ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Autumn Term 2023 Prof. R. Wattenhofer

Distributed Systems
Exam

Date: 22-01-2024

Do not open or turn before the exam starts!
Read the following instructions!

This examination lasts for 90 minutes and comprises 90 points.

You may answer in German, English, or combine German and English.

Unless explicitly stated, you do **not** have to justify your answers. Writing down your thoughts (math, text, or annotated sketches), however, might help with the understanding of your approach. This may then result in points being awarded even if your answer is not correct. Please write legibly. Unreadable answers will not be graded.

Please write your name and student number on every additional sheet. Please write your name and student number in the following fields on this cover sheet.

Family Name	**First Name**	**Student Number**

Task	**Achieved Points**	**Maximum Points**
1 - Weak Byzantine Agreement		20
2 - Approximate Agreement		19
3 - Game Theory		18
4 - Internet Computer		19
5 - Blockchain		14
Total		**90**

1 Weak Byzantine Agreement (20 points)

In class, we studied Byzantine Agreement (BA) protocols, which satisfy All-Same Validity, Agreement and Termination. In this exercise, we study protocols achieving a weaker variant called Weak Byzantine Agreement (WBA). For simplicity, we restrict ourselves to nodes holding binary inputs 0 or 1. In a WBA protocol, nodes are additionally allowed to output $\perp$, with the meaning "I don't know!". A WBA protocol should satisfy All-Same Validity and Termination, while Agreement is relaxed to the following weaker requirement:

- (Weak Agreement) If two correct nodes output y and y', either $y = y'$, or $y = \perp$, or $y' = \perp$.

a) [4] Consider $n = 3$ nodes with inputs $\mathbf{x} = (x_1, x_2, x_3)$ and outputs $\mathbf{y} = (y_1, y_2, y_3)$. Nodes 1 and 2 are correct while node 3 is byzantine. Which of the following could be correct outcomes of running a WBA protocol:

○ True ○ False $\mathbf{x} = (1, 0, 0)$ and $\mathbf{y} = (\perp, \perp, \perp)$

○ True ○ False $\mathbf{x} = (0, 0, 1)$ and $\mathbf{y} = (\perp, \perp, \perp)$

○ True ○ False $\mathbf{x} = (1, 0, 1)$ and $\mathbf{y} = (\perp, 0, 1)$

○ True ○ False $\mathbf{x} = (1, 0, 1)$ and $\mathbf{y} = (0, 1, \perp)$

Alice proposes a simple protocol to solve WBA in the synchronous model. We will look for **tight** resilience thresholds. A protocol achieves some property up to some resilience threshold, such as $f < n/100$, if this property holds in every execution where $f < n/100$. This resilience threshold is tight if there is some execution where $f \leq n/100$ and the property fails.

Algorithm 1 Alice's Protocol

```
1: x := input value
2: Send x to all nodes. Let c_y be the number of received y's for y ∈ {0,1}.
3: if c_0 > c_1 then
4:     Output 0.
5: else if c_1 > c_0 then
6:     Output 1.
7: else
8:     Output ⊥.
9: end if
```

b) [6] Alice's protocol satisfies the following. Fill in the blanks with tight conditions; e.g., $f < n/100$. For each statement, give a short counterexample showing that your threshold is tight. **Full marks will be awarded only with a valid counterexample.**

All-Same Validity for: $f <$ _____

Weak-Agreement for: $f <$ _____

Bob proposes another protocol to solve WBA in the synchronous model:

Algorithm 2 Bob's Protocol

1: $x :=$ input value
2: Send x to all nodes. Let c_y be the number of received y's for $y \in \{0, 1\}$.
3: **if** $c_0 \geq n - f$ **then**
4: Output 0.
5: **else if** $c_1 \geq n - f$ **then**
6: Output 1.
7: **else**
8: Output $\perp$.
9: **end if**

c) [8] Bob's protocol satisfies the following. Fill in the blanks with tight conditions; e.g., $f < n/100$. For each statement, give a short counterexample showing that your threshold is tight. **Full marks will be awarded only with a valid counterexample.**

All-Same Validity for: $f <$ _____

Weak-Agreement for: $f <$ _____

d) [2] Name a protocol from the lecture achieving synchronous BA. Give a quantifiable advantage of using Bob's protocol over the previous (not just "it's simpler").

a)
- **(True)**
- **(False)** Fails All-Same Validity.
- **(True)**
- **(False)** Fails Weak Agreement.

