Dept. Comp. Science **Midterm Distributed Sys.**
Vrije Universiteit **25.10.2005**

1a Give an example of a 3-tiered client-server architecture. *5pt*

One, nowadays very common example, is the organization of a Web site. It consists of a Web server (first tier), which talks to an application server (second tier), which, in turn, communicates with a database server (third tier). Other examples include a clint machine with the user interface communicating with an application server, in turn communicating with a database server.

1b Despite the fact that multi-tiered architectures do not really solve any problems inherent to distributed systems, they have one practical advantage. What is this advantage? *5pt*

These architectures considerably simplify systems management by physically decoupling the logical application layers across multiple machines.

2a What is the difference between synchronous and isochronous transmission mode? *5pt*

In synchronous transmission mode, only a maximum end-to-end delay for each unit of a data stream is given. This is useful for real-time sensor data to guarantee that observations are accurate. In isochronous transmission mode, there is also a minimum end-to-end delay specified generally to ensure that a receiver can process incoming data units in real time. It is primarily used for multimedia streaming.

2b What is meant by stream synchronization? *5pt*

Stream synchronization refers to the fact that packets of two or more different (continuous or discrete) data streams should be delivered in a specific order at the receiver. So, for example, if PDU[i,j] denotes the jth packet from stream i, synchronization may dictate that packets are to be delivered in the order PDU[1,1], PDU[2,1], PDU[1,2], PDU[2,2], PDU[1,3], PDU[2,3], etc.

3a What is the difference between an iterative and a concurrent server? *5pt*

An iterative server handles a single request at a time, mostly by means of a single thread. A concurrent server can handle multiple requests at the same time, often by means of one thread that picks up incoming requests in order to pass it on to another thread that completes the request processing.

3b Explain how you would organize an object server so that it can *simultaneously* support the iterative and concurrent way of handling requests (for different objects). *5pt*

The key to this answer are object adapters with their activiation policy. By simply having one adapter follow the policy that a request is to be handled by the adapter, and another that it spawns a new thread on every request, both types of request handling can be handled.

4a Explain how Lamport's way of adjusting logical clocks works. Be precise! *5pt*

You need to explain that every process P_i maintains a local counter C_i that is incremented at every event, notably when sending and receiving a message. When process P_i sends a message m, it receives a timestamp $T(m) = C_i$. When m is received by P_j, the latter sets its local clock to $\max\{C_j, T(m)\} + 1$.

4b Explain how Lamport timestamps can be used to realize totally ordered multicasting. *10pt*

Grading: *The final grade is calculated by accumulating the scores per question (maximum: 45 points), and adding 5 bonus points. The maximum total is therefore 50 points.*

Dept. Comp. Sc.	Distributed Systems
Vrije Universiteit	20.12.2005

Part I

This part covers the same material as the midterm exam.

1a Explain the principal working of a remote procedure call (RPC). *5pt*

Your answer should include the fact that a client is provided with a stub (or proxy) that implements the interface of the procedure as implemented at the server. The stub is repsonsible for transforming the call into a message, specifying the procedure and the parameter values. The stub sends the message to the server, where the server-side stub unmarshalls it and subsequently calls the local implementation. The result is sent back to the client stub as a message. This message is again unmarshalled, after which the result is passed back to the calling client.

1b Give two compelling arguments why an RPC can never provide the same semantics as a local procedure call. *5pt*

Access transparency cannot be achieved by RPCs as, in most cases, it is difficult or impossible to handle pointers to local data structures. Also, RPCs generally operate across a network so that masking failures becomes a major issue. In theory and practice, completely hiding the occurrence and recovery from a failure is impossible.

1c What is the main difference between a remote method invocation (RMI) and an RPC? *5pt*

RMI generally provides a systemwide object reference mechanism. This means that objects can be referenced from any (remote) machine. As a consequence, it becomes possible to pass "pointers" (i.e., references) as parameters in an invocation.

2a Imagine a file server keeping a table of (client,file)-pairs, identifying which clients accessed which files. Is this a stateful server? Morivate your answer. *5pt*

It all really depends whether keeping this information is crucial for the proper working of the file server. If not, that is, the table as such may be lost without affecting the functional behavior of the file server, then clearly the table itself does not indicate a stateful design. If maintaining the table is crucial, then obviously we are dealing with a stateful design.

2b Explain what a message-queuing server is and whether it can be designed to be stateless. *5pt*

A message-queuing server is responsible for storing and forwarding messages in a message-queuing system. For a stateless design, the issue is whether the server can permit to loose all its information regarding clients. In fact, this is possible. Of course, it may be the case that it also looses the messages it had stored, but this has to do with the reliability and fault tolerance of the server, not whether it adheres to a stateless or stateful design.

2c The X Window system refers to the X kernel running on the client machine as the "X server", and the application running on the compute server, as the "X client." Why is this actually a correct usage of client/server terminology? *5pt*

The fact is, that the X kernel simply regards the client machine as the one holding the resources it should control. These resources may be used by applications, which them become clients. To use the resources, they send messages to the X server who will then react and respond. On the other hand, the X server will report on events, such a keystrokes and mouse moevements, which are essentially called back *to the client application.*

3a Consider a client that attempts to synchronize its clock with that of a time server once every minute. It sends a number of requests, with the results shown below. How will the client adjust its clock after receiving a response? Time is given in (hr:min:sec:msec). Processing time at the server is zero. *5pt*

Sent at (local time)	Round-trip delay	Response
10:54:00:00	18 ms	10:54:00:10
10:55:00:00	24 ms	10:55:00:12
10:56:00:00	22 ms	10:55:00:10

The first response is received at 10:54:00:18, and apparently it took 18/2 units to get the answer to the client. The latter is running slow and will adjust its clock to 10:54:00:19. Likewise, the second measurement will show the clock running in sync with the server. The third response is received at 10:56:00:22, which took 11 ms to get to the client. The actual time should thus have been 10:55:00:21. The client will slow down *its clock, e.g., by skipping the next 1 ms.*

3b Explain the principle of the Berkeley algorithm for adjusting the clocks in a distributed system. *5pt*

The time server probes each machine, asking for its local time. After that, it computes an average and tells every member how it should adjust its clock.

