ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Autumn Term 2018 Prof. R. Wattenhofer

Distributed Systems
Exam

Monday, 21st January 2019, 09:00 - 10:30

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

The maximum points for all tasks and subtasks are indicated in brackets. **Justify all your answers** except if the task explicitly relieves you of it. Mark drawings precisely. Answers which we **cannot read** are not awarded any points!

At the beginning, fill in your name and student number in the corresponding fields below. Label each extra sheet with your name and student number, too.

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

Task	**Achieved Points**	**Maximum Points**
1 - Multiple Choice		8
2 - Binary Value Broadcast		28
3 - Localization		22
4 - Blockchain		14
5 - Selfish Caching with a Byzantine		18
Total		**90**

1 Multiple Choice (8 points)

For each statement, indicate whether it is true or false. Each correct answer gives 1 point. Wrong (or double) answers and unanswered statements give 0 points. **In this task, you *cannot* justify your answers.**

Statement	**true**	**false**
Paxos would still be correct if clients require an answer from 70% of the servers instead of a simple majority.	☐	☐
Assume that the initial configuration C_0 is bivalent and there are $f < n/2$ crash failures. The Ben-Or algorithm will visit precisely one (not more, not less) critical configuration during its execution.	☐	☐
Let $f < n/3$. Then, any byzantine agreement algorithm that satisfies Median Validity also satisfies All-Same validity.	☐	☐
A shared coin always yields the same output for all nodes.	☐	☐
At any given time during the PBFT protocol, only a single correct node considers itself to be primary.	☐	☐
In the PBFT protocol, the requests contained in the new-view-certificate were already executed by every correct node.	☐	☐
When using consistent hashing, with high probability no machine must store too many movies.	☐	☐
Happened-before consistency is composable.	☐	☐

Statement	true	false
Paxos would still be correct if clients require an answer from 70% of the servers instead of a simple majority. *Reason: Yes, any fraction of servers above 1/2 guarantees that two different commands cannot be accepted simultaneously.*	✓	
Assume that the initial configuration C_0 is bivalent and there are $f < n/2$ crash failures. The Ben-Or algorithm will visit precisely one (not more, not less) critical configuration during its execution. *Reason: For $f < n/2$ crash failures, the Ben-Or algorithm solves consensus (Theorem 11.20). By Lemma 11.12, we are guaranteed to reach a critical configuration C within finite time. Let C be the first critical configuration that the Ben-Or algorithm passes. All children of C in the configuration tree are univalent and thus the entire subtree rooted in C consists of univalent configuration (except for C). Hence, none of the reachable configurations are bivalent and thus none of them are critical.*	✓	
Let $f < n/3$. Then, any byzantine agreement algorithm that satisfies Median Validity also satisfies All-Same validity. *Reason: Assume that all $n-f$ correct nodes have the same input value. Then, byzantine values can either be the same, larger or smaller. In any case, the median will be one of the correct values, and the median condition implies that it should be taken.*	✓	
A shared coin always yields the same output for all nodes. *Reason: A shared coin is a random variable that is 0 for all nodes with constant probability, and 1 for all nodes with constant probability. No requirement for these two probabilities to add up to 1.*		✓
At any given time during the PBFT protocol, only a single correct node considers itself to be primary. *Reason: If the node with id $v \mod n$ is in view v and the node with id $(v+1) \mod n$ is in view $v+1$, then both consider themselves to be primary.*		✓
In the PBFT protocol, the requests contained in the new-view-certificate were already executed by every correct node. *Reason: It could contain prepared-certificate of the requests that did not have time to commit before the view change occurred.*		✓
When using consistent hashing, with high probability no machine must store too many movies. *Reason: The Chernoff bound in the script gives a bound with high probability.*	✓	
Happened-before consistency is composable. *Reason: Happened-before consistency = sequential consistency, and sequential consistency is not composable (Lemma 13.17).*		✓

