Using Weighted Voting to Optimise Streamlined Blockchain Consensus Algorithms

Diana Micloiu[1]

Responsible Professor: Jérémie Decouchant[1]
Supervisor: Rowdy Chotkan[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Micloiu Diana
Final project course: CSE3000 Research Project
Thesis committee: Jérémie Decouchant, Rowdy Chotkan, Kaitai Liang

Abstract

Streamlined Byzantine Fault Tolerant (BFT) protocols, such as HotStuff [PODC'19], and *weighted voting* represent two possible strategies to improve consensus in the distributed systems world. Several studies have been conducted on both techniques, but the research on combining the two is scarce. To cover this knowledge gap, we introduce a weighted voting approach on Hotstuff, along with two optimisations targeting weight assignment distribution and leader rotation in the underlying state replication protocol. Moreover, the weighted protocols developed rely on studies proving the effectiveness of a specific voting power assignment based on discrete values. We generalise this approach by presenting a novel continuous weighting scheme applied to the Hotstuff protocol to highlight the effectiveness of this technique in faulty scenarios. We prove the significant latency reduction impact of weighted voting on streamlined protocols and advocate for further research.

1 Introduction

Many distributed system paradigms, such as state machine replication [1] and blockchains, have a common core concept: *consensus*, which denotes the collective agreement of network participants. Distributed systems are known to be prone to hardware and software failures that can compromise availability or even change the system's normal behaviour. Hence, consensus is needed as a mechanism for coordinating the system's critical actions and ensuring its functionality.

In the blockchain world, consensus algorithms lay at the basis of distributed ledger technologies. They are classified into permissioned and permissionless, distinguishing each other by either limiting participation to a predetermined set of nodes or allowing anyone to join. Out of the two, permissionless systems gained more popularity, with the seminal Nakamoto consensus relying on Proof-of-Work. However, its significant impact on energy consumption revealed the system's limitations and urged researchers to look for alternative consensus algorithms [2]. In turn, interest in permissioned systems grew as their efficiency in terms of throughput, latency, and finality was observed. Thus, the focus shifted towards finding ways to optimise their performance. Strategies such as system size reduction [3] and leader rotation mechanisms [4] have been explored to enhance scalability and resilience. Notably, the Practical Byzantine Fault Tolerance algorithm (PBFT) [5] has been a focal point of research in permissioned systems.

PBFT is part of a more prominent family of protocols: Byzantine Fault Tolerant (BFT), which enables systems to tolerate arbitrary node failures [6–8]. In particular, the protocol requires $3f + 1$ nodes in the system to withstand f failures. Hence, the capability of the system to resist failures comes with the cost of managing the demand for the increased number of nodes and higher communication complexity. In the efficient scenario, the leader is truthful. However, this is not always the case, and protocols need to support intricate fallback strategies, which usually imply node synchronisation and state transfer.

The main disadvantages of BFT protocols, namely that they are slow and expensive to run, support the research of streamlined and cluster-based algorithms. In this sense, researchers developed Hotstuff [9], a streamlined protocol that assumes partial synchrony and uses leader rotation on each block proposal to shift the communication burden from the leader. By using a star-type communication pattern, the protocol achieves linear message complexity and faster response times. Additionally, current research is being conducted to optimise the features of streamlined algorithms [10], such as Pili [11], Pala [12], Streamlet [13], Tendermint [14] and, previously mentioned, Hotstuff [9]. For instance, DAMYSUS improves on top of Hotstuff by reducing the number of communication phases using trusted components, thus achieving better performance [15].

Reaching consensus represents a critical point of improvement for distributed protocols. In this sense, the idea of using a weight metric as voting power gained popularity with Proof-of-Stake (PoS) and reputation-based protocols [16]. Building on top of this kind of mechanism, WHEAT achieved higher performance for state machine replication in geographically distributed settings [17]. Next, researchers put together BFT-SMaRt [18] (an enhanced version of PBFT) and the weighted voting mechanism behind WHEAT to create AWARE, a deterministic, self-monitoring and self-optimising algorithm for reducing the latency of the blockchain [19].

So far, research on the benefits of weighted voting has only studied PBFT in AWARE. This paper seeks to address this literature gap by *investigating the impact of weighted voting on streamlined consensus algorithms*. By extending the principles established by AWARE to Hotstuff [9] and evaluating the robustness in facing node failures, we aim to contribute to the broader understanding of weighted voting's efficacy in streamlined blockchain systems.

This study consists of a latency prediction model which emulates Hotstuff behaviour to gather data on whether or not applying weighted voting decreases the latency of the blockchain algorithm. By analysing different optimisation techniques, this paper points out possible performance improvements and presents conclusive results that encourage the development of an actual deployment in a further study.

As an overview, our contributions can be summarised as follows.

1. We apply AWARE's weighting scheme [19] to Hotstuff and Chained Hotstuff, using two latency prediction models to estimate the algorithms' performance.
2. We analyse how optimising the weight distribution to replicas and/or leader rotation impacts latency by employing separate *Simulated Annealing* [20] methods.
3. We explore the possibility of using continuous weight values instead of AWARE's discrete weights. We apply this weighting scheme on Hotstuff whilst ensuring quorum safety and assessing its effectiveness in reducing latency.

The rest of this paper is organised as follows. §2 reviews the academic advancements in weighted voting and streamlined algorithms. §3 presents a thorough description of Hotstuff for the state replication protocol and WHEAT for the underlying weighting scheme, which is also exploited in AWARE. §4 and §5 describe our contributions by outlining the latency optimisation methods and prediction models. §6 delves into the experimental setup and presents our findings. §7 reflects on the ethical side of this research. §8 provides a critical overview of our experiments and discusses the next steps that could be taken in this area of research. §9 concludes this paper with an overview of the impact of studying weighted voting on streamlined blockchain algorithms in the research field.

2 Related work

The literature related to the aforementioned scientific gap revolves around two key concepts: **weighted voting** and **streamlined algorithms**. Therefore, we provide an overview of each of the two research areas to gather a better understanding of the improvements that have been achieved over the years.

Weighted voting Inspired by the popularity of Proof-of-Stake protocols, researchers have adapted this idea into using a weight metric as the voting power of nodes in permissioned systems. One of the first projects highlighting the advantages of weighted voting was the Cosmos Network, which used a Tendermint-based blockchain protocol and a stake-based voting approach for reaching consensus [21]. Next, the possibility of using weights for electing the leader was researched in credit-based PBFT (CPFT), a blockchain algorithm that tunes the weights based on nodes' past behaviour such that the probability of electing a good leader increases [22]. Three years later, new research on using vague sets and credit rating for optimising the consensus of credit-based PBFT blockchain algorithms appeared [23]. Later on, starting from the same idea of assigning credits to nodes, researchers developed CG-PBFT, a blockchain algorithm that uses a novel credit evaluation model together with a three-way quick sorting algorithm to achieve around 50% increase in throughput [24]. Moreover, current research has tackled the idea of a reward and punishment system based on node ranking in D-PBFT [25].

