COMP28112 Model Answers

Todd Davies

May 22, 2015

These are my own answers to the COMP281 papers. Use them at your own risk; they are probably all wrong.

2011 Paper

Describe one key difference between client-server and peer-to-peer applications. 2011.1.a (2 marks)

P2p does not have a central point of coordination; all nodes are the same in terms of functionality (though they may have different roles). A client server architecture is dissimilar to this in the fact that the machines that make up the network have different capabilities and responsibilities that cannot be changed.

Explain briefly what failures are known as Byzantine failures and what the Byzantine generals problem refers to. 2011.1.b (2 marks)

Byzantine failures are ones where processes send failed messages since they are malicious or faulty.

The Byzantine Generals problem refers to a situation where two generals are situated on the sides of a valley and want to attack a city in the valley. To win the battle, they must attack at the same time, but they cannot coordinate their attack with any guarantee of timing it properly since they can only send messages through the city where the messengers may be killed or the contents of the message changed.

Describe briefly the two-phase commit protocol. 2011.1.c (2 marks)

When a client wants to commit, it sends a message to a server. The server then asks every client whether they are ready for a commit. Each client replies with a yes/no. If there is a no, then a `global_abort` message is sent, otherwise a `global_commit` message is sent. Upon receiving a global commit or abort message, each client does exactly that, which ensures that only one operation is done for all clients.

Explain briefly why somebody would choose to use Cloud Computing. 2011.1.d (2 marks)

- You dont have to buy and own hardware, you can just rent time from other peoples hardware to do computing.
- Clouds are large and therefore provide capacity for horizontal scaling.
- Clouds make it easier to have the computing resources to apply techniques such as load balancing, replication and parallelization. This can make it easier to cope with variable demand.

Why is it practically impossible to achieve strict consistency in a distributed system? 2011.1.e (2 marks)

It is practically impossible to achieve strict consistency in a distributed system since latency is not zero, so you can never know exactly what is going on in a remote system at any one time. Messages can get lost or mutated too, which further complicates the sending and updating of state between systems.

Suppose the C function below is to be made available to remote processes using RPC. What particular implementation problem does this highlight? Why is the normal solution only partly satisfactory? 2011.1.f (2 marks)

```
void doIt (int *p, int *q) { (*p)++ ; (*q)-- ; return ; }
```

You would have to serialise and deserialise the values of p and q to have this function run over RPC, because it uses pointers as opposed to actual values. You would also have to update the value of the pointers either as they are updated during the RPC functions execution or after the function has finished executing.

Distinguish carefully between a name server and a directory server. 2011.1.g (2 marks)

A name server takes the name of a resource and resolves it into the address of the resource (e.g. a URL). A directory server takes attributes about an object (e.g. the amount of RAM of a server or the age and course studied of a student) and returns information about objects that have the same attributes (think LDAP)

What is meant by the term causally ordered multicast? 2011.1.h (2 marks)

Ummmmm? Maybe take a look here: `http://www.cse.buffalo.edu/~stevko/courses/cse486/spring13/lectures/12-multicast2.pdf`

Looks like every process has a vector clock and when a message is received, the process waits until it can preserve the causal ordering before accepting the message. This means it waits until messages that had been previously sent to it (according to the vector clock) arrive before accepting the new message. Until then, the message is buffered.

What properties are provided by a secure channel? 2011.1.i (2 marks)

A secure channel is one that third parties cannot eavesdrop on. They might be able to see the data passing through, but it will be encrypted somehow so they wont be able to make sense of it.

The Andrew File System (AFS) uses callback promises. Explain what this means. 2011.1.j (2 marks)

Dunno? Google turns up not much...

Im guessing that it is a mechanism where if you send a remote server a message, it promises to reply back to you and will cause a callback to execute in your code when it does.

Explain briefly why some applications are not parallelisable. Describe Amdahls law and explain what it can be used for. 2011.2.a (3 marks)

Some applications are not parallelisable because they have operations that are interdependent. For example, if we were building a machine to compute how long it takes for the collatz conjecture to reach 1 given an input number, we could not parallelise it since each stage of the conjuncture depends on the last. Amdahls law dictates how much we can speed up the execution of a program given how much of the program is parallelisable. The law is:

$$\frac{1}{\text{serial portion} + (\frac{1}{\text{num threads}} \times \text{parallel portion})}$$

Explain briefly what the four properties commonly denoted by the acronym ACID are when referring to transactions. 2011.2.b (3 marks)

ACID stands for:

Atomic - All or nothing principle; either the transaction is committed and its changes are applied, or no state is changed as a result of the transaction.

Consistent - Each transaction moves the system from one valid and consistent state to another.

Isolated - Each transaction executes independently from all other transactions (even if they may be happening concurrently). The effect of transactions in progress is hidden from all other transactions.

Durable - Once a transaction is committed, it stays committed even in the event of failure such as power loss etc

A service is replicated onto 3 computers. 2011.2.c (6 marks)

- The first computer, A, has a mean time between failures of 2 days.
- The second computer, B, has a mean time between failures of 3.5 days.
- The third computer, C, has a mean time between failures of 12 days.

When a failure occurs, it takes on average 12 hours to fix.