3c The communication layer in distributed systems can keep track of causally related messages. What are two major objections against this functionality? *5pt*

(1) The system can keep track only of potential *causaliteit. Actual causality can only be captured at the application level, which is outside the scope of a distributed system. (2) In many systems, there is a lot of out-of-band communication, for example, by means of external databases. Such communication may make messages also causally related, but this cannot be captured by the communication layer.*

Part II

4a Is the following sequence of events allowed with a sequentially-consistent store? What about a causally-consistent data store? Explain your answer. *5pt*

P1:	W(x)a		W(x)c		
P2:		R(x)a	W(x)b		
P3:		R(x)a		R(x)c	R(x)b
P4:		R(x)a		R(x)b	R(x)c

Considering that P3 and P4 in the end read different values for x shows that this is not sequentially consistent. It does adhere to a causally-consistent store, as the causal dependency $a \rightarrow b$ for x is preserved when delivering x to processes.

4b Explain why *writes-follow-reads* consistency guarantees that causal relations are maintained when used for the implementation of a distributed news system. *5pt*

Assume that Alice reads an article A (at location L1) and decides to post a response B (at location L2). At that moment, the original article A will first be moved to location L2 as well, implying that at L2 the postings and possible reactions will always be jointly available.

4c What are the conditions to prevent read-write and write-write conflicts in a quorum-based system. *5pt*

Let N_R denote the number of servers to access for reading; N_W the number for writing, and N the total number of servers. Then (1) $N_W > N/2$ and (2) $N_R + N_W > N$.

5a Show that with 4 processes of which one is faulty, that 3 processes can come to an agreement irrespective of the message communicated by the faulty process. *5pt*

The easiest solution is to reproduce Fig. 7.4 and explain that processes send each other their value, as well as forward what they have been sent.

5b Consider a print server with three possible events: (M) notify the client that printing is done; (P) print; (C) crash. When a client is notified that the print server has just recovered from a crash, it can follow 4 different strategies: (1) Never reissue the print request, (2) Always reissue the print request; (3) Reissue only if the delivery of the print request has not been acknowledged; (4) Reissue only when delivery of the request has been acknowledged. Show that if the server always notifies the client after printing, it is impossible to devise a scheme in which the print job is never lost or never duplicated. *10pt*

The answer to this question lies in Fig. 7.7, following the server strategy $M \rightarrow P$.

6a NFS is arguably not a file system. Explain why. *5pt*

NFS offers facilities to access remote file systems, and relies on the Virtual File System layer as implemented by many operating systems. The VFS provides access to the locally available file system.

6b Coda allows clients to cache files, but sends invalidation messages when a file is modified. What is the main reason for doing these callbacks in parallel? *5pt*

If doen sequentially, a faulty client may contribute considerably to completion of the notification. By doing it in parallel, the Coda server can decide after a single round which clients are not responding withut hindering the ones that are nonfaulty.

6c Coda allows a reading client to continue to operate on its local copy, even if a concurrent write takes place at the server. Why is this behavior considered correct? *5pt*

The whole issue is that as long as timing is not crucial, the reading client is considered to work on a consistent copy. That, after opening the file for reading, another process started to modify the file is less important: this could equally have happened after the client had finished its session.

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

1a Give five examples of policies that have been separated from mechanisms in distributed systems. *5pt*

(1) telling what your browser should do when it comes to caching Web pages. (2) deciding on which security keys you use for SSH, (3) setting protection bits in distributed file systems, (4) deciding how often your machine should synchronize with a time server, (5) decding on how and when you want your packages of system software to be automatically checked for updates. Similar examples easily come to mind.

1b Explain what is meant by geographic scalability problems and give an example how such a problem can be solved. *5pt*

The essence is that the behavior or performance of a system is influenced by the fact that components are placed far apart, incurring high communication latencies. There is not really that much you can do, except perhaps by replicating data close to clients, or concealing waiting times by letting the client do something else.

1c Web sites are often organized as a multi-tiered client-server system consisting of a Web server, and application server, and a database server. What are the advantages of such a scheme? *5pt*

The main advantage is that it can be easily managed. In addition, its modularity often makes it easier to replace parts, or to, for example, replicate and distribute the database.

2a What is the major disadvantage of using RPCs in comparison to messaging as in message-queuing systems? *5pt*

The main disadvantage is that due to the synchronous and nonpersistent nature of RPCs, whenever a fault occurs it has to be dealt with immediately. In MQ systems, message delivery can simply be delayed until the problem is solved.

2b Explain what subject-based routing means and how message brokers can be used to implement it. *5pt*

Subject-based routing refers to the fact that messages can be tagged with a subject as address and that only processes who have subscribed to the subject will be able to receive it. What is needed is that senders and subscribers are matched, which is typically something message brokers are good at. All messages are simply sent to the broker who maintains a list of (subject,receiver) pairs and simply forwards messages to the subscribed processes.

2c Consider a worldwide overlay network of many message brokers connected through IBM MQ series. Sketch a straightforward solution for supporting multicasting in such an environment. *5pt*

One possible solution is to organize the brokers into a single multicast backbone tree. Subscriptions for subject S are broadcast to all brokers, where each of them simple registers along which links messages on S should be forwarded when published. This induces a multicast subtree.

