1. Remote Procedure Calls

1. 1 An RPC system operates in an unreliable environment, and each message has to be acknowledged by the receiver. Also, when a client message is too long, it is divided up into smaller blocks. (The size of a block is a system parameter, and we assume it be constant in the system, i.e. it is always the same.) To accelerate the RPC protocol, however, the server does not acknowledge each block of the client message, but sends an acknowledgement (ACK) only after receiving *n* blocks, and after the last block. So a message consisting of *2n+1* blocks will have three ACKs, one after the first *n* blocks, one after the second *n* blocks and one at the end.
The total size of the message is sent in one block before the actual message, and it is acknowledged immediately by the server. Upon the receipt of this initial ACK the client starts to send the message.

The server's reply to an RPC client request is one-block long.

In the system the maximum transmission time of a block is *T1*, the time it takes to send an ACK is *Tack*, the execution time of the remote method on the server side is *Texec*. Each message has to be processed (marshalling, unmarshalling, queuing the message etc.) once at the sender side and once at the receiver side. Processing a message once takes *Tproc* time, and it is the same at client and at server. The whole message is processed in one step, before any transmission starts or after the reception of the whole message. Both client and server are single-processor machines.

Questions

1.1.1 Assuming no failure during transmission, how long will it take for the client to return from two consecutive remote procedure calls? You have a one-threaded client, one-threaded server and synchronous request-reply RPC.

6 marks

1.1.2 Will your answer change if you have a two-threaded client, a two-threaded server, and asynchronous request-reply-acknowledgement RPC? If yes, how, if not, why not? Justify your answer and explain the differences (or the lack of them) clearly.

4 marks

1.2 The above RPC method is used in a distributed database.

Questions

1.2.1 Data is accessed via transactions. There are frequent updates in the database, and very often several clients want to access the same data concurrently. The server can be stateful or stateless. Suggest two possible

solutions in no more than half a page. Which one would be the most efficient in terms of rollbacks?

1+4 marks

2. Distributed Operating Systems

2.1 Process Migration

Questions

2.1.1 What are the reasons for (aims of) migrating a process?

2 marks

2.1.2 Explain the steps of migrating a process from one host to another.

2 marks

2.2 Local RPC

Question

2.2.1 Explain how lightweight RPC optimises local invocations.

3 marks

3. Time in Distributed Systems

3.1 In a distributed system we want to implement causal ordering of events. Explain how a vector clock keeps track of individual events, and how it can ensure that events are processed in causal order.

2 marks

3.2 In a multiuser game each user has the ability to send out a challenge, and the first respondent will have a duel with the issuer of the call. Describe an algorithm that will ensure that the strongest opponent will be picked for the duel when two or more players respond to the challenge at the same time.

2 marks

4. Replication

In a gossip system, a front end has a vector timestamp (3, 5, 7) representing the data it has received from members of a group of three replica managers. The three replica managers, RM1, RM2 and RM3 have value timestamps of (5, 2, 8), (4, 5, 6) and (4, 5, 8), respectively. RM2 has a replica timestamp of (5,5,8).

Questions

4.1 There are two timestamps given for RM2. What do they represent and why do they differ?

3 marks

4.2 Which replica manager(s) could have answered the last query of the front end?

2 marks

4.3 Which replica manager could accept an update from the front end? Which could incorporate an update from the front end immediately?

3 marks

5. Transaction Processing

5.1 Consider the following interleavings of transactions R and S:

	Scenario (1)	
Steps	R	S
1		Open transaction
2	Open transaction	
3	Read(i)	
4		Read(i)
5	Write(i,11)	
6	Read(i)	
7	Close transaction	
8		Write(k,22)
9		Close transaction

Scenario (2)	
R	S
Open transaction	
	Open transaction
	Read(i)
Read(i)	
	Write(i,22)
Write(i,11)	
	Close transaction
Read(k)	
Close transaction	

Questions

Using the two-phase locking protocol, explain for each step

5.1.1 if the operation can be performed at that point

5.1.2 what locks are set/released if the operation can be performed, or what happens if it cannot be performed

5.1.3 the result of the operation, including the value of data items read or written, or the outcome of the transaction.

1.5 + 2 + 1.5 marks

5.2 Distributed deadlocks

Questions

5.2.1 Explain how phantom deadlocks can be mistaken for real deadlocks in a distributed system.

2 marks

5.2.2 How will a distributed deadlock be resolved? Describe the steps of an algorithm.

3 marks

6. Advanced topics

6.1 Explain the role of an object proxy and a skeleton in Java RMI. What is the difference between passing an object by value and passing it by reference in Java?

3 marks