Speeding up One-Step Consensus in Weakly Byzantine Environments

Aleksander Kampa

ak@sikoba.com

December 2021

Abstract

By relaxing fault tolerance thresholds, both for the number of crash-prone and Byzantine nodes, and simultaneously waiting for more than the canonical $n - t$ messages from other nodes before proceeding, one-step consensus can be obtained more often under favourable circumstances.

1 Introduction

Our context is an asynchronous distributed system composed of nodes which seek to reach consensus despite being subject to possible failures. The system consists of n nodes, of which t can be faulty. Among the faulty nodes, t' can by Byzantine, the others being crash-prone. We call this a weakly Byzantine environment.

It has been shown in [Kam21] that binary consensus in weakly Byzantine environments is possible when $n > 2t + t'$. This result is a simple application of a consensus protocol proposed by Tyler Crain in [Cra20]. When $t = t'$, the inequality reduces to the well known lower bound $n > 3t$ for the (fully) Byzantine setting.

It has been further been shown in [Kam19] how the requirements for one-step binary consensus in the (fully) Byzantine setting change in the weakly Byzantine environment. This extends results obtained by Song and Renesse in [SvR08] by introducing an element of choice: if less Byzantine failures can be tolerated, crash resilience can be increased.

In this paper, we examine how one-step consensus can be accelerated when tolerating not only less Byzantine, but also less crash-prone nodes, and additionally by waiting for more than the usual $n - t$ messages before proceeding.

2 Measurements

This is an introduction to binary measurements in weakly Byzantine environments.

2.1 The model

We have a set of nodes, each of whom is expected to hold a value 0 or 1. We consider a measurement device $\mathcal{M}$ that "pings" all the nodes, waits for responses and returns a measurement as a pair of values $\{v_0, v_1\}$ corresponding to the number of 0 and 1 messages received.

The measuring device knows only the values n, t and t', while the configuration being measured may actually have less than the maximum allowed number of crash-prone and Byzantine nodes. Because there are up to t faulty nodes, $\mathcal{M}$ is only guaranteed to receive $n - t$ responses.

We assume that the behaviour of faulty nodes can change at every measurement. A crash-prone node may respond to a first measurement and fail to respond to another one. A Byzantine node may respond 0 to one measurement, 1 to another and not respond at all to a third.

2.2 Measurement Modes

We define the following measurement modes of $\mathcal{M}$.

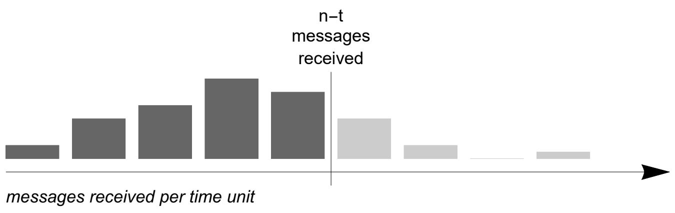
- *Sample Measurement (with time factor Δ)*
  $\mathcal{M}$ waits for a time period Δ before returning its measurement. The number of responses received ranges between 0 and n.
- *Standard Measurement*
  $\mathcal{M}$ waits for exactly $n - t$ responses.
- *Extended Measurement (with time factor τ)*
  After receiving $n - t$ responses, $\mathcal{M}$ collects responses for an additional time period τ. The number of messages received ranges between $n - t$ and n.
- *Complete Measurement*
  This is a hypothetical measurement in which $\mathcal{M}$ receives all messages sent by the nodes in repose to its ping. The number of messages received will be at least equal to the number of correct nodes.

If we add the assumption that there is a maximum delay D for message delivery, then the Complete Measurement would correspond to a Sample Measurement with time factor $2D$.

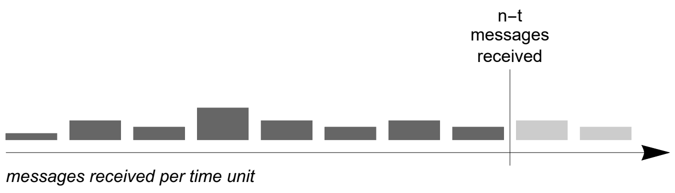
2.3 Choosing the time factor in Extended Measurement

In Extended Measurement, the time factor τ could be a function of the distribution of messages received over time. Let's consider some scenarios.

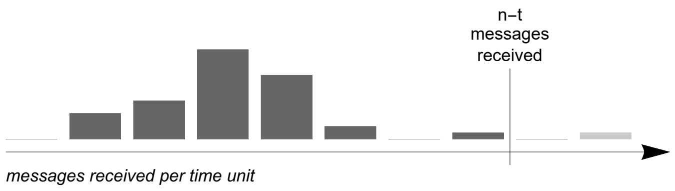
Under normal circumstances, the graph of messages received per time unit should have a clear peak and should already be dropping off as we reach $n - t$ messages, assuming (as is usually the case) that t that is less than $n/3$:



In this favourable scenario, we can extrapolate this distribution and set τ accordingly. In another scenario, we could have a slow "trickling in" of messages, indicating a slow network:



In that case, it may make sense to set a larger τ, as we would expect that messages will continue to be received, albeit slowly. Finally, we may 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, setting a large τ may just be a waste of time. Whatever the value of τ, $\mathcal{M}$ could also have a rule to stop the measurement if no messages have been received for a certain period of time.

