Exam Advanced Network Security - August 21, 2017

Jaap-Henk Hoepman

jhh@cs.ru.nl

August 22, 2017

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 60, as specified alongside the questions. The final grade equals \( 1 + \frac{9 \times \text{points}}{60} \), 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. What will happen to the following program: will it run forever, or will it (eventually) terminate. Assume that reading or writing a single variable is atomic. (6 points)

\[
\begin{align*}
    a &= 1 \\
    b &= 0 \\

    \text{thread while } a \neq b \\
    &\quad \text{do if } a < b \text{ then } a := (a + 1) \mod 3 \\

    \text{thread while } b \neq a \\
    &\quad \text{do if } b < a \text{ then } b := (b + 2) \mod 3
\end{align*}
\]

b. What are the three properties a mutual exclusion protocol must satisfy? (3 points)

*Version: Sun Aug 20 14:02:32 2017 +0200 / b1f33e8 / exam-2017-08-21.tex*
c. Find below Lamport’s bakery algorithm for mutual exclusion (code shown for node i).

\[
\begin{align*}
\text{num}[i] & := 0 \\
\text{choosing}[i] & := \text{false} \\
\text{while} & \text{ true} \\
\text{do & choosing}[i] & := \text{true} \\
\text{do & num}[i] & := 1 + \max_j \text{num}[j] \\
\text{choosing}[i] & := \text{false} \\
\text{for & } j \neq i \\
\text{do & while & choosing}[j] & \text{ do /* wait */} \\
\text{while & } (\text{num}[j] > 0) \land ((\text{num}[j], j) < \text{num}[i], i) & \text{ do /* wait */} \\
\text{/* critical section */} \\
\text{num}[i] & := 0
\end{align*}
\]

It is assumed that the lottery numbers \text{num}[i] are unbounded. What goes wrong if we need to store these variables in a fixed size register that can only store bounded values? (6 points)

**Answer:**

a. Initially \( b < a \). As a result, thread 1 will not change \( a \) until after thread 2 changed \( b \). Consider the first write to \( b \). This is the result of (the enabled) first iteration of thread 2 that sets \( b := (b + 2) \mod 3 \), so \( b = 2 \) right after this write. Hence thread 2 will not change \( b \) until after thread 1 changed \( a \). Consider the first write of \( a \). This is the result of (the now enabled) first iteration of thread 1 that sets \( a := (a + 1) \mod 3 \), so \( a = 2 \) right after this write, and hence \( a = b \) right after this write. In between the write to \( b \) by thread 2 and the current time (where thread 1 is back to the start of the while loop and now \( a = b \)), we have \( a \leq b \). As a result, thread 2 either saw \( a = b \) and terminated already, or it saw \( b \neq a \) and skipped the assignment and returned to the start of the while loop (and repeated this several times, until thread 1 assigned \( a \) the value 2 too). In other words, \( b \) couldn’t possibly have changed and we now have both threads at the start of the while loop with \( a = b \). We conclude the program always terminates.

b. A mutual exclusion protocol needs to satisfy the following three properties:

- **Mutual exclusion:** there is at most one processor in the critical section,
- **Progress:** if there is at least one processor enters, and the critical section is empty, then one of these processors will eventually get access to the critical section, and
• No starvation: if a processor enters, and if all processors that get access to the critical section release it, then it will eventually get access.

c. In Lamport’s bakery algorithm, an entering node reads all \( num[j] \) and sets its own value to the maximum plus 1. Consider the following sequence of steps: node \( i \) enters, sees node \( j \) already entered and sets \( num[i] \) to \( num[j] + 1 \). Node \( j \) enters the critical section, leaves, and wants to enter again. It now sees node \( i \) already entered and sets \( num[j] \) to \( num[i] + 1 \) (i.e. its previous value plus 2). This can be repeated at infinitum until at some point the register will overflow. If we decide to remedy this by computing modulo the size of the register, then we break the property that nodes that enter later will not overtake nodes already contending for the critical section, thus violating the no-starvation requirement.