For the following two parts, only the right bounds **(bold)** and counterexamples were required for full marks. Here we also briefly explain why the protocols work for the given bounds.

b)
- *All-Same Validity:* As long as $\mathbf{f < n/2}$ the property holds: the most efficient way the Byzantine nodes can break pre-agreement is if they all propose the opposite value. This succeeds whenever $f \geq n - f \iff f \geq n/2$.
- *Weak-Agreement:* Only $\mathbf{f = 0}$ works, since already for $f = 1$ the property can fail for $n = 3$: two correct nodes have inputs 0 and 1, respectively. The Byzantine node sends 0 to one node and 1 to the other, making them output conflicting answers. The same construction works for any odd $n > 1$.[1]

c)
- *All-Same Validity:* The asymmetry between 0 and 1 in the algorithm matters here. If all correct nodes hold input 0, then they all execute line 4, even for $f < n$. However, if they all hold input 1, then it might be the case that someone still gets $n - f$ zeros and outputs 0 as a result (because that "if" takes precedence). This can only (and indeed does) happen if Byzantine nodes propose at least $n - f$ zeros, requiring $f \geq n - f \iff f \geq n/2$. Hence, the answer is $\mathbf{f < n/2}$.
- *Weak-Agreement:* The protocol works for $\mathbf{f < n/3}$: if two correct parties got outputs $y = 0$ and $y' = 1$, this means that at least $n - f$ parties sent at least one 0 and at least $n - f$ parties sent at least one 1. Therefore, there are at least $n - 2f > f$ parties that sent both a zero and a one. However, this can't happen with only f byzantine parties. The bound is tight: for $n = 3f$ one can use the same $n = 3, f = 1$ example from above.

d) King's algorithm. Bob's protocol only needs one round of communication, while for full BA we know from the lecture that $f + 1 = \Omega(n)$ rounds are required.

[1] For even n, Weak-Agreement holds for $f = 1$ but fails again at $f = 2$ for similar reasons. This observation wasn't required for full marks.

2 Approximate Agreement (19 points)

In the Approximate Agreement chapter, you have learned about a special validity condition called **Correct-Range Validity**: the correct nodes' outputs are within the range of their inputs. In this exercise, we will achieve asynchronous Byzantine Agreement that satisfies this condition, assuming that the **the correct nodes hold inputs in** $\mathbb{Z}$ (that is, integers). We have n nodes in a fully connected **asynchronous** network. Out of the n nodes, $f < n/3$ are **byzantine**. Before moving forward, we recall the following properties.

a) [1] In an Approximate Agreement algorithm:

- (Correct-Range Validity) Correct nodes' outputs are within the range of their inputs.
- (ε-Agreement) For any given $\varepsilon > 0$, if two correct nodes output y and y', _____.
- (Termination) Every correct node outputs some value.

b) [1] In an algorithm achieving Byzantine Agreement with All-Same Validity:

- (All-Same Validity) If all correct nodes have the same input x, _____
- (Agreement) Correct nodes output the same value.
- (Termination) Every correct node outputs some value (with probability 1).

Let us take a look at a few additional properties of the correct nodes' outputs in Approximate Agreement. These properties will help us design our asynchronous Byzantine Agreement algorithm that achieves Correct-Range Validity. Hence, in the following, we assume that correct nodes run Approximate Agreement with $\varepsilon := 1/3$ and inputs $x \in \mathbb{Z}$, and they obtain outputs y.

c) [2] Assuming that correct nodes' inputs are in $\mathbb{Z}$, show that, for every correct node, both $\lfloor y \rfloor$ and $\lceil y \rceil$ satisfy Correct-Range Validity.

d) [2] Assume that there is some $a \in \mathbb{Z}$ such that, for every correct node, $a < y < a + 1$. Show that all correct nodes hold the same value $\lfloor y \rfloor$, and that all correct nodes hold the same value $\lceil y \rceil$.

e) [2] Every node defines z as the closest integer to its output y. That is, if $y - \lfloor y \rfloor \leq \lceil y \rceil - y$, $z := \lfloor y \rfloor$. Otherwise, $z := \lceil y \rceil$.