2 Binary Value Broadcast (28 points)

Consider a distributed system with n nodes, $f < n/3$ of which can be byzantine. Assume that each node has a binary input value and the nodes communicate via a synchronous network. Algorithm 1 presents a possible broadcast abstraction which the nodes can use to communicate their inputs. This abstraction is similar to reliable broadcast. However, instead of forwarding messages, the nodes forward single values:

Algorithm 1 Synchronous Binary Value Broadcast (code for node u)

$x_u \in \{0, 1\}$ ◁ input bit

Round 1:

1: Broadcast own message $\texttt{msg}(u, x_u)$

Round 2:

2: **if** at least $f + 1$ messages $\texttt{msg}(v, x_v)$ contain the same bit x_v **then**
3: Broadcast $\texttt{echo}(u, x_v)$
4: **end if**
5: **if** at least $2f + 1$ messages $\texttt{echo}(w, x_w)$ contain the same bit x_w **then**
6: Accept(x_w)
7: **end if**

For $f < n/3$ byzantine nodes, show that ...

a) [5] ... if a correct node has not broadcast a value, this value will not be accepted by any correct node.

b) [6] ... each correct node will accept at least one value.

c) [5] ... if one correct node has accepted *exactly* one bit x, no other correct node will accept *only* the opposite bit $1 - x$.

Algorithm 2 presents a randomized algorithm for synchronous byzantine agreement, similar to Ben-Or. This algorithm makes use of a shared coin subroutine as a blackbox, which is assumed to be computationally efficient in this case. Unfortunately, some lines of the algorithm have been erased.

Algorithm 2 Synchronous Byzantine Agreement (Ben-Or)

$x_u \in \{0, 1\}$ ◁ input bit
$r = 1$ round

1: decided = false
2: **repeat**
3: Binary Value Broadcast $\texttt{msg}(u, x_u, r)$ (according to Algorithm 1)
4: Let X_u be the set of accepted bits
5: Broadcast X_u
6: Let c be the outcome of a shared coin subroutine
7: [Erased lines]

8: $r = r + 1$
9: **until** decided (Erased part)
10: decision = x_u

d) [12] Suggest a replacement for the missing lines in Algorithm 2, such that the resulting algorithm satisfies termination, agreement and all-same validity. Note that no additional communication rounds are required.

[more space for d)]

a) Since byzantine nodes can only echo the same value f times, no correct node will be able to echo a value that is not from a correct node. There will be at most f echoes for a non-correct value and therefore no correct node will accept such a value.

b) Since $n > 3f$, there are at least $2f + 1$ correct nodes in the system. One of the values 0/1 will be held by a majority of the nodes, i.e. by at least $f + 1$ correct nodes. Since all correct nodes will receive the correct $f + 1$ values, they will echo them. Since there are at least $2f + 1$ correct nodes, all nodes will accept the value.

c) Let 0 be the only bit that the node accepted. If less than $f+1$ correct nodes echoed 0, then 1 was the value that was held by the majority of the correct nodes, which is a contradiction to 0 being the only accepted value. Since at least $f + 1$ nodes had 0 as the input value, all correct nodes will echo 0 and thus every correct node will accept it.

d) The full algorithm is presented below. The nodes decide by entering the loop in Line 7. If there are $n - f$ sets which contain exactly one value, each correct node will receive at least $n - 2f > f + 1$ correct sets with exactly one value. This means that no correct node will find the opposite bit inside $n - f$ sets. Moreover, since at least one correct node has accepted only this value, it will be inside all correct sets according to task b). This gives agreement. The algorithm will terminate, because with constant probability the shared coin will have the same value as the value which some correct nodes have adapted. If all nodes have the same input, all $n-f$ correct sets will contain only this input and the nodes will decide in Line 7 of the algorithm. Since Binary Value Broadcast can tolerate $f < n/3$ byzantine nodes and all previous analysis only requires f and n to satisfy $n - 2f > f + 1$, the algorithm can tolerate $f < n/3$ byzantine nodes.

