Dept. Comp. Sc.	Distributed Systems
Vrije Universiteit	20.12.2005

Part I

This part covers the same material as the midterm exam.

1a Explain the principal working of a remote procedure call (RPC). *5pt*

Your answer should include the fact that a client is provided with a stub (or proxy) that implements the interface of the procedure as implemented at the server. The stub is repsonsible for transforming the call into a message, specifying the procedure and the parameter values. The stub sends the message to the server, where the server-side stub unmarshalls it and subsequently calls the local implementation. The result is sent back to the client stub as a message. This message is again unmarshalled, after which the result is passed back to the calling client.

1b Give two compelling arguments why an RPC can never provide the same semantics as a local procedure call. *5pt*

Access transparency cannot be achieved by RPCs as, in most cases, it is difficult or impossible to handle pointers to local data structures. Also, RPCs generally operate across a network so that masking failures becomes a major issue. In theory and practice, completely hiding the occurrence and recovery from a failure is impossible.

1c What is the main difference between a remote method invocation (RMI) and an RPC? *5pt*

RMI generally provides a systemwide object reference mechanism. This means that objects can be referenced from any (remote) machine. As a consequence, it becomes possible to pass "pointers" (i.e., references) as parameters in an invocation.

2a Imagine a file server keeping a table of (client,file)-pairs, identifying which clients accessed which files. Is this a stateful server? Morivate your answer. *5pt*

It all really depends whether keeping this information is crucial for the proper working of the file server. If not, that is, the table as such may be lost without affecting the functional behavior of the file server, then clearly the table itself does not indicate a stateful design. If maintaining the table is crucial, then obviously we are dealing with a stateful design.

2b Explain what a message-queuing server is and whether it can be designed to be stateless. *5pt*

A message-queuing server is responsible for storing and forwarding messages in a message-queuing system. For a stateless design, the issue is whether the server can permit to loose all its information regarding clients. In fact, this is possible. Of course, it may be the case that it also looses the messages it had stored, but this has to do with the reliability and fault tolerance of the server, not whether it adheres to a stateless or stateful design.

2c The X Window system refers to the X kernel running on the client machine as the "X server", and the application running on the compute server, as the "X client." Why is this actually a correct usage of client/server terminology? *5pt*

The fact is, that the X kernel simply regards the client machine as the one holding the resources it should control. These resources may be used by applications, which them become clients. To use the resources, they send messages to the X server who will then react and respond. On the other hand, the X server will report on events, such a keystrokes and mouse moevements, which are essentially called back *to the client application.*

3a Consider a client that attempts to synchronize its clock with that of a time server once every minute. It sends a number of requests, with the results shown below. How will the client adjust its clock after receiving a response? Time is given in (hr:min:sec:msec). Processing time at the server is zero. *5pt*

Sent at (local time)	Round-trip delay	Response
10:54:00:00	18 ms	10:54:00:10
10:55:00:00	24 ms	10:55:00:12
10:56:00:00	22 ms	10:55:00:10

The first response is received at 10:54:00:18, and apparently it took 18/2 units to get the answer to the client. The latter is running slow and will adjust its clock to 10:54:00:19. Likewise, the second measurement will show the clock running in sync with the server. The third response is received at 10:56:00:22, which took 11 ms to get to the client. The actual time should thus have been 10:55:00:21. The client will slow down *its clock, e.g., by skipping the next 1 ms.*

3b Explain the principle of the Berkeley algorithm for adjusting the clocks in a distributed system. *5pt*

The time server probes each machine, asking for its local time. After that, it computes an average and tells every member how it should adjust its clock.

3c The communication layer in distributed systems can keep track of causally related messages. What are two major objections against this functionality? *5pt*

(1) The system can keep track only of potential *causaliteit. Actual causality can only be captured at the application level, which is outside the scope of a distributed system. (2) In many systems, there is a lot of out-of-band communication, for example, by means of external databases. Such communication may make messages also causally related, but this cannot be captured by the communication layer.*

Part II

4a Is the following sequence of events allowed with a sequentially-consistent store? What about a causally-consistent data store? Explain your answer. *5pt*

P1:	W(x)a		W(x)c		
P2:		R(x)a	W(x)b		
P3:		R(x)a		R(x)c	R(x)b
P4:		R(x)a		R(x)b	R(x)c

Considering that P3 and P4 in the end read different values for x shows that this is not sequentially consistent. It does adhere to a causally-consistent store, as the causal dependency $a \rightarrow b$ for x is preserved when delivering x to processes.

4b Explain why *writes-follow-reads* consistency guarantees that causal relations are maintained when used for the implementation of a distributed news system. *5pt*

Assume that Alice reads an article A (at location L1) and decides to post a response B (at location L2). At that moment, the original article A will first be moved to location L2 as well, implying that at L2 the postings and possible reactions will always be jointly available.

4c What are the conditions to prevent read-write and write-write conflicts in a quorum-based system. *5pt*

Let N_R denote the number of servers to access for reading; N_W the number for writing, and N the total number of servers. Then (1) $N_W > N/2$ and (2) $N_R + N_W > N$.

5a Show that with 4 processes of which one is faulty, that 3 processes can come to an agreement irrespective of the message communicated by the faulty process. *5pt*

The easiest solution is to reproduce Fig. 7.4 and explain that processes send each other their value, as well as forward what they have been sent.

5b Consider a print server with three possible events: (M) notify the client that printing is done; (P) print; (C) crash. When a client is notified that the print server has just recovered from a crash, it can follow 4 different strategies: (1) Never reissue the print request, (2) Always reissue the print request; (3) Reissue only if the delivery of the print request has not been acknowledged; (4) Reissue only when delivery of the request has been acknowledged. Show that if the server always notifies the client after printing, it is impossible to devise a scheme in which the print job is never lost or never duplicated. *10pt*

The answer to this question lies in Fig. 7.7, following the server strategy $M \rightarrow P$.

6a NFS is arguably not a file system. Explain why. *5pt*

NFS offers facilities to access remote file systems, and relies on the Virtual File System layer as implemented by many operating systems. The VFS provides access to the locally available file system.

6b Coda allows clients to cache files, but sends invalidation messages when a file is modified. What is the main reason for doing these callbacks in parallel? *5pt*

If doen sequentially, a faulty client may contribute considerably to completion of the notification. By doing it in parallel, the Coda server can decide after a single round which clients are not responding withut hindering the ones that are nonfaulty.

6c Coda allows a reading client to continue to operate on its local copy, even if a concurrent write takes place at the server. Why is this behavior considered correct? *5pt*

The whole issue is that as long as timing is not crucial, the reading client is considered to work on a consistent copy. That, after opening the file for reading, another process started to modify the file is less important: this could equally have happened after the client had finished its session.

Final grade: *(1) Add, per part, the total points. (2) Let T denote the total points for the midterm exam ($0 \leq T \leq 45$); $D1$ the total points for part I; $D2$ the total points for part II. The final number of points E is equal to* $\max\{T, D1\} + D2 + 10$.