Streamlined algorithms The research area for streamlined BFT protocols gained interest with the introduction of Hotstuff [9], urging the study of possible optimisations performed on this protocol. Considering multiple points of improvement, researchers came up with variations of Hotstuff targeting a better overall performance of the system. Sync Hotstuff represents one notable research, which shifts the focus from the partially synchronous model of Hotstuff to a fully synchronous one to highlight the trade-offs and impact on performance and security [26]. Next, researchers targeted improving performance by introducing DAMYSUS, a blockchain protocol that enhances Hotstuff by using trusted components to increase resilience and decrease the number of communication phases [15]. On the same path of decreasing the number of communication steps, Hotstuff-2 represents a two-phase variant which explores the relation between leader responsiveness and liveness for achieving optimal results [27]. Furthermore, building on top of DAMYSUS, Oneshot is the first streamlined hybrid BFT protocol which achieves the minimal number of communication phases by exploiting the knowledge of the system's state available to the nodes [28].

Weighted voting on streamlined algorithms Current research also explored applying the weighted voting scheme showcased in AWARE [19] to Hotstuff. By leveraging the idea of an adaptive resilience threshold introduced in ThreatAdapt [3], FLASHCONSENSUS [29] is optimising AWARE using this mechanism of dynamically reducing the number of replicas that actively participate in the protocol's execution. The research also mentions experiments combining this new algorithm with the weighted voting mechanism. FLASHCONSENSUS-flavoured HotStuff uses weights, best leader selection and smaller quorums to prove the effectiveness of using weighted voting on Hotstuff. However, the research uses the enhanced AWARE algorithm rather than the original, and it also leaves out details regarding the implementation of weighted voting on the streamlined algorithm. In addition, the impact on its chained version and the generalisation from discrete weighting remains unexplored.

3 Background

This research aims to combine the state replication protocol of Hotstuff with the weighted scheme introduced by WHEAT [17] whilst also considering the optimisation approaches established by AWARE [19]. Hence, this section delves into the streamlined approach of Hotstuff in §3.1 and presents the two-weight scheme of WHEAT in §3.2.

3.1 Hotstuff: the streamlined approach

Inspired by the simplicity of the theoretical protocol Streamlet [13] and part of the BFT family, Hotstuff is a blockchain protocol that sets itself apart by achieving linear (in the number of nodes) communication complexity [9]. The protocol benefits from the mechanism of switching leaders in each successive round and requires at least $3f + 1$ nodes to tolerate f Byzantine faults.

There are two protocol versions, namely the *Basic Hotstuff* and the *Chained Hotstuff*. The difference between the two comes from the latter's enhanced voting mechanism, which enables a pipelined approach of moving forward multiple blocks in only one round (*view*). For completely processing a block, Hotstuff uses five communication phases, with three core: **prepare, pre-commit,** and **commit** and two additional ones: **new-view**, for sending the last prepared block at the beginning of the protocol, and **decide**, for actually executing the block at the end (see Figure 7 in Appendix A.1).

New-view In this first phase, the leader receives the last prepared block and its corresponding view number from each node. This step consists of gathering all the new-view messages triggered at the end of the precedent view.

Prepare This phase concerns finding the next proposal. The leader awaits for $2f+1$ quorum of nodes with their block-view number information and chooses the block with the highest view number to be extended. Next, the leader sends to all replicas its proposal, and they each provide back a vote if the SAFENODE condition is satisfied [9]. That is, a replica will accept the proposal if the proposed block extends either its latest locked block or a prepared block from a view higher than the one of its last locked block [15]. These checks will ensure the safety and liveness of the protocol, respectively.

Pre-commit The proposed block is marked *prepared* as the leader gets $2f + 1$ votes from the replicas, forming a quorum certificate. The leader sends this certificate to all the replicas so that each can verify it, mark the proposed block as prepared and vote for it in the pre-commit phase.

Commit The leader collects again $2f + 1$ votes and forms a quorum certificate of the prepared block, which becomes *locked* at this step. Next, the leader sends this certificate to all replicas to lock the same block, followed by each sending back a vote.

Decide The leader reaches consensus on the locked block and then *executes* it. All replicas perform the same action after they receive the certificate from the leader.

Locking mechanism The efficiency of communication complexity in the Hotstuff protocol comes with the cost of implementing this locking mechanism. The block is prepared and then locked in the commit phase to preserve the *liveness* of the blockchain consensus algorithm, and only afterwards is it considered *safe* to execute in the final phase.

3.2 WHEAT: the weighting mechanism

Based on the BFT-SMaRt protocol of state machine replication, WHEAT solves the problem of optimising the system's latency in geo-replicated settings [17]. The algorithm is able to attain better performance by introducing the concept of additional replicas. In the BFT family, quorums are formed by gathering a majority of replica responses. In contrast, in WHEAT, the size of a quorum is smaller or equal to that of the Byzantine majority.

Effectively, WHEAT uses weighted voting to achieve consensus faster by leveraging the heterogeneity of the wide-area network (WAN). That is, the protocol assigns higher weights to the replicas that yield the lowest end-to-end latency. In this way, they form smaller quorums but account for the majority needed to move forward. The rest of the replicas constitute a fallback strategy since they form a larger quorum that is in place if the faster replicas become idle. Furthermore, AWARE enhances this technology by introducing mechanisms for self-monitoring (deterministic latency prediction) and self-optimisation (voting weights tuning and leader relocation), such that latency is decreased by giving more power to better-performing replicas [19].

In the Byzantine Fault Tolerant world, a quorum system represents a collection of subsets of replicas that could possibly form a quorum, and any two subsets intersect by $f + 1$ replicas [30]. By adding Δ extra replicas and enforcing a quorum formation mechanism relying on weighted replication, WHEAT imposes the subsequent safe weight distribution scheme.

Consider a BFT system of n replicas withstanding a maximum of f faulty and including Δ additional ones. Hence, n can be expressed as follows:

$$n = 3f + 1 + \Delta \tag{1}$$

Furthermore, regarding consensus, each replica should wait for a quorum formation of Q_v weighted votes:

$$Q_v = 2(f + \Delta) + 1 \tag{2}$$

The voting powers take the form of a binary weight distribution over the replicas of WHEAT. Each node has value either V_{max} or V_{min}, which are computed as follows:

$$V_{\text{max}} = 1 + \frac{\Delta}{f} \tag{3}$$

$$V_{\text{min}} = 1 \tag{4}$$

In the system, the $2f$ replicas that are best performing in terms of latency are attributed weight V_{max} and all the others take V_{min}. Consequently, the weighted quorum contains at most $n - f$ and at least $2f + 1$ replicas.

