



58 - RI6S1OPTDX496SC



0406186522

FÖRSÄTTSLAD TENTAMEN/ EXAMINATION COVER

Jag intygar att mobiltelefon och annan otillåten elektronisk utrustning är avstängd och förvaras på anvisad plats. / I hereby confirm that mobile phones and other unauthorized electronic equipment is shut off and placed according to instructions

MARKERA MED "X" /
MARK WITH "X"



IFYLLES AV STUDENT OCH TENTAMENSVAKT /
TO BE FILLED IN BY THE STUDENT AND THE INVIGILATOR:

KURSKOD / COURSE CODE I D 2 2 0 1		EFTERNAMN / FAMILY NAME Stahl	
KURSNAMN / COURSE NAME Distribuerade system, grundkurs		FÖRNAMN / FIRST NAME Emil	
PROVKOD / TEST CODE T E N 1		NAMNTECKNING / YOUR SIGNATURE Stahl Emil / [Signature]	
TENTAMENSDATUM / EXAMINATION DATE Y/Y/Y/Y M/M D/D 2 0 1 9 - 1 0 - 1 8		SIGNATUR TENTAMENSVAKT / SIGNATURE INVIGILATOR: [Signature]	
PROGRAMKOD / PROGRAM CODE:	INLÄMNINGSTID / TIME SUBMITTED: 12.30	ANTAL BLAD / NO OF SHEETS: 13	3/26

MARKERA BEHANDLADE UPPGIFTER MED "X" OCH EJ BEHANDLADE UPPGIFTER MED "-". /
MARK WITH "X" PROBLEMS SOLVED. MARK WITH "-" PROBLEMS NOT ATTEMPTED

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
X	X	X	X	X	X	X	X	X	X										

IFYLLES AV INSTITUTIONEN / TO BE FILLED IN BY THE DEPARTMENT:

BEDÖMNING / ASSESSMENT																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	2	4																	

BONUSPOÄNG /
BONUS POINTS:

10,

SLUTSUMMA /
FINAL POINTS:

,

BETYG /
GRADE:

A

Godkänns av examinator /
approved by Examiner.....

[Signature]

Exam Distributed Systems ID2201 – Emil Ståhl

Part A

Code for part A: 3412

A:1

Answer: D

A:2

Answer: B

A:3

Answer: C

A:4

Answer: A

A:5

Answer: C

A:6

Answer: D

A:7

Answer: A

A:8

Answer: D

A:9

Answer: B

A:10

Answer: C

A:11

Answer: A

A:12

Answer: ~~D~~

A

A:13

Answer: ~~D~~

C

A:14

Answer: A

A:15

Answer: B

A:16

Answer: B

A:17

Answer: A

A:18

Answer: C

A:19

Answer: A

A:20

Answer: A

A:21

Answer: D

A:22

Answer: B

A:23

Answer: B

A:24

Answer: C

A red handwritten signature, possibly reading 'Stähli', is written in the bottom right corner of the page.

Part B

B:1

At time 82 request sent by client

At time 117 reply received at client

At time 111 request received at server

At time 120 reply sent by server

I'm using the following formula to calculate the halfway RTT:

$$\frac{((T2 - T1) - (T4 - T3))}{2}$$

$$\frac{(117 - 82) - (120 - 111)}{2} = \frac{35 - 9}{2} = \frac{26}{2} = 13$$

Now I calculate the difference between client and server:

$$82 + 13 = 111 + x$$

$$95 = 111 + x$$

$$x = -16.$$

Answer: The client should adjust its local time 16 steps.



B:2

Total order but not FIFO order.

Assume a process A sending two messages $a1$ and $a2$.

Now assume another process B sending two messages $b1$ and $b2$.

These messages are delivered at the destination as $a2, a1, b2, b1$.

And at another destination in the same order $a2, a1, b2, b1$.

This is a situation that meets the requirements of total order multicast but NOT FIFO order. Because $a1$ was sent before $a2$ and $b1$ was sent before $b2$, but they got delivered in the exactly same sequence at both destinations, i.e. total order.