3a When using DNS in the following way, the user apparently is getting inconsistent answers. Explain this difference. *5pt*

```
seuss > host flits.cs.vu.nl; host 192.31.231.65
flits.cs.vu.nl has address 192.31.231.65
65.231.31.192.in-addr.arpa domain name pointer flits.few.vu.nl.

seuss > host entry.van-steen.net; host 213.10.169.34
entry.van-steen.net has address 213.10.169.34
34.169.10.213.in-addr.arpa domain name pointer ipd50aa922.speed.planet.nl.
```

It is actually very simple. Although one would expect that entry.van-steen.net is registered as uniquely belonging to 213.10.169.34, this is not the case. The fact is that the address belongs to the planet.nl domain, for which it has been registered under ipd50aa922.speed.planet.nl. However, the domain name entry.van-steen.net is registered with a separate DNS server, where it is resolved to 213.10.169.34.

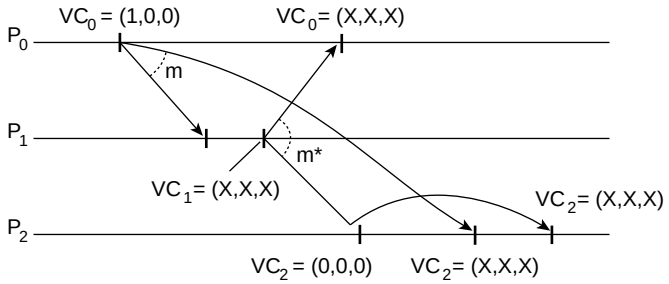
3b User `maarten` in domain `van-steen.net` has a forwarding mail address to `steen@cs.vu.nl`. Show which messages are exchanged between a user agent sending a message to `maarten@van-steen.net`, DNS, mail servers, and the recipient user agent. *10pt*

1. SND requests MX record for van-steen.net. 2. DNS returns domain name. 3. SND requests IP address of mail server of van-steen.net; DNS returns IP address (4 is often returned along with 2). 5. SND sends mail to mail server MS1 at van-steen.net. 6. MS1 looks up mail server for cs.vu.nl (as before). 7. MS1 sends mail to mail server MS2 at cs.vu.nl. 8. MS2 forwards mail to user's mail box/user agent.

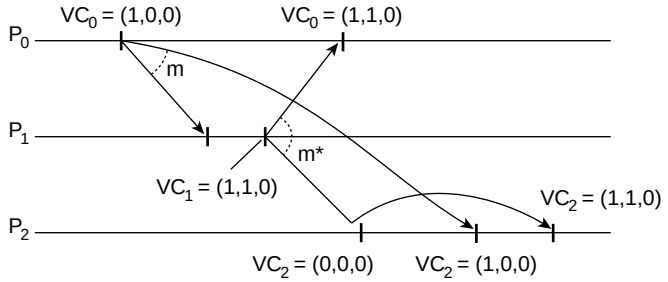
4a Explain how Lamport timestamps work and show by example that they do not necessarily capture (potential) causality. *5pt*

Lamport timestamps are explained in the book. For the example, simply take two concurrent events with different timestamps. It should be obvious that there is no potential causality.

4b Vector timestamps can be used for capturing potential causality. Fill in the vector clocks in the following figure and explain why the delivery of message m* at process P_2 is delayed. *5pt*



Because $VC_1 = (1,1,0)$ at the time m arrives at P_2, the latter notices it is missing a message that causally preceded m*. It will have to wait until that message (m) arrives.*



5a Explain what is meant by sequential consistency. Be precise! *5pt*

See book.

5b What is the essence of demanding that the set of synchronization variables adheres to sequential consistency when considering weak consistency models? *5pt*

This demand ensures that all processes see the same lock/release behavior for synchronization variables. As a consquence, everyone will have the same view on whether it is allowed to operate on the data protected by a synchronization variable.

5c Is it possible to have write-write conflicts in client-centric consistency models? Explain your answer. *5pt*

Yes. These models say nothing about concurrent access at all. Therefore, it could very well be possible that while Alice is modifying x at location A, that Bob is modifying x as well, but at another location B.

6a Explain what is meant by a virtual synchronous multicast. *5pt*

In such a multicast, the sender will have a view of who the recipients are. For VSM, it is guaranteed that a message that sent to such a view is delivered to all nonfaulty members of that view, or to none at all.

6b In virtual synchrony we can separate the ordering of message delivery at a single recipient from the fact that messages are subject to total-ordered delivery. What is the difference between the two? *5pt*

As already partly mentioned, the difference is between what happens at a single recipient, and what happens with respect to the whole group. Total-ordered delivery means that all recipients see the same messages in the same order. However, this delivery may not necessarily mean that, for example, potential causality is preserved.

7a What is the difference between two-phase locking and two-phase commit? *5pt*

The two have nothing to do with each other. 2PL refers to the fact that once a transaction releases a lock, it can no longer acquire one. 2PC refers to the way a distributed transaction instructs participants to abort or commit.

7b Two-phase commit (2PC) is said to be a blocking protocol for which reason three-phase commit was devised (3PC). How can this blocking happen? *5pt*

When the coordinator crashes in 2PC, the remaining participants may not be able to come to a final decision on the next step and will have to wait until the coordinator recovers.

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.*

Department Computer Science	**Distributed Systems**
Vrije Universiteit	**24.10.2006**

1a Explain the difference between request-level and message-level interceptors. *5pt*

1b Sketch the general organization of a self-managing distributed system as a feedback-control loop. Explain the various aspects of this organization. *5pt*

2a Explain the difference between a process virtual machine and a virtual machine monitor. *5pt*

2b Give an example illustrating that a multithreaded client can improve distribution transparency in comparison to a single-threaded client. *5pt*

Just think of a client communicating with an actively replicated server. Obviously, by executing the (assumed synchronous) calls simultaneously we can improve the response time in comparison to calling each replica one at a time. The parallel call mimics the behavior of synchronously calling a single server.