4 Weighted voting for streamlined algorithms

To sense the impact of weighted voting on streamlined algorithms, this research funnels on a representative blockchain algorithm, namely Hotstuff [9]. This paper evaluates the effectiveness using **latency decrease** as a measurement metric. Hence, to approach the research problem, a latency prediction model is used to emulate the behaviour of Hotstuff, such that intricate implementation details are ignored, and the focus lies on the predicted time of completing a block proposal.

Hotstuff employs five communication phases (which together form a view) to execute a block. To predict the latency from the start of the view, when the leader proposes a block, to the end, when the block is executed, the latency prediction model follows the **new-view, prepare, pre-commit, commit and decide** phases. By analysing a Hotstuff run step-by-step, four quorum formation events can be identified. The leader waits sequentially for **new-view, prepare, pre-commit and commit** messages to complete a block proposal. Since weighted voting is defined within the consensus mechanism, the latency prediction model concentrates on the quorum formation completion times, which can be optimised.

Given the required information on the network topology and the weighting scheme (weight assignment for replicas), a latency prediction for a Hotstuff run depends on the latencies registered by the leader. That is, for each message type x, the leader retains latency vector L_x. In particular, $L_x[i]$ represents the latency, reported by the leader, of receiving the message of type x from replica i. Hence, we can use the corresponding latency vectors to compute the times $t_{PREPARE}, t_{PRECOMMIT}, t_{COMMIT}, t_{DECIDE}$ it takes the leader to form a quorum of messages in order to advance to the next phase (see Algorithm 1). By adding together these timestamps, we get the overall predicted latency of completing a Hotstuff view.

Furthermore, to compute the time it takes to form a quorum, the model uses the voting power of a replica expressed by its corresponding weight. In short, when gathering messages for consensus, we first consider the messages that arrive faster, adding up their weights until the quorum formation condition is satisfied. In this way, the latency of the last message that helped constitute the needed quorum equals the overall time it took the leader to reach a consensus, hence advancing to the next phase.

4.1 Weighted Basic Hotstuff

The basic version of **Weighted Hotstuff** entails using the V_{max} and V_{min} weights (see Equations (3) and (4)) by assigning the highest one to $2f$ replicas. To ensure the safety of the quorum system, the algorithm follows WHEAT [17] and introduces the use of Δ, namely the number of additional replicas. In order to get the predicted latency on a given network setting, the weights assigned are fed into Algorithm 1.

Best weight assignment The weight assignment to replicas represents a critical point of improvement. Hence, **Best Assigned Weighted Hotstuff** employs a *Simulated Annealing* [20] approach to find the proper configuration for a network scenario. Starting from the weight distribution used in Weighted Hotstuff, a candidate solution is generated using a perturbation function as follows: find a replica having V_{min} weight and assign it V_{max} instead, whilst its voting power becomes V_{min}. This way, multiple weighting schemes

Algorithm 1: Latency prediction model for Weighted Hotstuff

```
Data: weightingScheme, leaderRotation, numberOfViews
Result: total latency for running in the given network setup
latency ← 0
for viewNumber ← 0 to numberOfViews do
    currentLeader ← leaderRotation[viewNumber]
    $L_{new-view}$ ← getLatencyOfMessages("new-view", currentLeader)
    $t_{PREPARE}$ ← timeToFormQuorum($L_{new-view}$, weights)
    $L_{prepare}$ ← getLatencyOfMessages("prepare", currentLeader)
    $t_{PRECOMMIT}$ ← timeToFormQuorum($L_{prepare}$, weights)
    $L_{precommit}$ ← getLatencyOfMessages("precommit", currentLeader)
    $t_{COMMIT}$ ← timeToFormQuorum($L_{precommit}$, weights)
    $L_{commit}$ ← getLatencyOfMessages("commit", currentLeader)
    $t_{DECIDE}$ ← timeToFormQuorum($L_{commit}$, weights)
    latency ← latency + ($t_{PREPARE}$ + $t_{PRECOMMIT}$ + $t_{COMMIT}$ + $t_{DECIDE}$);
return latency
```

are tested by predicting the latency under each possible set of weights as an energy function. Ultimately, the algorithm converges to one weighting scheme, yielding the lowest latency for the network scenario.

Optimal Leader Rotation The Hotstuff blockchain algorithm benefits from a linear message complexity due to the novel leader rotation scheme it introduces in ledger technologies. Choosing the best possible succession of leaders is a potential improvement point since it strongly correlates with the algorithm's latency (see Figure 1).

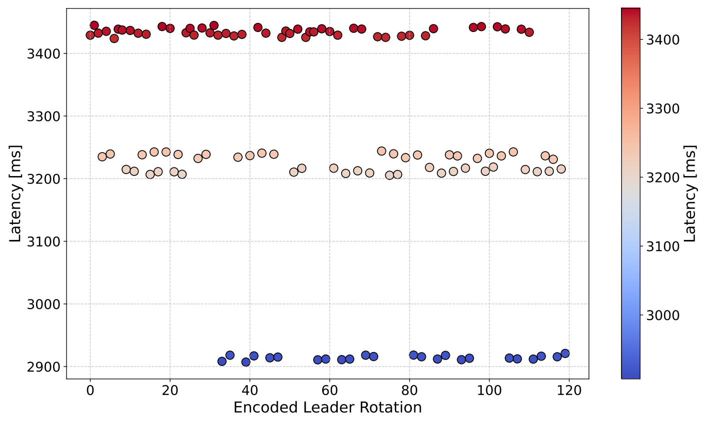

Figure 1: Analysis of the impact of leader rotation on Hotstuff's latency performance for $f = 1, \Delta = 1$, 4 views executed.

However, the leader election itself does not introduce a significant delay directly, but it can indirectly affect the algorithm's behaviour due to leaders' network connectivity, which impacts communication latency. A poorly connected leader can lead to delays in message propagation, and in faultiness cases, it can jeopardise the overall view completion.

