Please answer your questions in the order specified. Write legibly, and use proper sentences; an unreadable answer is a wrong answer. Answer questions concisely, but with sufficient detail and precision. Always explain your answers. Please leave space in the margin for correction marks. Use a separate piece of scratch paper for draft answers, private computations or remarks.

You are not allowed to use books, notes, tablets, PC’s and (smart)phones during the exam.

The total number of points that can be scored is 55, as specified alongside the questions. The final grade equals \( 1 + 9 \times \frac{\text{points}}{55} \), rounded to the nearest half grade (except for grades between 5 and 6 which are rounded to nearest full grade).

You can keep this exam. Good luck!

**Question 1 (15 points):**

a. Explain the difference between the leader election problem and the mutual exclusion problem \( (5 \text{ points}) \)

b. What happens if you would run a leader election protocol on a disconnected graph? \( (2 \text{ points}) \)

c. This is LeLann’s leader election protocol (code shown for node \( i \)) to elect a leader on a ring.

\[
\begin{align*}
    I_i &:= \emptyset \\
    \text{send right } C[i].id \\
    \text{while } C[i].id \notin I_i \\
    \text{do begin} \\
    \quad I_i = I_i \cup \{id\} \\
    \quad \text{send right } id \\
    \text{end} \\
    C[i].leader := (C[i].id = \min_j j \in I_i)
\end{align*}
\]
In this protocol it is assumed that message passing is FIFO. Explain what this means exactly, and explain why this is necessary. (8 points)

Note: the condition in the while loop was missing in the exam. This means the question could not be answered. Therefore, this part of the question was not graded. The total number of points for this question was reduced to 7.

Answer:

a. In leader election, one node needs to be elected as leader, and once the leader is elected, it remains leader forever. In mutual exclusion, there is a critical section access to which is restricted to one node at the time. It is assumed that nodes repeatedly require access to the critical section, and that once nodes reach the critical section, they stay in the critical section only for a limited time. You could say the node in the critical section is the current leader, but that this node relinquishes leadership for other nodes to take over. Leadership circulates forever, to those nodes that ask for it. In leader election, all nodes have a chance to become leader; there is no need to ask.

b. If you run a leader election protocol on a disconnected graph, each connected component of the graph would elect its own leader.

c. FIFO message passing means that messages on a single transmission link between two nodes are always received in the same order as they are sent.

Assume message passing is not FIFO, and that we run LeLann’s leader election algorithm. Recall that in LeLann’s algorithm nodes start by sending their own identifier to their clockwise neighbour, and that nodes j store identifiers they receive in a set Ij, and that they forward identifiers they receive in a clockwise direction, until they receive their own identifier back. If this identifier is the minimum among all identifiers in Ij, then node j is leader.

Now assume the following situation: there are three nodes, a, b and c ordered like that in clockwise fashion as a ring of three nodes. Node a has identifier 1, node b has identifier 0, and node c has identifier 2. All nodes start by sending their identifiers to their clockwise neighbour. Node b is fast and immediately forwards the identifier 1 to its clockwise neighbour c. If the link between b and c is not FIFO, node c may receive the value 1 before the value 0. Node c then sends value 1 to node a before sending value 0 to node a. This means node a will stop looking for a leader, decide it is leader (as among the set {1, 2} of values it received, 1 is the lowest value), and stop forwarding values! This means the value 0 is not forwarded, which means node 0 waits forever (and does not know whether it is leader or not).
Question 2 (10 points):

a. In bitcoin, transactions are signed using a digital signature based on public key cryptography, but blocks are not. Please explain how blocks are secured, and explain why transactions must be signed, while blocks need not, in order to keep Bitcoin secure. (5 points)

b. In bitcoin, the verification of the signature over transactions is not hardcoded, but implemented using scripts. Explain how Bitcoin scripts implement this. (5 points)

Answer:

a. If transactions were not signed, anybody could transfer value from one bitcoin address to another or could tamper with the contents of a transaction. Blocks need not be signed as their contents are protected against tampering through the use of hashes: transactions in a block are hashed using a Merkle tree, correctness of the block is verified by checking that the nonce, together with the root block data, hashes to a value smaller than the target, and the integrity of previous blocks is protected as their hash is included in the next block in the blockchain.

b. In Bitcoin this is done using the “Pay-to-PubKeyHash” (P2PKH) script. The script (associated with an output) expects a signature to be associated with the connected input along with the public key associated with the owner of the output. The script contains the hash of the public key. The script then checks that the public key provided matches to the hash stored in the script, and then verifies that the signature over the transaction verifies against this public key.

Question 3 (15 points): This is Dijkstra’s self-stabilizing mutual exclusion algorithm.

Node 0: if \( x[N] = x[0] \) then \( x[0] := x[N] + 1 \mod K \)

Node \( i \neq 0 \): if \( x[i - 1] \neq x[i] \) then \( x[i] := x[i - 1] \)

Prove that it always stabilises on a ring with \( N + 1 \) nodes in at most \( O(n^2) \) steps under the central daemon, provided that \( K \geq N \)

Answer: The central daemon gives every node a turn separately.

We have to specify the legitimate states and prove closure and convergence.