Algorithm 3 Synchronous Byzantine Agreement

$x_u \in \{0, 1\}$ ◁ input bit
$r = 1$ round

1: decided = false
2: **repeat**
3: Binary Value Broadcast $\texttt{msg}(x_u, r)$ (according to Algorithm 1)
4: Let X_u be the set of accepted bits
5: Broadcast X_u
6: Let c be the outcome of a shared coin subroutine
7: **if** $n - f$ sets X_v each only contain the value x **then**
8: $x_u = x$, decided = true
9: **else if** $n - f$ sets X_v contain the bit x, but no $n - f$ sets contain the opposite bit of x **then**
10: $x_u = x$
11: **else**
12: $x_u = c$
13: **end if**
14: $r = r + 1$
15: **until** decided (Erased part)
16: decision = x_u

3 Localization (22 points)

In this task, we look at a system using audio speaker signals for localization similar to GPS. All speakers and a receiver are located on a line and the receiver wants to position itself on this line. Assume that the signal propagation speed is $c = \frac{1000}{3}$ m/s $= \frac{1}{3}$ m/ms.

a) [8] Given the speaker signals below, which were all received at handset time t=0, write down the system of equations for computing the receiver's position h and its time offset θ to the system time.

Label	Sent from position	Sent at system time
A	1 m	-11 ms
B	3 m	-11 ms
C	5 m	1 ms

b) [8] Consider the two potential receiver states (h = 4 m, θ = - 1 ms) and (h = 4 m, θ = 1 ms). Compute the sum of the squared residuals (in ms^2).

Which of the states of more likely, in the least squares sense?

	$h = 4$
$\theta = -1$ ms	
$\theta = 1$ ms	

c) [6] Assuming that one signal is spoofed, which one must it be? You can use the knowledge that the true receiver location is $h = 6$.

a)

$$|1\text{ m} - h|/c = 11\text{ ms} - \theta$$
$$|3\text{ m} - h|/c = 11\text{ ms} - \theta$$
$$|5\text{ m} - h|/c = -1\text{ ms} - \theta$$

$+\theta$ works as well, but will flip the result in b).

b) E.g. $h = 4, \theta = 1 : r_A = |1\text{ m} - 4\text{ m}|/(1/3)\text{ m/ms} - 11\text{ ms} + 1\text{ ms} = -1\text{ ms}$.
$R = r_A^2 + r_B^2 + r_C^2 + r_D^2$

	$h = 4$
$\theta = 1$ ms	$1 + 49 + 25 = 75$
$\theta = -1$ ms	$9 + 81 + 9 = 99$

The more likely receiver state among the options is therefore ($h = 4$ m, $\theta = 1$ ms).

c) Generally, the signal which leads to a perfect solution (residual error = 0) if it is excluded from the system of equations. Since the receiver position is known, it must the signal whose computed time offset does not agree with any of the other two. By simplifying the system of equations, we get:

