Department Computer Science	**Distributed Systems**
Vrije Universiteit	**22.02.2012**

BE SURE THAT YOUR HANDWRITING IS READABLE

1a Explain Lamport's happened-before relationship and provide a distributed implementation of the associated logical clocks. *5pt*

Consider events a, b, and c. (1) If a preceded b in the same process, then $a \rightarrow b$. (2) If a is the sending of a message m, and b is the receipt of that message, then $a \rightarrow b$. (3) If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$. This relation can be implemented by letting each process P_i maintain a local counter C_i.

1: *For any two successive events that take place within P_i, C_i is incremented by 1.*

2: *When a message m is sent by process P_i, the message receives a timestamp $ts(m) = C_i$.*

3: *When a message m is received by P_j, P_j adjusts its local counter C_j to $\max\{C_j, ts(m)\}$; then executes step 1 before passing m to the application.*

1b Give an example of how Lamport's logical clocks can be used to realize distributed mutual exclusion. *5pt*

There are several answers possible, but an easy one is briefly explaining the Ricart-Agrawala algorithm:

- *When a process wants to access a resource, it sends a request to the other processes, and to itself.*
- *Upon receipt of a request, a process can return a grant when not interested itself. If its own request had a lower timestamp, it will just place the received request in the queue. If its own request had a higher timestamp, it will return a grant to the sender.*

1c Show by an example that Lamport's clocks cannot generally capture causal relations. *5pt*

Simply provide an example with two independently sent messages having different timestamps.

2a Explain how TCP-handoff works. *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. It is important that you mention that the server spoofs the switch.

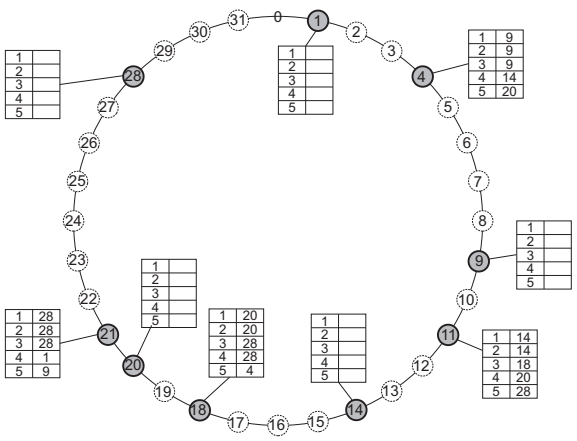
2b Sketch a solution for an efficient, content-aware, TCP-handoff. *5pt*

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.

2c How would you realize TCP handoff if the servers are placed in different autonomous systems? *5pt*

Again, there are different solutions possible, but one could be to tunnel the connection between switch and Web server, effectively emulating that the two are operating on the same LAN.

3



3a Fill in the missing Chord finger table for nodes 1, 9, 14, 20, 28. *5pt*

Node 1: (4,4,9,9,18); Node 9: (11,11,14,18,28); Node 14: (18,18,18,28,1); Node 20: (21,28,28,28,4); Node 28: (1,1,1,4,14).

3b Show all hops for the following key lookups for the shown Chord-based P2P system: *5pt*

source	key
1	12
9	3
14	20
14	31
20	18

12@1: [9, 11, 14]; 3@9: [28, 1, 4]; 20@14 [18, 20]; 31@14 [28, 1]; 18@20: [4, 14, 18]

3c Explain how node 7 can join the Chord network, assuming it knows only node 21. *5pt*

It simply asks node 21 to lookup who's responsible for 7, and inserts itself before that node.

4a Explain how a content-aware Web cache works by considering the processing of database queries. *5pt*

Consider a query asking for a list of items (such as all albums by Evan Parker). A content-aware cache will store the result, but the next time a query is forwarded, it executes a containment procedure. Specifically, if a client would require all Evan Parker albums since 1975, the content-aware cache would be able to see that it already stores those results.

4b Web-hosting services such as Akamai redirect clients to a best replica server using DNS. Explain, by example, how this redirection works. *5pt*

When a client asks for a page, we can assume that this page contains references to other files accessible through a service-specific URL like page.example.cdn.net. The important thing to note is that URL will eventually be partly resolved by a DNS server that is under control of the Web-hosting service. In this example, we can assume that cdn.net will point to a server in the CDN, which will continue with resolving page.example. In any case, when reaching the server for resolving the last part of the name ("page"), that server will have the client's IP address and can use that for deciding which server's IP address it should return.

4c Provide details on how Web clients can get redirected to a replica server in the case of a flash crowd. *5pt*

What is important is that you realize that the origin server can do the redirection (and that it, in general, will actually do that). What it can't do in most cases, is return the content. Therefore, the origin server will keep a list of replica servers and more or less randomly redirect clients to one of those servers.

5 Consider an implementation of numerically bounded continuous consistency. Assume that each write $W(x) > 0$. Let $origin(W)$ denote the server S_i to which the write operation was submitted first, and $log(S_i)$ the log of server S_i. We assume there are N servers. For data item x, $TW[i,j]$ is defined as

$$TW[i,j] = \sum\{W(x) | origin(W) = S_j \& W \in log(S_i)\}$$

5a Give an expression for the value $v(x)$ of x assuming its initial value was 0. *5pt*

$v(x) = \sum_{k=1}^{N} TW[k,k]$

5b Give an expression for the value of $v_i(x)$ of x at server S_i. *5pt*

$v_i(x) = \sum_{k=1}^{N} TW[i,k]$

5c To be numerically bounded, we demand that $|v(x) - v_i(x)| < \delta$. What can a server do to guarantee this constraint? *5pt*

Denote by $TW_k[i,j]$ what server S_k believes the value of $TW[i,j]$ is. To keep matters simple, a server S_k can then decide to start forwarding logged writes to S_i when $\Delta_k = TW[k,k] - TW_k[i,k] \leq \delta/(N-1)$. If all servers do this, we know for sure that $\sum \Delta_k \leq \delta$.

6a Consider a set of N replica servers. Show, by counter example, that for quorum-based replication, the write set needs to be larger or equal to $\lfloor N/2+1 \rfloor$. *5pt*

Simply take a set of 2 (or 3) servers and show that if the write-set is only 1 server, you will create a conflict.

6b Assuming crash/performance failure semantics, how large must the write set be in the case of quorum-based replication in case we want to survive the failure of k servers? *5pt*

What we need is at least the normal write set, thus $\lfloor N/2+1 \rfloor$ servers. This already forms a majority. As a consequence, if we need to survive 1 failing server, we need to have 1 extra server to the write-set, and so on, leading to a total of $\lfloor N/2+1 \rfloor + k$ servers.

6c Consider the previous question. How large must the read set be? *5pt*

As the read set necessarily involves only nonfaulty servers, we merely need to guarantee that $N_W + N_R > N$, where N_W is as large as needed to survive k failing servers.

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.*