

Introduction to Consensus in Weakly Byzantine Asynchronous Environments

Aleksander Kampa¹

¹*Aragon ZK Research*

Abstract

We provide a summary of results for binary consensus in *weakly Byzantine* asynchronous environments, in which up to t out of n nodes are faulty, and up to $t' \leq t$ nodes are Byzantine. Binary consensus is shown to be possible when $n < 2t + t'$. Conditions for one-step symmetric binary consensus are presented. As a new contribution, the concept of asymmetric binary one-step consensus is introduced, and it is shown that in this setting a weakly one-step protocol can be achieved when $n > 2t + 2t'$, while a strongly one-step protocol requires $n > 2t + 3t'$. Using asymmetric consensus in a leader-based protocol has the potential to allow for one-step consensus in favourable conditions, at the cost of only a moderate weakening of fault tolerance assumptions.

Keywords

Binary Consensus, Asynchronous Systems, Weakly Byzantine Environments, One-step Consensus, Asymmetric Consensus, Weakly One-step Protocol, Strongly One-step Protocol

1. Introduction

In many real-world settings, it can be acceptable to weaken fault tolerance assumptions in order to speed up consensus in favourable circumstances. In this paper, we provide an overview of some not-yet-published results of consensus in weakly Byzantine asynchronous environments, in which only a subset of faulty nodes is Byzantine, the rest being crash-prone. A particular focus is on one-step protocols, as well as concepts of "strongly one-step" and "weakly one-step" which were introduced by Song and Van Renesse in [1].

An original contribution of this paper is an analysis of asymmetric one-step protocols, which can be used to increase the likelihood of fast consensus in some settings.

2. Background

In the context of distributed systems, such as a blockchain, the speed at which consensus is reached is important. Lowering the number of communication rounds not only decreases block time, but also reduces communication and computation overhead. Significant work has therefore been done to speed up consensus by reducing the number of communication steps in favourable circumstances.

The concept of two-step consensus in an asynchronous crash-prone environment appears to have been introduced by Schiper in [2], improving upon the Chandra-Toueg algorithm [3] which requires three steps at a minimum. Other two-step algorithms followed, e.g. [4].

The concept of one-step consensus was introduced by Brasileiro et al. [5], again in a crash failure model. One-step consensus in an asynchronous Byzantine environment was achieved by


5th Distributed Ledger Technology Workshop (DLT 2023), May 25–26, 2023, Bologna, Italy

✉ alex.kampa@aragon.org (A. Kampa)

🌐 <https://research.aragon.org/> (A. Kampa)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

the Bosco algorithm proposed by Yee Jiun Song and Robbert van Renesse [1], which is discussed below. Other efforts in that direction have followed, e.g. [6].

The downside on one-step protocols is that they require a weakening of security assumptions. While standard protocols tolerate up to $\frac{1}{3}$ -rd of Byzantine nodes¹, the Bosco algorithm tolerates only $\frac{1}{5}$ -th of Byzantine nodes in its "weak", and only $\frac{1}{7}$ -th in its "strong" version. Such a weakening of Byzantine fault tolerance can be acceptable for consortium blockchains or for systems where failures are expected to be rare.

One way to mitigate the weakening of security assumptions is to use asymmetric consensus. In an asymmetric binary protocol, one value is favored over the other, meaning that it is preferred for the consensus to converge on that value. The favored value is typically referred to as the "privileged" value. Asymmetric protocols are used in distributed systems where it is more beneficial for the participants to agree on a specific value than on the other one. This could be due to various reasons, such as efficiency, security, or other application-specific considerations. For example, in some cases, the preferred value might represent a "safe" or "conservative" choice that minimizes the risk of undesired outcomes.

One obvious application of asymmetric consensus are leader-based protocols, which are used in many public blockchains. In such protocols, every consensus round has a well-defined "leader" responsible for proposing a value or a set of values that the participating nodes can either accept or reject. Often, the set of nodes responsible for reaching consensus on the proposal is also well known. In the vast majority of cases, the leader's proposal is valid and ends up being accepted. This is therefore a setting in which the "ACCEPT" value is clearly favoured over the "REJECT" value.