Inspired by AWARE'S established leader relocation mechanism [19], this variant of Weighted Hotstuff aims to provide a latency prediction for using an optimised leader rotation. Thus, a decrease in the measurement metric would support further study of a Hotstuff implementation with alternative leader selection strategies. As of its current implementation, Hotstuff uses a *round-robin* leader rotation [31]. However, a leader selection based on network health, similar to that of AWARE, could be beneficial. Illustrating the optimisation opportunity in Hotstuff, *Simulated Annealing* is used by **Optimal Leader Rotation Weighted Hotstuff** to run the blockchain protocol against different leader rotation schemes, collecting the best performance that can be achieved in a given network setting.

The metaheuristic method starts from the classical leader rotation of Hotstuff, which serves as a baseline for comparing alternative solutions. From the existing state, the algorithm moves to a neighbouring one by swapping two leader positions, generating a new possible leader rotation scheme. If the new one proves to be better, the simulation moves to this new state and continues in the same manner until it converges to a leader sequence that is most suitable in terms of latency for a proposed scenario.

Optimal Leader Rotation + Best Assigned A combined *Simulated Annealing* approach targeting both optimisation strategies illustrates their latency reduction impact on Weighted Hotstuff. The algorithm generates a candidate that follows with probability $\frac{1}{2}$ the weighting distribution optimisation model and, with the same probability, a leader rotation one. Hence, the solution indicates the best weighting scheme and leader rotation based on the network environment.

4.2 Weighted Chained Hotstuff

The Chained Hotstuff blockchain algorithm represents a pipelined version of the original one. It benefits from the similarity of the five communication phases to introduce a unique approach to advancing to the next phase on multiple blocks. In a network with n replicas, each one votes for the progress of at most n block proposals in one view. Hence, a leader proposes a block in view i and forwards the responsibility to the leader of the upcoming view until, ideally, it is executed in view $i + 4$ by the corresponding leader.

The latency prediction model used in Weighted Hotstuff is tweaked accordingly to account for these changes in state replication behaviour. For one view, we have a set of block proposals $b_1, b_2...b_n$ for which we consider simultaneous advancement. Since the focus still lies on quorum formation times, the $L_{x,b_j}[i]$ vector encapsulates the latency reported by the leader of the view for receiving the message of type x for block proposal b_j from replica i. The latency vectors are then used to predict the time it takes to reach the required quorum weight for consensus. Note that due to its state replication nature, the adapted model exhibits a warming-up period of n views until the simulation reaches the point of having a maximum capacity of block proposals in each upcoming view.

As in the Weighted Hotstuff algorithm, its chained version also employs a weight metric as voting power in the consensus mechanism. The **Weighted Chained Hotstuff** assigns V_{max}

and V_{min} weights to $2f$ replicas in the network. Furthermore, the blockchain algorithm benefits from the same possible performance improvements: weighting distribution and leader rotation. In this sense, this research explores their impact by developing two *Simulated Annealing* approaches: **Best Assigned Weighted Chained Hotstuff** and **Optimal Leader Rotation Weighted Chained Hotstuff**, which follow the same paradigm presented in §4.1, but with the required changes for emulating the proper chaining behaviour of the protocol.

5 Continuous Weighting Scheme

WHEAT [17] presents the effectiveness of using two-weight vote power assignments in optimising blockchain systems' performance in geographically distributed environments. The use of V_{max} and V_{min} can be interpreted as a limitation of the algorithm. Following this idea, **Continuous Weighted Hotstuff** extends on top of the Weighted Hotstuff algorithm by introducing a weighting scheme generalisation using *Simulated Annealing*. This research only tests its efficacy on Hotstuff to compare its performance to the other variants of protocol optimisations. Nevertheless, the continuous weighting scheme is not limited to the streamlined blockchain world and could be applied to any voting power-based consensus algorithm.

The *Simulated Annealing* approach for finding the continuous weighting scheme that achieves minimal latency works as follows. The latency prediction model of Algorithm 1 is used as an energy function to evaluate the fitness of alternative weighting schemes. To generate a new candidate from the current state, the set of weights uses a $perturbationStep = 0.1$ hyperparameter to draw a new weight from the uniform distribution $U(currentWeight - perturbationStep, currentWeight + perturbationStep)$. Moreover, the voting power of replicas is capped between zero and two to reduce the searching area of the annealing process. The required sum of weights equals the weighted quorum condition of a consensus process to progress on the block proposal safely. Hence, the weighted quorum size depends on the replicas' specific weighting distribution scheme. The choice of capping the weight values does not represent a limitation of the research conducted and does not endanger the validity of the optimisation since the weights are relative to the network scenario.

For a quorum to be *safe*, the following properties of the quorum system need to be validated:

1. **Availability:** Even when the most powerful f replicas fail, there is at least one quorum to reach consensus.
2. **Consistency:** Every two quorums overlap by at least one correct replica.

The probabilistic model uses an additional functionality to deem the correctness of the two quorum system conditions when predicting latency. When a new candidate is used to predict the latency, its corresponding quorum size must be computed. First, considering the f replicas having the highest voting power faulty implies that the sum of weights of the remaining replicas is an upper bound on the quorum size. *Availability* is satisfied by considering this upper bound as the required weighted consensus condition. To ensure *consistency*, only the subsets of replicas with total voting power greater or equal to the weighted quorum size are valid. The algorithm takes any two valid quorums and verifies if they overlap by $f+1$ replicas. Suppose the candidate continuous weighting scheme passes all of these steps. In that case, the quorum system is well-founded, and the latency predicted by the model is taken into account by the annealing process in moving to a subsequent state. Otherwise, the current state remains unmodified for the next step of the algorithm.

6 Evaluation

This paper analyses the impact of weighted voting on streamlined algorithms by gathering data from multiple experiments to determine whether or not introducing the voting power assignment generates significant latency improvements.

6.1 Experimental setup

We evaluate the performance of Weighted Hotstuff (§4.1), Weighted Chained Hotstuff (§4.2) and their multiple optimisation variants to weigh up against those of basic and chained Hotstuff. Thus, we perform experiments in two scenarios: non-faulty, when the network behaves normally, and faulty, when some replicas are idle.

To run the latency prediction models, we use Python scripts to test the behaviour of protocols in a given network scenario. Hence, each experiment depends on the specified distributed environment. Moreover, the latency prediction algorithm for Weighted Hotstuff requires providing a set of weights. The algorithm's behaviour matches that of Best Assigned Weighted Hotstuff in case of optimal weight assignment. Thus, the experiments are conducted by tailoring the weighting distribution to the network scenario. That is, data on the distance within all replicas is gathered, and the f best and f worst connected replicas are assigned V_{max} voting power.

