Department Computer Science	**Distributed Systems**
Vrije Universiteit	**20.12.2006**

1a Give at least one technique other than replication or caching that can be applied to tackle geographical scalability problems. *5pt*

Geographical scalability comes from the fact that components of a distributed system may lie far apart, and that communication is subject to high latency and unreliable message transfer. Hiding latencies is therefore important, and can be accomplished through caching and replication of processes and data. What may also help is to use asynchronous communication, but this will need to be combined with running multiple threads, which in turn may be highly dependent on the application.

1b Caching and replication are important scaling techniques, but introduce another scalability problem. What is that problem and how can it be tackled? *5pt*

The problem is keeping copies consistent. In principle, this requires global synchronization of the replicas each time an update takes place, which is virtually impossible. The only solution is to weaken the consistency requirements for replicas.

1c Distributed systems are often designed under the false assumption that the topology of the underlying network does not change. Give an example where making this assumption simplifies the design, but adversely affects the system when the topology does change. *5pt*

When designing overlay networks, such as in the case of peer-to-peer systems, optimizing the overlay by taking network proximity into account can be very important. If we can assume that latencies between nodes are stable, then finding the nearest best peer can be relatively simple. However, if the network topology does change regularly, which may happen when nodes are mobile or when churn is relatively high, we will see that performance may rapidly drop.

2a One particular group of two-tiered client-server architectures is moving away from fat clients to thin clients, effectively placing a much higher burden on the server side. What class is this and what is the reason for this shift? *5pt*

We're referring to C/S architectures in which the client software is placed on end-user machines. The problem with this configuration is that end users are expected to be able to manage their machines. In practice, this turns out to be cumbersome and less cost-effective. As an alternative, by moving functionality to the server side, management of the distributed systems becomes easier, leading to thin-client solutions.

2b Give an example of the organization of a Web site into a three-tiered architecture. Be sure to make clear what each tier is expected to do. *5pt*

A typical example consists of a site supporting search facilities. In that case, you may see a standard Web server (such as Apache), an application server that processes queries that have been passed on from the Web server, along with a relational database from which answers are retrieved. The application server may be extended with a process that generates HTML from the answers, which it then passes on to the Web server.

2c Explain the fundamental difference between a Web service and a traditional Web site. *5pt*

A Web service effectively allows clients other than Web browsers to access it through fairly traditional communication facilities (typically SOAP). In this way, it becomes possible to compose services from other Web services, and perhaps offer these composed services through a Web site.

3a Give an epidemic algorithm to estimate the size of a network. *5pt*

The solution is in the book: let node i start with $x_i = 1$, all other nodes $j \neq i$ with $x_j = 0$. Every time node k selects another node j at random, $x_j, x_k \leftarrow (x_j + x_k)/2$. In the end, each x_j will converge to $1/n$, with n the size of the network.

3b Would your algorithm from the previous question still work for a *wireless* network in which each node can communicate only with its neighbors? Explain your answer. *5pt*

Yes, albeit slower. The key observation is that no matter with whom a node i communicates (say j), $x_i + x_j$ will remain constant, while $|x_i - x_j|$ becomes smaller, or stays the same. However, it is not difficult to see that eventually *$|x_i - x_j|$ will become zero, no matter which nodes communicate as long as the network is connected.*

3c Epidemic algorithms generally require that a node can select another random node. How can this randomness be achieved even for very large networks? *5pt*

The crux is that each node maintains a limited-size list of neighbors from which it randomly selects one each time. That selected neighbor is then also used to modify the list of neighbors by exchanging references to neighbors (which effectively means that the topology of the overlay network changes.

4a Assume nodes in the Chord peer-to-peer system do not maintain a separate reference to their predecessor in the ring. How would you look up the predecessor of a node i? *5pt*

Initiate a lookup for key $i-1$. This will lead to a series of nodes $n_1, n_2, \ldots, n_k$, where n_k is responsible for $i-i$ (i.e., we are looking for $succ(i-1)$). If $n_k \not\equiv i$, then n_k is the predecessor. Otherwise, we will need to look up key $i-2$, and so forth, until we find a value p for which $succ(i-p) \not\equiv i$.

4b In Chord, there is a difference between iterative and recursive lookups. Which one performs better? *5pt*

If we assume that Chord does not optimize its finger tables to take network proximity into account, there is no performance difference: all operations take place on a logical overlay anyway. With network proximity, recursive lookup will generally be cheaper as the distance to the target will gradually decrease. Note that such a decrease need not always take place.

4c In Chord, a finger table entry $FT_p[i]$ always points to $succ(p+2^{i-1})$. How can this scheme be extended to take network proximity into account? *5pt*

The crucial observation here is that $FT_p[i]$ points to the first *existing node in the interval $[p+2^{i-1}, p+2^i-1]$. It cannot do any harm to add more references to existing nodes in that interval, and then subsequently choose the one that is nearest, but also satisfies the constraint that its ID is greater or equal to the key that is being looked up.*

5a Explain what is meant by data-centric causal consistency. *5pt*

A data store is said to provide causal consistency if writes that are potentially *causally related are seen by all processes in the same order. Concurrent writes may be seen in a different order by different processes.*

5b Explain why letting the middleware preserve causality when delivering messages is not necessarily a good idea. *5pt*

There are essentially two reasons. The first is that the middleware can preserve only potential *causality. It is only at the application level that we can really decide whether one message depends on another. The second reason is that the middleware may not be able to capture* all *relationships. For example, there may be out-of-band communication between two users such that Bob reacts to a message sent by Alice, but which he sees only because Alice phones him.*

5c Consider a system that maintains vector clocks for enforcing causal communication. Let $ts(m)$ denote the (vector) timestamp sent with message m, and VC_i the vector clock for process i. A message from process i is delivered to process j only if (1) $ts(m)[i] = VC_j[i]+1$ and (2) $ts(m)[k] \leq VC_j[k]$ for all $k \neq i$. Give the correct interpretation for these two conditions. *5pt*

(1) This is the next message that process j expects from process i; (2) process j has seen all messages that process i had seen before sending message m.

```
(01) main(int argc, char* argv[]) {
(02)    Ice::Communicator   ic;
(03)    Ice::ObjectAdapter  adapter;
(04)    Ice::Object         object;
(05)    ...
(06)    ic = Ice::initialize(argc, argv);
(07)    adapter = ic->createObjectAdapterWithEnd Points( "MyAdapter","tcp -p 10000");
(08)    object  = new MyObject;
(09)    adapter->add(object, objectID);
(10)    adapter->activate();
(11)    ic->waitForShutdown(); }
```

6a Explain what is happening in the (incomplete) code fragment given above. *5pt*

In this example, we see that an adapter is created (07), after which we place a newly created object under its regime (09), activate it, meaning that messages can now be sent to that object (10), and then subsequently wait until the server is shut down. Overall, the code fragment shows the minimal code for creating an object server.

6b What information will the objectID parameter contain in line (09)? *5pt*

That ID will most likely contain (1) the IP address of the server, (2) the port to which the adapter containing the object is listening to (10000), as well as a unique index for the object itself.

6c What is meant by an *activation policy* for an object adapter? *5pt*

An activation policy specifies how an object under the regime of a specific adapter is invoked by a remote client. There are various arlternatives: the thread running the adapter invokes the object; a new thread is created, or incoming requests are queued to wait for their turn.

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.*