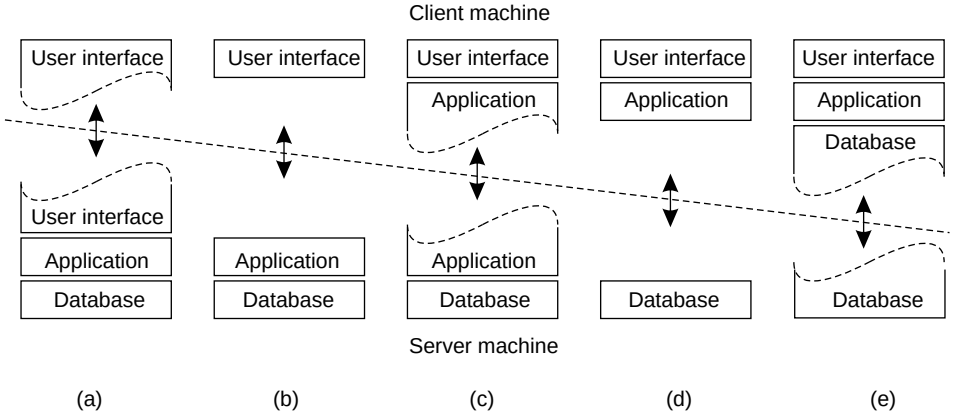
Part I

This part covers the same material as the midterm exam.

1a What is meant by the statement that distributed systems are characterized by the occurrence of *partial failures*? *5pt*

Unlike most nondistributed systems, a failure in a distributed system does not bring the entire system to a complete halt. In contrast, most of the components continue to operate and it may take considerable effort to mask a failure and the recovery from that failure. In many cases, this masking only partly succeeds, so that the behavior of non-failing components can be affected in unpredictable ways.

1b Consider the following client-server architecture. Give a representative example for each of the five organizations. *5pt*



(a) Dumb-terminal connected to a mainframe. (b) X-terminal connected to a regular server. (c) Web browser in which an applet is executing as part of a Web interface to an e-commerce site. (d) Traditional organization such as followed in NFSv3: all the files are stored at the server, but applications and such run on the client machine. (e) Arguably what happens in modern distributed file systems such as NFSv4. Another example is where a Web browser makes use of a local proxy cache.

1c What is meant by a multi-tiered client-server architecture? *5pt*

In such an architecture, different functions are placed at servers on different machines, where each server can act as a client for another server. In a traditional two-tiered architecture, there is only a client machine and a server machine. In three-tiered architectures, the functions belonging to the user-interface level, the processing level, and the data level are separated and carried out by processes at three different machines.

2a Communication between processes can be characterized along different dimensions. Give an example for each of the following combinations. *5pt*

	transient	persistent
synchronous	...	...
asynchronous	...	...

	transient	*persistent*
synchronous	*RPC/RMI*	*message queuing with acknowledgment*
asynchronous	*socket-based messaging*	*message queuing without acknowledgment*

2b Explain what is meant by causally-ordered multicasting. *5pt*

In a communication system that provides causally-ordered multicasting, if a process P multicasts a message m_2 after the receipt of a previously multicasted message m_1, then m_2 will always be delivered to a receipient only after *the delivery of m_1 to that recipient.*

2c Many communication subsystems provide at best only FIFO-ordered multicasting, meaning that only the messages from the same sender are delivered in the order they were sent. What is the argument against provide stronger delivery semantics, such as causally-ordered or totally-ordered multicasting? *5pt*

The problem with stronger semantics is caused by the fact that it requires the subsystem to consider all *communication between different processes. In practice. lots of communication may happen through so-called external channels, also referred to as out-of-band communication. Such communication can establish causal relationships that can never be perceived by the communication susbsystem, for which reason it might as well be handled at the application level.*

3a Explain what a true identifier is, and give an example. *5pt*

A true identifier is a name such that (1) it refers to at most one entity, (2) each entity has at most one identifier, and (3) it is never reused. Social IDs are a typical example of a true identifier.

3b Is an object's name containing the MAC address of the machine where that object was created location independent? It is essential that you motivate your answer. *5pt*

It all depends whether the naming system uses that address to actually locate the object. If it does, the name is definetely not location independent. However, if the MAC address is otherwise not used, the name may be considered to be location independent.