To make the analysis reliable, experiments are conducted using real data collected from cloudping, a tool for latency monitoring of deployed AWS clusters [32]. For the results presented later in this section, the following clusters were used: *Cape Town (af-south-1), Hong Kong (ap-east-1), Canada (ca-central-1), London (eu-west-2) and Northern California (us-west-1)*. This choice of network setting supports the research of weighted voting efficacy on geographically distributed settings since it reflects the heterogeneity of wide area network (WAN) environments. Moreover, simulations were performed using a blockchain system with $f = 1$ and $\Delta = 1$ (having a total of $n = 5$ replicas). Since Hotstuff is a protocol that sets itself apart from the others in the BFT family by using a new leader in each view, the views represent a critical aspect of the influence of weighted voting. Thus, we vary the number of views by performing simulations for all values from 5 to 20, focusing on the average latency per view.

The latency prediction models depend on the values generated for latency vectors L_x. In an actual deployment, the latency at which the leader receives messages from each replica is influenced by two factors: distance from leader to replica, reflected in link latency and the payload of the transmitted data, namely message delay. To emulate the network behaviour properly, for a leader i, we create its corresponding latency vectors L_x as follows: distance within clusters (for the link latency) plus an offset drawn from a uniform sample $U(0, 5)$ for $L_{new-view}$ (since new-view messages encapsulate the block proposal, whereas the rest have lower payload containing just hashes) and $U(0, 2)$ for the rest. If the link is fast enough, the message payload latency can be neglected. However, for a better analysis of weighted voting, we went for a uniform distribution with significantly lower values than the cluster latency. In this way, message complexity is considered without severely influencing the protocol's behaviour but with the perks of mocking the heterogeneity of real environments.

6.2 Non-faulty scenario

The experiments performed considered as baseline Basic and Chained Hotstuff whose latencies are obtained by running the prediction model of Weighted Hotstuff with all weights equal to $V_{normal} = 1$. In this way, we highlight the effectiveness of weighted voting on streamlined protocols (see Table 1).

Table 1: Performance comparison between Hotstuff and Weighted Hotstuff variants based on Figures 2 and 3.

	Basic	**Chained**
No optimisation	-7.13%	-7.28%
Best Assigned	-19.48%	-19.64%
Optimal Leader Rotation	-10.52%	-9.78%
Continuous	-18.30%	n.a
Optimal Leader Rotation + Best Assigned	-24.36%	-22.48%

Weighted Hotstuff Figure 2 showcases the impact of weighted voting on Hotstuff. It is clear from this visual representation of latency performance that introducing weighted voting decreases the latency of running the streamlined blockchain algorithm. Compared with the Basic Hotstuff in which all weights are assigned equal voting power, Weighted Hotstuff exhibits around 7% decrease in latency. Adding to the V_{max}/V_{min} weight assignment, the Optimal Leader Rotation variant reduces latency by up to 10%. However, there is room for improvement as Best Assigned and Continuous Weighted Hotstuff both indicate almost 20% latency decrease. Furthermore, by combining the two optimisation approaches, (Optimal Leader Rotation + Best Assigned) Weighted Hotstuff reveals around 25% latency reduction. Hence, the prediction models developed on multiple protocol variants support the idea established in AWARE of introducing leader relocation and weight-tuning mechanisms for improving the performance of blockchain algorithms.

Figure 2 also shows that the optimal leader rotation variant has the same latency as its underlying weight optimisation Hotstuff variant over multiple of n views. That is a consequence of the simulation framework using a static network setting for the experiments conducted. Since for such a simulation consisting of a multiple of n views, each replica is the leader for an equal amount of times, the cumulative latency over all the views is the same.

Chained Weighted Hotstuff Figure 3 showcases the impact of weighted voting on Chained Hotstuff. The figure highlights the warming-up period of n views and supports the research of weighted voting on streamlined algorithms with a 7% latency improvement gained by just assigning weights. Furthermore, adding the leader rotation optimisation on top of it accounts for an additional 3% decrease in latency. As presented in Best Assigned Weighted Hotstuff, its chained version follows the same trends, exhibiting an average latency per view with 20% better than the baseline and performing best with an up to 25% reduction for the combined version of the two optimisations.

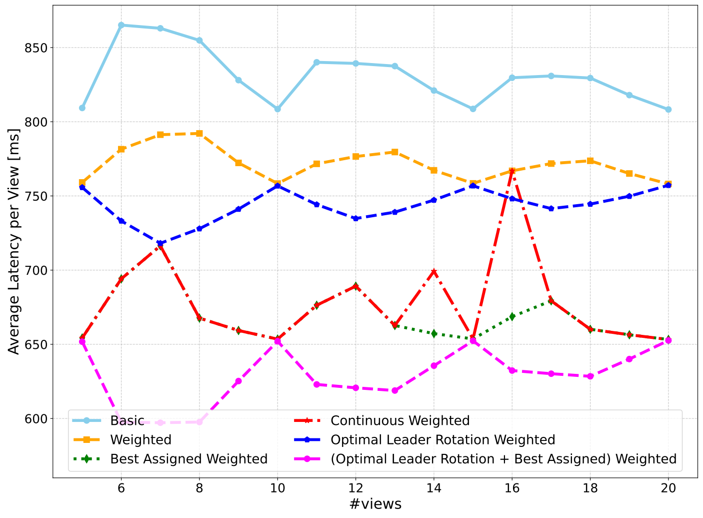

Figure 2: Average latency per view in Hotstuff protocol variants for $f = 1, \Delta = 1$.

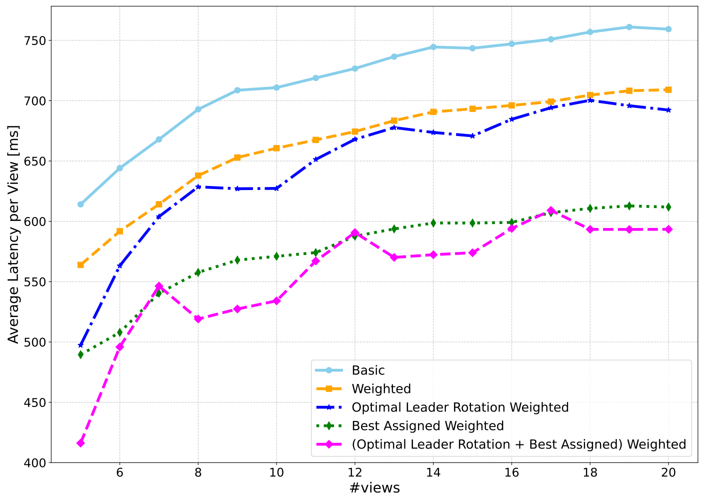

Figure 3: Average latency per view in Chained Hotstuff protocol variants for $f = 1, \Delta = 1$.