2c Active replication generally requires totally ordered message delivery. Is this condition sufficient in the case of actively replicated multithreaded servers? Explain your answer. *5pt*

No. The problem is that although messages are delivered in the correct order to the server, we also need to guarantee that they are processed in the same order. In a multithreaded server, this means that thread scheduling needs to be deterministic as well: all threads are scheduled in the same order at every server.

3a Explain the relationship between a node identifier and a key in Chord. *5pt*

Node identifiers and keys are assumed to be drawn from the same m-bit identifier space. For a given key k, the node with the smallest identifier id $\geq k$ is responsible for handling whatever is associated with key k.

3b In Chord, the finger table for node p is defined by $FT_p[i] = succ(p+2^{i-1})$. Assume a 32-bit identifier space and consider the following finger table for node 18. Explain to where node 18 forwards a lookup request for the following keys: $k = 26, 20, 18, 17, 29$. *5pt*

$$FT_{18} = [20, 20, 28, 28, 4]$$

The requests are forwarded, respectively to: 20, 20, 18, 4, and 28.

3c Explain how a node (with unique identifier p) can easily join a Chord ring. *5pt*

It is important that you note that we assume that p knows at least one node of the Chord system, say, q. It then simply sends a lookup request for $succ(p+1)$ to q, that is, it requests lookup(p+1). This will return the node with smallest id $\geq p+1$, which will be the successor of p in the ring. If that node keeps track of its predecessor, insertion is then straightforward.

4a What is meant by sequential consistency? *5pt*

This type of consistency prescribes that if we consider a collection of concurrently executing processes that share a data item x, that the order of operations on x as specified by each program that is executed by the processes is respected, and that the effect of all operations as seen by every process are consistent with a serial execution of one program after the other.

4b What is the most important effect in terms of performance when using synchronization variables? Explain your answer. *5pt*

The most important effect is that multiple read and write operations can be performed by a single process without the need for synchronization with other processes. The result is a performance improvement as there is no need for communication as long as the rules for accessing and modifying synchronization variables are respected.

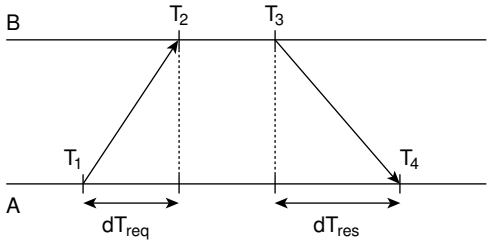
4c Explain why entry consistency and distributed objects form a natural match. *5pt*

Entry consistency is all about associating synchronization variables with a group of data. This is exactly what objects actually do. By guaranteeing that object invocations are atomic, we actually provide entry consistency in a way that naturally matches the logical organization of data by objects.

4d Consider a system in which remote objects are replicated and which guarantees entry consistency. To that end, *all* method invocations are multicast systemwide in a totally ordered fashion. Does this implementation provide stronger consistency guarantees? Explain your answer. *5pt*

In fact, it does. Where entry consistency requires total ordering of operations per (replicated) object*, totally ordered multicast of all invocations effectively guarantees sequential consistency at the granularity of method invocations.*

5a Explain by means of the following diagram, how *A* can estimate the offset of its clock relative to that of *B*. *5pt*



A will have to send a message timestamped with T_1 to B. B will record T_2 as well as its transmission time T_3, which are both put into the response. At T_4 A can compute its offset as

$$\theta = \frac{(T_2 - T_1) + (T_4 - T_3)}{2}$$

5b Suppose that in the previous figure, we cannot assume that $dT_{req} \approx dT_{res}$, as is often the case in wide-area networks. How does this affect the accuracy of *A*'s estimation? *5pt*

Go for extremes, and you will see what happens. With dT_{req} being very small and dT_{res} very large, then the long time it takes for the response to make it back to A, will make A believe its clock offset is very large. In fact, this is not the case. If the request took very long to reach B, while the reponse was returned in a jiffy, A will come to the conclusion that its offset is relatively low.

6a Akamai's CDN deploys replication of documents through standard Web proxy caching techniques. Explain how Akamai does this replication. *10pt*

It boils down to explaining Fig. 12-20.

6b What is strongest kind of consistency would you argue that Akamai provides. Explain you answer! *5pt*

First, it is important to note that updates are always carried out through an origin server. This means that there are no write-write conflicts. Second, because updates lead to modifying the name of an embedded document, and *because each initial request is always forwarded to the origin server, clients will, in principle, never get to see a stale document. Therefore, it can be argued that Akamai provides data-centric consistency, and notably sequential consistency at the granularity of whole-document updates. Anther acceptable answers are continuous consistency with no deviations in value, time, and ordering. However, mentioning client-centric consistency is just too weak (although strictly correct): it provides systemwide consistency. Note that these consistencies become weaker if the main page is allowed to be cached.*

6c To support replication of Web applications, a CDN can deploy *content-aware caching*. Explain how. *5pt*

When deploying a content-aware cache, an edge server or proxy cache server assumes that queries sent to the origin server adhere to only a limited number of templates. This allows the server to store the results of queries in a local cache such that it can easily see whether a next query can be answered from local data only. For example, a query for selecting all elements in a range [0,100] returns an answer that can also be use to answer a query for all elements in a range [10,90].

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.*

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.*

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.