For the legitimate states define \( P(i,c) \) to be true if and only if for all \( j \) such that \( 0 \leq j \leq i \) we have \( x[j] = c + 1 \mod K \) and for all \( j \) such that \( i < j \leq N \) we have \( x[j] = c \). A state is legitimate if \( P(i,c) \) holds for some \( i \) and \( c \).
Closure is proven as follows. Assume $P(i, c)$ holds for some $i$ and $c$. If $i \neq N$ then in the state in which $P(i, c)$ holds, node $i + 1$ is privileged and the only one that can take a step. After that step, $x[i + 1] = x[i]$ while all other nodes still have the same state. We conclude that $P(i + 1, c)$ holds. If $i = N$ then for all $j$ we have $x[j] = c + 1 \mod K$ and in particular $x[0] = x[N] = c + 1 \mod K$. Hence node 0 is privileged and the only one that can take a step. After the step, $x[0] = c + 2 \mod K$ while $x[j] = c + 1 \mod K$ for all $j \neq 0$. We conclude that $P(0, c + 1 \mod K)$ holds. This proves closure.

Convergence is proven as follows. Initially colour all nodes white. Colour node 0 blue the first time it takes a step; after that it stays blue forever. Nodes colour blue when they copy a blue value from their counterclockwise neighbour (and then stay blue forever). Let $h$ be the number of times node 0 takes a step while node $N$ is still white. Then $h \leq N$: after the first step of 0, there are at most $N - 1$ white nodes that can provide $N$ with at most $N - 1$ new white states.

W.l.o.g. let $x[0]$ initially be $K - 1$; after the first step $x[0]$ becomes 0; so after $i$-th step, $x[0] = i - 1$. Starting at node 0, we observe that all blue nodes form a decreasing chain of values. Now let 0 be about to take the $h + 1$-th step (i.e. $N$ is blue). Then before that step $x[0] = h - 1$. As $h \leq N \leq K$ we see that $x[0]$ did not wrap around. Also, all nodes are blue at this point, and $x[N] = x[0]$. As now all nodes are blue, and have decreasing values, we must have $x[j] = h - 1$ for all nodes $j$. In other words $P(N, h - 2)$ holds.

Question 4 (15 points):

(a) The validity condition for the consensus protocol tolerating crash failures reads: “If all processors have the same input value $b$, then all correct processors must decide $b$”. Show that the consensus protocol tolerating crash failures as discussed in class does not satisfy the following, stronger, validity condition: “If all correct processors have the same input value $b$, then all correct processors must decide $b$” (5 points)

(b) Consider the consensus protocol tolerating at most $f < n/3$ byzantine failures discussed in class. This protocol needs to run $f + 1$ rounds distributing values in the tree before (recursively) computing the decision value. Show what goes wrong if it runs only $f$ rounds and then (recursively) computes the decision value. Use $n = 4$ and $f = 1$ to create an execution with a wrong decision in this case. (10 points)

Answer:

(a) Consider the following situation: suppose all correct processors hold the value 1, but there is one (faulty) processor that has the value 0. It sends the value 0 to all other processors in the first round, after which
it crashes. As a result all correct processors $q$ will see two values in their tree, hence $|V_{q}| = 2$ and so they will all decide on the default value $v_{def} = 0$. This violates the alternative validity condition that in this case demands their decision value is 1.

b. Let us write $v^i$ for $v^p_i$. Assume node 1 and 2 have input 0 and node 3 and 4 have input 1. Assume node 4 is faulty.

Before round 1, all $v^j_\sigma = \perp$ for all $j \in \{1, 2, 3, 4\}$ and all $\sigma$ of length 1.

For $\sigma = \epsilon$ we have $v^1_\epsilon = 0, v^2_\epsilon = 0, v^3_\epsilon = 1,$ and $v^4_\epsilon = 1$.

We have $f = 1$, so the (wrong) protocol has 1 round.

In round 1 each processor $p_i$ sends $v^i_j$ to all other processors; except $p_4$ that crashes and sends arbitrary values. Let us write $?_i$ for the value it sends to processor $p_i$. In other words, processor

- $p_1$ sends $m^1_1 = m^2_1 = m^3_1 = m^4_1 = 0,$
- $p_2$ sends $m^1_2 = m^2_2 = m^3_2 = m^4_2 = 0,$
- $p_3$ sends $m^1_3 = m^2_3 = m^3_3 = m^4_3 = 1,$ and
- $p_4$ sends $m^1_4 = ?_1, m^2_4 = ?_2, m^3_4 = ?_3, m^4_4 = ?_4$.

Then after round 1 processor

- $p_1$ has $v^1_1 = 0, v^1_2 = 0, v^1_3 = 1$ and $v^1_4 = ?_1$
- $p_2$ has $v^2_1 = 0, v^2_2 = 0, v^2_3 = 1$ and $v^2_4 = ?_2$.
- $p_3$ has $v^3_1 = 0, v^3_2 = 0, v^3_3 = 1$ and $v^3_4 = ?_3$.
- $p_4$ has $v^4_1 = 0, v^4_2 = 0, v^4_3 = 1$ and $v^4_4 = ?_4$.

Here the protocol stops distributing values, and starts the recursive decision phase.

At the leaves we have $d^1_\sigma = v^1_\sigma$. Hence

- $p_1$ has $d^1_1 = 0, d^1_2 = 0, d^1_3 = 1,$ and $d^1_4 = ?_1$.
- $p_2$ has $d^2_1 = 0, d^2_2 = 0, d^2_3 = 1,$ and $d^2_4 = ?_2$.
- $p_3$ has $d^3_1 = 0, d^3_2 = 0, d^3_3 = 1,$ and $d^3_4 = ?_3$.
- $p_4$ has $d^4_1 = 0, d^4_2 = 0, d^4_3 = 1,$ and $d^4_4 = ?_4$.

Now suppose that $?_1 = 0$ while $?_2 = 1$. Then using $d^\sigma_q = \text{Majority}(\{d^\sigma_{\tau, q} | q \notin \sigma\})$ we have $d^1_\epsilon = 0$ while $d^2_\epsilon = 1$ (assuming that the majority function returns 1 if there is no clear majority). This contradicts the agreement condition.