B:3

View synchronous communication layer is necessary in order to ensure that all backend replicas is delivered the state changes before electing a new leader, if the current one has crashed. It also gives us total order so that all replicas receive the state changes in the same order and thus are consistent with each other.

Move? To elect a new leader from view.

1

B:4

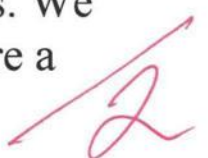
A phantom deadlock occurs as follows:

Assume we look at a process A that is waiting for a message from a process B .

Before we have time to look at / query the process B . Process A receives the message from B and then B is waiting for a message (for example an acknowledgment on delivery) from process A .

So when we look at process B it looks like it's waiting for a message from process A at the same time as process A is waiting for B .

This is the problem of examine individual/local processes. We can't get the whole picture. That's why we need to capture a global state.



B:5

Quorum based mutual exclusion works in the way of that a node doesn't need to get votes from all of the nodes in the system, but only from a quorum/part of them.

The most important thing is that two quorums share at least one node.

The advantage with this is that the system can proceed even if some of the nodes are not active / has crashed.

As compared to normal mutual exclusion algorithms where all nodes votes are needed in order to enter the critical section.

B:6

In a larger ring it's a higher risk of that multiple nodes after each other crashes at the same time. i.e. the crash of a nodes successor and successor of successor.

So that the most important thing to consider when deciding on this matter is the size of the ring structure. What works and may be a great solution for 50 nodes may not be as good of a solution for 500 nodes.

B:7

The advantages of using the gossip protocol is the good turnaround time of $O(1)$, if we assume we have multicasting.

We would need reliable failure detectors though for gossip protocol to work.

We would need to sacrifice the ability of nodes changing priority between elections, which is only possible with a ring based implementation.

move? Incomplete

B:8

I've difficulties drawing on paper but will try to explain this as good as possible.

The definition of a consistent cut is:

“For every event e in the cut,
If an event f happened before e , then f is also in the cut.”

So if I would make a consistent cut in the figure below I would make the cut to the right of event p_0 , q_0 , r_0 , q_1 and p_1 .

So that both the sending event of q_1 and the receiving event of p_1 is in the cut.

In order to make an inconsistent cut I would make the cut so event p_0 , q_0 , r_0 and p_1 is in the cut but q_1 is outside of the cut.



This would result in an inconsistent cut as of q1 happened before p1 but q1 is not in the cut.

Part C

C:1

Lamport clocks can give us a partial look of the happened before order, but not a complete description.

It works as each process keeps its own clock / counter and increments it on each event (internal event, sending, receiving). When sending a message the counter is included in the message sent. Upon receiving a message the counter of the receiving process is set to the greatest of the internal and the received counter before incrementing it. Advantages of Lamport clocks is that they do not take up so much space and that processes can come and go.

Vector clocks works in a similar way but the counters are kept in one shared vector where each process only increments its own counter. Upon receiving the two counters are merged.

The advantages of vector clocks is that each process has access to all of the other processes counters so we can get a complete view of the happened-before order.

The disadvantages of vector clocks is that they can take up big amount of space (bits) and are not optimal if we have a big churn rate / processes that come and go.

Vector clocks will always give a causal order but Lamport can sometimes have problem with this as I noticed in my homework on this matter.

I'm not sure about real-time clocks but I guess it is when every node keeps its own clock and are synchronized with other nodes in a synchronous system.

This approach would also respect the causal order of events. The advantage with real-time systems is that you wouldn't need a secondary algorithm as Lamport or vector clocks.

Real-time clocks doesn't give us a happened-before order in an asynchronous system because clocks cannot be synchronized.

Lamport and vector clocks works in an asynchronous system. But only vector clocks gives the complete happened before order.

C:2

Not checked

The first way I would implement total order multicast is with a single server that multicasts the messages. I would start with to give every message a sequence number in order to be able to order all messages at the receiver.

I'm not sure what b-multicast gives us but if reliable multicast is not included we would need to have acknowledgements on received messages, timeouts and a history of received messages.

