Department Computer Science	**Distributed Systems**
VU University	**15.12.2009**

MAKE SURE THAT YOUR HANDWRITING IS READABLE

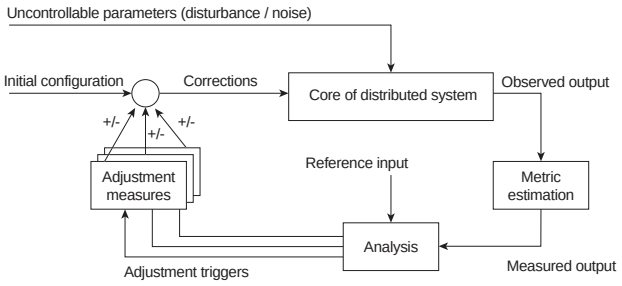
1a Explain what is meant by request-level and message-level interceptors in middleware. *5pt*

Request-level interceptors are special local components to which an invocation request is passed before passing it to the underlying middleware. Such an interceptor is invocation aware in the sense that it knows with which invocation it is dealing, and for which server it is intended. Typically, such interceptors can be used to implement replicated calls. A message-level interceptor is a component that is logically placed between the middleware and the underlying operating system. It can thus handle only basic network messages, for example, by fragmenting them into smaller parts (and assembling these parts at the receiver side).

1b Where does the need for adaptive middleware come from? *5pt*

Middleware is intended to incorporate general-purpose, i.e., application-independent, mechanisms for distributed computing. The problem is that for practical purposes, it is very difficult to separate policy from mechanism, with the effect that many middleware solutions are not right for specific applications. The result is the need to be able to tweak the middleware for the specific needs of an application.

1c In the underlying feedback control loop, give an example of the analysis component in combination with the reference input. *5pt*



An example that is also discussed in the book, is analyzing whether measured performance is as good as it could have been when another replication scenario would have been used. In this case, the reference input is a cost function that needs to be minimized.

2a What is the difference between transport-layer switching and content-aware request distribution? *5pt*

With transport-layer switching, a front end to a server cluster accepts incoming TCP connections and hands these off to one of the back-end servers using only information that is available at the TCP-level: client address and destination port. In the case of content-aware request distribution, the switch can also inspect the content of requests (such as an HTTP URL) and use that information to decide to which back-end server the request should be forwarded.

2b Explain how TCP handoff works and why it is difficult to apply to wide-area networks. *5pt*

With TCP handoff, an incoming connection request is forwarded by a switch to a specific server, which then sends the response back directly to the client, using the network address of the switch. This last issue is problematic is a wide-area system, as it essentially involves spoofing the switch, which is often difficult to do across administrative domains.

2c Explain how the content-aware request distribution can be combined with TCP handoff. *5pt*

Your answer should explain what is happening in Figure 12-9. Essential is that you mention the initial handoff to a distributor or dispatcher to decide what the best server could be based on content, after which the TCP connection is handed off to that server. The switch is subsequently informed.

3a Traditional RPC mechanisms cannot handle pointers. What is the problem and how can it be addressed? *5pt*

The problem is that pointers passed as parameters refer to a memory location that is local to the caller. That location is not only often meaningless to the receipient, but more important is that the recipient will most likely not have the data structure in its memory that the caller has. There are not many things you can do about this, except copying the entire (dynamic data structure) from the caller to the callee when doing the RPC. An alternative is to replace pointers by global systemwide references, as is done with Java object references.

3b Where does the need for at-least-once and at-most-once semantics come from? Why can't we have exactly-once semantics? *5pt*

