Part I

This part covers the same material as the midterm exam.

1a Explain what is meant by administrative scalability and why it is often such a difficult problem to solve. *5pt*

Administrative scalability refers to the problem of expanding a distributed system across multiple administrative organizations. Difficulties arise from the fact that different organizations each have their own policies regarding security, replication, admission, and so on. Policies between different organizations may easily conflict.

1b What are the main differences between a network operating system and a (middleware based) distributed system? *5pt*

A NOS provides a low degree of distribution transparency; its communication is often based only on files. A DS offers more transparency and a specific model of communication. The scalability of a DS varies and is often dependent on the model that is offered.

2a Give at least two examples of a middleware protocol. *5pt*

(1) Distributed commit. (2) Distributed mutual exclusion. (3) Virtual synchronous multicast.

2b What is the difference between passing an object by reference or an passing object by value in the case of a remote object invocation? *5pt*

Passing an object by reference means that only the object reference is used in the parameter. As a consequence, the referred object is left untouched. Passing an object by value can be done only with references to a locally available (possibly remote) object, which is subsequently copied. The result is that a second object is created. Due to transparency, the difference between the two is not always clear, often leading to serious programming mistakes.

3a Motivate the extensive use of threads in distributed systems. *5pt*

The main reason is that it allows for building well-structured efficient servers, by having a simple mechanism handle concurrent requests that each may lead to blocking I/O operations (or calls to other servers).

3b Consider a mobile agent having its own thread of control. Are we dealing with weak or strong mobility when the agent moves to another machine? Motivate your answer. *5pt*

Whether the agent has its own thread of control is really not that interesting when it comes to weak or strong mobility. What counts is wether the execution segment of the thread is transferred or not. If the thread can be migrated along with the rest of agent, we will likely be dealing with strong mobility. However, it may also be the case that only the data and code segment are transferred, and that a new thread of control is started at the destination machine. In that case, we are dealing with weak mobility.

4a Explain what a closure mechanism is in naming systems, and give at least one example. *5pt*

A closure mechanism is the (implicit) mechanism by which name resolution starts. A typical example is sending a DNS query to your (well known) local DNS server.

5 Explain what is meant by a consistent global state when dealing with distributed snapshots. *5pt*

It means that the receipt of only those messages is recorded for which the sending has been recorded as well.

Part II

6a A file is replicated on 10 servers. List all combinations of read quorum and write quorum that are permitted by the quorum-based voting protocol when dealing with replicated writes. *5pt*

The following possibilities of (read quorum, write quorum) are legal: (1, 10), (2, 9), (3, 8), (4, 7), (5, 6). Note that we assume that write-write conflicts should be avoided.

6b Explain how an epidemic-based distribution protocol works for propagating updates. *5pt*

See book. You should at least mention random selection of peer to communicate with, and preferably also the difference between anti-entropy and gossiping.

6c Give an example of a causal-consistent execution of read and write operations in a distributed system. *5pt*

See Figure 6-9.

6d It has been argued that middleware protocols for achieving causal consistency should be replaced by application-level protocols. What is a strong argument in favor of such a replacement? *5pt*

The problem is that middleware protocols capture only potential *causality by looking at when a message is sent, and taking into account the messages that have been received so far. Applications can tell precisely (at the semantic level) whether sending a message depends on a previously received one.*

7a Explain what a virtually synchronous reliable multicast is and what the advantage of this scheme is. *5pt*

This type of multicast guarantees that a message that is multicast to a group G of processes, is either delivered to every non-faulty member of G, or to none at all. The advantage is that when a message is delivered to a process P, that process knows exactly to which other processes the message has been delivered. It makes programming fault-tolerant applications much easier.

7b What is a pessimistic message logging scheme and why is it preferred over optimistic approaches? Be precise. *5pt*

In a pessimistic message logging scheme, each message m sent to a process P is logged to stable storage before P is allowed to send another message. In this way, it is ensured that only P depends on m until it sends another message. Logging can be done by either the sender of m or P. The method is preferred over optimistic approaches due to its simplicity.

8a Sketch two different approaches for implementing a subject-based publish/subscribe system. *5pt*

Use broadcasting as in TIB/Rendezvous, or a message-brokerage approach as in IBM MQSeries.

8b How does Jini realize referential decoupling of communicating processes? Can this method scale? *5pt*

By means of JavaSpaces. This approach allows processes to write tuple instances to a storage, but more importantly, allows other processes to read those tuples using a simple associative search strategy. However, scalability problems arise, because, in principle, a reading process needs to continue to inspect every tuple until a match is found. If tuple instances are distributed across multiple servers, searching may become prohibitively expensive.

8c What is the (semantic) difference between writing a tuple instance to a transparently N times replicated JavaSpace, and writing that tuple instance N times to a nonreplicated JavaSpace? *5pt*

There is hardly any difference except when removing a tuple. In the first case, you need to remove all replicas; in the latter, removing one instance is enough.

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