Department Computer Science	**Distributed Systems**
Vrije Universiteit	**24.10.2006**

1a Explain the difference between request-level and message-level interceptors. *5pt*

1b Sketch the general organization of a self-managing distributed system as a feedback-control loop. Explain the various aspects of this organization. *5pt*

2a Explain the difference between a process virtual machine and a virtual machine monitor. *5pt*

2b Give an example illustrating that a multithreaded client can improve distribution transparency in comparison to a single-threaded client. *5pt*

Just think of a client communicating with an actively replicated server. Obviously, by executing the (assumed synchronous) calls simultaneously we can improve the response time in comparison to calling each replica one at a time. The parallel call mimics the behavior of synchronously calling a single server.

2c Active replication generally requires totally ordered message delivery. Is this condition sufficient in the case of actively replicated multithreaded servers? Explain your answer. *5pt*

No. The problem is that although messages are delivered in the correct order to the server, we also need to guarantee that they are processed in the same order. In a multithreaded server, this means that thread scheduling needs to be deterministic as well: all threads are scheduled in the same order at every server.

3a Explain the relationship between a node identifier and a key in Chord. *5pt*

Node identifiers and keys are assumed to be drawn from the same m-bit identifier space. For a given key k, the node with the smallest identifier id $\geq k$ is responsible for handling whatever is associated with key k.

3b In Chord, the finger table for node p is defined by $FT_p[i] = succ(p+2^{i-1})$. Assume a 32-bit identifier space and consider the following finger table for node 18. Explain to where node 18 forwards a lookup request for the following keys: $k = 26, 20, 18, 17, 29$. *5pt*

$$FT_{18} = [20, 20, 28, 28, 4]$$

The requests are forwarded, respectively to: 20, 20, 18, 4, and 28.

3c Explain how a node (with unique identifier p) can easily join a Chord ring. *5pt*

It is important that you note that we assume that p knows at least one node of the Chord system, say, q. It then simply sends a lookup request for $succ(p+1)$ to q, that is, it requests lookup(p+1). This will return the node with smallest id $\geq p+1$, which will be the successor of p in the ring. If that node keeps track of its predecessor, insertion is then straightforward.

4a What is meant by sequential consistency? *5pt*

This type of consistency prescribes that if we consider a collection of concurrently executing processes that share a data item x, that the order of operations on x as specified by each program that is executed by the processes is respected, and that the effect of all operations as seen by every process are consistent with a serial execution of one program after the other.

4b What is the most important effect in terms of performance when using synchronization variables? Explain your answer. *5pt*

The most important effect is that multiple read and write operations can be performed by a single process without the need for synchronization with other processes. The result is a performance improvement as there is no need for communication as long as the rules for accessing and modifying synchronization variables are respected.

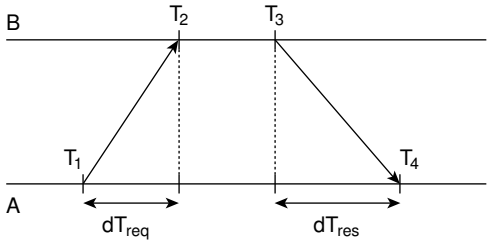
4c Explain why entry consistency and distributed objects form a natural match. *5pt*

Entry consistency is all about associating synchronization variables with a group of data. This is exactly what objects actually do. By guaranteeing that object invocations are atomic, we actually provide entry consistency in a way that naturally matches the logical organization of data by objects.

4d Consider a system in which remote objects are replicated and which guarantees entry consistency. To that end, *all* method invocations are multicast systemwide in a totally ordered fashion. Does this implementation provide stronger consistency guarantees? Explain your answer. *5pt*

In fact, it does. Where entry consistency requires total ordering of operations per (replicated) object*, totally ordered multicast of all invocations effectively guarantees sequential consistency at the granularity of method invocations.*

5a Explain by means of the following diagram, how *A* can estimate the offset of its clock relative to that of *B*. *5pt*



A will have to send a message timestamped with T_1 to B. B will record T_2 as well as its transmission time T_3, which are both put into the response. At T_4 A can compute its offset as

$$\theta = \frac{(T_2 - T_1) + (T_4 - T_3)}{2}$$

5b Suppose that in the previous figure, we cannot assume that $dT_{req} \approx dT_{res}$, as is often the case in wide-area networks. How does this affect the accuracy of *A*'s estimation? *5pt*

Go for extremes, and you will see what happens. With dT_{req} being very small and dT_{res} very large, then the long time it takes for the response to make it back to A, will make A believe its clock offset is very large. In fact, this is not the case. If the request took very long to reach B, while the reponse was returned in a jiffy, A will come to the conclusion that its offset is relatively low.

6a Akamai's CDN deploys replication of documents through standard Web proxy caching techniques. Explain how Akamai does this replication. *10pt*

It boils down to explaining Fig. 12-20.

6b What is strongest kind of consistency would you argue that Akamai provides. Explain you answer! *5pt*

First, it is important to note that updates are always carried out through an origin server. This means that there are no write-write conflicts. Second, because updates lead to modifying the name of an embedded document, and *because each initial request is always forwarded to the origin server, clients will, in principle, never get to see a stale document. Therefore, it can be argued that Akamai provides data-centric consistency, and notably sequential consistency at the granularity of whole-document updates. Anther acceptable answers are continuous consistency with no deviations in value, time, and ordering. However, mentioning client-centric consistency is just too weak (although strictly correct): it provides systemwide consistency. Note that these consistencies become weaker if the main page is allowed to be cached.*

6c To support replication of Web applications, a CDN can deploy *content-aware caching*. Explain how. *5pt*

When deploying a content-aware cache, an edge server or proxy cache server assumes that queries sent to the origin server adhere to only a limited number of templates. This allows the server to store the results of queries in a local cache such that it can easily see whether a next query can be answered from local data only. For example, a query for selecting all elements in a range [0,100] returns an answer that can also be use to answer a query for all elements in a range [10,90].

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.*