UMEÅ UNIVERSITET
Institutionen för Datavetenskap

UMEÅ UNIVERISTY
Dept of Computing Science
Code: 15

EXAMINATION

Course: **5DV020/Distributed Systems**

Teacher in charge: Yvonne Löwstedt/Lars Larsson

Semester: HT-08
Date: 2009-04-16
Time: 09.00–15.00

Name: _____

Personal ID number: _____

Unique code for this examination: **15**

Note!

This examination will be graded anonymously. This sheet will be removed before the teacher receives the rest of the examination. The above code must therefore be on all other pages when you submit the examination to the examination supervisory staff. **Memorize** your code since it will be used as reference when the results are published.
Furthermore,

- Write the answers on the answers on the same paper as the question (the back of the paper may also be used).
- Mark the questions you have solved with a cross on the next page.
- The solutions should be neatly written. The train of thought should be easy to follow. All non-obvious assumptions must be explicitly stated.

Till skrivningsbevakaren: Avskilj detta försättsblad och stoppa i kuvert som skickas till Yvonne Löwstedt, Datavetenskap.

Umeå University, 901 87 Umeå. Ph: 090 786 6131. Fax: 090 786 6126. Email:

UMEÅ UNIVERSITET
Institutionen för Datavetenskap

UMEÅ UNIVERISTY
Dept of Computing Science
Code: 15

EXAMINATION

Course: **5DV020/Distributed Systems**

Teacher in charge: Yvonne Löwstedt/Lars Larsson

Semester: HT-08

Date: 2009-04-16

Time: 09.00–15.00

Unique code for this examination: **15**

Problem	Solved	Points
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
Sum		
Grade		

Question 1 (1 + 2 + 3 points)

Lars recently got a new laptop. He only uses it to connect via wireless networks. Sadly, the Linux driver for the WIFI network interface is pretty bad, and his connection to networks is unreliable: the signal quality is low and he experiences packet loss from time to time. Also, (most likely) because it was a cheap laptop, his clock is drifting noticeably. Thus, he needs to synchronize his clock. And it is up to you to help him!

a. Name a good clock synchronization scheme.

b. Explain how the clock synchronization scheme works. You don't need to go into minute detail, but clearly show what messages are sent back and forth in the distributed system, what the nodes are responsible for, and a few words about how the scheme works.

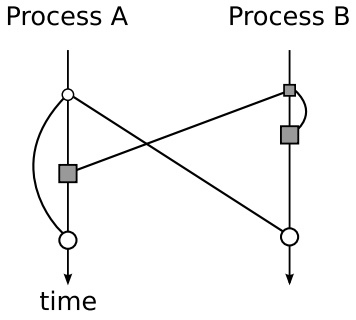
c. Motivate why the clock synchronization scheme you suggested is good for the particular case of Lars' laptop, given the circumstances presented in the text above.

We can be generous with paper — we have unlimited printer quotas!

Question 2 (2 + 2 + 2 points)

GCom requires support for several message orderings: FIFO, causal, total, and causal-total. Hybrid orderings such as the last one are interesting from many points of view, since they allow us to gain benefits of several types of orderings — without requiring much effort from us as implementers. Let us therefore devote some attention to such hybrid orderings!

For the following, assume that there are at least three processes in the system and that there is a minimum total of five messages sent between them. All processes must send at least one message (to avoid trivial solutions). You do not have to explicitly show sequencer or sequence number voting messages. Also, please provide your answer as figures with additional text (not only in writing, as the written word is open to ambiguous interpretations). The figures must be drawn in the following fashion:



A small circle denotes a send event, whereas a big circle of the same color denotes a *delivery* event. Recall the difference between receiving a message and delivering it: received messages are put in queues until they are ready to be delivered according to the chosen ordering.

a. If possible, show a message ordering that is **total**, but **not FIFO-total**. If such an ordering is impossible to show, provide a sound argument for why this is the case.

b. If possible, show a message ordering that is **total**, but **not causal-total**. If such an ordering is impossible to show, provide a sound argument for why this is the case.

c. If possible, show a message ordering that is **FIFO**, but **not causal-total**. If such an ordering is impossible to show, provide a sound argument for why this is the case.

Feel free to use this page as well...

. . . and this one.

Question 3 (4 + 2 points)

Election algorithms are useful in many scenarios, since it is often natural to regard one process as the "leader" and grant it special powers over the other processes. There are two requirements on any particular run of election algorithms to ensure that they give a correct result:

E1 *Safety*: A participant process p_i has $\mathsf{elected}_i = \perp$ or $\mathsf{elected}_i = P$, where P is chosen as the non-crashed process at the end of the run with the largest identifier.

E2 *Liveness*: All processes p_i participate and eventually set $\mathsf{elected}_i \neq \perp$ — or crash.

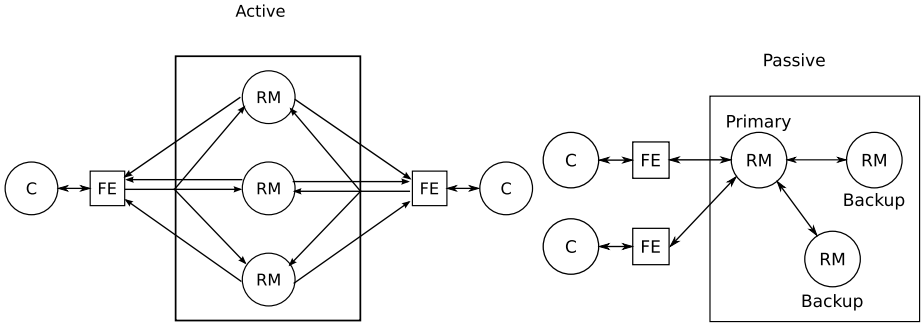
a. Choose either the ring-based or the bully election algorithm. State your choice, and describe how it works. Make sure that you do so in an unambiguous and precise way (algorithms in pseudo code are excellent for this... *hint, hint*).

b. Both algorithms guarantee E1 and E2 in the best case. However, explain how compliance to E1 and E2 of the algorithm you chose is affected by a client that crashes and is restarted. Make no assumptions about the client being able to save any kind of state (in particular, it does not know what identifier it used prior to crashing).