When messages are received I would place them in a data structure (for example a hold-back queue) before delivering them to the application layer in order to make sure that messages are ordered on sequence numbers. If we only deliver messages to the application layer that are in the right order all nodes shall deliver the messages in exactly the same order, i.e. total order.

I'm not sure what kind of link-layer structure is given or if I should suggest a structure. But if you know the IP-adress of each node I think my solution above would somehow work.

This is the first way I would implement it, not sure how correct it is.

Not so sure about finding another solution but I'm thinking about a peer-to-peer network or forming an overlay network that has the structure of a ring.

When one node wants to send a message to all other nodes in the network, it would simply just send it to its successor that forwards it to its successor and so on. When the node that sent the message gets it back from its predecessor it knows that all nodes has received it.

We would of course need to implement most of the features from the last homework (Chordy). That is pointer to a nodes successor and predecessor. Procedure for stabilization, monitoring of nodes and the ability to add new nodes etc. The most important part would be that every node must keep a copy of the last message it received from its predecessor so the message do not get lost if a node crashes.

The disadvantages of this implementation is to perform stabilization and have multiple pointers to successors. It would also take some time to send messages around the ring, a problem that the first implementation doesn't have. The ring structure wouldn't need sequence number though because the message would propagate the ring one node at a time, so all node gets the messages in the same order/total order.

Advantages with a ring structure would be to have a dynamic structure that can change if we have a high churn rate.

Advantages with the first implementation would be that we do not need to perform stabilization or change the structure.
Faster message delivery.

C:3

Not answered.

Not sure how Part C went but I hope my 10 bonus points from the homework assignments can cover some of it.

Thank you for a great course! I've really enjoyed it!

/Emil

Part A

A:1 1p. What would we call a system where one node is always reacting on requests and other nodes only communicate with this node?

- ☐ A a peer-to-peer system
- ☐ B a synchronous system
- ☐ C an asynchronous system
- ☐ D a client server system

A:2 1p. What do we call a system where the upper bound of the time for operations and message delivery are known?

- ☐ A an asynchronous system
- ☐ B a synchronous system
- ☐ C a fault tolerant system
- ☐ D a high speed system

A:3 1p. What is not provided by TCP?

- ☐ A flow control, not to overflow the receive buffer
- ☐ B congestion control, to avoid network congestion
- ☐ C a guaranteed delivery of messages
- ☐ D a full-duplex stream between two processes

A:4 1p. What is a good reason for choosing UDP rather than TCP?

- ☐ A you have small messages that should be sent with little delay
- ☐ B UDP will guarantee the delivery of a message
- ☐ C you need to know that a message is handled by the remote application
- ☐ D you have large messages or a sequence of messages

A:5 1p. What is significant for synchronous communication?

- ☐ A send operations can only be performed with certain intervals
- ☐ B sender and receiver must have synchronized clocks
- ☐ C the send operation blocks and waits for the receive operation
- ☐ D a sent message will always be received

A:6 1p. What is meant by *time uncoupling* in a communication framework?

- ☐ A A sender does not need to know the name or identifier of the receiver.
- ☐ B Messages are resent if lost, providing a reliable service.
- ☐ C The sender does not wait for a acknowledgment from the receiver.
- ☐ D Sender and receiver need not be active at the same time.

A:7 1p. How are arguments passed in Java RMI?

- ☐ A remote objects as reference, all other as copies
- ☐ B serialized objects as reference, all other as copy
- ☐ C call by reference only
- ☐ D call by copy only

A:8 1p. What is given by *at least once* RPC semantics?

- ☐ A even if a failure occurs the call has been invoked at least once
- ☐ B if no failure is reported the call has been invoked exactly once
- ☐ C invocation is guaranteed to happen once
- ☐ D if no failure is reported the call has been invoked at least once

A:9 1p. How can we make two computer clocks perfectly synchronized?

- ☐ A send a message containing a time stamp every microsecond
- ☐ B we can not

- ☐ C use atomic clocks
- ☐ D use the Stanford network synchronization protocol

A:10 1p. What does the reply from a NTP server contain?