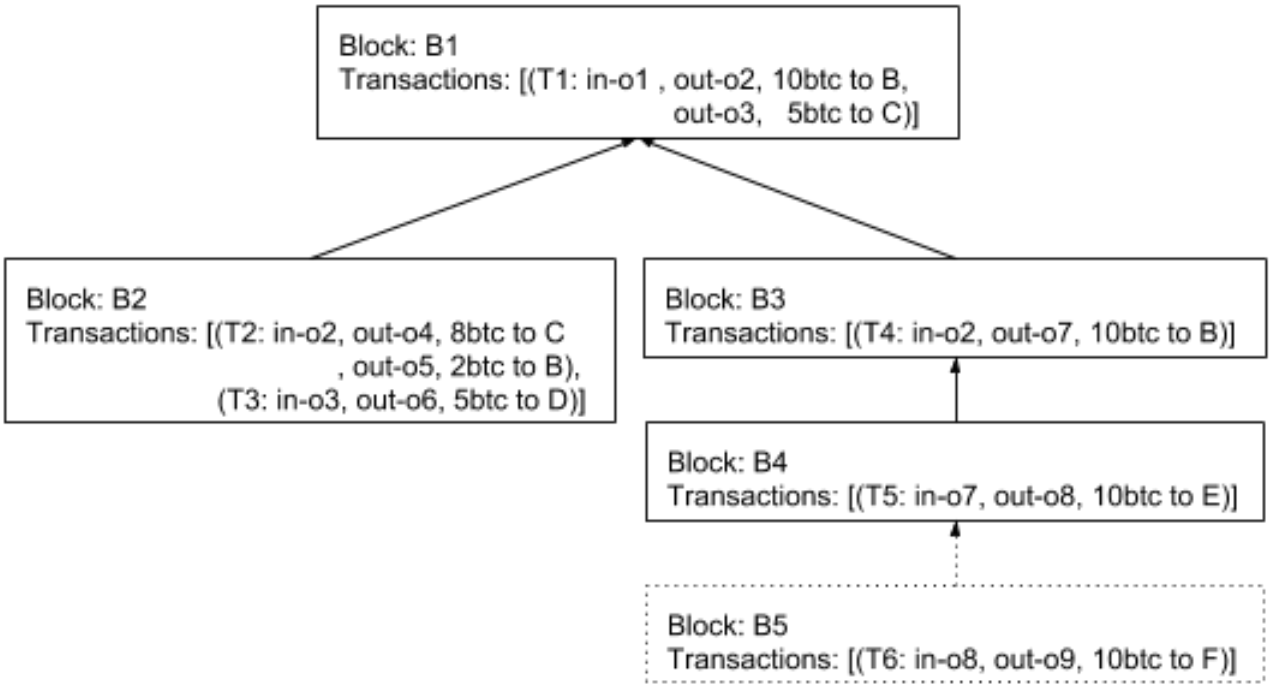
$$4\text{ ms} + \theta = 0$$
$$-2\text{ ms} + \theta = 0$$
$$4\text{ ms} + \theta = 0$$

With $\theta = -4$ ms, the residuals for both A and C are zero. So, **the signal from B** must be the spoofed one. (The receiver state is ($h = 6$ m, $\theta = -4$ ms), which means that the receiver time is behind the true time.)

4 Blockchain (14 points)

In this task, we look at a hypothetical segment of the Bitcoin blockchain. A transaction is represented as (T1: in-o1, out-o2, 10btc to B), which means:

- Transaction input is o1
- Transaction output is o2, whose value is 10btc, and can be spent by B



a) [5] A Bitcoin node starts up from scratch, and syncs itself till (and including validating) block B4. Which of the numbered outputs from the set {out-o1, ..., out-o9} make it in and/or out of the UTXO set during this syncing process? Check the correct cells in the following table – no justification needed.

output	**in**	**out**
out-o1		
out-o2		
out-o3		
out-o4		
out-o5		
out-o6		
out-o7		
out-o8		
out-o9		

b) [3] Merchant D got paid 5btc in transaction T3 by C, which ended up in block B2. Should D deliver the goods now?

c) [3] Do any participants look like they are cheating? If so, which?

d) [3] Give a technical explanation for why Satoshi Nakamoto might have included the headline "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks" in the coinbase transaction of the genesis block of Bitcoin?

a)

output	in	out
out-o1		x
out-o2	x	x
out-o3	x	
out-o4		
out-o5		
out-o6		
out-o7	x	x
out-o8	x	
out-o9		

b) No, D should not deliver anything to B yet. T3 is currently mined in B2, which is an orphan block. D could either wait for the T3 to get confirmed in a block that has 6 confirmations. But given that T3 has missed 2 blocks already, D should ask C to bump up the fees and rebroadcast a new transaction.

c) B could possibly be malicious, as it is attempting a double spending transaction spending o2 in both T2 and T4.