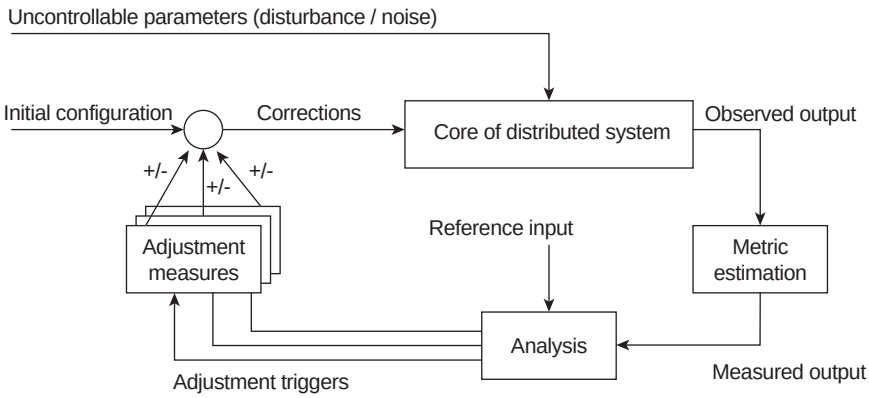
1b What do you see as the main scalability problems with a video conferencing application? *5pt*

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.

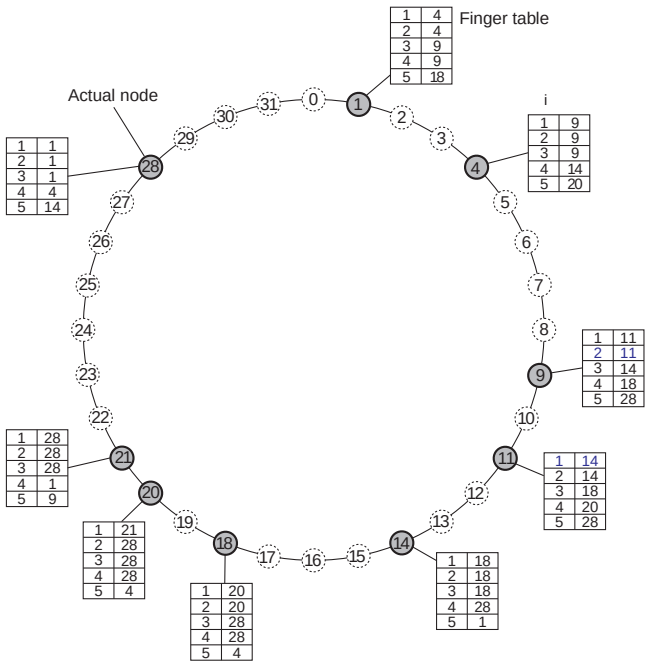
2b An RPC is a transient, synchronous form of communication. What does this mean and what are the main drawbacks? *5pt*

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*



$FP_p[i] = p + succ(p + 2^{i-1}$, *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.*

Department Computer Science	**Distributed Systems**
VU University	**08.01.2008**

1a Some distributed systems can simply not scale, no matter which techniques are applied. Give an example of such a system and explain why scalability is (close to) impossible. *5pt*

Think of a centralized database system that needs to handle many concurrent updates (transaction systems often have this). In order to improve scalability, one could consider replicating the database, but that would instantly lead to a serious synchronization problem when updates need to take place.

1b Explain how code shipping as is done with Java applets in the case of Web services, can help improve scalability. Which scalability problem is actually being tackled? *5pt*

With code shipping, a client will be allowed to prepare, for example, the input for a remote database. As such, we reduce the communication between client and server, and as such are overcoming latency problems that hinder scalability.

2a Explain the difference between a process virtual machine and a virtual machine monitor. *5pt*

A PVM is essentially an interpreter or emulator that takes the executable of an application and runs it on an operating system. An example of a PVM is the one used in Java. A VMM is a subsystem that mimics a specific instruction set, thereby hiding the underlying hardware architecture. Typically, a VMM is capable of running a complete operating system along with several of its applications.

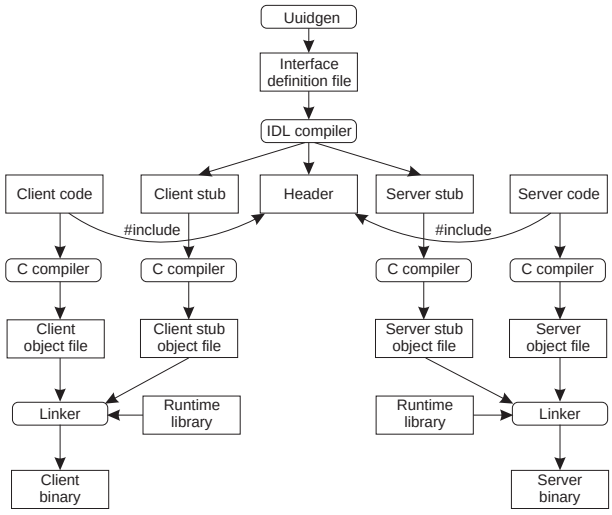
2b Explain how virtual machine technology is used in the PlanetLab system, and how it helps to support multiple applications. *5pt*

PlanetLab makes use of so-called vservers, which is essentially a specific and isolated Linux environment. Applications can have their own private environment, implemented by means of a chroot equivalent. This technology helps in setting up parallel vservers across multiple PlanetLab nodes. An application in that case will be allocated a vserver on different machines, after which it can make use of the resources available to those respective machines. In this sense, vservers help to run distributed applications concurrently, while keeping them isolated from each other.