What is the availability of the replicated service? 2011.2.c.i (2 marks)

We need to find the chance that all the computers will fail at the same time. It is easy to find the up-times for each individual computer:

A - 75%

B - 85.7%

C - 95.83%

The chance that these will all fail at the same time is: $(1 - 0.75) * (1 - 0.875) * (1 - 0.9853) = 0.00046 = 0.046\%$
Therefore, the uptime is going to be $100 - 0.046 = 99.954\%$

What would the availability of the replicated service be if only computers A and B were used? 2011.2.c.ii (2 marks)

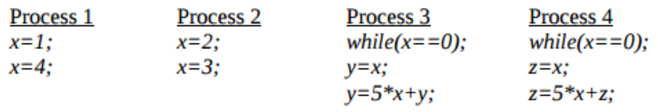
If only A and B were used, it would be:

$1 - ((1 - 0.75) * (1 - 0.875)) = 0.96875 = 96.88\%$

Describe how in the general case of n computers, each with a mean time between failures f_i and an average time to fix a failure t_i you would choose the two computers that provide the highest availability. 2011.2.c.iii (2 marks)

Rank the computers in ascending order of t_x/f_x, and take the top n elements.

The following four processes access a shared variable x. Each process accesses a different replica of the store used to hold this variable. Before any process starts executing, the value of x is 0 in all the replicas. 2011.2.d (8 marks)



```
Process 1    Process 2    Process 3        Process 4
x=1;         x=2;         while(x==0);     while(x==0);
x=4;         x=3;         y=x;             z=x;
                          y=5*x+y;         z=5*x+z;
```

When all four processes have completed executing the statements given, are 7 and 14 possible values of y and z respectively, if the replication uses the sequential consistency model? Justify your answer. 2011.2.d.i (4 marks)

With a sequential consistency model, it is not possible for the final values to be $y = 7$ and $z = 14$, since for y to equal 7, x must change state while process 3 is running, and this is not possible in a sequential consistency model since only one process can execute at once.

The possible values of x when process three actually starts are $x = 4$ and $x = 3$, none of which let $y = 7$.

When all four processes have completed executing the statements given, are 7 and 14 possible values of y and z respectively, if the replication uses the causal consistency model? Justify your answer. 2011.2.d.ii (4 marks)

Using a causal consistency model, we still can't get $y = 7, z = 14$, since we need $y = 5 \times (x = 1) + (y = 2)$ and $z = 5 \times (x = 2) + (z = 4)$, but this can never happen, since for $z = 4$, we need to have done $x = 1$, but we need to have done $y = 2$ before that, which means that we would have already done $x = 2$.

Describe in detail the Bully algorithm for the election of a leader in a distributed system. 2011.3.a.i (8 marks)

The Bully Algorithm works like this:

- An initiating client will send a message to all other clients with a higher identifier than itself.
- Any client receiving an election message will then start its own election, sending messages to clients with higher identifiers than itself.
- Only when a process gets no replies (within a timeout) is it considered elected. It then sends a message to all other processes announcing its election.

In this manner, any process that receives a message should always send a message back to the sender

Carefully explain all the assumptions made in the Bully algorithm. 2011.3.a.ii (4 marks)

- The timeouts are reasonable
- All processes know about all other processes (and their loads)
- Messages are delivered quickly
- Messages are delivered reliably

Describe one application for which the Bully algorithm might be applied, indicating why a leader is needed. 2011.3.b (2 marks)

In a peer-to-peer protocol (such as bit torrent), the Bully algorithm can be used to elevate a node to coordinator status if the client that was coordinating the network went offline or started to get a too high load.

In a system containing 6 computers, identified by the integers 1-6, the leader is chosen by the Bully algorithm to be the live one with the highest identifier. Assume for this part that all messages are delivered promptly, and that the computers and the network are entirely reliable. 2011.3.c (6 marks)

How many messages in total are sent so that the computer with identifier 1 after it is rebooted can learn the identity of the leader by triggering an election? Take care to explain your working! 2011.3.c.i (6 marks)

Since identifier 1 is the lowest node, it will exhibit worst case behaviour for the bully algorithm, $O(n^2)$.

This is because 1 will send five messages out (to each other process) to start the election. Client 2 will then send out four messages (to the ones bigger than it) etc, until there are $5 + 4 + 3 + 2 + 1 + 0 = 15$ messages sent. Since every client receiving an election message replies to the sender, then 15 replies will be sent (in addition to the 15 initial message). Then, the winner of the election (6) will send a message out to say that it is elected, which is another 5 messages, bringing the total to 35.

Repeat (i) with the computer rebooted and needing to discover the leader being that with identifier 5 instead. 2011.3.c.ii (2 marks)

If 6 wasn't online anymore, then the number of messages sent would be $(5+4+3+2+1)+(4+3+2+1)+4 = 29$,

2012 Paper

What is a Java servlet? 2012.1.a (2 marks)

A Java Servlet is a Java program with the capabilities of a server. It could host web pages, provide an API endpoint or other services and is most commonly used to serve content via the `HTTP` protocol.

What is the main assumption on which Cristians clock synchronisation algorithm is based? 2012.1.b (2 marks)

The time for a message to go from machine A to machine B is (roughly) the same as the time it would take for a message to go from machine B to machine A. This is most often accurate for routes with small round trip times.

Explain the difference between a name server and a directory server. 2012.1.c (2 marks)

A name server takes a name, matches it to an object and returns attributes about the object. A directory server takes attributes, matches them to an object and returns more attributes about the object.

Explain briefly what is meant by the term middleware. 2012.1.d (2 marks)

Middleware is software that sits between a client application and the operating system. It provides services to the client application that the operating system does not, such as RPC stubs.

It can also provide an abstraction from the OS to mask the heterogeneity of platforms used in distributed systems (some apps will run on Windows, others on Linux, some on x64, some on ARM architectures etc).