Assume that there is some $a \in \mathbb{Z}$ such that a correct node v has obtained $y \leq a$, while a correct node v' has obtained $y' \geq a$. Show all the correct nodes hold the same value z.

We may now focus on achieving asynchronous Byzantine Agreement with Correct-Range Validity. We assume an asynchronous algorithm achieving Approximate Agreement (denoted by `AA`), and an algorithm achieving Byzantine Agreement **on bits** (denoted by `BBA`).
We will be working with the template provided by Algorithm 3: the nodes first run `AA` on their input values $x \in \mathbb{Z}$, with $\varepsilon := 1/3$. This way, the correct nodes obtain values y that satisfy Correct-Range Validity and ε-Agreement, but not Agreement. To make the step from ε-Agreement to Agreement, we want to make use of `BBA`.

Algorithm 3 Byzantine Agreement with f-Median Validity

1: $x :=$ input value in $\mathbb{Z}$
2: Join `AA` (for $\varepsilon = 1/3$) with input x. Upon obtaining output y:
3: Define $z := \lfloor y \rfloor$ if $y - \lfloor y \rfloor \leq \lceil y \rceil - y$, and $z := \lceil y \rceil$ otherwise.
4: Bit $b :=$ _____
5: Join `BBA` with input b. Upon obtaining output bit b':
6: _____
7: Output _____

f) `[11]` Complete the missing lines in Algorithm 3 such that **asynchronous** Byzantine Agreement with Correct-Range Validity for inputs in $\mathbb{Z}$ is achieved. Alternatively, you may design your own algorithm achieving these guarantees. Either way, prove that the algorithm is correct.

a) (ε-Agreement) For any given $\varepsilon > 0$, if two correct nodes output y and y', $\underline{|y - y'| \leq \varepsilon}$.

b) (All-Same Validity) If all correct nodes have the same input x, $\underline{\text{no correct node outputs } y \neq x}$.

c) Approximate Agreement ensures that $y \in [x_{\min}, x_{\max}]$, where $x_{\min}$ and $x_{\max}$ denote the lowest resp. highest correct inputs. Since $x_{\min}, x_{\max} \in \mathbb{Z}$, $x_{\min} \leq \lfloor y \rfloor$ and $\lceil y \rceil \leq x_{\max}$.

d) All correct nodes obtain $\lfloor y \rfloor = a$ and $\lceil y \rceil = a + 1$.

e) ε-Agreement ensures that all correct nodes obtain values in $[a - 1/3, a + 1/3]$, and therefore all correct nodes obtain $z = a$.

f) We can complete the algorithm as follows:

Algorithm 4 Asynchronous Byzantine Agreement with Correct-Range Validity

1: $x :=$ input value
2: Join AA (for $\varepsilon = 1/3$) with input x. Upon obtaining output y:
3: Define $z := \lfloor y \rfloor$ if $y - \lfloor y \rfloor \leq y - \lceil y \rceil$, and $z := \lceil y \rceil$ otherwise.
4: $b :=$ $\underline{0 \text{ if } z \text{ is even and 1 otherwise}}$.
5: Join ByzantineAgreement with input b. Upon obtaining output b':
6: $\underline{\text{If } b \neq b', z := \lceil y \rceil \text{ if } y - \lfloor y \rfloor \leq y - \lceil y \rceil, \text{ and } z := \lfloor y \rfloor \text{ otherwise.}}$
7: Output $\underline{z}$.

Correct nodes' values y satisfy Correct-Range Validity and ε-Agreement for $\varepsilon := 1/3$.

Then, if there is some $a \in \mathbb{Z}$ such that $a < y < a + 1$ for all correct nodes, Task d) ensures that all correct nodes obtain the same rounding options $\lfloor y \rfloor$ and $\lceil y \rceil$. Regardless of b', the correct nodes output the same rounding option, which satisfies Correct-Range Validity according to Task c).

Otherwise, there is some $a \in \mathbb{Z}$ such that $y \leq a$ for some correct node v, while $y' \geq a$ for some correct node v'. Task e) ensures that correct nodes obtain the same z, which satisfies Correct-Range Validity according to Task c). Therefore they join BA with the same bit b, obtain $b' = b$ due to All-Same Validity, and output the initial rounding option z.

3 Game Theory (18 points)

In the Game Theory chapter, you saw that the (Optimistic) Price of Anarchy of Selfish Caching can be $\Theta(n)$. Figure 1 shows the example network from the lecture. The problem is that only a single node caches in the Nash Equilibria (NE) instead of two like in social optima.

