| Dept. Math. & Comp. Sc. | Distributed Systems |
|---|---|
| Vrije Universiteit | 08.01.2002 |

## Part I

***This part covers the same material as the midterm exam.***

*1a* Explain the principle of a system that supports message-oriented persistent communication. *5pt*

*Messages are sent by passing them to a queue manager that provides storage guarantees as long as the destination is not capable of receiving the message. Queue managers may be connected to each other in the form of an overlay network, having routing capabilities to allow forwarding of messages toward their destination.*

*1b* What would be an important drawback of using an e-mail system as the technology for implementing a message-queuing system? *5pt*

*An important drawback is that e-mail systems are tailored to end users instead of applications in general. Therefore, there is often less need for issues such as fault tolerance and guaranteed delivery of messages. As a consequence, these extras need to be added to the e-mail system in order to provide the same functionality as general-purpose message-queuing systems.*

*1c* Sketch a design of an RPC-based system that supports data streams for isochronous transmission mode. *5pt*

*Essentially, use the RPC call to set up an additional channel (e.g., a TCP pipe) by passing a pointer to a buffer from which data should be read. As a side effect, the RPC call sets up the pipe to the receiver, starts a thread for reading data from the buffer and passing it to the receiver. If the RPC call is synchronous, a return indicates the connection has been torn down, for whatever reason. With asynchronous RPC, use separate calls to setup and tear down the connection.*

*2a* Why is using DNS for locating objects that change location regularly not such a good idea? *5pt*

*DNS works efficiently only if we can assume that a name-to-address binding does not change regularly. This allows for effective caching of such bindings so that lookups are fast. However, this assumption is not valid for mobile objects.*

*2b* When using forwarding pointers to locate mobile objects, we need to keep chains short. How can this be accomplished? *5pt*

*The first (incomplete) approach is to adjust a client's reference to the object's new address as soon as the object has been located. Alternatively, the new address can be sent along the lookup path allowing each intermediate node/pointer to be adjusted. This approach can be combined with leases by which clients are forced to dereference their current reference so that long paths can be shortened, but also pointers that have no source can be removed.*

*2c* Explain how hierarchical location services exploit locality by considering the general implementation of update and lookup operations. *10pt*

*A hierarchical location service is constructed as a tree. An address of an entity is stored in the leaf node; an intermediate node stores a pointer for an entity $E$ to a child node, if and only if that child node stores an address or a forwarding pointer for $E$. A lookup is initiated at a leaf node. If the node does not store an address for the requested entity, the operation is forwarded upwards until a node is reached that stores a forwarding pointer, after which the lookup is forwarded "downwards." An update operation is initiated at a leaf node and forwarded upwards to the first node that stores a pointer (or the root), unless the leaf already stored an address for the entity. From there on, a path of forwarding pointers to the leaf node is created; the address is stored in the leaf node.*

*3a* Consider a single-threaded object server. How would you implement a remote object that is to be hosted by this server such that remote invocations are atomic? *5pt*

*A simple solution is to make a (partial) copy of the object each time it is invoked. Only if the invocation succeeds to completion will the results be made permanent. This is the same as using a private workspace in the case of transactions.*

*3b* Answer the same question as in (a), but now for a multi-threaded object server. *5pt*

*The only difference is that we are now dealing with concurrency. A simple solution is to make use of a lock by which we effectively allow only a single thread at a time to execute a method. A more advanced solution is to consider a method implementation as a series of read and write operations, and turning each invocation into a local, nondistributed transaction.*

---

## Part II

*4a* Explain what is meant by active replication. *5pt*

*In active replication, each replica is represented by a process that carries out the operation causing the update. In contrast, in passive replication the updated state is shipped to each replica where it replaces the replica's current state.*

*4b* Active replication generally requires a totally-ordered execution of update operations. Explain when such an ordering requirement can be relaxed. *5pt*

*There is essentially only one situation in which total ordering is not necessary: when operations are commutative (i.e., $f \odot g = g \odot f$), or when two calls $f(x)$ and $f(y)$ are commutative because they operate on different parts of the state.*

*4c* Explain how replicated invocations are caused and how they can be avoided. *10pt*

*Replicated invocations are caused by a replicated object invoking another (possibly replicated) object. Each replica of the caller object will invoke each replica of the called object. Unless execution of these operations is idempotent, we have a problem. A solution is assign a coordinating replicas at the caller object that invokes each replica of the called object. Likewise, the called object will need a coordinator to return a response to each replica of the calling object. Alternatively, the coordinator of the calling object can distribute the received results.*

*5a* Explain the principle of subject-based addressing, and sketch its implementation in TIB/Rendezevous. *5pt*

*In subject-based addressing, messages are tagged with a subject instead of an address. Messages are delivered to only those processes that have subscribed to the subject as named in the message. In TIB, this scheme is implemented by broadcasting a message to all receivers where it is subsequently filtered by a daemon process. Only those messages for which a subscriber exists on the receiving machine will be stored locally. The subscribers are notified when a message arrives.*

*5b* Sketch two alternatives for implementing a distributed version of JavaSpaces, and explain how read and write operations would work for each. *10pt*

*See figures 12-17 and 12-18 in the book.*

*6a* Alice can delegate access rights to Bob by means of a certificate. How can a server verify whether Bob received the certificate in a legitimate way without the server having to contact Alice? *5pt*

*Bob needs to prove that he got the certificate from Alice. This can be done by having Alive pass a secret key $K^-$ to Bob through a secure channel that can be used in conjunction with a public key $K+$ stored in the certificate. The server will ask Bob to decrypt the challenge $K^+(N)$ and return $N$.*

*6b* To protect a mobile agent against a malicious host, Ajanta uses a log to which servers can append information allowing an agent's owner to detect whether the log has been tampered with. Explain how this works. *5pt*

*Initially, the log has an associated checksum $C_{init} = K^+_{owner}(N)$. Each time data $X$ is added by server $S$, a new checksum is computed $C_{new} = K^+_{owner}(C_{old}, S, [X]_S)$. Tampering the log can be detected by checking whether the checksum matches the data. What the owner does is compute $K^-_{owner}(C)$, which reveals $C_{next}$, along with $S$ and $[X]_S$. This data can be matched against the then-last appended element to the log.*

***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* $\max\{T, D1\} + D2 + 10$.