Why is it practically impossible to achieve strict consistency in a distributed system? 2012.1.e (2 marks)

Strict consistency is when any read to a shared data item returns the most recent write operation on that data item. This means there must be an absolute time ordering of all accesses. Unfortunately, since (as we know from the eight fallacies of Distributed Computing), the latency of any network is not zero and messages are not reliable. This means that any message we send to update other machines about a change of state may not be sent. Since there is no way of getting an absolute global clock or getting any global state, then we cannot achieve real time memory consistency across all nodes, and therefore cannot achieve strict consistency in the system.

Traditional RPC mechanisms cannot handle pointers. What is the problem? 2012.1.f (2 marks)

In order to handle pointers with RPC, you must serialise the datastructure into a message so that it can be constructed from the same message at the other machine. When the reply is received back at the sender, it can be deserialised again and the values in the datastructure that were changed on the remote machine can be updated in local memory. Traditional RPC mechanisms didn't have this facility.

What is meant by parameter unmarshalling? 2012.1.g (2 marks)

Extracting the parameters from an RPC message (can be binary data, ascii etc) that was sent from another machine into memory of the local machine.

What is the key difference between caching and replication 2012.1.h (2 marks)

Replication is keeping two fully fledged entities up to date with each other in order to provide redundancy or distribute the load of a service over more machines.

Caching is merely a small piece of hardware or a small program that remembers the results of expensive computations after they've been done once and returns the result without doing any computation if the same request comes again. Rarely is there any business logic in a cache.

Explain briefly why somebody would like to use Cloud Computing. 2012.1.i (2 marks)

- You don't have to buy and own hardware, you can just rent time from other people's hardware to do computing.
- Clouds are large and therefore provide capacity for horizontal scaling.
- Clouds make it easier to have the computing resources to apply techniques such as load balancing, replication and parallelization. This can make it easier to cope with variable demand.

In the context of lab exercise 2, what would you do to launch a denial of service attack against the server? 2012.1.j (2 marks)

I would find the most computationally expensive operation on the server (maybe list free slots, since it access the database a lot) and set up a tight loop to continuously send requests for this operation to the server. To increase the impact of my attack, I would try and execute it on a machine with a high network bandwidth and low latency, as well as possibly using multiple threads or machines to launch the attack.

Explain briefly what is wrong with the assumption latency is zero in the context of distributed computing. Why is it considered a common fallacy? 2012.2.a (3 marks)

The latency between two computers can never be zero, for a number of reasons. From a purely physical standpoint, information cannot travel faster than the medium through which it is being carried, and even the fastest medium available to us, light has a speed limit. Most of the time, latency is significantly greater than the speed of light divided by the distance travelled, which is because most of the time, the data passes through a network as packets, and is forwarded on from node to node each time. Since multiple computers are used to facilitate this, each one having some small amount of lag, the time between the start point and the destination increases further. At any point along its route, the data could get stuck in a buffer, have to take a detour etc.

Many (new) programmers haven't created applications intended to use distributed systems before, and thus don't realise that waiting for a remote resource, RPC call etc can (will) take orders of magnitude more time than a load from memory might.

Explain briefly what the four properties commonly denoted by the acronym ACID are when referring to transactions. 2012.2.b (3 marks)

Atomicity - Each operation/transaction must either succeed or fail; there is no in-between. If the operation fails, then the state of the system must be exactly as it was before the start of the operation.

Consistency - Each operation or transaction must bring the state of the system from one valid state to another.

Isolation - Operations or transactions happening in parallel must execute as though they were running in a serial manner.

Durability - Once a transaction is committed or an operation is completed, the system will not roll back to a previous state in the event of failure or power loss etc.

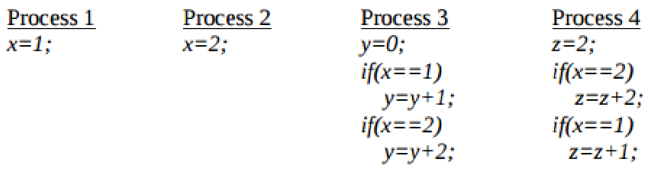
Outline the Byzantine Generals problem, and illustrate how one of three being a traitor makes a solution impossible, whereas with one of four it is achievable. 2012.2.c (6 marks)

There are three Byzantine generals on the battlefield. One is the commander and the other two are his deputies. Each of the 'good' generals knows that the others could be corrupt, but not which one (or any of them). In order to make sure they never do the wrong thing, they agree that a commander must receive the same message from both other commanders before he does anything.

However, if one commander is corrupt, they can change every message they receive and make it so that the other two commanders will never do anything since the other two commanders will always receive conflicting messages.

However, if there is one corrupt commander within four (or more) commanders, then they can never cause a deadlock like that, because they will always be a minority among the other non-corrupt commanders.

The following four processes access a shared variable x. Each process accesses a different replica of the store used to hold this variable. Before any process starts executing, the value of x is 0 in all the replicas. 2012.2.d (8 marks)



```
Process 1    Process 2    Process 3     Process 4
x=1;         x=2;         y=0;          z=2;
                          if(x==1)      if(x==2)
                            y=y+1;        z=z+2;
                          if(x==2)      if(x==1)
                            y=y+2;        z=z+1;
```

When all four processes have completed executing the statements given, are 3 and 5 possible values of y and z respectively, if the replication uses the sequential consistency model? Justify your answer. 2012.2.d.i (4 marks)

No, because for $y = 3$ and $z = 5$, the values of x must change during the execution of process 3 and process 4. This cannot occur in a sequential consistency model, since each process must run one after another, and process 3 and 4 don't change the value of x.