- ☐ A the send time of the reply
- ☐ B the latency of the request and the send time of the reply
- ☐ C send and receive time of request and send time of reply
- ☐ D the delta of the client time stamp and the server time

A:11 1p. What is true if A happened *real-time before* B?

- ☐ A A could have *happened before* B
- ☐ B B could have *happened before* A
- ☐ C B is a consequence of A
- ☐ D A and B occurred in the same process

A:12 1p. What can we know if we use Lamport clocks?

- ☐ A if, but not only if, a *happened before* b then $L(a) < L(b)$
- ☐ B if $L(a) = L(b)$ then $a = b$
- ☐ C if, and only if, $L(a) < L(b)$ then a *happened before* b
- ☐ D if, but not only if, $L(a) < L(b)$ then a *happened before* b

A:13 1p. What is the most that we know if we use vector clocks?

- ☐ A if a *happened before* b then $V(a) < V(b)$
- ☐ B if $V(a) = V(b)$ then a and b are unordered
- ☐ C $V(a) < V(b)$ if and only if a *happened before* b
- ☐ D if $V(a) < V(b)$ then a *happened before* b

A:14 1p. What is the definition of a unstable global state predicate?

- ☐ A the predicate could hold true in a state but then be false in future states
- ☐ B the predicate will never hold true in any consistent state reachable from the original state
- ☐ C if a system enters a state where the predicate holds true it will remain true in all future states
- ☐ D the predicate holds true in all consistent global states

A:15 1p. What do we know if we record a snapshot using the algorithm by Chandy and Lamport?

- ☐ A if the algorithm terminates then the execution will also terminate
- ☐ B there is a linearizations from the state described by the snapshot and a state that did occur in the execution
- ☐ C the execution passed through the state described by the snapshot
- ☐ D a non-stable predicate that is true for the snapshot has also been true during the execution

A:16 1p. Can we implement a reliable multicast using only basic multicast?

- ☐ A no, we need a synchronous communication network
- ☐ B yes, by re-sending each received message to all other nodes
- ☐ C no, we need a reliable failure detector
- ☐ D yes, but only if we have network supported multicast

A:17 1p. What would you call a multicast service that would never deliver a message m_2 before another message m_1 , if m_2 was sent as a response to m_1 ?

- ☐ A causal order
- ☐ B random order
- ☐ C total order
- ☐ D FIFO order

A:18 1p. What is the advantage of a ring based election algorithm compared to the bully algorithm?

- ☐ A better best case turnaround time
- ☐ B more reliable if nodes fail
- ☐ C nodes can easily change priority in-between elections
- ☐ D better worst case turnaround time

A:19 1p. What is a dirty-read during a transaction?

- ☐ A reading a value that has not been committed
- ☐ B reading ahead of a write operation
- ☐ C reading a value that has been written by the same transaction
- ☐ D reading an old value that will be over written

A:20 1p. What is two-phase commit?

- ☐ A a protocol that ensures atomicity in a distributed transaction
- ☐ B a protocol where other transactions are allowed to see temporary value before final commit
- ☐ C a protocol that uses a global lock that is taken by each server that wants to commit
- ☐ D a protocol where sub-transactions are allow to commit independently of each other

A:21 1p. What problem could we still have even if two transactions are serially equivalent?

- ☐ A inconsistent retrieval if both transactions write to the same object
- ☐ B no problems since all conflicting operations are performed in the same order
- ☐ C lost updates if both transactions read the same object
- ☐ D dirty read if one transaction reads a value that is not yet committed

A:22 1p. In a view-synchronous group membership protocol, a process that enters the group, and is included in the delivered view, will be guaranteed to be delivered:

- ☐ A only the messages delivered before it joined

Part C

C:1 4p. What we are often looking for is what caused something to happen in a distributed system. To our help we have Lamport clocks, vector clocks and real-time clocks. How are these clocks related to each other and how are they related to causality? Describe the pros and cons of using these clocks, and how good they are in helping us track down the events that caused something to happen.