3c What is the main advantage of using true identifiers as intermediates in the mapping of names to addresses? *5pt*

If an object can have N different names, and be replicated at P places, we need to maintain $N \times P$ name-to-address bindings. Every a name changes, we need to change P bindings, and likewise, a change of address requires N updates. Using a true identifier as intermediate solves this problem: every change requires only 1 update in the mapping. Moreover, using true identifiers allows for extensive replication in the naming service if we can assume that name-to-identifier mappings do not change often (as is assumed in DNS).

Part II

4a What is sequential consistency? *5pt*

Sequential consistency defines the effect of operations in the presence of concurrent writes on a shared (possibly distributed) data store. Such a data store is said to be sequentially consistent if any execution is the same as if operations by all processes were executed in some sequential order, and the operations of each indvidual process are carried out in the order as specified by that process's program.

4b What is the difference between a client-centric consistency model, and a data-centric consistency model? *5pt*

In a client-centric model, you are interested only in the consistency as observed by an individual process. In contrast, a data-centric consistency model prescribes what all *processes will see, and thus provides systemwide consistency.*

4c In a system that guarantees eventual consistency, how are write-write conflicts handled? *5pt*

This is a tricky question. The whole issue is that eventual consistency says in the absence of updates, all processes will eventually see the same value. As a consequence, when conflicts occur, they need to be resolved in such a way that only one update wins. In other words, although there is no general rule that says how these conflicts are to be handled, eventual consistency does mean that of all conflicting updates, one update will win.

5a Consider two machines each running NFSv3 servers. Each machine executes the following instructions: (1) create a file, (2) mount the root of the other machine's file system in `/other`, and (3) list the directory contents. What is the result of executing (3)? Assume both machines have executed (2) before any of them executes (3). Explain your answer! *5pt*

machine1	*machine2*
`touch /other/a.txt`	`touch /other/b.txt`
`mount machine2:/ /other`	`mount machine1:/ /other`
`ls /other/other`	`ls /other/other`

Consider machine1. Because NFSv3 does not allow imported file systems to be exported, `ls /other/other` *will refer to the original content of* `/other` *on machine2, which contains only the file* `b.txt`.

5b Consider a simple Web site with only static pages. The site is hosted on a machine running NFSv3, exporting the file system containing the pages. If another machine mounts this file system, and subsequently hosts a Web proxy server, what kind of consistency can clients of that proxy server expect (assuming it makes use of the mounted file system)? *5pt*

The Web proxy need no longer fetch pages through HTTP, but can immediately fetch them locally through NFS. Because NFS provides strong consistency, clients will always see up-to-date pages. BTW, this system has actually been implemented by AFS, a predecessor to Coda.

5c When a client opens a file for reading only, the server can pass a copy to the client. Coda allows the client to continue reading that file, even if it is updated in the meantime. Why can this behavior be considered correct? *5pt*

It has everything to do with timing. As long as timing issues do not play a role, the client can logically be considered to be reading a file that will only be updated after *that client is done. In other words: the data that client is accessing is consistent during that client's session, independent of any concurrent updates. Of course, when time starts playing a role, this argument will no longer hold.*

6a What is meant by subject-based addressing, and how can it be implemented using message brokers? *5pt*

In subject-based addressing, senders pass messages to the communication subsystem that are identified by a subject only. Receivers can only get the messages with the subjects for which they have subscribed. In a simple message-broker implementation, all messages are forwarded to the broker. It then matches message subjects against the subscribed receivers, and subsequently forwards messages to subscribers.

6b Explain the semantics of Jini/JavaSpace's `read`, `write`, and `take` operations. *5pt*

In Jini/JavaSpace, process operate on a shared data space containing tuples. A tuple is a record containing object references of which an instance can written to the shared space. To read a tuple, a process provides a template, which corresponds to a tuple's record, but possibly specifying some fields to be NULL. A read succeeds if a match is found: a template's filled-in fields are identical to the corresponding fields in a tuple instance. In that case, a copy of the matched instance is returned. A take removes the instance. If no match is found, the process is blocked.

6c Explain why it is so difficult to implement an efficient distributed JavaSpace. *5pt*

The crucial issue is that for either read, write, or take operations, you will, in principle, need to do at least a partial search that may expand across several machines.

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