When all four processes have completed executing the statements given, are 3 and 5 possible values of y and z respectively, if the replication uses the causal consistency model? Justify your answer. 2012.2.d.ii (4 marks)

No, since for $y = 3$ and $z = 5$ require all of the if statements to evaluate to true and execute all of the additions, since $0 + 1 + 2 = 3$ and $2 + 2 + 1 = 5$. Since process 3 requires x to be 1 first and process 4 requires it to be 2 first, we can only ever get three of the four if statements to execute. This means we can get either $y = 3$ or $z = 5$ (or none of them), but not both.

Explain briefly what is the role of a client stub and a server stub in RPC 2012.3.a (3 marks)

A stub is middleware that sits between an application and the OS. Its role is to facilitate an RPC call, but its exact 'job description' varies based on whether its a client or server stub:

Client:

1. First, the stub will marshal the parameters to the remote method into a text or binary format.
2. The client stub then sends the request to the server (where the server stub will handle it) and waits for a reply.
3. Once it receives a reply the client stub unmarshals the data from the servers reply back into memory and returns back to the client application.

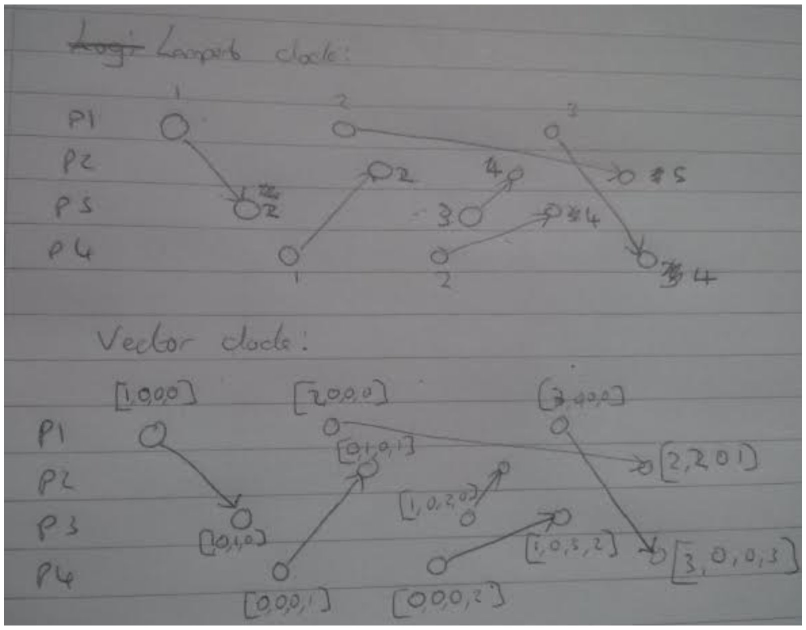
Server:

1. When the server receives a message from a client sub, it unmarshals the parameters and puts them in memory.
2. Then it calls the desired procedure on the server application
3. When the procedure has finished, it marshals the data back into a message and sends it back to the client.

Describe in detail how the two-phase commit protocol can implement distributed transactions 2012.3.b (5 marks)

- There is a server that acts as a coordinator.
- When a client wants to commit a transaction, it sends a message to the server asking to commit.
- The server then sends a message to all clients asking if they are in a state where they can commit.
- Each client sends a reply (either yes or no) as to whether they can commit.
- If all of the clients are in a position to commit, then the server will send out a `global_commit` message and all clients will commit, otherwise, it will send out a `global_abort` message and all clients will abort the transaction.

Consider the figure below, which shows 4 processes and a number of communication events taking place over a period of time. Calculate the value of Lamport clocks and vector clocks for each of the 12 events. You can assume that all logical clocks start initially with zeros. 2012.3.c (6 marks)



In a system containing 7 computers, identified by the integers 1-7, the coordinator is chosen by the Bully algorithm to be the live one with the highest identifier. Assume for this part that all messages are delivered promptly, and that the computers and the network are entirely reliable. At a certain point in time the coordinator (computer 7) and the computer with the second-highest identifier (computer 6) crash. How many messages in total are sent if the computer with identifier 1 is the computer discovering the crash and triggering an election? You need to count all three types of messages that the algorithm sends. You can assume that computers with identifiers 6 and 7 remain crashed during the election. 2012.2.d (6 marks)

First, 1 will send a message to all computers higher than it $(2, 3, 4, 5, 6, 7)$ totalling six messages. Each live process receiving a message from 1 will then send messages to processes higher than it:

Computer	**Election messages sent**
1	$2, 3, 4, 5, 6, 7$
2	$3, 4, 5, 6, 7$
3	$4, 5, 6, 7$
4	$5, 6, 7$
5	$6, 7$
6	Crashed!
7	Crashed!

This totals $6 + 5 + 4 + 3 + 2 = 20$ messages sent.

Each process then receives a reply from the live ones above it:

Computer	**Replies received**
1	$2, 3, 4, 5$
2	$3, 4, 5$
3	$4, 5$
4	5
5	
6	
7	

This totals $4 + 3 + 2 + 1 = 10$ messages.

The winner of the election (5) then sends a message to all other processes to announce its win (six more messages), so a total of $20 + 10 + 6 = 36$ messages are sent.

2013 Paper

Explain briefly what is meant by the term middleware. 2013.1.a (2 marks)

Software that sits in between client applications and the operating system that abstracts the details of implementing programming tasks and masks the heterogeneity of the underlying platforms from the client application.

Explain briefly what failures are known as Byzantine failures. 2013.1.b (2 marks)

When a remote system either:

- Does not reply to messages
- Sends faulty replies to message with miscellaneous data
- Sends maliciously crafted messages
- Acts inconsistently when interacting with different components

Describe briefly the two-phase commit protocol. 2013.1.c (2 marks)

When a client wants to commit, it sends a message to a server. The server then asks every client whether they are ready for a commit. Each client replies with a yes/no. If there is a no, then a `global_abort` message is sent, otherwise a `global_commit` message is sent. Upon receiving a global commit or abort message, each client does exactly that, which ensures that only one operation is done for all clients.