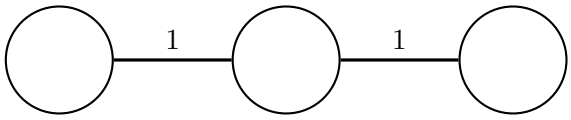
d) Including a time-specific headline proves that the genesis block was mined after 20090103. Without such an external time marker, the genesis block could have been created earlier, allowing the existence of an entire alternate blockchain with precomputed proof-of-work. Such a blockchain could be used by Satoshi to make a double-spending attack in tandem with the main public blockchain which uses the same (bad) genesis block.

5 Selfish Caching with a Byzantine (18 points)

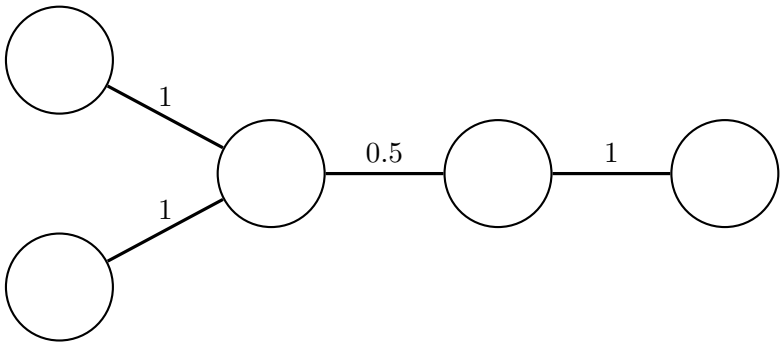
Recall the selfish caching game on graphs, where the cost of node v is 1 if v caches the file, and $c_{v,u} \cdot d_v$ otherwise, where d_v is the demand of v and $c_{v,u}$ is the shortest path length to a caching node. In the lecture, we have seen that selfish behavior can lead to a higher total cost than the Social Optimum.

We now investigate an even worse scenario: Assume that there is exactly one byzantine node who deliberately tries to increase the total cost. However, its only power is to not have to play rationally, i.e. it can decide to cache or not cache freely. All other nodes still act selfishly.

a) [8] Assume that the byzantine node does *not* cache the file. Find demands in the following graph and place the byzantine node such that the cost of the NE with the byzantine node is higher than in any NE with only selfish nodes.

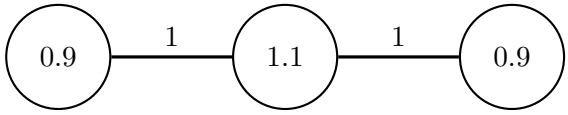


b) [10] Assume that the byzantine node *does* cache the file. Find demands in the following graph and place the byzantine node such that the cost of the NE with the byzantine node is higher than in any NE with only selfish nodes.



a) With the labels chosen as below, only the central node would be cashing in the NE, thus resulting in a weight of $2 \cdot 0.9 + 1 = 2.8$. However, if the central node is byzantine and decides not to cash, the total cost results in $2 \cdot 1 + 1.1 = 3.1$.

(Various other correct solutions exist. Note: if the byzantine NE is worse than at least one selfish NE in the graph, but not worse than *every* selfish NE, then at most 5 points can be given.)



b) With the labels chosen as below, the rightmost node would always cash in the NE. Therefore, its neighbor with demand 0.9 will never cash, and that nodes neighbor will have to cash again, since its distance from any cashing node would be more than 1. This results in a weight of $1 + 1 + 3 \cdot 0.9 = 4.7$.

On the other hand, let us choose the second node from the right as the byzantine. If this node cashes, its left neighbor does not have to cash anymore. The two nodes on the left are this way forced to cash. This results in a weight of $4 \cdot 1 + 0.9 = 4.9$, which is larger than the cost of the NE.

(Again, there are numerous other correct solutions. If the byzantine NE is worse than at least one selfish NE in the graph, but not worse than *every* selfish NE, then at most 5 points can be given.)