3. Preliminaries

3.1. Model

We are in the classic context in which n nodes, each with an initial value of 0 or 1, aim to reach consensus using a predetermined randomised protocol. This is called binary consensus. The nodes that follow the protocol are said to be correct. At most t nodes are faulty. Among the faulty nodes, at most t' are Byzantine, the remaining $t - t'$ being crash-prone. When $t' < t$, this is called a weakly Byzantine environment.

The nodes communicate over a reliable asynchronous network. An adversary has control over the faulty nodes, can see the content of messages, can decide the order in which messages are delivered and can delay message delivery. However, the adversary cannot impersonate nodes² and must eventually deliver all messages.

We will require the usual conditions from our protocol: *agreement* (all correct nodes decide the same value), *validity* (a correct node can only decide a value initially proposed by a correct node) and *termination* (all correct nodes decide with probability 1). Under these assumptions, *unanimity* (if all correct nodes propose the same value, then this must be the consensus value) is implied by *validity*.

3.2. Definitions and notations

We denote by **AWB** an asynchronous weakly Byzantine environment.

A **symmetrical consensus protocol** treats both input values in a similar way, without giving preference to one or the other. An **asymmetric protocol** favours one of the two values.

¹This is for protocols where a decision, once taken, is irreversible. In the context of blockchains, this is called "block finality" or simply "finality". Proof-of-work tolerates up to 50% of Byzantine nodes, at the cost of block finality.

²This is usually achieved by cryptographically signing messages

A **one-step protocol** allows a decision in one communication step under certain conditions. A protocol is **strongly one-step** if it allows a one-step decision when all correct nodes have the same initial value, which means that a node must be able to decide upon receiving $n - t - t'$ identical values. It is **weakly one-step** if it allows a one-step decision when all correct nodes have the same initial value and there are no faulty nodes in the system³. In this case a node must be able to decide upon receiving $n - t$ identical values.

Zugzwang At any stage of a consensus protocol, a node may have to make a decision after receiving only $n - t$ messages. A protocol may call for a node to wait for some period of time after having received $n - t$ messages, but it is never guaranteed that it will receive more messages.

Trust Threshold Sometimes we want to be certain that at least one of the messages received has been sent by a correct node. This is the case when at least $t' + 1$ messages with the same value are received.

Overlap When two correct nodes receive m_1 and m_2 messages for the same value, the (minimal) overlap $\max\{0, m_1 + m_2 - n\}$ represents the minimum number of messages that both nodes have received from the same processes. Often, we will want that overlap to be at least equal to $t' + 1$.

Ranges of possible values (unconstrained) Denote by (m_0, m_1) the number of messages received by a node for values 0 and 1 respectively. The range of the number of values that another node may receive is as given in Table 1. It will always be possible to obtain the entire range in the first communication step of a consensus protocol. In later steps, the protocol may restrict outcomes in some way; for example, if a node received a majority of values 0 in the first step, the protocol may prevent it to send 1 in the second step.

	# of 0 values	# of 1 values
minimum	$\max\{0, m_0 - t - t'\}$	$\max\{0, m_1 - t - t'\}$
maximum	$n + t' - m_1$	$n + t' - m_0$

Table 1
Range of values received by another node

To see how these values are obtained, consider a node receiving m_0 messages for 0. It is possible that only $m_0 - t'$ of these messages are from correct nodes, t' messages 0 are from Byzantine nodes and all remaining $n - m_0$ correct nodes are sending 1. In that case, the adversary will be able to schedule message delivery so that another node receives $n - m_0 + t'$ messages with value 1 (all Byzantine nodes are now sending 1), and only $(n - t) - (n - m_0 + t') = m_0 - t - t'$ values 0, and then delays delivery of the remaining t messages with value 0.

Threshold values We now define some threshold values that are useful in certain protocols.