2.4 Threshold Measurements

In some cases, we may be interested in reaching some fixed threshold T. Once that threshold is reached, i.e. at least T messages with the same value have been received, the measurement can stop. Thresholds can be used both in Standard and Extended measurements to speed up the measuring process.

2.5 Interpreting Measurements

There are a number of questions that can be asked, for example: under which circumstances can measurements actually yield useful information? And what kind of information? Below, we will deal with one such question, specifically: what kind of measurement result ensures that any other standard measurement will have a majority of a certain value.

3 Speeding up consensus by waiting

3.1 Condition for one-step consensus

In order to decide in one step, a node must receive a sufficient number of messages with the same value v to ensure that no other node can receive $> \frac{n-t}{2}$ messages with an alternative value. All other nodes will therefore, after having received at least $n-t$ messages, have a majority of messages with value v. Then all correct nodes will set their proposed value in the next step to v, and this will eventually result in a consensus value of v by unanimity.

To see how this works, let's suppose that node i has received $m_i > t'$ messages with the value v. The situation is as follows:

- node i has received at least $m_i - t'$ messages with value v from correct nodes
- node i has received at most t' messages with value v from Byzantine nodes
- node i knows nothing about the remaining $n - m_i$ messages, which could all be from correct nodes, and all have a value other than v

If we now take the viewpoint of another node j, we find that the maximum number of values other than v that this node could receive is the sum of:

- t' messages from Byzantine nodes, which sent value v to node i, but are all sending some other value to node j
- $n - m_i$ messages from correct nodes not received by node i

To ensure that, after receiving $n - t$ messages, node j can never have a majority of messages other than v, we must have:

$$n - m_i + t' < \frac{n-t}{2} \tag{1}$$

This implies:

$$m_i > \frac{n+t+2t'}{2} \tag{2}$$

3.2 One-step consensus in the standard approach

In most protocols, a node waits only for $n-t$ messages before proceeding, as it is never guaranteed that more messages will be received. In that case, it will only ever be possible to reach one-step consensus if:

$$n - t > \frac{n+t+2t'}{2} \tag{3}$$

This provides a lower bound for n as a function of t an t':

$$n > 3t + 2t' \tag{4}$$

3.3 Restating the rules for one-step consensus

As $t' \leq t$, we can write:

$$t' = t - \delta, \quad 0 \leq \delta \leq t \tag{5}$$

An then, because $n > 2t + t'$, we can also write:

$$n = 2t + t' + 1 + e = 3t + 1 + e - \delta, \quad 0 \leq e \tag{6}$$

The one-step threshold inequality $m > \frac{n+t+2t'}{2}$ can then be restated as as:

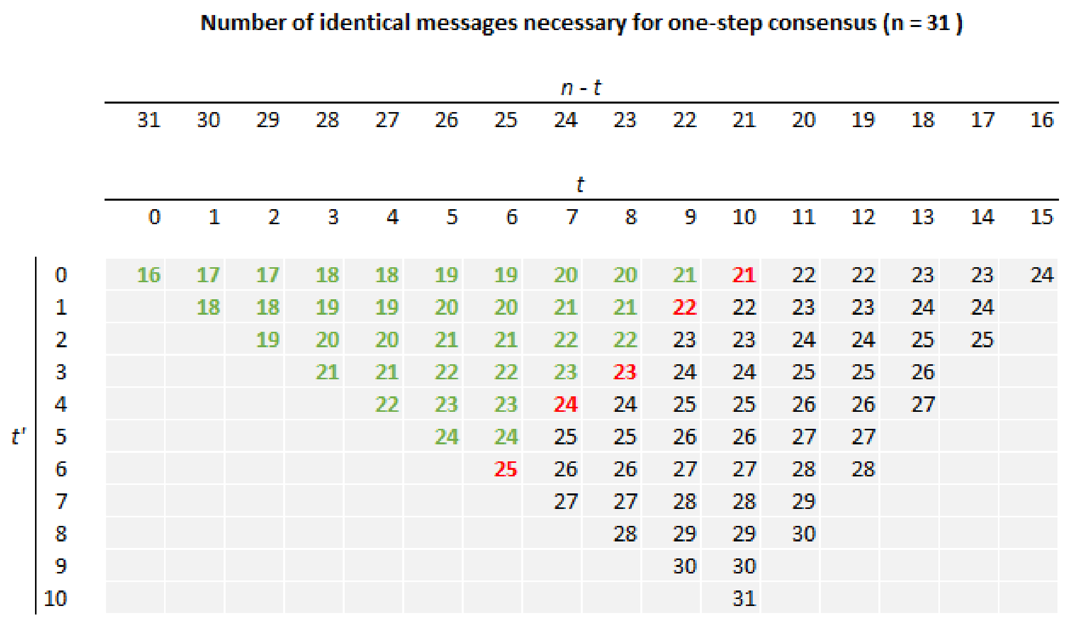
$$m > n - \frac{1 + e + \delta}{2} \tag{7}$$

So we can see that as e and δ grow, the threshold for reaching one step consensus gets easier to reach. Given a fixed upper limit of faulty nodes t, increasing e means adding more correct nodes to the network, while increasing δ means reducing the number of faulty nodes that can be Byzantine.

