Department Computer Science	**Distributed Systems**
VU University	**11.02.2010**

MAKE SURE THAT YOUR HANDWRITING IS READABLE

1a Explain what access transparency is and give an example of a middleware mechanism that achieves it. *5pt*

Access transparancy is the ability to hide the differences in invoking and accessing objects. Although RPC is the canonical example, access transparency is better achieved through Java's RMI.

1b Give an example where the extent to which distribution transparency is achieved needs to be traded off against another desirable feature of distributed systems. *5pt*

In general, aiming at a high degree of failure transparency is going to have an adverse effect on performance: as long as the system is busy masking (the recovery from) failures, no progress can be made.

1c Explain how code migration can help in achieving scalability. *5pt*

Typically, by moving code from the server to the client, we can reduce communication, which not only offloads the server, but also increases client-perceived performance. This is where Java applets play an important role.

2a What is the essential difference between an event-based and a shared data-space architectural style? *5pt*

In the event-based style, processes are decoupled only in space, not in time. In a shared data-space style, they are also decoupled in time.

2b Shared data spaces are difficult to scale across dispersed networks. Why? *5pt*

The problem is that shared data spaces should allow for efficient matching of subscriptions against publications. In practice, this boils down to efficient searching in real time, which is extremely difficult, if not impossible to realize in wide-area settings.

2c Sketch an implementation of a replicated JavaSpace server. *5pt*

Your answer should explain that all tuples are replicated and that by means of an atomic broadcast are removed again when a local match is found. See also Figure 13-12.

3a Give a general framework for a push-pull anti-entropy information dissemination network. *5pt*

You need to describe the normal epidemic cycle: (1) select a random peer, and (2) select the data to exchange, (3) exchange data, (4) update local data, (5) wait ΔT time units.

3b Consider an epidemic-based application with N peers in which peer i has a local variable $x_i \geq 0$. When peer i gossips with peer j, $x_i, x_j \leftarrow (x_i + x_j)/2$. Show that, eventually, x_i converges to $\frac{1}{N}\sum_{k=1}^{N} x_k$. *5pt*

Crucial for the correctness of your answer is that you mention that after an exchange, $\sum x_k$ is not affected, yet that both i and j have the same value. This can only mean that all peers will eventually have the same value, while $\sum x_k$ has not changed. Ergo, they can each have only the average $\bar{x}_k$.

3c There are two important assumptions we need to make for convergence in (b) to take place. What are these? *5pt*

(1) The exchange is atomic and (2) the set of peers remains the same.

4a Explain how iterative name resolution in DNS works and why it may incur high latency costs. *5pt*

See Figure 5-15 and Figure 5-18.

4b When resolving name www.distributed-systems.net, DNS will return an address, but not when trying to resolve distributed-systems.net. How can this be? *5pt*

DNS maintains domain names; not every domain name is necessarily associated with an IP address. In fact, many names are really just like directories. Obviously, each domain name does have an associated name server, but that's something else.

4c Sketch a Chord-based implementation of DNS. What would you see as the main drawback of such an implementation? *5pt*

Each domain name should be uniquely hashed, so that a lookup can be directed to the responsible peer. That peer would simply maintain the DNS records associated with the DNS name. The main drawback is that the hierarchical structure of DNS is completely flattened. For example, it is virtually impossible to list all names that fall under distributed-systems.net *without taking special measures.*

5a When applying a primary-backup protocol for replicating a service, we may actually experience a *scale down* in performance. Why is this so? *5pt*

In the PB protocol, all backups need to have acknowledged an update before the updating client can proceed. As a consequence, an update may take much longer in comparison to a nonreplicated service.

5b Consider a quorum-based replication scheme with N servers. Show how inconsistency may be introduced if the write quorum $N_{write} \leq N/2$. *5pt*

Let N be even and name servers $S_1, S_2, \ldots, S_N$. Perform a write on servers S_{2i} and another concurrent one (i.e., with the same timestamp, perhaps by another client) on servers S_{2i+1}. Obviously, it is now unclear which update is the most recent. This could have never happened if $N_{write} > N/2$ as in that case at least one server would have received the two updates in a specifc order.

5c If the read-write ratio in a quorum-based replication scheme is high, what should the write-quorum N_{write} be? Explain your answer. *5pt*

Obviously, if we have relatively many reads, it makes sense to go for a high write-quorum so that as few as possible servers need to be contacted for a read operation.

6a Web applications can be replicated following an edge-server architecture. What does this architecture look like? *5pt*

There will be a single origin server and numerous replica servers, known as edge servers. Clients always connect the same edge server, which is responsible for caching and maintaining replicated data.

6b In an edge-server architecture, it is important that a client is always redirected to the same edge server. Why? *5pt*

This is primarily done to ease consistency: that edge server will always appear to offer responses that are consistent with previous requests. If a client can be redirected to different edge servers, those servers will need to be kept (strongly) consistent.

6c How can a client transparently "discover" its most appropriate edge server? *5pt*

There are different alternatives, but a straightforward one is to deploy DNS-based redirection, by which the DNS server associated with a site always returns the address of the best edge server for that client.

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.*