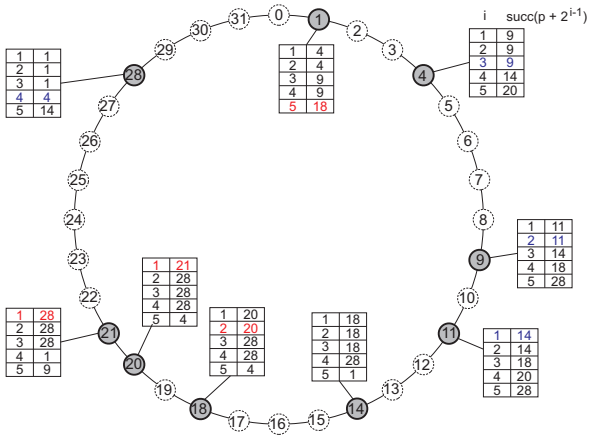
The problem originates from having a (suspected) server crash, detected by the lack of a response in the case of an RPC. What the client-side software can do is either resend the request until it finally gets a response (at-least-once semantics) or immediately reports the failure to the client application, thus providing at-most-once semantics. Guaranteeing exactly-once semantics is, in principle, impossible, because you cannot know in general whether the server crashed before or after executing the requested operation.

3c Consider a client/server system based on RPC, and assume the server is replicated for performance. Sketch an RPC-based solution for hiding replication of the server from the client. *5pt*

Simply take a client-side stub that replicates the call to the respective servers. It is essential that you mention that these calls should be done in parallel and that (for example) the first response is immediately passed to the client. Serializing the RPCs or waiting for all responses is OK for fault tolerance, but certainly not for performance.

4a Resolve the following key lookups for the shown Chord-based P2P system: *5pt*

source	key
4	15
4	22
21	30
21	27
20	18



15@4: 14–18; 22@4: 20–21–28; 30@21: 28–1; 27@21: 28; 18@20: 4–14–18

4b Adjust the finger tables of nodes 18 and 14 when a node with ID 24 enters the ring. Also give the finger table of node 24. *5pt*

Node 18: [20,20,24,28,4]; Node 14: [18,18,18,24,1]; Node 24: [28,28,28,1,9].

4c Chord allows keys to be looked up recursively or iteratively. Explain the differences, as well as the main advantage of iterative over recursive lookup. *5pt*

With recursive lookups, a message is forwarded from peer to peer until it reaches its destination. In contrast, with an iterative lookup, the requester is returned the next peer it should ask for the key. One can argue that in the case of Chord, iterative lookups are much better: recursive lookups do not have the advantage of proximity-awareness. Also, note that iterative lookups have the advantage of letting the client handle failures more easily.

5a Explain how two-phase commit works. *5pt*

Make sure that you explain (1) coordinator sends vote-request; (2) participants respond; (3) coordinator sends decision; (4) participants ack.

5b Explain what happens when a participant, who is in the READY state, times out because it hasn't received a response from the coordinator yet. *5pt*

In that case, P can check whether any of the other particpants has made a transition to either ABORT or INIT (in which case P can abort) or COMMIT (and commit as well). The difficulty is when all others are in READY: they all need to wait until the coordinator recovers.

5c If we use two-phase commit for a distributed transaction, can we allow a coordinator to issue two distributed transactions (involving the same participants) at the same time? *5pt*

Yes: the local transaction managers at the participants will handle any concurrency issues. What is seen here is that the use of 2PC is completely independent of the semantics of specific transactions.

6a How can a Web hosting service help in handling flash crowds? *5pt*

Crucial for a correct answer is that you not only state that content is replicated, but that the origin server is assumed to be capable of redirecting requests, but perhaps no longer in also returning content-rich responses. Note that distributed request distribution is really tricky business.

6b Akamai uses DNS-based redirection. Explain how resolution of the name www.philips.akamai.net would work. *5pt*

The trick is that the regular DNS will resolve the name philips.akamai.net, pointing to a DNS server that is controlled by Akamai. If we use iterative DNS name resolution, that server will know the IP address ot the requesting client, and be able to decide to which server (with logical name www.philips.akamai.net) it can forward the request.

6c Explain the difference between content-aware and content-blind caching for Web applications by means of an example. *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. For example, if an edge server had once received the query "select ALL FROM books WITH author=Irving", it can cache that result. Later, when receiving a query "select ALL FROM books WITH author=Irving AND date<2008", the edge server should be able to recognize that this is a subquery, and that it can thus look into its local cache. 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. In our example, the two queries would each get a unqiue ID, which is then used to do a cache lookup.

***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.*