6.3 Faulty scenario

Resilience against failures is critical in any distributed environment. Hence, we compare the performance of different Weighted Hotstuff optimisations under faulty conditions. Since streamlined protocols are part of the BFT family, the system would need at least $3f + 1$ nodes to withstand f failures. Thus, for a given network scenario, the f replicas holding the highest voting power are considered idle, jeopardising faster consensus. In this way, we force the analysed streamlined algorithms to mimic their fallback scenario strategy.

Weighted Hotstuff Figure 4 shows that, out of all, the (Optimal Leader Rotation + Best Assigned) Weighted Hotstuff performs the best. Conversely, Best Assigned and Continuous Weighted Hotstuff express their over-fitting nature by showcasing significantly higher fallback latency. Since these two variants optimise the weight assignment for a given network scenario, the system would inevitably take longer to recover from an idle actor. Additionally, they perform worse

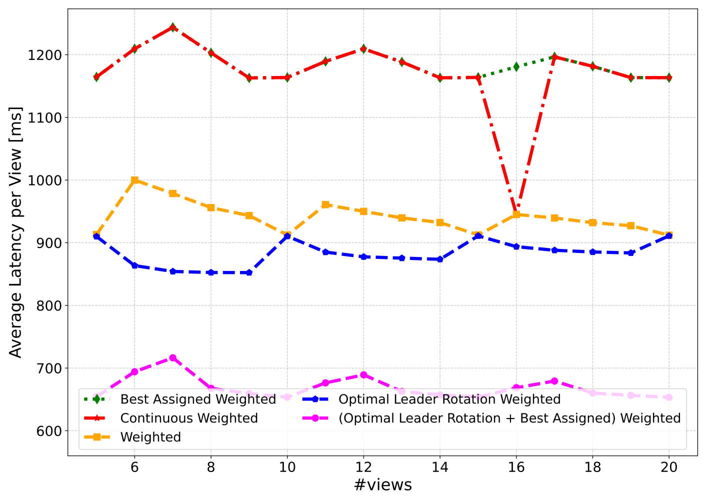

Figure 4: Average latency per view in Hotstuff protocol variants for *faulty scenario*, $f = 1, \Delta = 1$.

than Weighted Hotstuff and its leader rotation optimisation version under normal network conditions and far worse under faulty ones.

Continuous Weighting Scheme As expected, Continuous Weighted Hotstuff performs at least as well as the Best Assigned one in non-faulty environments. However, we introduce this generalisation to investigate its impact in faulty scenarios. For this, we experimented with multiple simulations on randomly generated network topologies with within clusters latency ranging between 0 and $400\,\mathrm{ms}$. Figure 5 showcases the difference in latency between the Best Assigned and Continuous Weighted Hotstuff and proves that the latter performs better or equal in 85% of the performed simulations (see Figure 8 in Appendix A.2). Notably, the continuous variant fails to surpass the discrete one in all simulations due to the *Simulated Annealing* algorithm terminating before reaching the global minimum. Hence, this research supports further study of the continuous metric for voting power assignment.

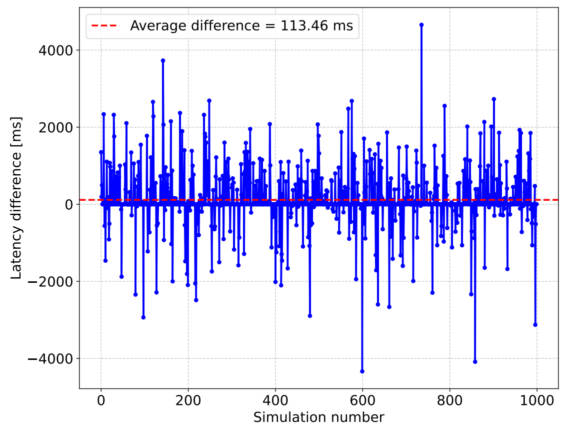

Figure 5: Difference in latency performance between Best Assigned and Continuous Weighted Hotstuff variants for 1000 *faulty scenario* simulations, $f = 1, \Delta = 1$, 10 views executed.

Chained Weighted Hotstuff Despite behaving similarly to its basic version, Chained Weighted Hotstuff and its optimisations show major differences in fallback behaviour, as presented in Figure 6. Best Assigned Weighted Chained Hotstuff has the lowest fallback latency. That is due to the communication pattern of the chained protocol. Because simulations are performed under a specific set of weights, Chained Hotstuff cannot overfit when optimising for best weight assignment since the pipelined block proposal mechanism heavily influences the algorithm performance. With voting quorums for multiple blocks in one view, some weight assignments might benefit the consensus of one block whilst requiring more time to gather messages for the other. Hence, the Best Assigned Chained Hotstuff has higher resilience against idle nodes. Furthermore, the Weighted and Optimal Leader Rotation Weighted Hotstuff have higher fallback latency but still within the expected limits of $200\,\mathrm{ms}$ delay, with the latter performing better under this faulty environment setting.

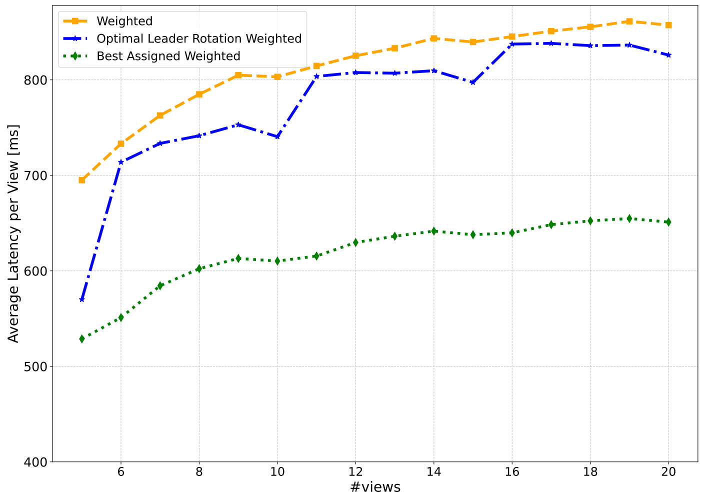

Figure 6: Average latency per view in Chained Hotstuff protocol variants for *faulty scenario*, $f = 1, \Delta = 1$.

7 Responsible Research

This section reflects upon the ethical aspects of the research, presenting the measures taken to adhere to the Netherlands Code of Conduct for Research Integrity [33].

Datasets Given the area of research, namely blockchain algorithms, this paper does not use predefined datasets or extend upon implemented algorithms of previous academic work. This research employs designing latency prediction models for Weighted Hotstuff and Weighted Chained Hotstuff alongside their variants. As detailed in §6, we used data from cloudping [32] to get the latency between different AWS clusters. Moreover, we motivated our choice of clusters, providing transparency in our research. Hence, the experiments aim to emulate a real-life scenario using reliable data to support the study of weighted voting on streamlined algorithms.