What is meant when a service is provided with at least once semantics? 2013.1.d (2 marks)

The application may send the RPC call multiple times before it receives word from the server that the RPC call has taken place and was successful. This may result in the RPC call being executed multiple times.

Why is it practically impossible to achieve exact synchronisation of clocks in a distributed system? 2013.1.d (2 marks)

The latency in a network is variable, so you can never know exactly how long a message took to reach a server. Because of this, if one server sends its current clock to another server, the latter one won't know how long it took for the message to reach it and therefore it won't know how much to increment the clock by.

Methods such as Cristian's algorithm and the Berkeley algorithm attempt to solve this problem, but can only do so with some assumptions and to a limited accuracy.

When using Java RMI, what is the purpose of the RMI registry? 2013.1.f (2 marks)

To act as a central repository for computers to access remote objects. Clients can interrogate the registry using a string and get an object back. This allows RMI methods to have parameters that are strings that point to objects in the repository so you can pass arbitrary data structures and objects easily.

What is meant by parameter marshalling? 2013.1.g (2 marks)

When an RPC call is made, the parameters cannot be sent directly (since they may be pointers etc), so they must be serialised into a blob of text or binary data before they are sent (marshalled) and deserialised at the other end by the server (unmarshalled).

What is the key difference between caching and replication? 2013.1.h (2 marks)

Replication is keeping two fully fledged entities up to date with each other in order to provide redundancy or distribute the load of a service over more machines.

Caching is merely a small piece of hardware or a small program that remembers the results of expensive computations after they've been done once and returns the result without doing any computation if the same request comes again. Rarely is there any business logic in a cache.

Explain briefly what Littles Law is. 2013.1.i (2 marks)

Little's law dictates that the average number of elements in a queue is:

$$\text{av. number} = \text{av. time between arrivals} \times \text{time to process one}$$

This allows us to estimate how a system will handle load and what its queue size will be (and if its appropriate).

In the context of lab exercise 2, what would you do to launch a denial of service attack against the server? 2013.1.j (2 marks)

- Estimate the most expensive server operation (maybe listing slots)
- Create a program that loops indefinitely with a request for the expensive operation.
- Run the program on a fast computer with a high bandwidth

Hopefully, the server will be too busy processing my requests to have time for other users, therefore I would (indirectly) be denying them service.

Explain briefly why some applications are not parallelisable. Describe Amdahls law and explain what it can be used for. 2013.2.a (4 marks)

Amdahl's law can tell us the maximum speed up we can achieve if we parallelised all parts of a program that don't need to be run in a serial manner.

Some parts of applications are not parallelisable because they have interactions and data dependencies that must take place in a specific order. For example, if we were computing the Fibonacci numbers, we couldn't parallelise it because each new number depends on the last two, and therefore we need to know the last two numbers before generating the next.

However, if we used the output from the Fibonacci number sequence for some operation, say to render an image of a snail shell with Fibonacci proportions, then the rendering could take place in parallel (since it's probably a more intensive operation). If the Fibonacci portion of our program was about 20% of computing time, and the rendering was 80%, then our maximum speed up would be:

$$\frac{1}{0.2 + (\frac{1}{n} * 0.8)}$$

When n (the number of threads) tends to infinity (if we could use an infinite level of parallelism), then the speed up tends towards 5.

Explain briefly what the four properties commonly denoted by the acronym ACID are when referring to transactions. 2013.2.b (4 marks)

Atomicity:
A transaction either occurs (commit) or it is does not (abort), there is no in between. Either all of the effects of a transaction happen, or none of them do, which is the all or nothing principle.

Consistency:
Each transaction moves the state of the system from one valid state to another.

Isolation:
Even if transactions happen in parallel, they must behave as though they happened in a sequential manner. No transaction in progress can affect another transaction.

Durability:
Even in the event of a power loss, system failure or other event, once a transaction is committed, it should stay committed, and not be rolled back.

A service is replicated onto 3 computers. 2013.2.c (6 marks)

- **The first computer, A, has a mean time between failures of 2 days.**
- **The second computer, B, has a mean time between failures of 3.5 days.**
- **The third computer, C, has a mean time between failures of 12 days.**

When a failure occurs, it takes on average 12 hours to fix.

What is the availability of the replicated service? 2013.2.c.i (2 marks)

Up time of A: $\frac{48}{52}$

Up time of B: $\frac{84}{96}$

Up time of C: $\frac{288}{300}$

The down time chance is $(1 - \frac{48}{52}) * (1 - \frac{84}{96}) * (1 - \frac{288}{300}) = \frac{1}{2600}$.

The availability is $1 - \frac{1}{2600} = 99.96\%$

What would the availability of the replicated service be if only computers A and B were used? 2013.2.c.ii (2 marks)

If only A and B were used, then the downtime would be:

$$d = (1 - \frac{48}{52}) * (1 - \frac{84}{96}) = \frac{1}{104}$$

And the availability would be $1 - \frac{1}{104} = 99.04\%$

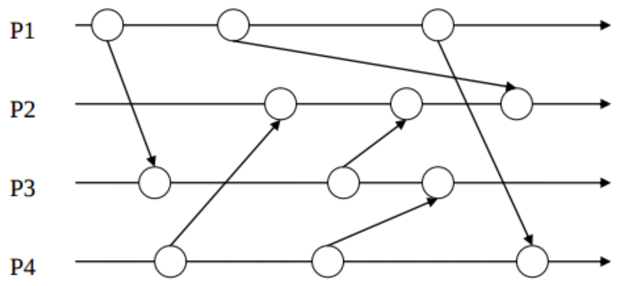
Describe how in the general case of n computers, each with a mean time between failures if and a time to fix the failures ti you would choose the two computers that provide the highest availability. 2013.2.c.iii (2 marks)