**Question 2 (10 points):** This question concerns Bitcoin.

a. What is the so-called genesis block in Bitcoin? \( \text{(2 points)} \)

b. What does a miner do when it creates a new block and starts mining it, in a blockchain based on the proof of work principle? \( \text{(5 points)} \)

c. How does Bitcoin ensure that the average time between the addition of a new block to the blockchain is roughly 10 minutes? \( \text{(3 points)} \)

**Answer:**

a. The genesis block is the origin of the block chain, i.e. the very first block in it, and provides the initial supply of 50 BTC to the system.

b. Collect valid transactions into a block, compute the merkle hash tree, find the current head of the blockchain, and find a nonce such that the hash over the nonce, the merkle root and the pointer to the previous block is lower than a target value (i.e starts with a number of leading zero’s) that correspond with a certain level of difficulty.

c. The target value is adjusted every 2016 blocks to ensure this, by setting

\[
T_{new} = T_{prev} \cdot \frac{t_{actual}}{2016 \times 10 \text{minutes}}
\]

where \( t_{actual} \) is the actual time it took to mine the last 2016 blocks.

**Question 3 (20 points):** Consider an undirected graph \( G = (V, E) \) with \( N \) nodes in \( V \) and edges in \( E \). Assume the graph is connected (i.e. there is a path from every node \( u \in V \) to every other node \( v \in V \)). Let \( V = \{0, \ldots, N-1\} \), i.e. nodes have the natural numbers as identifier.
Let $C[u]$ denote the state of a node $u$. This state can only be read by all neighbouring nodes $u$ of $v$ (i.e. if $(u, v) \in E$). Let $N(u) = \{v \mid (u, v) \in E\}$ denote this set of neighbours.

Consider the following self-stabilizing leader-election protocol, running under the central daemon. We assume node $u$ has access to its identity $u$ which, as far as the node is concerned, is a fixed, unaltered constant. The state of each node has two fields, $C[u].value \in \mathbb{N}$ (i.e. its value is always a non-negative integer) and $C[u].leader \in \{\text{true}, \text{false}\}$. The code below shows the program for a node $u$. All nodes run the same program.

```
\text{minval} := \min\{C[v].value \mid v \in N(u)\}
\text{minval} := \min(u, \text{minval})
C[u].value := \text{minval}
C[u].leader := \text{true} \text{ if } u \text{ is } \text{minval} \text{ else false}
```

(a) Explain why the legitimate states of this protocol are given by

\[ \forall v \in V : P(v) \]

where

\[ P(v) \triangleq (C[v].value = 0 \land C[v].leader = (v = 0)) . \]

(b) Prove the protocol is self-stabilising.

(c) What is the worst-case stabilisation time of this protocol?

(d) Would the protocol still work if $V$ was an arbitrary subset of $\mathbb{N}$?

(e) Can you think of a much simpler self-stabilizing protocol for this problem, given this particular setting? And how fast does your solution stabilize?

**Answer:**

(a) In this state there is exactly one leader, namely node 0, who stays leader forever.

(b) We have to prove closure and convergence.

Closure follows because if before a step of a node $u$ it holds that $\forall v \in V : C[v].value = 0 \land C[v].leader = (v = 0)$, then $\min\{C[v].value \mid v \in N(u)\} = 0$, hence after the step still $P(u)$ holds.

Convergence follows from the following observation. Colour all nodes blue. Colour node $u = 0$ red once it takes a step for the first time.
Colour a node red once it takes a step while one of its neighbours is red. We show by induction that for all red nodes \( u \), \( P(u) \) holds.

By the protocol, if node 0 takes the first step, \( C[0].value = 0 \) (and \( C[0].leader = true \)). This is the base case. In the induction step, if a neighbour \( w \) of node \( u \neq 0 \) is red, \( w \) has \( C[w].value = 0 \). Hence

\[
\min\{C[v].value \mid v \in N(u)\} = 0 \quad \text{and} \quad C[u].value = 0 \quad \text{(and} \quad C[0].leader = false)\]

after the step. Because the scheduler is fair, every blue node gets infinitely many turns. As the graph is connected, the set of red nodes is extended by all their neighbours until all blue nodes eventually turn red.