Results Due to the nature of this research project, results are heavily influenced by the prediction models. Hence, we provide a Gitlab repository that describes the codebase extensively in its included README file [34]. Furthermore, all design choices are mentioned and explained in §4 and §6. Besides, the results can be obtained by running the Python experiment files. In interpreting the results, we objectively point out the performance aspects presented in the figures and explain any odd behaviour that the plots showcase. This way, we target transparency and reproducibility of our research, complying with the FAIR (Findable, Accessible, Interoperable and Reusable) principles.

Research process This research process was conducted responsibly by ensuring a proper literature review. §2 describes all the research performed in recent years on our study's two areas of interest: weighted voting and streamlined algorithms. Additionally, we mentioned a research study that briefly looked into the possibility of applying weighted voting on Hotstuff, and we discussed its limitations to put our research endeavours into perspective. The latency prediction models were developed after an in-depth analysis of Hotstuff and Chained Hotstuff communication protocols to ensure the validity of our solutions. Moreover, for Continuous Weighted Hotstuff, we followed the axiomatic quorum system properties and provided an implementation that would comply with the availability and consistency requirements. Throughout the research process, we tried to be as meticulous as possible to guarantee the ethics and correctness of our paper.

8 Discussion

The results presented in §6 showcase the impact of our research on the study of streamlined blockchain algorithms. Providing multiple prediction models for different optimisations of weighted voting in Hotstuff and Chained Hotstuff, this paper states the efficacy of the voting power mechanism in such ledger technologies. Thus, we provide real arguments towards the broader study of this research area.

Limitations Some experiments, such as simulations of the *Simulated Annealing* approaches for $n > 15$ replicas, are impractical due to the high code complexity. Moreover, for the continuous weighting scheme, the function for checking that the properties of the quorum are satisfied is computationally expensive, making simulations for $n > 4$ infeasible (see Figure 9 in Appendix A.2). Hence, further research could benefit from extensive analysis for latency prediction in a broader network context.

Following the same limitation pattern, the latency prediction models only treat the scenario of running protocols on a specific set of weights. The next step would be dynamically changing the weights from one view to the other to accommodate the leader rotation better. However, to ensure the availability and consistency of the quorum system, this approach entails checking that every two quorums formed on a protocol run overlap by at least one correct replica. This check only would severely delay the overall simulation time of the experiment. Thus, a study should be conducted to circumvent this limitation and research a safe way of changing the voting powers.

Furthermore, the experiments benefit from using specified reliable data of within AWS clusters latency extracted from cloudping [32]. Even though the setting supports this research, it also introduces some limitations. The most notable one is for the study of the continuous weighting scheme approach, which points out the significant impact of the technique only when run on a randomly generated network scenario. Hence, the next step would be extending this research by running the prediction models on timestamped data from AWS cluster monitoring to observe how weighted voting would impact performance given real-time network fluctuations.

Future work This paper aims to provide the means to understand the impact of weighted voting on streamlined blockchain algorithms. Hence, we advocate for future studies in this area by showing results of possible latency improvements generated through the weighted voting approach. In this sense, upcoming research should apply weighted voting on the actual implementation of Hotstuff and Chained Hotstuff protocols. Instead of using the cloudping data, the research should deploy AWS clusters to observe the real-time behaviour of the blockchain algorithms. Furthermore, the research should follow AWARE's steps of establishing leader relocation and vote assignment fine-tuning mechanisms for improving latency performance in geographically distributed settings whilst taking into account the unreliable side of distributed environments.

9 Conclusions

This paper explored the potential of weighted voting in optimising streamlined blockchain algorithms and analysed, in particular, the latency impact on Hotstuff and its chained variant [9]. It investigated possible protocol improvements in weight assignment and leader rotation. Besides, it looked into extending from AWARE's weighting scheme [19] to a generalised continuous approach that would improve performance whilst maintaining the safety of the quorum system. To this end, we introduced latency prediction models that emulate Hotstuff and Chained Hotstuff behaviour and assign voting powers to the system's nodes, hence simulating a **Weighted (Chained) Hotstuff** protocol. Furthermore, we used *Simulated Annealing* to estimate the impact of optimising the weight assignment distribution in **Best Assigned Weighted** and the leader rotation in **Optimal Leader Rotation Weighted** for both basic and chained versions of the streamlined protocol. Using these latency prediction models, we conducted experiments on reliable data of within AWS clusters latency from cloudping [32] and also analysed the faulty scenario of best-performing nodes becoming idle. In this way, our experiments are based on real data and consider the fallback scenario, producing results which support future research in the area. Hence, we showcased that only applying weighted voting to a streamlined blockchain algorithm reduces latency by around 7%. Moreover, combining best weight assignment and optimal leader rotation achieves minimal latency, almost 25% lower than one of the classic protocols. As for the **Continuous Weighted Hotstuff**, the enhancement from the discrete set of weights performs in faulty settings equally well or better than Best Assigned Weighted Hotstuff in around 85% of the simulations.

In short, this research represents the standing proof that weighted voting decreases the latency of Hotstuff and Chained Hotstuff, and optimisations of the weight distribution and leader rotation are critical for further performance improvement. As for the generalisation of AWARE's weighting scheme [19], this paper introduces a novel approach of using continuous weights as the voting power of the nodes, which is set to improve performance in recovery scenarios. Even though applied in this study only on the Hotstuff algorithm, any blockchain algorithm can employ the continuous weighting scheme. The results provided in this research, together with the novel ideas described, are a founding base for the study of weighted voting in streamlined algorithms and its shift from the discrete model.

References

[1] F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: A tutorial,” *ACM Computing Surveys (CSUR)*, vol. 22, no. 4, pp. 299–319, 1990.

[2] R. Asif and S. R. Hassan, “Shaping the future of Ethereum: exploring energy consumption in Proof-of-Work and Proof-of-Stake consensus,” *Frontiers in Blockchain*, vol. 6, 2023.

[3] D. S. Silva, R. Graczyk, J. Decouchant, M. Völp, and P. Esteves-Verissimo, “Threat adaptive byzantine fault tolerant state-machine replication,” pp. 78–87, 2021.

[4] Y. Amir, B. Coan, J. Kirsch, and J. Lane, “Prime: Byzantine replication under attack,” *IEEE transactions on dependable and secure computing*, vol. 8, no. 4, pp. 564–577, 2010.