I would sort the computers by:

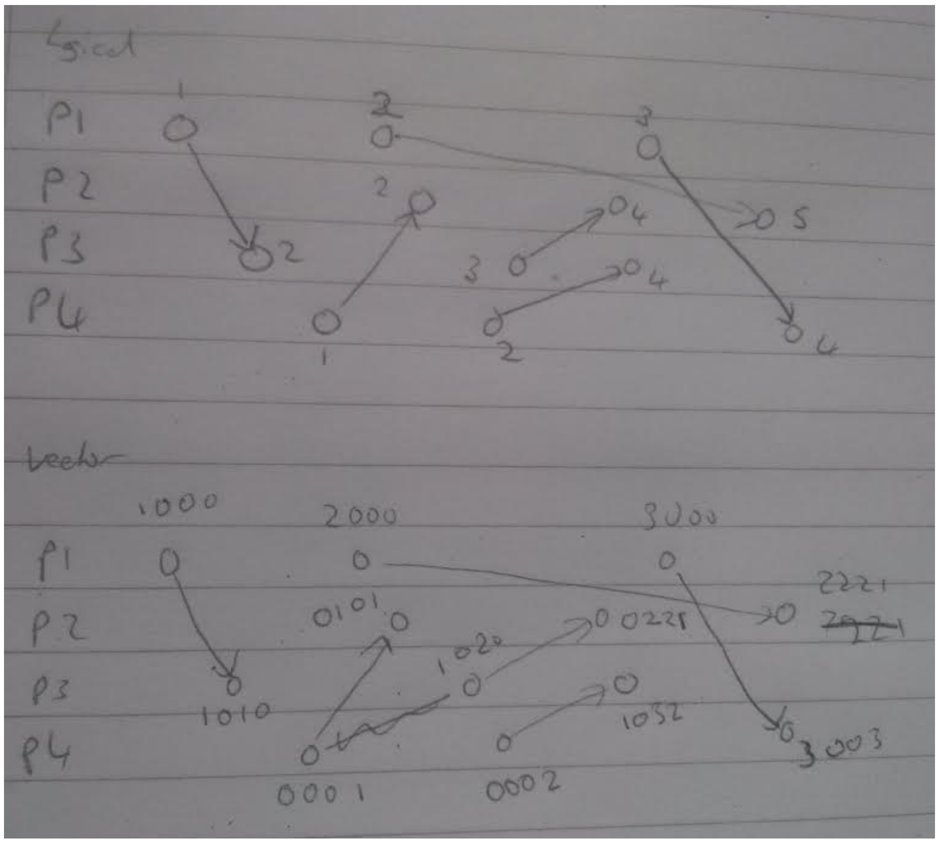
$$\frac{f_x}{f_x + t_x}$$

Then I would use the top n computers.

Consider the figure below, which shows 4 processes and a number of communication events taking place over a period of time. 2013.2.d (6 marks)



Calculate the value of Lamport clocks and vector clocks for each of the 12 events shown above. You can assume that all logical clocks start initially with zeros.



Explain briefly why the assumption latency is zero is considered a common fallacy in distributed computing. 2013.3.a (2 marks)

The latency between two computers can never be zero, for a number of reasons. From a purely physical standpoint, information cannot travel faster than the medium through which it is being carried, and even the fastest medium available to us, light has a speed limit. Most of the time, latency is significantly greater than the speed of light divided by the distance travelled, which is because most of the time, the data passes through a network as packets, and is forwarded on from node to node each time. Since multiple computers are used to facilitate this, each one having some small amount of lag, the time between the start point and the destination increases further. At any point along its route, the data could get stuck in a buffer, have to take a detour etc.

Many (new) programmers haven't created applications intended to use distributed systems before, and thus don't realise that waiting for a remote resource, RPC call etc can (will) take orders of magnitude more time than a load from memory might.

Describe all the operations that take place during a Remote Procedure Call (RPC). 2013.3.b (5 marks)

First, the client application will call the client stub with the procedure it wants to execute and the parameters that are needed to do so. The client stub then marshalls the parameters into a blob which is then sent to the server. The server receives the message and the server stub unmarshalls the blob back into parameters, which are then passed to the procedure on the server and the procedure is ran. When the procedure is finished the server stub marshalls the (possibly mutated) parameters back into a blob and sends it back to the client, where the client stub will then unmarshal the parameters, update their values in main memory and return to the client application.

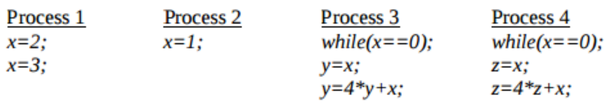
Describe clearly how the Bully algorithm to elect a coordinator works. 2013.3.c (5 marks)

The Bully Algorithm works like this:

- An initiating client will send a message to all other clients with a higher identifier than itself.
- Any client receiving an election message will then start its own election, sending messages to clients with higher identifiers than itself, and also send an answer message back to the election message's sender.
- Only when a process gets no replies (within a timeout) is it considered elected. It then sends a message to all other processes announcing its election.

In this manner, any process that receives a message should always send a message back to the sender. The upper bound on the number of messages is $O(n^2)$, and timeouts can also make the election slow.