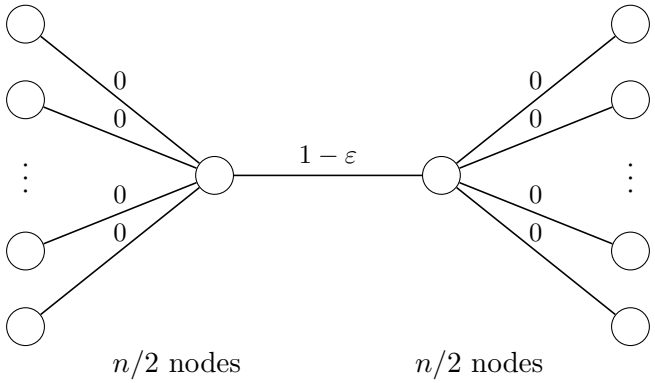

Figure 1: A network with a Price of Anarchy of $\theta(n)$ in the Selfish Caching model.

You have an idea to encourage more nodes to cache: Paying for file requests! In Selfish Caching with Payments (SCWP), if a node v requests a file from u, then it must pay u a fixed price p. So a node v can either:
(a) request the file from another node u, which costs $c_{v \leftarrow u} + p$, where $c_{v \leftarrow u}$ is the usual cost given by the distance between v and u (we assume unit demands) or
(b) cache the file locally, which costs $1 - n_v p$, where n_v is the number of non-caching nodes that request the file from v.
Note that non-caching nodes request the file from the nearest caching node. If multiple nearest caching nodes exist, the payment p is shared equally among them. If no nodes cache, the cost is infinite for all nodes. You want to check whether payments improve the Price of Anarchy.
Consider the setting of a fully connected graph on n nodes with all distances equal to zero.

a) [2] In the fully connected graph, what is the socially optimal total cost with $n = 8$ and $p = 0.2$?

b) [5] Find all pure Nash Equilibria (NE) and their total costs when $n = 8$ and $p = 0.2$.

c) [6] What are the pure Nash Equilibria (NE) for general n and p? You may assume $0 < p < 1$.

d) [5] What is the Price of Anarchy for Selfish Caching with Payments? You can choose any n, any p, and even any graph. Do payments help, i.e., is the PoA in the worst case better than $\Theta(n)$?

a) $SO = 1$

b) Let n_c be the number of nodes caching nodes. We consider three cases:

$n_c = 0$: All nodes have infinite cost. Therefore all nodes prefer to cache. Not an NE.

$n_c = 1$: Cost of caching $= 1 - 7(0.2) = -0.4$, cost if caching node switches ∞.
Cost of not caching $= 0.2$, cost if non-caching node switches $= 1 - \frac{6}{2}(0.2) = 0.4$.
Therefore, a single node caching is an NE.

$n_c = 2$: Cost of caching $= 1 - \frac{6}{2}(0.2) = 0.4$, cost if caching node switches 0.2.
A caching node will switch so this cannot be an NE. The same holds for $n_c > 2$.

Therefore, any single node caching is a pure NE. There are no other pure NEs. The total cost is 1 since all payments cancel out when summing the costs over all nodes.

c) Let n_c be the number of nodes caching. Cost of caching:

$$c^{\text{cache}} = 1 - \left(\frac{n - n_c}{n_c}\right) p$$

Cost of not caching:

$$c^{\text{not cache}} = p$$

Therefore cache if

$$1 - \left(\frac{n - n_c}{n_c}\right) p < p$$

$$n_c < np = 16 \times 0.2 = 3.2$$

Therefore, any set of three nodes caching is a pure NE. The total cost is 3 since all payments cancel out when summing the costs.

d) Given a fixed p, we can always choose n large enough such that $\lfloor np \rfloor = \mathcal{O}(n)$ nodes cache in a pure Nash Equilibrium, with total cost $\lfloor np \rfloor$. But the social optimum is $SO = 1$. So the Price of Anarchy is still linear in n.

4 Internet Computer (19 points)

a) [3] Imagine that a user sends message m to canister A and later message m' to canister B. Describe three circumstances (in one sentence each) in which m' can be replied before m.

b) [6] Consider the canisters as objects, and the computation done by them as operations that are initiated by messages. The output of the consensus algorithm of each subnet can be simplified as operations coming from a single (virtual) node. Operations are performed atomically inside canisters.

