Department Computer Science	**Distributed Systems**
Vrije Universiteit	**21.12.2011**

BE SURE THAT YOUR HANDWRITING IS READABLE

1a Explain why we cannot claim that remote procedure calls provide complete access transparency. *5pt*

The best compelling argument against complete access transparency, is that RPCs provide no support for global references, meaning that special measures are needed when dealing with pointers.

1b Give an example of the implementation of global references that can be used in RPCs. *5pt*

In the case of Java, a client stub can be completely shipped to another machine, and passed by value as a parameter in an invocation. By coding the actual contact address for an object inside the stub, and realizing that a stub is just a local object, we then have an implementation of a global object reference that is indistinuishable from a local object reference.

1c Explain how request-level interception works by considering the replication of an object across three object servers. *5pt*

It helps if you would provide a diagram. The basic idea is that call at the client stub is intercepted before passing it to the middleware. The interceptor simply calls the middleware three times, once for every server, and preferably in parallel.

1d If an object is replicated three times for fault tolerance, how many responses must an invoking client await before returning a response to its application? Be sure to explain your answer. *5pt*

It depends on the failure semantics. In the case of crash/performance failure semantics, and servers operating otherwise independently, a single response is enough. In the case of arbitrary failures, you need at least two responses (voting). It is impossible to reach agreement in the case of a single failing server when the servers are also allowed to communicate to each other.

2a DNS supports iterative as well as recursive name resolution. Explain the difference, and make clear why iterative name resolution is generally preferred. *6pt*

See figures 5-15 and 5-16. Iterative name resolution is generally preferred as it will provide the IP address of the DNS client resolver, which can be used for specific redirection. More important, is that iterative resolution offloads the higher-level name servers, which is good as they generally get many more requests than lower-level servers.

2b Content Delivery Networks can exploit DNS-based redirection to replica servers. How does this work? *6pt*

Consider the URL `cdn.company.com/image` *and assume that the name server for the* `cdn` *domain is run by that specific CDN. In that case, if the name server has a list of where replica servers are placed, it can redirect a request to resolve the URL to the server nearest to the requesting client. There, the client will ask the server to provide the file* `image`*.*

2c Outline how one could implement DNS using a DHT-based peer-to-peer system such as Chord. Consider resolving URLs. *5pt*

Simply take a URL (or specifically, the part containing the DNS domain name), hash it to a key, and look up the corresponding server by asking the peer responsible for that key. Note that this peer would get the address from the server when the URL comes to life. The remainder of the URL containing the path/filename is passed to the identified Web server.

2d Assuming a DHT-based implementation of DNS, how does iterative and recursive name resolution work? Is there any obvious benefit of one over the other? Explain your answer. *5pt*

In the case of iterative name resolution, when looking up a peer through the finger table, that peer will return the next peer to contact, if needed. In the recursive case, the contacted peer will forward the request to the next peer. There is no obvious advantage, except perhaps when counting the total number of messages sent. In any case, because there is no notion of network proximity, requests and responses will continue to show erratic behavior from the network's perspective.

3a Explain how the primary-backup protocol works. *5pt*

See book, notably Fig. 7-20.

3b Explain how the variant of the primary-backup protocol with local writes works. How could this protocol be made more fault tolerant? *5pt*

See book, notably Fig. 7-21. The trick here is to replicate each backup server with a few extra backup servers, and perform a PBP when the local writes take place. Note that it is not necessary to do this for all backups. It is only needed for the backup that is currently being written to.

3c To what extent does the primary-backup protocol provide sequential consistency? What about the local-write variant provided as an answer to (b)? *6pt*

Obviously, PBP provides sequential consistency as there can be at most one writer at a time, and all those writes are propagated to the other servers. This is also the case for the local-writes variant, provided that a client is always directed to the same replica server, and *subsequent writes by different clients are propagated in the same order to all replicas.*

3d Explain how the CODA file system handles client-side caching of files when dealing with a reading and writing client. Does CODA provide sequential consistency? *5pt*

You need to provide and explain Fig. 11-20, or 11-23. CODA provides sequential consistency as all writes are essentially totally ordered through the server.

4a Sketch a middleware organization in which communicating processes are referentially decoupled. *5pt*

Any description of a publish-subscribe architecture will suffice.

4b In a shared data space, processes are also decoupled in time. What does this mean and why is it so difficult to find scalable solutions for communication? *5pt*

Decoupled in time means that processes need not be up and running at the same time in order for communication to succeed. It is difficult to scale, as we need to have a matching system that takes a look at the tags or content of messages stored in the data space. This requires searching, which is difficult to scale when needed to be done in real time and when the data space can change a lot.

5a Explain the two-phase commit (2PC) protocol. *5pt*

See book, pages 355-356.

5b What happens in 2PC when the coordinator fails and all participants are ready to commit? *5pt*

The protocol essentially blocks. Even if the participants reach agreement that they can commit, 2PC prescribes that we need the coordinator's decision to make the next move.

5c If you were to implement 2PC, what would you do in the case that the coordinator does not receive an answer from a participant? *7pt*

We need to distinguish four cases. (1) When the vote-commit message is not received, the coordinator can simply decide to abort. (2) When the vote-abort message is not received, the coordinator can also safely decide to abort. (3) When an ACK message is not received after a global-commit, the coordinator is already in the commit state, but it is unclear if the participant (who was ready) did not receive the global-abort, or that its ACK was lost. The coordinator can simply resend the global-abort, and in any case log the decision. (4) The ACK message was not received after a global-abort. This is analogous to (3): simply try again, perhaps several times, but in any case log the decision.

Grading: *The final grade is calculated by accumulating the scores per question (maximum: 90 points), and adding 10 bonus points. The maximum total is therefore 100 points.*