The following four processes access a shared variable x. Each process accesses a different replica of the store used to hold this variable. Before any process starts executing, the value of x is 0 in all the replicas. 2013.3.d (8 marks)



Process 1	Process 2	Process 3	Process 4
x=2;	*x=1;*	*while(x==0);*	*while(x==0);*
x=3;		*y=x;*	*z=x;*
		*y=4*y+x;*	*z=4*z+x;*

When all four processes have completed executing the statements given, are 6 and 13 possible values of y and z respectively, if the replication uses the sequential consistency model? Justify your answer. 2013.3.d.i (4 marks)

2014 Paper

Explain briefly why the lack of homogeneity is a challenge when developing distributed systems. 2014.1.a (2 marks)

Since most of the individual computers that make up distributed systems will differ, we need to come up with ways for them to communicate in such a way that they can have meaningful interactions. This involves creating Interface Definition Languages, middleware etc to help them inter-operate even when they might be running different architectures.

Explain briefly why the assumption latency is zero is considered a common fallacy in distributed computing. 2014.1.b (2 marks)

Many (new) programmers haven't created applications intended to use distributed systems before, and thus don't realise that waiting for a remote resource, RPC call etc can (will) take orders of magnitude more time than a load from memory might since it must travel through (many) different systems over different protocols and even through different countries, where at any stage it may be delayed and thus the latency increased.

Explain briefly what publish-subscribe messaging is. 2014.1.c (2 marks)

Public-Subscribe messaging is when a 'pub-sub' server is designated to take messages from publishers and forward them to subscribers. The server provides a central point at which systems can push their messages to and receive their messages from and know that the message will be sent to/from the correct places.

This is especially handy if one client wishes to push a message to many (or unknown amounts of) other clients. Instead of having to send n messages, it can send one message to the pub-sub server and the server will send out the correct amount of messages to the subscribed clients. This helps abstract the network topology away from clients.

In a distributed system, what is the purpose of an IDL? 2014.1.d (2 marks)

An Interface Description Language defines an interface with which different systems who may be running different architectures and different operating systems etc can serialise data to and from and communicate safely.

They are commonly used by RPC software to facilitate the actual message sending of the RPC calls.

In the context of RPC, what is copy-restore and what is it used for? 2012.1.e (2 marks)

Since a local client and a remote server (almost certainly) do not share memory space, copy-restore is a way of letting one mutate the memory of another remotely. On making an RPC call, the parameters for the method are serialised and sent to the receiver, where they are deserialised again. While the procedure is running, the parameters may be changed, and when the RPC call has finished, they are serialised and sent back to the original machine, where their new values are written back to memory.

What must a server do to provide at most once semantics to its clients? 2014.1.f (2 marks)

It must provide an acknowledgement to clients when it has received an RPC call and successfully unmarshalled the parameters. This means that clients can know not to re-sent the RPC call (since otherwise they wouldn't know that it was received).

See `http://stackoverflow.com/questions/13330067/rpc-semantics-what-exactly-is-the-purpose` for (what looks like) good description of at most once and at least once semantics.

Explain briefly what failures are known as Byzantine failures. 2014.1.g (2 marks)

When a remote system either:

- Does not reply to messages
- Sends faulty replies to message with miscellaneous data
- Sends maliciously crafted messages

Then it counts as a Byzantine failure.

In the context of data replication, explain briefly what eventual consistency is. 2014.1.h (2 marks)

When a protocol guarantees that data will eventually be consistent across multiple systems, but it does not specify a time duration before the data will be consistent (where consistency means to have the same data values).

When using Java RMI, what is the purpose of the rmiregistry? 2014.1.i (2 marks)

As a map between a String name of an object and the object's value. Clients can query the rmiregistry with the name of an object and get the object back. In this way, it acts as a distributed memory space for distributed systems, since all systems can read and update the values stored in it.

The rmiregistry means that you aren't limited to just passing values in RPC calls, you can pass references to objects and datastructures in the rmiregistry and the system you're calling with the RPC call can look them up in the rmiregistry.

In the context of lab exercise 2, what would you do to launch a denial of service attack against the server? 2014.1.j (2 marks)

Find the most expensive operation I can on the server (maybe listing the free slots), and set up a program on a machine with a fast processor and high bandwidth to run in a loop sending a request to the server every time it loops. Thousands or tens of thousands of requests per second could be sent in the right conditions, which could overload the server, cause it to cease responding (or at least increase its response times) and deny its service to other users.

Explain briefly what the role of a client stub and a server stub is in RPC. 2014.2.a (2 marks)

Client stub:

1. The client stub receives a request from the client application to execute an RPC call.
2. It marshalls the parameters of the call into a binary or text blob and sends it (often in a format specified by an IDL) to the server.
3. The client waits for a reply.
4. When a reply is received, the client unmarshalls the parameters and updates them in the main memory.

Server stub:

1. When the server receives an RPC call from a client, it calls the server stub.
2. The stub unmarshalls the parameters of the call and calls the local procedure with them.
3. Once the procedure is finished, the parameters are marshalled back again and sent back to the client.

Explain briefly what is meant by logical (Lamport) clocks and vector clocks. What property is captured by vector clocks that is not if Lamport clocks are used? 2014.2.b (3 marks)

Logical and vector clocks aim to provide a happens-before ordering of events in distributed systems. A logical clock is incremented every time an event occurs in a distributed system, and it attached to all messages. If a client receives a message with a higher logical clock than it, then its clock is updated to be that value, and then it is incremented to signify that an event occurred (a message received event).

A vector clock is like a logical clock, except each system keeps track of the clock of each other system in a vector. In this manner, causality is captured by vector clocks and not by logical clocks.