2c Virtual machines help in migrating applications between different servers. Why is this easier than general solutions for strong mobility? *5pt*

The real issue here is that strong mobility requires that a complete execution environment (consisting of the process stack and such) is migrated to a different machine. That can only be done easily if the target machine and operating system is exactly the same as the source. With virtual machines this problem goes away: the execution environment as a whole, including the parts on which it depends, are migrated, effectively reducing the migration to a special case of weak mobility.

3a Explain which functions are implemented in the runtime library of an RPC system. See also the following figure. *5pt*



The library typically consists of (configurable) send and receive primitives, such as parameterised primitives for socket communication. In addition, we should expect to see all kinds of routines for converting machine-dependent data structures into network- and machine neutral representations.

3b Executing an RPC requires that a client can contact a server. How does it find the contact point for a server, and what does that contact point consist of? *5pt*

Looking up a contact point is typically done by contacting a name server at a well-known address and passing it an identifier of the RPC server that the client wants to contact. The name server, such as a port mapper, returns a transport-level address, consisting of an IP address and port number.

3c RPC systems cannot support local references (such as pointers), as these refer to objects only locally accessible. Instead, global object references should be used, if possible. Outline an implementation of such a reference. *5pt*

One possible implementation is that it refers to an RPC call running on the machine wanting to provide a local reference, such as a get or put statement. In Java, this problem is solved by implementing the reference as a complete client-side stub, with the contact address hard-coded in the stub. This solution is possible because Java stubs can be migrated across the network and directly executed at the receiving side.

4a Explain the principle of an epidemic protocol. *5pt*

Your answer should at least include that a process P randomly selects another process Q to exchange new data items, following either a pull, push, or push-pull protocol.

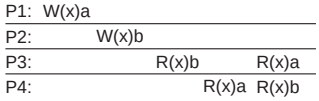
4b Sketch how the nodes in a distributed system can each compute the size of the system (i.e., the total number of nodes) using an epidemic protocol. *5pt*

Let each process i maintain a variable x_i. Every time a process i contacts another process j, $x_i, x_j \leftarrow (x_i + x_j)/2$. In this case x_i will converge to $\bar{x}$. If we initialize x_1 to 1 and all the others to 0, each process will compute $1/n$, where n is the number of processes. Your answer may also include the concurrent computation of max, so that we do not have to determine who is allowed to start with 1.

4c What is the problem with removing a data item in an epidemic system, and how can this problem be solved? *5pt*

The problem is that if there is only a single process left that still holds the item to be removed, it will almost instantly start to spread that item as something new: the others don't have it. The standard solution is to spread an update on the data item stating that is officially declared dead, which is the equivalent of stating that it should be considered obsolete, and thus as have being removed.

5a Why is the following data store not sequentially consistent? Is it causally consistent? Be sure to explain your answer. *5pt*



It is not sequentially consistent because P3 and P4 are reading the effects of concurrent writes (by respectively P1 and P2) in different orders. It is causally consistent because there are no causal relationships that need to be obeyed.

5b Consider a system that combines read-your-writes consistency with writes-follow-reads consistency. Is this system also sequentially consistent? Explain your answer. *5pt*

No, it is not sequentially consistent. Although the combination effectively provides location-independent consistent behavior for a single process, it does not guarantee that when there are two concurrent write operations at different locations, that the effect of those writes will be seen everywhere in the same order. As a side note: if we combine all client-centric consistencies, it turns out that you will have a sequentially consistent system.

5c Consistency can also be formulated in terms of numerical deviations. Give an example of such a form of consistency, and sketch how that consistency can be enforced. *5pt*

Typically, one may want to specify for a stock exchange system that the values of copies of the same share that are replicated at different locations, deviate no more than 1% from each other. One way

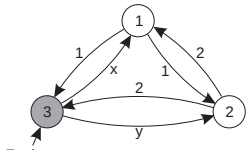
to enforce such a consistency is to gossip update information between nodes, such that each node maintains information on how far others are deviating from the value of the freshest value. You can easily obtain points if you make this more concrete by mentioning the vectors $TW[i,j]$ and $TW_k[i,j]$ that record what k knows about how far i is in processing updates that were initiated at j and showing that $TW[k,k] - TW_k[i,k]$ exceeds a limit, it's time to propagate updates again.

6a What is a *k* fault-tolerant group, and how does *k* depend on failure semantics? *5pt*

A k fault-tolerant group of processes is a group that continues to operate as expected even in the presence of k failing processes. In the case of crash/performance failures, you need $k+1$ processes; when dealing with Byzantine failures but processes do not communicate with each other, you need $2k+1$ members so that you can perform majority voting. With communication between processes, $3k+1$ members are needed.

6b Show that having three processes of which one is faulty, is not enough to guarantee agreement between the two nonfaulty ones in a Byzantine setting. *5pt*

The easiest thing to do is to prove this by contradiction: let the three processes pass their information in two rounds, and see whether you can get a majority. Essentially, this means drawing Figure 8-6:



(a)

1 Got(1, 2, x)
2 Got(1, 2, y)
3 Got(1, 2, 3)

(b)

1 Got	2 Got
(1, 2, y)	(1, 2, x)
(a, b, c)	(d, e, f)

(c)

7c Explain what a flash crowd is, and why it is so difficult to develop general-purpose predictors. *5pt*

A flash crowd is a sudden burst in requests for a specific Web document. Although predictors can be made, they are very specific to Web request traffic, making it virtually impossible to come up with solutions that will automatically configure them for an unknown trace of requests.

7b What is the best measure against flash crowds? Explain briefly how it works. *5pt*

The best measure is simply to keep a number of replicas available that will return the responses to client requests. However, we still need to be able to process those requests. This means that the original site will accept incoming requests and subsequently redirect them to the resepctive replicas.