Trusted Origin threshold τ_{tor} . What number of values received guarantees that every other node will receive at least $t' + 1$ such values? The requirement $\tau_{tor} - t - t' > t'$ implies:

$$\tau_{tor} > t + 2t'$$

Majority threshold τ_{maj} . What number of values v received guarantees that every other node will receive a strict majority of such values? This is equivalent to saying that no other node can receive more than $\frac{n-t}{2}$ values $\neg v$, i.e. $n + t' - \tau_{maj} < \frac{n-t}{2}$. This implies:

$$\tau_{maj} > \frac{n + t + 2t'}{2}$$

³While we allow for at most t faulty nodes, a scenario in which there are in fact no faulty nodes is of course possible

Commitment threshold τ_{com} . What number of value v received guarantees that no other node will receive that same number of messages for $\neg v$? We require $n + t' - \tau_{com} < \tau_{com}$ which implies:

$$\tau_{com} > \frac{n + t'}{2}$$

In any given round, a node:

- can receive at most n messages⁴
- is guaranteed to receive only $n - t$ messages
- is guaranteed to receive only $n - t - t'$ messages from correct nodes

Table 2 shows the conditions under which a given threshold can be reached if we want a node to reach this threshold after receiving n , $n - t$ or $n - t - t'$ messages respectively. For example, we must have $n > 3t + 2t'$ to be able to reach τ_{maj} with $n - t$ (identical) messages received.

Table 2
Threshold "reachability" conditions

	$\tau = n$	$\tau = n - t$	$\tau = n - t - t'$
$\tau_{tor} > t + 2t'$	T1 $n > t + 2t'$	T2 $n > 2t + 2t'$	T3 $n > 2t + 3t'$
$\tau_{maj} > n+t+2t'/2$	M1 $n > t + 2t'$	M2 $n > 3t + 2t'$	M3 $n > 3t + 4t'$
$\tau_{com} > n+t'/2$	C1 $n > t'$	C2 $n > 2t + t'$	C3 $n > 2t + 3t'$

Correct value condition. Another rather natural condition is the following: after receiving $n - t$ messages, can a node always know at least one value that has been sent by a honest node? Given that a node is only guaranteed to receive $\lceil n-t/2 \rceil$ values for any one value, this constraint can be expressed as $n-t/2 > t'$ which implies the following condition:

$$\mathbf{CV} \quad n > t + 2t'$$

4. Binary consensus in AWB environments

A question one can immediately ask is: what are the conditions for reaching binary consensus in an asynchronous weakly Byzantine environment. With less Byzantine nodes, it seems clear that we should be able to improve on the usual $n < 3t$ condition. Indeed, in [7] it is shown that binary consensus can be obtained with the following condition (although it remains an open question if this is an optimal bound):

$$n > 2t + t'$$

The above-mentioned paper provides an analysis of a protocol by Tyler Crain [8]. The key observation is that Crain's protocol relies on two essential devices:

- the *correct value condition* **CV**, which requires $n > t + 2t'$
- the *commitment threshold* **C2** for $n - t$ messages, requiring $n > 2t + t'$

The conclusion follows immediately, as $t \geq t'$. Note that the "correct value condition" is called "trust threshold" in [7].

⁴counting the message sent by the node itself

5. Symmetric one-step consensus: W-Bosco

One-step binary consensus in an asynchronous Byzantine environment is achieved by the Bosco algorithm proposed by Yee Jiun Song and Robbert van Renesse [1]. In that paper, which uses the standard Byzantine setting in which $t = t'$ (all faulty nodes can be Byzantine) it is proven that:

- A weakly one-step protocol requires $n > 5t$ nodes
- A strongly one-step protocol requires $n > 7t$ nodes

Note that the Bosco protocol meets these conditions and is therefore optimal. Also note that there is a significant cost involved: instead of tolerating the usual $\frac{1}{3}$ -rd of Byzantine nodes, we can tolerate only $\frac{1}{5}$ -th or $\frac{1}{7}$ -th of Byzantine nodes. Later, [9] showed how the results achieved by Song and van Renese can be modified in a weakly Byzantine environment and introduced W-Bosco, a slightly modified version of Bosco.

5.1. Properties of symmetrical one-step binary consensus protocols

In [9] it is noted that Bosco is implicitly a symmetric consensus protocol, meaning that the possible consensus values 0 and 1 are treated equivalently, without giving a preference to either. The following facts are established for such protocols in the weakly Byzantine setting:

- The decision threshold m in a symmetrical one-step binary agreement protocol must satisfy $m > \frac{n+t+2t'}{2}$ which corresponds to the τ_{maj} condition
- A weakly one-step protocol requires $n > 3t + 2t'$ nodes **M2**
- A strongly one-step protocol requires $n > 3t + 4t'$ nodes **M3**

The proof of these statements relies mainly on the following *Lemma*: In the context of a symmetric binary one-step consensus protocol, assume that one correct node reaches the decision threshold for a value v . Then every node, after receiving $n - t$ messages, will receive a strict majority of messages for v .

