Department Computer Science	Distributed Systems
Vrije Universiteit	24.10.2006

1a Give an example of how a high degree of distribution transparency can have an adverse effect on performance. *5pt*

One simple example is when the middleware continues to attempt to recover from failures during an RPC while completely hiding these. Obviously, the RPC is going to take very long.

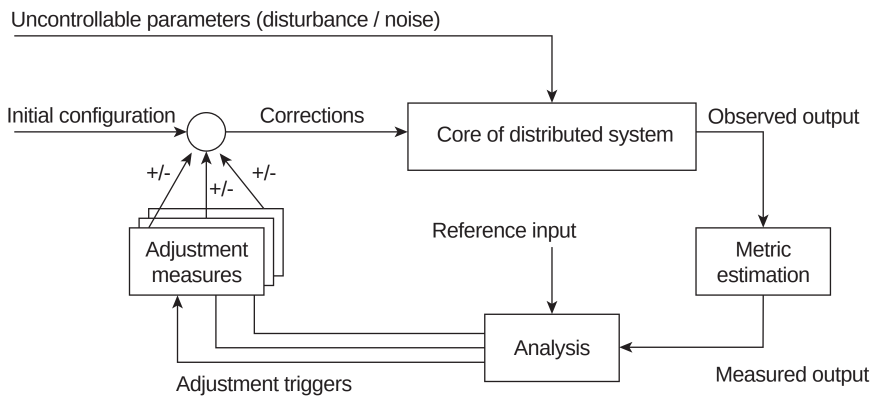
1b What do you see as the main scalability problems with a video conferencing application? *5pt*

It's the synchronization between two communicating parties when the application needs to run across a wide-area network: it is virtually impossible to hide the long latencies.

1c What do you see as the main scalability problem in realizing a system according to a shared data space architecture. *5pt*

The main problem here is that an SDS essentially requires that we match providers and consumers of the shared data based on content instead of identifiers. This matching imposes an inherent search through, in principle, all available data. Such a search will never scale for very large of dispersed systems.

1d Give a concrete, simple example of a feedback control loop in a distributed system. Be explicit about each of the components in the following figure. *5pt*



Consider Globule. It collects traces from various Web replica servers and analyses those to find the best replication for Web pages. The latter is done by comparing various what-if situations through simulation, and choosing the one with the lowest overall cost. Globule supports three measures: change the replication degree, change the location of replicas, and consistency enforcement policies (i.e., choosing the appropriate consistency protocol).

2a Explain the principal operation of a remote procedure call (RPC) *5pt*

You should more or less explain Fig. 4.7.

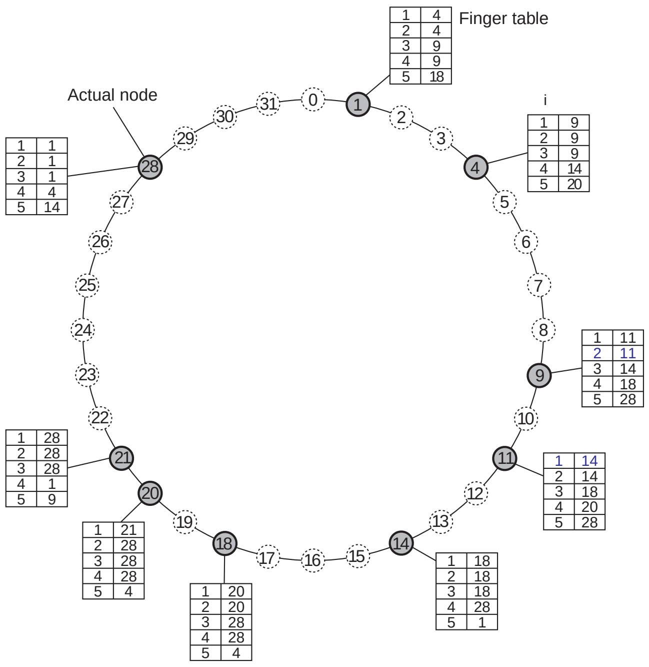
2b An RPC is a transient, synchronous form of communication. What does this mean and what are the main drawbacks? *5pt*

Transient synchronous communication means that sent messages cannot and will not be stored by the communication system if delivery cannot proceed. In addition, the sender and receiver are assumed to be both active at the moment communication takes place, which implies that if something goes wrong, it should be instantly restored. The latter makes this form of communication often more difficult to implement efficiently.