This page is intentionally left blank. Go on, write more about the election algorithms — you know you want to!

Question 4 (2 + 2 + 2 points)

These two pictures show the flow of communication under the active and passive replication schemes:



Important concepts related to fault tolerance and replicated shared objects are those of *linearizability* and *sequential consistency*. Both relate to interleavings of operations that act upon the replicated shared objects. For both, it is a requirement that the interleaving of operations such that the interleaved sequence of operations meets the specification of a (single) correct copy of the objects. Linearizability requires that the order of operations in the interleaving is consistent with the *real times* at which the operations occurred in the actual execution, whereas sequential consistency is more relaxed and merely requires that the order of operations in the interleaving is consistent with the *program order* in which each individual client executed them.

a. It is possible to increase the efficiency of an *active* replication scheme by allowing the clients to contact a single replica manager for read-only queries. Is such a scheme (a) linearizable, (b) sequentially consistent, or (c) neither? Motivate your answer.

b. It is possible for clients to offload the primary replica manager and contact a single backup replica manager for read-only queries in a *passive* replication scheme. Is such a scheme (a) linearizable, (b) sequentially consistent, or (c) neither? Motivate your answer.

c. A customer wants to create a site for online live betting, and wants you to suggest a scheme for the distributed database that contains all the information (things to bet on, the odds, the bets, etc). Would you recommend active or passive replication, and would you use the optimizations suggested in the two previous questions? Motivate your answer clearly.

Use this page too (if you want — you probably do).

Question 5 (2 + 4 points)

Transactions is a very important concept in distributed systems — in particular ones that deal with databases of various kinds. The idea is to define not mere operations, but operation "groups" that can be applied to a distributed object. In that context, ACID is an acronym that is well-known and used in distributed systems as well as in databases. It describes properties that one requires from a transaction-aware system for the system to be useful.

a. What does ACID stand for? Name the four properties that make up the acronym, and give a brief description of them in the context of distributed systems.

b. There are some problems related to distributed objects that can be accessed using transactions. Three classical problems are the *dirty read*, *lost update*, and *inconsistent retrievals* problems. Describe **two** of these (you may choose which two of the three), and give simple and clear examples that illustrate what goes wrong.

Question 6 (2 + 2 + 2 points)

Concurrency control can be either *pessimistic* or *optimistic*. The former works under the assumption that concurrent access to resources often occur, and must be avoided, whereas the latter works under the assumption that such clashes are relatively few, and may instead be reactively detected (and handled in some suitable way).

a. Explain the optimistic *forward validation* scheme.

b. Explain the optimistic *backwards validation* scheme.

c. Suppose that you have an application where the amount of read operations greatly outnumber the amount of write operations. Which optimistic concurrency control scheme would you use, and why? Clearly motivate your answer.

Question 7 (2 + 2 + 2 points)

A problem with distributed systems is that it is hard to get an overview of the current global state. But, thankfully, there are algorithms that we can use to determine the global state, even if it is not done in real time.

a. Explain at least two problems inherent in distributed systems related to determining the global state — what are the difficulties?

b. Why is global state important? What functionality is enabled or made possible by being able to obtain the global state?

c. Briefly (but clearly, obviously) explain an algorithm for determining global states where marker messages are used.

Question 8 (4 points)

Identify and explain (motivate) a situation where Java RMI would be a better choice for distributed computing than Web Services.

Question 9 (2 + 1 points)

Threats within Computer security are often divided into several categories. There are several different ways of categorizing them, but during the course we discussed one proposed by Shirey, namely:

- Disclosure
- Deception
- Disruption
- Usurpation

a. Give at least one brief example of a kind of attack that would fit into each category.

b. Apart from categorization above, it is common to talk about Active and Passive threats. Explain the difference between active and passive threats. Feel free to use the answers from (a) as examples of passive and active threats.

Question 10 (3 points)

Explain what a Certification Authority does, briefly mention its primary functions, and most importantly describe how it fits into the big security picture.

Question 11 (5 × 1 points)

Understanding the terminology used within a subject can sometimes (as within computer security) be fundamental. Not only does the proper use of terminology make someone sound trustworthy and educated, it is also central in understanding new material concerning the subject, and relate this to earlier experiences.
For each pair of security related terms below:

- briefly explain *both* terms and;
- explicitly state if and how they relate to each other.

The terms are:

a. LAN and DMZ

b. Biometrics and Digital Signatures

c. IDS and Chinese Wall Model

d. Buffer overflow and Key escrow

e. SSL and SYN flooding

Question 12 (-3 to 3 points)

The following questions require only a true or false answer. Correct answers give 0.5 points, whereas incorrect answers are penalized with -0.5 points. Note that the total from the question may be negative, and this will impact your final score. No answer is the safest option, and counts as 0 points. Any text besides "true" or "false" will not be taken into consideration.

Remote processes can bind new references into a Java RMI registry	
A message encrypted using private key X can be decrypted by either the private key X or the public key associated to X	
A transaction that has made a provisional commit may never be aborted	
Clock *skew* is a measurement of the rate at which a clock counts time increments (frequency of oscillations)	
A clock has to be accurate to be *correct*	
There are no mutual exclusion algorithms that can survive a single crash failure	