To see why this holds, note that in the first step of a consensus protocol, all nodes must be prepared to set their value based on only $n - t$ messages received. Unless one value is favoured over the other, the only possible criterion is the majority of values received.

5.2. W-Bosco

The W-BOSCO algorithm is then introduced, being a slight modification of Bosco. The original Bosco algorithm is shown in Table 3, while W-BOSCO is shown in Table 4. The only difference is the decision threshold in line 3.

Both algorithms assume the existence of an *Underlying-Consensus* primitive which is called when one-step consensus cannot be achieved. The main idea is that if the condition in line 3 is true, then it is possible to decide for value v immediately because all other correct nodes will either also decide in line 3, or submit value v to Underlying-Consensus. There is an implicit assumption that deciding for v in line 3 means that the node in question will signal this as its decision value to Underlying-Consensus. As a result, all honest nodes will propose the same value v to Underlying-Consensus, which must therefore decide v because of unanimity.

The following results are obtained:

- W-Bosco is weakly one-step if $n > 3t + 2t'$
- W-Bosco is strongly one-step if $n > 3t + 4t'$.

This means that W-Bosco is optimal in the AWB setting.

Table 3
BOSCO

	Input: v_p
1	Broadcast (VOTE , v_p) to all nodes
2	Wait until $n - t$ VOTE messages have been received
3	if $> \frac{n+3t}{2}$ VOTE messages contain the same value v then
4	DECIDE(v)
5	if $> \frac{n-t}{2}$ VOTE messages contain the same value v ,
6	and there is only one such value v then
7	$v_p \leftarrow v$
8	Underlying-Consensus (v_p)

Table 4
W-BOSCO

	Input: v_p
1	Broadcast (VOTE , v_p) to all nodes
2	Wait until $n - t$ VOTE messages have been received
3	if $> \frac{n+t+2t'}{2}$ VOTE messages contain the same value v then
4	DECIDE(v)
5	if $> \frac{n-t}{2}$ VOTE messages contain the same value v , then
6	$v_p \leftarrow v$
7	Underlying-Consensus (v_p)

5.3. Comparison of Bosco and W-Bosco

Let's take a concrete example: a network of 50 nodes. To be weakly one-step, Bosco requires $n > 5t$ and therefore tolerates 9 Byzantine nodes. This means that when all nodes have the same initial value, consensus in one step will be possible if there are in fact no faulty nodes, so that the power of the adversary is limited to delaying message delivery.

The equivalent formula for W-Bosco is $n > 3t + 2t'$, and all the possible weakly one-step configurations of (t, t') when $n = 50$ are shown in Table 5. For example, W-Bosco can tolerate up to 13 faulty nodes, if we assume that only 5 of these can be Byzantine.

Table 5
Options for weakly one-step consensus with $n=50$

	t	t'	$3t+2t'$
Bosco	9	9	45
W-Bosco	10	9	48
	11	8	49
	12	6	48
	13	5	49
	14	3	48
	15	2	49
	16	0	48

We again consider a network of 50 nodes. To be strongly one-step, Bosco requires $n > 7t$ and therefore tolerates 7 Byzantine nodes. This means even if 7 nodes are actually Byzantine and controlled by an adversary intent on delaying or preventing consensus, consensus will still be reached in one step if all correct nodes have the same initial value.

The equivalent formula for W-Bosco is $n > 3t + 4t'$, so W-Bosco also tolerates 7 Byzantine nodes when $t = t'$. But other configurations are possible, as shown in Table 6. For example, W-Bosco can tolerate a total of 11 faulty nodes, if we assume that only at most 4 of these can be Byzantine.

Table 6
Options for strongly one-step consensus with $n=50$

	t	t'	$3t+4t'$
Bosco	7	7	49
W-Bosco	7	7	49
	8	6	48
	9	5	47
	11	4	49
	12	3	48
	13	2	47
	15	1	49
	16	0	48

W-Bosco does not improve on Bosco, but it introduces an element of choice: if less Byzantine failures can be tolerated, crash resilience can be increased.