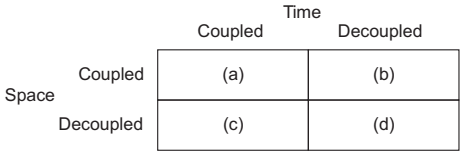
***Grading:** The final grade is calculated by accumulating the scores per question (maximum: 90 points), and adding 10 bonus points. The maximum total is therefore 100 points.*

Department Computer Science	**Distributed Systems**
VU University	**19.12.2008**

1 Distributing data and processes may help to address size scalability, but may easily introduce geographical scalability. Give a well-known example to illustrate this point, as well as a solution. *5pt*

A well-known example is DNS by which the complete name space is distributed such that name resolution is also distributed. However, DNS may easily suffer from geographical scalability as many long-haul connections need to be crossed during name resolution. The solution to that problem is extensive result caching, which works because DNS data is mostly only read.

2a Processes can be decoupled in time and space. For each combination of decoupling (see figure), characterize the type of distributed system through an example. *5pt*

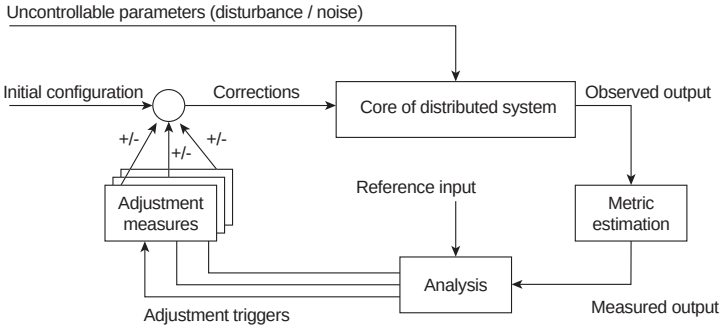


(a) Distributed systems in which processes communicate only through TCP connections. (b) Systems based on e-mail communication. (c) Group communication using multicast addressing, such as in event-based publish-subscribe systems. (d) Systems deploying shared data spaces in which subject-tagged messages are stored in a database.

2b Explain the concept of interceptors as used in middleware and why they can be useful. *5pt*

An interceptor allows an application to break in the usual flow of control between an application, middleware, and the underlying communication network. A request-level interceptor is used for modifying the request as sent between an application and the middleware; a message-level interceptor is used to modify the requests sent between middleware and transport-level communication interface. Interceptors are used to transparently modify the request-reply policy as deployed by the middleware. As such, they are useful as a mechanism to modify the behavior of the middleware to satisfy the needs of applications without having to modify those applications.

2c Explain how a feedback control loop works by providing an example of a distributed system that fits the following figure. Explain each component, as well as each connection to/from a component. *10pt*

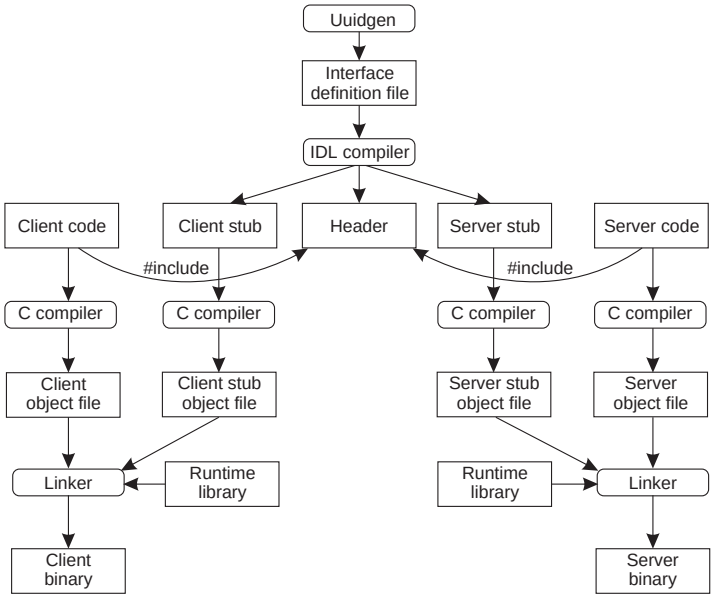


See book. A good example is a web hosting service such as Globule, which supports automatic replication of Web pages.

3a Unlike local pointers, having systemwide object references helps to improve access transparency in RPCs. How can such object references be implemented? *Hint: think of how Java realizes remote method invocations.* *5pt*

Taking Java as our example, we can simply take a complete client-side stub as an object reference. The stub contains the contact address of the remote object, but because the stub is a piece of code that can be interpreted by any Java virtual machine, it can be copied and moved between different processes. In this way, it is indeed a systemwide object reference.

3b The figure below shows how an RPC system works in practice. Explain what is in the runtime library. *5pt*

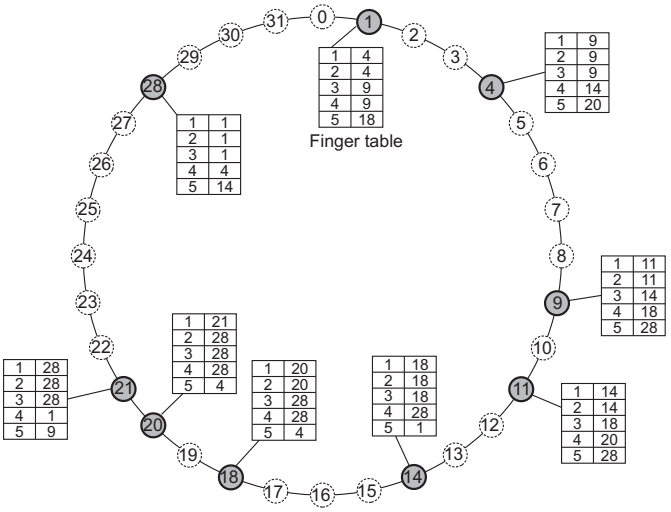


Typically parameterizable calls to the underlying transport-level interface, but notably also library routines for converting data structures to host-independent representations.