Which statements are true? For each statement explain your answer in at most two sentences.

○ True ○ False Execution is linearizable inside a subnet.

○ True ○ False Execution is sequentially consistent across subnets.

c) [6] In this question we compare the Internet Computer to Bitcoin and Ethereum. For each statement explain your answer in one sentence.

○ True ○ False Cyclic arbitrage is easier to accomplish on the Internet Computer than on Ethereum.

○ True ○ False Contrary to Bitcoin, the Internet Computer never experiences reorgs. In other words, confirmed blocks will never be reverted.

○ True ○ False Contrary to Bitcoin, the Internet Computer never experiences forks.

d) [4] In a subnet consisting of 12 nodes, an adversary controls 4 nodes. First, can the adversary negatively influence the execution of the subnet? Second, can the adversary negatively influence any other subnet? For each, in at most two sentences, illustrate a possible attack vector, or explain why no attack is possible.

a) Multiple factors can make it so m' is replied before m:

- Canisters A and B can be in different subnets, in which case consensus algorithms are independent, and might pick up one message before the other.
- They might be in the same subnet, but canister A can have a much longer queue.
- Messages can be re-ordered on the way to nodes.
- Nodes can reorder messages when creating blocks.

b)

- True. Every execution is restricted to a single node. Thus every execution is a sequential execution, and trivially linearizable.
- True. Just like regular messages, xnet messages also go through consensus first. Every execution $E \mid o$ is linearizable, since canisters are accessed through queues, and thus have linearization points. Thus, by the composability of linearizability E is linearizable. Since linearizability implies sequential consistency, the statement is true. Alternatively, this can also be argued through happened-before consistency, which is equivalent to sequential consistency.

c)

- False. Both on Ethereum and on the Internet Computer computation is being kicked off by a user message. On Ethereum, all resulting computation is done atomically, while the Internet Computer has no such guarantees. Thus, on Ethereum, the exact price of each trade in the cycle can be computed at the very start of a transaction, while the Internet Computer gives no such guarantees.
- True. The IC does not use eventual consistency, i.e., blocks that are finalized will remain finalized deterministically.
- False. The IC consensus algorithm still builds a DAG that can have forks.

d) The adversary can finalize two blocks inside the compromised subnet. Furthermore, it can create conflicting or even just duplicated cross-net messages that might negatively influence other subnets: Any application that relies on the consistency of a canister in the compromised subnet might also be compromised.

5 Blockchain (14 points)

a) [6] You just finished building a practical quantum computer in your garage. When testing it, you realize that it can create forgeries for Bitcoin's ECDSA signature scheme in just a few minutes of computing time.

Which of the following attacks are possible with your newly gained power? For each statement explain your answer in at most two sentences.

○ Possible
○ Impossible

Double spend by forking the blockchain.

○ Possible
○ Impossible

After observing a transaction t in the memory pool, create a conflicting transaction sending the funds of t to you.

○ Possible
○ Impossible

Drain a wallet that is reusing a public key across transactions.

b) [8] For each statement regarding Ethereum, mark whether it is true or false.

○ True ○ False An Ethereum transaction, if not immediately included in a block, remains valid and may be included in a future block.

○ True ○ False Smart contracts can be configured to periodically and automatically initiate a transaction.

○ True ○ False Smart contracts can only be programmed in one specialized programming language (Solidity).

○ True ○ False A user can spam the network with transactions that run out of gas for free, as transactions are aborted once too much gas is consumed.

This page is intentionally left blank.

a) Impossible, because the resistance against forks of the proof-of-work system only relies on the security properties of the underlying cryptographic hash functions.

Possible, because the transaction contains the public key and you can use that to create another transaction with a forged signature.

Possible, because any of the existing transactions contains the public key and thus allows you to craft a transaction draining the wallet.

b) True. Before being included in a block a transaction is not in any way bound to a specific block. This is also why the account nonce is needed.

False. Every transaction is initiated by an externally-owned address, smart contracts only act upon incoming calls.

False. Smart contracts on Ethereum are deployed as EVM bytecode, which can be the product of compiling any of multiple higher-level languages.

False. The gas fee must be paid for any transaction even if it is aborted, reverted, especially if it out of gas.