C:2 4p. Describe two different ways how we can implement total order multicast using only b-multicast with no link-layer multicast support. Compare the two strategies.

C:3 4p. You should implement a distributed messaging system (no central server) for a known set of client nodes. You can see it as a chat service where all chat clients communicate with each other.

What is important is that when the messages are presented to the user, they are presented in a logical order i.e. we don't want to see a comment on a message if we have not seen the original message. Assume that all nodes are correct and describe an implementation.

Now assume that we have a dynamic set of clients. Clients can join the group and leave the group. Joining and leaving the group should be done in total order with respect to all messages. How would you have to change your implementation to support this?

Part B

B:1 2p. At local time 117 you receive a NTP reply with the following information: request sent at 82, received at 111, reply sent at 120. How should you adjust your local time?

B:2 2p. Briefly describe a situation that meets the requirements of total order multicast but not to FIFO order multicast.

B:3 2p. In a passive replicated system we have a primary that uses a view synchronous multicaster to deliver state changes to the back-end replicas. Why do we use a view synchronous communication layer?

B:4 2p. Show by example how a dead-lock situation is falsely detected, a phantom lock, if one only consider information gathered from the nodes in a system one by one.

- ☐ B all messages starting from the view where it is included in the group
- ☐ C all messages in the history of the group
- ☐ D all messages starting from the active view when it issues its request to join

A:23 1p. In an active replicated server, how do we know that the replicas are in a consistent state?

- ☐ A the primary replica sends update messages to all other
- ☐ B they reliably receive all requests in a total order
- ☐ C they do not change state and are thus by definition consistent
- ☐ D a coordinator uses two-phase commit for each request

A:24 1p. What is the purpose of hashing in a DHT?

- ☐ A to reduce the original identifiers to a smaller key space
- ☐ B to authenticate a request
- ☐ C to generate uniformly distributed keys
- ☐ D to more quickly find the responsible node

ID2201 Distributed Systems

2019-10-18

3412

The exam consists of three parts: A, B and C. Part A consists of twenty-four multiple choice questions where a correct answer is rewarded one point and a wrong answer zero points. Answer the questions by writing A, B, C or, D in the boxes below. It's the boxes below that count, not the things you write elsewhere. If you think the question is wrong or if you think that two answers are correct then make a small note of this but do write one answer in the boxes below.

Part B consists of eight short answer questions that could give you two points each. You should use the the space given in the exam to answer these questions. Make sure to write short answers that are to the point.

Part C consists of three essay question where I want you to reason about what problems we might have, different solutions to the problems and the pros and cons of your solutions. You should use the the space given in the exam to answer these questions i.e. the page the question is written on. The answers should be well formulated and well written. You should show that you can identify the critical problems and make use of the knowledge gained in the course.

Write your name on all pages and hand in the whole exam.

You are allowed to have a dictionary (book, not digital) but this must be examined by the staff at the beginning of the exam.

The requirements for grades are as follows (part A, B, C)

- grade E (16, -, -)
- grade D (20, 6, -)
- grade C (20, 10, -)
- grade B (20, 10, 4)
- grade A, (20, 10, 8)

A1: A2: A3: A4: A5: A6:

A7: A8: A9: A10: A11: A12:

A13: A14: A15: A16: A17: A18:

A19: A20: A21: A22: A23: A24:

B:5 2p. Describe briefly quorum based mutual exclusion and its advantages.

B:6 2p. In a DHT a large numbers of nodes are connected in a logical ring structure. In order to maintain the ring even if nodes dies, one will keep track of a number of successors, not only the closest one. The question is how many successors to keep track of, the more successors we keep track of the more resilient we are to node failures, but it also means more links to check. What do we need to consider when deciding how many successors to keep track of?

B:7 2p. What would the advantage be to use a gossip protocol when building a replicated service? What would we maybe have to sacrifice?

B:8 2p. Draw two lines, one that represents a *consistent cut* and one that represents an *inconsistent cut*.

