.

Interpolation between these intervals can help estimate actual positions.

Extrapolation can be useful but may result in choppy movement in the case of sudden movement changes.