3c In RPC, a client needs to **bind** to a server. What does this mean and how can it be realized? *5pt*

Binding in this case means that the client makes all the necessary preparations to allow it to call procedures maintained by the server. It is accomplished by first having the server register, one way or the other, to which network-level and transport-level address it is accepting incoming requests. This can be done through a separate, well-known directory server. A client asks for this contact information, after which it can, for example, set up a TCP connection to the appropriate server.

4a Explain how name resolution works in Chord by resolving $k = 30$ starting from node 21 in the following example. Do the same for $k = 19$ from 21. *5pt*



(1) $21 \rightarrow 28 \rightarrow 1$. *(2)* $21 \rightarrow 9 \rightarrow 18 \rightarrow 20$.

4b In Chord, the finger table entry $FT_p[i]$ of peer p is equal to $succ(p + 2^{i-1})$. Explain how Chord's finger tables can be extended to incorporate proximity routing. *5pt*

There is no reason why p can't just keep a whole number of references to nodes in the range $[p + 2^{i-1}, p + 2^i - 1]$. *In that case, when it is required to look up a key k, it can decide to route that request to the peer with the smallest id* $\geq k$ *that it knows, but which is also closest to itself.*

4c As in any other naming system, it is possible to look up up a key in Chord recursively or iteratively. Explain the differences, as well as some advantages and disadvantages of either approach. *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 and are also more vulnerable to security attacks. On the other hand, if the response follows the same route as the recursive lookup, replication can be done moer effectively.

4d Give two approaches to using Chord for implementing a distributed file system. *5pt*

The simplest one is by hashing a file name fn to $h(fn)$ and storing the file at the peer with the smallest id $\geq h(fn)$. Alternatively, we can also store blocks at peers instead of whole files. For example, a block with content d can be stored, together with its hash $h(d)$ at the peer with id $\geq h(d)$.

5a Explain how a blocking primary-backup protocol works, as well as its nonblocking variant. Why is a primary-backup protocol in which the primary moves to the location of the writer, never blocking? *5pt*

See Figures 7-20 and 7-21, respectively. The variant in which the primary moves to the writer is always nonblocking as there is otherwise hardly any advantage in having such a variant. With moving the primary, it becomes possible to efficiently execute a series of updates locally.

5b Sketch the design of a simple, centralized consistency protocol for active replication that realizes sequential consistency. *5pt*

Simply install a sequencer: when an operation needs to be carried out at multiple replicas, first fetch a sequence number from the sequencer and then forward the operation to the replicas.

6a One can argue that NFS is not really a file system. Explain why. *5pt*

NFS is actually a protocol and its implementation to make an existing local file system available to remote clients. This is best illustrated by Figure 11-2.

6b Mention two different measures that were designed into NFS version 4 in order to let it operate better in wide-area networks. *5pt*

(1) Moving to a stateful design by allowing clients to locally cache files an operate on them. (2) Supporting compound procedures by which multiple requests could be sent in a single message.

7a Explain how Akamai uses standard Web-caching techniques to effectively implement server-initiated replication of Web pages. *5pt*

In essence, you need to explain Figure 12-20.

7b Explain how a content-aware cache works in edge-server systems that support replication of Web applications. *5pt*

With a content-aware cache, the client's edge server assumes that queries can be classified according to a limited number of so-called templates. Each template is essentially the same as a prototype function declaration. When a query with template T is issued for the first time, the server's response is stored locally. A next time, the edge server can execute a so-called containment check to see whether the query addresses a subset of what was requested before. If so, the response can be looked up locally.

***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.*

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

MAKE SURE THAT YOUR HANDWRITING IS READABLE

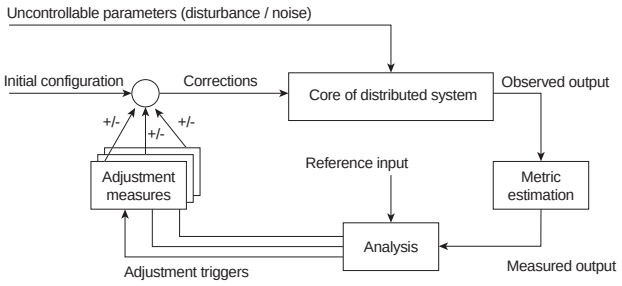
1a Explain what is meant by request-level and message-level interceptors in middleware. *5pt*

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*

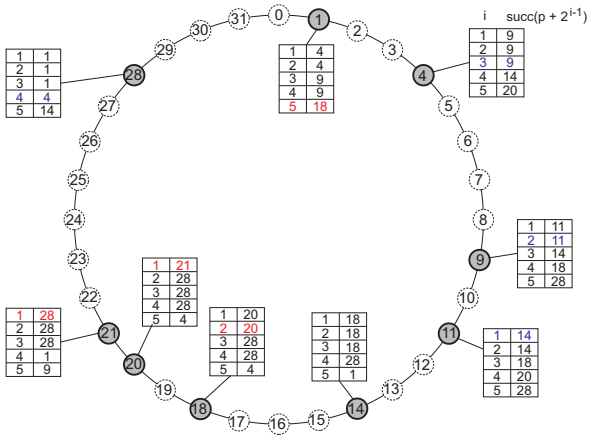
The problem originates from having a (suspected) server crash, detected by the lack of a response in the case of an RPC. What the client-side software can do is either resend the request until it finally gets a response (at-least-once semantics) or immediately reports the failure to the client application, 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.*