6. Asymmetric one-step binary consensus

In some cases, it is known in advance that one of the binary values is more likely to occur as the consensus value. This could be the case, for example, when the nodes must decide if the proposal of a “round leader” is valid or not. In most cases, such a proposal is expected to be accepted. One-step consensus can then be achieved more easily for that value, as we can use τ_{tor} as the decision threshold. The protocol W-Bosco-0, which favours the value 0, is presented in Table 7

Table 7
W-BOSCO-0

	Input: v_p
1	Broadcast (VOTE , v_p) to all nodes
2	Wait until $n - t$ VOTE messages have been received
3	if $> t + 2t'$ VOTE messages contain the same value 0 then
4	DECIDE(0)
5	if $> t'$ VOTE messages contain the value 0, then
6	$v_p \leftarrow 0$
7	else
8	$v_p \leftarrow 1$
9	Underlying-Consensus (v_p)

We find that:

- W-Bosco-0 is weakly one-step for the value 0 if $n > 2t + 2t'$ **T2**
- W-Bosco-0 is strongly one-step for the value 0 if $n > 2t + 3t'$ **T3**

This is an immediate consequence of the reachability thresholds for τ_{tor} , which were provided in Table 2.

To see how this can speed up consensus, consider a network of $n = 50$ nodes, with $t = 13$ and $t' = 5$, which allows for weakly one-step consensus in the symmetric setting as per Table 5. Achieving a one-step consensus requires $\lceil n+t+2t'/2 \rceil = 37$ identical messages which is feasible but only under close to optimal conditions (very few faulty or Byzantine nodes and/or more than $n - t = 37$ messages received in total). However, if we know that the value 0 is privileged, we can use W-Bosco-0 in the very first round. The one-step threshold of $t + 2t' + 1 = 24$ will then be much easier to reach.

Another way to look at this is that with an asymmetric first round, we do not need to weaken security assumption as much and can tolerate more faulty nodes. Again taking a network of 50 nodes as an example, we have seen that W-Bosco is weakly one-step while tolerating up to 12 faulty nodes, of which half can be Byzantine, i.e. $(t, t') = (12, 6)$. With an asymmetric first round, the same can be achieved while tolerating 16 faulty nodes, again with up to half of them being Byzantine, i.e. with $(t, t') = (16, 8)$. This is a significant improvement.

Table 8

W-Bosco-0: Options for weakly one-step consensus on value 0 with $n=50$

	t	t'	$2t+2t'$
Bosco-0	12	12	45
W-Bosco-0	13	11	48
	14	10	48
	15	9	48
	16	8	48
	17	7	48
	18	6	48
	19	5	48
	20	4	48
	21	3	48
	22	2	48
	23	1	48
	24	0	48

Table 9

W-Bosco-0: Options for strongly one-step consensus on value 0 with $n=50$

	t	t'	$2t+3t'$
Bosco-0	9	9	45
W-Bosco-0	9	10	48
	11	9	49
	12	8	48
	14	7	49
	15	6	48
	17	5	49
	18	4	48
	20	3	49
	21	2	48
	23	1	49
	24	0	48

7. Waiting for more messages to speed up consensus

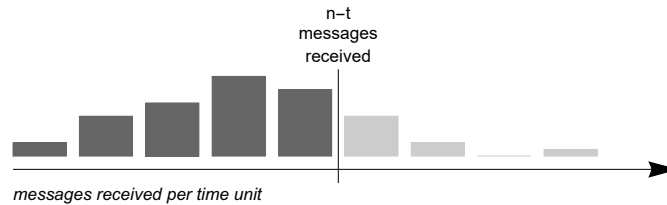
7.1. The benefits of waiting

We have seen that consensus can be sped up by relaxing assumptions about the number of faulty nodes. We have also seen that, with prior knowledge about the type of decision, we may be able to apply asymmetry to favour one of the values. This will increase the probability of achieving fast consensus on that value.

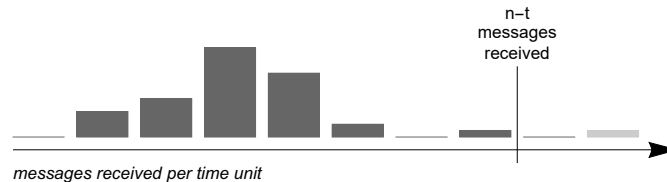
A third approach is to wait for more than the canonical $n - t$ messages. Of course, this method should only be used if there is an actual possibility that additional messages will expedite a decision - this will not always be the case.