2c An RPC to a replicated server can be made highly transparent to caller and callee with respect to access, replication, and failure transparency. How? Explain your answer! *5pt*

First, we need to place the replicated call inside the client-side stub, from where it can simply be executed in parallel to the servers. Nothing special needs to be done for the server, however, the client should not wait for answers per server, but first issue the call to each server individually, and then wait until answers come in. This is crucial for performance, but also failure transparancy as it can return the first *answer back to the client.*

3a Explain the values for the finger table of node 9 in the following Chord DHT-based system. *5pt*



1	4
2	4
3	9
4	9
5	18

1	9
2	9
3	9
4	14
5	20

1	11
2	11
3	14
4	18
5	28

1	14
2	14
3	18
4	20
5	28

1	18
2	18
3	18
4	28
5	1

1	20
2	20
3	28
4	28
5	4

1	21
2	28
3	28
4	28
5	4

1	28
2	28
3	28
4	1
5	9

1	1
2	1
3	1
4	4
5	14

$FP_p[i] = p + succ(p+2^{i-1}$, with $i \geq 1$. In this example, $FP_9[1] = succ(9+2^0) = succ(10) = 11$. Likewise, $FP_9[2] = succ(9+2) = succ(11) = 11$, and so on.

3b Assume node 4 is requested to look up key 29. How is this key resolved? You **must** explain your answer! *5pt*

Node 4 will first look for the entry that satisfies $FT_4[j] \leq 29 < FT_4[j+1]$, which in this example is entry 5, where we need to apply modulo arithmetic. Therefore, the request is forwarded to node 20, which will then, for similar reasons, forward it to node 28. Node 28 will find that 29 lies between 28 (its own ID) and $FT_{28}[1]$, so that it sends the request to node 1. The latter is responsible for key 29.

4a Show that logical clocks do not necessarily capture potentially causal relationships. *5pt*

Simply take two concurrent events, like the sending of two (first) messages by different processes. The attached timestamp will not indicate that one preceeded the other because of causality issues.

4b When using vector clocks for enforcing causally ordered multicasting, $VC_i[i]$ is incremented only when process P_i sends a message, and sends VC_i as a timestamp $ts(m)$ with message m. How should we interpret the following two conditions for delivering m when received by process P_j: *5pt*

1. $ts(m)[i] = VC_j[i] + 1$.
2. $ts(m)[k] \leq VC_j[k]$ for $k \neq i$.

The first condition states that this is the next *message that P_j is expecting from P_i; the second that it has seen all messages that P_i had seen when sending m.*

4c We always carefully talk about tracking "potentially" causal relationships by middleware. Why "potential?" *5pt*

Because the middleware cannot know for sure why an application sent a message after having received another. It may very well be that the two are completely independent. However, this is a semantic issue that cannot be observed by the middleware.

5a What is the crucial difference between data-centric and client-centric consistency models? *5pt*

The crucial difference is that data-centric models state how concurrent processes acting shared data will see the effects of concurrent read and write operations. With client-centric models, consistency *is formulated only for a single process. In particular, this may imply that concurrent writes in a client-centric model may eventually lead to conflicts because consistency was preserved only on a per-process basis.*

5b Explain why the *writes-follows-reads* client-centric consistency model ensures that causal relations in the case of a network news service are preserved. *5pt*

In this case, the WFR model guarantees that if you decide to post a reaction to a previously read news item, that news item will have to be present at the location where the posting takes place. In other words, all the data is locally available that had been previouslt read when a user decided to react to a news item.

6a The upload/download model for files is known from FTP sessions, but has found its way into distributed file systems. Where and how is it used? *5pt*

It is currently being applied for whole-file caching, notably in systems like NFSv4 and others that are designed to operate in wide-area networks. The basic idea is that a client is given a complete local copy of a file so that it can perform local operations until finished, after which it returns the updated file to the main server.

6b In Coda, a client is allowed to continue reading from a file despite that the server knows an update has taken place. Argue why this does not violate consistency. *5pt*

The principal idea is that when a client has access to a read-only local copy of a file, it is just a coincidence that the update took place while the client was reading. For that matter, the update could also have been considered to take place after the client was finished. So, from a logical point of view, the client is operating on perfectly valid data, and then only after it closes its read session, it should be given access to the updates when it requires to read or write the file again.

7a Explain the difference between content-aware and content-blind caching for Web applications. *5pt*

With content-aware caching, the cache has knowledge on the data model that is used by the Web application, and with that, can conduct query-containment procedures to see whether a query could possibly be addressed by the data that is already cached. With content-blind caching, the cache simply attaches a unique id to an entire, specific query in order to check whether that exact query had been issued before. If so, it can possibly return the previously stored response from its cache.

7b Replicating (the database of) a Web application across edge servers may actually slow down the application. Why is this so? *5pt*

When the read-write ratio is low, we will see that relatively many updates take place. Assuming a pessimistic consistency protocol such as primary-backup, then means that we'll on average be sending more control messages over the network as part of a 2PC protocol, and forcing the origin server to wait until all replica servers have reached the same conclusion. Obviously, this will adversely affect performance.

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.*