c. Define a round to be a part of an execution in which each node takes at least one step. Consider an arbitrary graph \( G = (V,E) \) with \( V = \{0,\ldots,N-1\} \). Let \( d \) be the largest distance (taken along a shortest path) between node 0 in \( G \) to any node in \( G \). The proof of convergence above essentially shows that after each round, the 'diameter' of the set of red nodes increases by 1. Or rather, after round \( k \), all nodes with distance \( \leq k \) to node 0 are red. Then after \( d \) rounds all nodes are red. This is the upper bound. (Note we cannot say anything about the total number of steps, as blue nodes may take arbitrarily many steps before turning red.)

d. No. For example if \( 0 \notin V \) but the system is started in a state where some node has \( C[u].value = 0 \) then this is what all nodes will converge to. But as \( 0 \notin V \), no nodes stays leader in the steady state.

e. The program can be simplified tremendously. Remove the value part of the state, and set \( C[u].leader = (u = 0) \). This stabilizes after all nodes have taken at least one step, i.e. after just one round.

**Question 4 (15 points):** Consider the anonymous bulletin board protocol by Hoepman. The original protocol to send a message is given by

\[
\begin{align*}
\text{function } & \text{send}_{AB}(m) \\
& \text{id}^\prime \in_R \{0,\ldots,n-1\} \\
& \text{tag}^\prime \in_R T \\
& u := \{[m \| \text{id}^\prime \| \text{tag}^\prime]\}_AB \\
& \text{add}(\text{id}_{AB}, u, B(\text{tag}_{AB})) \\
& (\text{id}_{AB}, \text{tag}_{AB}) := (\text{id}^\prime, \text{tag}^\prime) \\
& K_{AB} := \text{KDF}(K_{AB})
\end{align*}
\]

Here \( B() \) is a hash function, senders (and receivers) communicate with the bulletin board through a mix network using secure channels, and the bulletin board \( B \) itself is implemented as follows:

\[
\text{add}(i, v, t): \text{Add } \langle v, t \rangle \text{ to the set at cell } i: B[i] := B[i] \cup \{(v, t)\}.
\]
get\((i, b)\): Let \(t = B(b)\). If \(\langle v, t \rangle \in B[i]\) for some value \(v\), return \(v\) and remove \(\langle v, t \rangle\) from \(B[i]\). Otherwise return \(\bot\), and leave \(B[i]\) unchanged.

Suppose that the bulletin board \(B\) itself changes the implementation of the bulletin board as follows.

add\((i, v, t)\): Store \(\langle v, t \rangle\) in cell \(i\): \(B[i] := \langle v, t \rangle\).

get\((i, b)\): Let \(t = B(b)\). If \(\langle v, t \rangle = B[i]\) for some value \(v\), return \(v\) and set \(B[i] = \bot\). Otherwise return \(\bot\), and leave \(B[i]\) unchanged.

a. What goes wrong if we use this different implementation of the bulletin board for the protocol? (3 points)

b. The original protocol is susceptible to an active attack that can confirm a suspicion that Alice and Bob are communicating. How does the attack work? Does the attack require cooperation from the bulletin board operator? (6 points)

c. Why is the second mix, between the bulletin board and the receiver of a message, necessary? (6 points)

Answer:

a. If a writers happens to choose a location to write a value to that is still occupied by a value that is not yet received, this previous value is overwritten.

b. If Alice and Bob are suspected to communicate with each other, the adversary can block all messages from other users (to the bulletin board) and ask the bulletin board whether (after the block) it sees participants that both write to and read from the same cell(s). If that is the case, the suspicion is confirmed. Collaboration from the bulletin board is required, because the communication between Alice, Bob and the bulletin board is encrypted so that the adversary cannot see which cells are accessed.

c. Omitting the second mix first and foremost leaks (to the bulletin board) the fact that Bob is using the service, leaks how many messages he is receiving, and when he prefers to receive them. Moreover, the second mix is important to protect Bob’s identity whenever Alice becomes compromised and the adversary tries to identify her contacts (with the help of the bulletin board service, who may be legally compelled to cooperate). (Either one of the answers scores full points.)