3.4 Speeding up one-step consensus

There are essentially two ways to speed up one-step consensus. The first is to weaken fault tolerance assumptions by reducing the number of potentially faulty nodes. The second is to wait for more than $n - t$ messages. The two methods can of course be combined.

To see how this is done, let's consider a specific example: a network of $n = 31$ nodes. The following table shows how many identical messages are needed to achieve one-step consensus, depending on possible values of t and t'. The numbers in red show configurations where one-step becomes possible, in ideal conditions, when exactly $n - t$ identical values are received. When less than $n - t$ identical values are required to achieve one-step consensus, the numbers are shown in green.



Number of identical messages necessary for one-step consensus (n = 31)

	n - t	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	16	17	17	18	18	19	19	20	20	21	21	22	22	23	23	24
	1		18	18	19	19	20	20	21	21	22	22	23	23	24	24	
	2			19	20	20	21	21	22	22	23	23	24	24	25	25	
	3				21	21	22	22	23	23	24	24	25	25	26		
	4					22	23	23	24	24	25	25	26	26	27		
t'	5						24	24	25	25	26	26	27	27			
	6							25	26	26	27	27	28	28			
	7								27	27	28	28	29				
	8									28	29	29	30				
	9										30	30					
	10											31					

We now consider a few of these possible scenarios.

- With $t = 6$ and $t' = 3$, we have $m = 22$ while $n - t = 25$. This means that a node does not always have to wait for 25 messages, which by assumption it is guaranteed to receive, before deciding. In favourable circumstances, decisions can be made faster, at the cost of significantly less fault tolerance.
- With $t = 7$ and $t' = 4$, we have $m = n - t = 24$. This means that a node will be able to decide after having received $n - t$ identical messages.
- With $t = 8$ and $t' = 5$, we have $m = 25$ while $n - t = 23$. In this case, one-step consensus can be achieved but only if the node waits to receive at least two more messages than the number it is certain to receive. This means correctly setting the Extended Measurement time factor τ. At the same time, a reasonable number of faulty nodes can still be tolerated.
- In the extreme case of $t = t' = 10$, one-step consensus can only be achieved if a node receives identical messages from all other nodes. In that case, a single crash failure will completely invalidate this possibility. Temporary network delays will also have a significant impact. Attempting to reach one-step consensus here does not seem to be a reasonable goal.

In general, the parameters t and t' should be chosen so that a one-step decision can be reached "often" while still providing an acceptable level of fault tolerance. The protocol can then by fine-tuned to handle scenarios where the number of messages needed for one-step is almost reached: does it wait (and if yes for how long) or does it proceed to the next step regardless.

3.5 Knowing when to stop

Let's assume that m identical messages are needed to reach one-step consensus, and that $n' \in [n - t, n]$ is the maximum number of messages a node is expected to receive before moving on. Then, if a node receives at least $n' - m + 1$ messages each for two different values, it will not be able to reach the threshold for any value. In that situation, there is no advantage of waiting for more messages after receiving the minimum $n - t$ messages.

4 Conclusion

If the nodes of a distributed systems are well-configured machines located in state-of-the-art data centers, the likelihood of crashes can be quite low. In addition, if these nodes have strong incentives to follow the protocol, and strong disincentives no to, the likelihood of Byzantine behaviour will also be lowered. Under these circumstances, it may be possible to relax the requirements for fault tolerance in order to speed up consensus in favourable circumstances.

Given a fixed number of nodes n, we have shown how the possibility of achieving one-step consensus depends on the interplay of two variables: the maximum number of faulty nodes t and the maximum number of Byzantine nodes t', and how waiting for more than the canonical number $n-t$ of messages can be used as an additional factor.

References

[Cra20] Tyler Crain. A simple and efficient asynchronous randomized binary byzantine consensus algorithm. *CoRR*, abs/2002.04393, 2020.

[Kam19] Aleksander Kampa. One-step consensus in weakly byzantine environments. `http://research.sikoba.com/`, 2019.

[Kam21] Aleksander Kampa. An introduction to asynchronous binary byzantine consenus. `http://research.sikoba.com/`, 2021.

[SvR08] Yee Jiun Song and Robbert van Renesse. Bosco: One-step byzantine asynchronous consensus. In *DISC*, volume 5218 of *Lecture Notes in Computer Science*, pages 438–450. Springer, 2008.