Taking the example of the previous section, suppose that one-step consensus requires 37 identical messages, and that of 37 received messages, 35 have the value 1. In that case, it may make sense to wait for additional messages, as the receipt of only two more messages with value 1 will avoid additional communication rounds.

Another consideration is the way messages for a given round have been received. The following graphic shows a favourable scenario:



It is likely that additional messages will be received without any long wait. On the other extreme, we could have a situation where $n - t$ is reached after a significant decline of messages received per time unit:



This would suggest that some nodes have crashed or that parts of the network are extremely slow. In this situation, waiting for more messages before proceeding may not make sense.

7.2. Example 1 (13 nodes)

Consider a network of 13 nodes. A standard protocol will tolerate $t = t' = 4$ Byzantine nodes. In each round, a node will then be expected to decide after receiving $n - t = 9$ messages.

Now assume that we can use the asymmetric protocol W-Bosco-0 as the first step of our protocol. Setting $(t, t') = (4, 3)$ we find that a node will be able to decide in one step if it receives $t + 2t' + 1 = 11$ identical messages. This can be achieved if there are no Byzantine nodes and no particular network delays, even if one or two nodes have crashed. This is not unrealistic, and it shows that one-step consensus can actually be achieved while weakening fault-resistance only slightly.

7.3. Example 2 (50 nodes)

With 50 nodes, a standard protocol would tolerate $t = t' = 16$ Byzantine nodes, and a node would be expected to decide after receiving $n - t = 34$ messages.

Again assuming that W-Bosco-0 can be used, we find that by setting $(t, t') = (16, 12)$, a node will be able to decide in one step if it receives $t + 2t' + 1 = 41$ identical messages. This means waiting for 7 additional messages, which seems quite realistic in most scenarios. The weakening in Byzantine fault-tolerance is relatively small, so that this could be an acceptable setting for many systems.

8. Future work

It would be useful to develop a model for defining the optimal time to wait after having received the usual $n - t$ messages. This will have to depend on the time distribution of the messages received. It could also be interesting to analyse cases where it will be optimal to move on even before having received $n - t$ messages.

9. Conclusion

We have provided an introduction to consensus in weakly Byzantine environments and shown under which conditions one-step consensus is possible. We have further shown how asymmetric on-step consensus can improve the likelihood of one-step consensus. Finally, we commented on the usefulness of sometimes waiting for more than the canonical $n - t$ messages in a round of a consensus protocol. Because both crashes and Byzantine behaviour are actually relatively rare in many real-world scenarios, we believe that research into speeding up consensus has significant value.

References

- [1] Y. J. Song, R. van Renesse, Bosco: One-step byzantine asynchronous consensus, in: DISC, volume 5218 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 438–450.
- [2] A. Schiper, Early consensus in an asynchronous system with a weak failure detector, *Distributed Computing* 10 (1997) 149–157.
- [3] T. D. Chandra, S. Toueg, Unreliable failure detectors for reliable distributed systems, *J. ACM* 43 (1996) 225–267.
- [4] I. Keidar, S. Rajsbaum, On the cost of fault-tolerant consensus when there are no faults - A tutorial, in: LADC, volume 2847 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 366–368.
- [5] F. V. Brasileiro, F. Greve, A. Mostéfaoui, M. Raynal, Consensus in one communication step, in: PaCT, volume 2127 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 42–50.
- [6] N. Banu, T. Izumi, K. Wada, Doubly-expedited one-step byzantine consensus, in: DSN, IEEE Computer Society, 2010, pp. 373–382.
- [7] A. Kampa, An introduction to asynchronous binary byzantine consensus, <https://research.sikoba.com/>, 2021.
- [8] T. Crain, A simple and efficient asynchronous randomized binary byzantine consensus algorithm, *CoRR* abs/2002.04393 (2020). URL: <https://arxiv.org/abs/2002.04393>. arXiv:2002.04393.
- [9] A. Kampa, One-step consensus in weakly byzantine environments, <https://research.sikoba.com/>, 2019.