Explain briefly what the four properties commonly denoted by the acronym ACID are when referring to transactions. 2014.2.c (4 marks)

Atomicity - Each operation/transaction must either succeed or fail; there is no in-between. If the operation fails, then the state of the system must be exactly as it was before the start of the operation.

Consistency - Each operation or transaction must bring the state of the system from one valid state to another.

Isolation - Operations or transactions happening in parallel must execute as though they were running in a serial manner.

Durability - Once a transaction is committed or an operation is completed, the system will not roll back to a previous state in the event of failure or power loss etc.

Describe in detail how a centralised coordinating process can provide a mutual exclusive access service in a distributed system. 2014.2.d.i (3 marks)

A coordinating process acts as a lock server. If a process wants to enter a section designated to be executed only in a mutually exclusive manner, then it must first obtain a lock. It does this by asking the lock server for a lock, executing the mutually exclusive section when it gets it, and releasing the lock again once its done. If there is already a process in the mutually exclusive section when another process asks for a lock, the lock server will put the latter process in a queue until the first has finished with the lock.

When the machine supporting such a process gets overloaded with other tasks it needs to find the least loaded machine in the network, and pass over the provision of the mutual exclusive access service to a process on that machine. Two algorithms are being considered for this. The first is to have the server ask each machine about its workload and then notify all the clients with the identity of the new server. The second is to use a ring-based election, initiated by the current server. 2012.2.d.ii (8 marks)

Fully describe the latter, clearly stating any assumptions you make and compare it with the former with respect to the number of messages passed.

A ring based election proceeds as follows:

1. The initiator of the election will send a message to the next node containing its identifier (a pair of the name of the server and the reciprocal of its load).
2. Each node will then forward the message if its identifier (the load) is lower than that of the message, or send its own identifier if its higher.
3. The message goes around the circle until one node receives its own identifier.
4. This means that the message has gone once around the ring and no other machine is more eligible for the role of coordinator, so the receiving machine considers itself elected.
5. The newly elected coordinator then sends a message around the ring announcing its election.

The assumptions are that messages are sent reliably, and that no processes crash.

This takes $O(3n-1)$ messages in the worst case, whereas the more simpler one takes $O(3(n-1)) = O(3n-3)$ messages, which although two fewer messages are sent, is potentially worse since it requires each server to know about all other servers.

Describe clearly all the operations that take place during a Remote Procedure Call (RPC). 2014.3.a (4 marks)

First, the client application will call the client stub with the procedure it wants to execute and the parameters that are needed to do so. The client stub then marshalls the parameters into a blob which is then sent to the server. The server receives the message and the server stub unmarshalls the blob back into parameters, which are then passed to the procedure on the server and the procedure is ran. When the procedure is finished the server stub marshalls the (possibly mutated) parameters back into a blob and sends it back to the client, where the client stub will then unmarshal the parameters, update their values in main memory and return to the client application.

Two computers are used to provide a replicated service. Each computer has a mean time between failures of 12 days; a failure takes on average 12 hours to fix. What is the availability of the replicated service? 2014.3.b (3 marks)

Probability of failure $= 1 - \frac{12}{(12*24)+12} = 0.96$.

The probability of both machines failing at the same time is $0.96^2 = 0.9216 = 92.16\%$.

Consider a client-server application, which consists of 100 services provided by some server. Ten of these services must be executed strictly one after the other, not in parallel with any other services. The remaining 90 services may be executed concurrently and in any order. Assume that each service takes the same time to execute. What is the maximum speedup that can be obtained for the application if multiple identical servers are used to provide the required services? 2014.3.c (3 marks)

If ten services have to be executed in a serial manner, and the last 90 can be executed in parallel then the maximum speed up using a number of parallel processors is (using Amdahl's law):

$$\text{speedup} = \frac{1}{0.1 + (\frac{1}{\infty} * 0.9)} = 10\times$$

Consider a simple server that carries out client requests without accessing other servers. Explain why it is generally not possible to set a limit on the time taken by such a server to respond to a client request. What would need to be done to make the server able to execute requests within a bounded time? 2014.3.d (4 marks)

Since latency is variable, although we could try and ensure that the server sends a response back to the client n seconds after it has received a request, we cannot make guarantees about how long it will take for the message to get from the client to the server and back again.

Furthermore, we don't know how many requests the server might be dealing with, if it has a queue of 1000000 requests, then it probably won't reply in the bounded time.

The only way we might make such guarantees is by taking control of the network and implementing some path through which packets can get from the client to the server within a bounded time, and scaling the service horizontally so that more server processes are online if the queue of requests starts to get large.

The following two processes access the shared variables x, y, z. Each process accesses a different replica of the store used to hold these variables. Before any process starts executing, the value of all three variables, x, y, z, is 0 in all the replicas. 2014.3.e (6 marks)

```
Process A            Process B
x=1;                 y=1;
if (y==0) z++;       if (x==0) z++;
```

When both processes have completed executing the statements given, what are the possible values of z, if the replication uses the sequential consistency model? Justify your answer. 2014.3.d.i (3 marks)

If process A executes first, then the value of z will be 1, and if process B executes first then the value of z will still be 1. This is because in both cases, the `if` statement on the second process will fail.

When both processes have completed executing the statements given, what are the possible values of z, if the replication uses the causal consistency model? Justify your answer. 2014.3.d.ii (3 marks)

The possible values are $z = 0, 1$ since if the statements $x = 1; y = 1;$ are executed first, then both if statements will fail and $z = 0$, if one process fully executes before the other starts, then it will be the same case as in `d.i`, and $z = 1$, but z can never be 2 because either $x = 1$ or $y = 1$ before two of the if statements can be reached, so z can never be incremented twice.