[5] M. Castro, B. Liskov *et al.*, “Practical byzantine fault tolerance,” in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[6] L. Lamport, R. Shostak, and M. Pease, “The Byzantine generals problem,” in *Concurrency: The Works of Leslie Lamport*, 2019, pp. 203–226.

[7] C. Cachin and M. Vukolic, “Blockchain Consensus Protocols in the Wild,” *CoRR*, vol. abs/1707.01873, 2017.

[8] C. Natoli, J. Yu, V. Gramoli, and P. J. E. Veríssimo, “Deconstructing Blockchains: A Comprehensive Survey on Consensus, Membership and Structure,” *CoRR*, vol. abs/1908.08316, 2019.

[9] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, “HotStuff: BFT consensus with linearity and responsiveness,” in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[10] E. Shi, “Streamlined blockchains: A simple and elegant approach (a tutorial and survey),” in *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I 25*. Springer, 2019, pp. 3–17.

[11] T. H. Chan, R. Pass, and E. Shi, “Pili: An extremely simple synchronous blockchain,” *Cryptology ePrint Archive*, 2018.

[12] ——, “Pala: A simple partially synchronous blockchain,” *Cryptology ePrint Archive*, 2018.

[13] B. Y. Chan and E. Shi, “Streamlet: Textbook streamlined blockchains,” in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 1–11.

[14] E. Buchman, J. Kwon, and Z. Milosevic, “The latest gossip on BFT consensus,” *CoRR*, vol. abs/1807.04938, 2018.

[15] J. Decouchant, D. Kozhaya, V. Rahli, and J. Yu, “DAMYSUS: streamlined BFT consensus leveraging trusted components,” in *Proceedings of the Seventeenth European Conference on Computer Systems*, 2022, pp. 1–16.

[16] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, “Repucoin: Your reputation is your power,” *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1225–1237, 2019.

[17] J. Sousa and A. Bessani, “Separating the WHEAT from the chaff: An empirical design for geo-replicated state machines,” in *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2015, pp. 146–155.

[18] A. Bessani, J. Sousa, and E. E. Alchieri, “State machine replication for the masses with BFT-SMART,” in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 355–362.

[19] C. Berger, H. P. Reiser, J. Sousa, and A. Bessani, “AWARE: Adaptive wide-area replication for fast and resilient Byzantine consensus,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1605–1620, 2020.

[20] Baeldung, “Simulated Annealing,” Website, 2023, Accessed on May 30, 2024. [Online]. Available: https://www.baeldung.com/cs/simulated-annealing

[21] J. Kwon and E. Buchman, “The Cosmos Network: A Primer on Tendermint-Based Blockchains,” Tendermint Inc. and Interchain GmbH, White Paper, 2016. [Online]. Available: https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md

[22] Y. Wang, Z. Song, and T. Cheng, “Improvement research of PBFT consensus algorithm based on credit,” in *Blockchain and Trustworthy Systems: First International Conference, BlockSys 2019, Guangzhou, China, December 7–8, 2019, Proceedings 1*. Springer, 2020, pp. 47–59.

[23] Z. Zeng, B. Wen, W. Du, F. Zhang, and W. Zhou, “PBFT Consensus Algorithm Optimization Scheme Based on Vague Sets and Credit Rating,” in *2023 6th International Conference on Software Engineering and Computer Science (CSECS)*. IEEE, 2023, pp. 1–5.

[24] J. Liu, X. Deng, W. Li, and K. Li, “CG-PBFT: an efficient PBFT algorithm based on credit grouping,” *Journal of Cloud Computing*, vol. 13, no. 1, pp. 1–20, 2024.

[25] J. Wang, W. Feng, M. Huang, S. Feng, and D. Du, “Improvement of Practical Byzantine Fault Tolerance Consensus Algorithm Based on DIANA in Intellectual Property Environment Transactions,” *Electronics*, vol. 13, no. 9, p. 1634, 2024.

[26] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin, “Sync hotstuff: Simple and practical synchronous state machine replication,” in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 106–118.

[27] D. Malkhi and K. Nayak, “Hotstuff-2: Optimal two-phase responsive bft,” *Cryptology ePrint Archive*, 2023.

[28] J. Decouchant, D. Kozhaya, V. Rahli, and J. Yu, “OneShot: View-Adapting Streamlined BFT Protocols with Trusted Execution Environments,” in *IPDPS 2024*, 2024.

[29] C. Berger, L. Rodrigues, H. P. Reiser, V. Cogo, and A. Bessani, “Chasing the speed of light: Low-latency

planetary-scale adaptive Byzantine consensus," *CoRR*, vol. abs/2305.15000, 2023.

[30] D. M. M. Reiter *et al.*, "Byzantine quorum systems," *Distributed Computing*, vol. 11, no. 4, pp. 203–213, 1998.

[31] D. Yaga, P. Mell, N. Roby, and K. Scarfone, *Blockchain Technology Overview*. NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2018.

[32] "CloudPing," Website, Accessed on June 2, 2024. [Online]. Available: https://www.cloudping.co/grid/latency/timeframe/1D

[33] Netherlands Organisation for Scientific Research (NWO), "Netherlands Code of Conduct for Research Integrity," Accessed on June 19, 2024. [Online]. Available: https://www.nwo.nl/en/netherlands-code-conduct-research-integrity

[34] D. Micloiu, "Using Weighted Voting to Optimise Streamlined Blockchain Consensus Algorithms," 2024, Accessed on June 23, 2024. [Online]. Available: https://github.com/dmicloiu/weightedhotstuff.git

A Appendix

A.1 Hotstuff communication

Figure 7 depicts the five communication phases of Hotstuff [9] that together form one *view*. This view has n participating replicas, out of which one is the leader. As the illustration highlights, only the leader communicates with the other replicas, thus achieving linear communication rather than the quadratic one of PBFT [5].

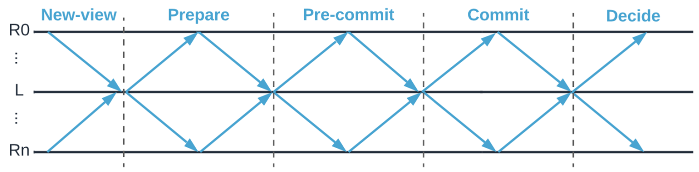

Figure 7: Hotstuff communication phases.

A.2 Continuous Weighted Hotstuff Analysis

Compared with the Continuous version, Best Assigned Weighted Hotstuff has higher latency in 13% of the simulations performed (see Figure 8). Even though the algorithms have matching performance in around 50% of the cases, the generalised weighting scheme still recovers faster in 30%. Since the optimisation model we developed relies on *Simulated Annealing* [20], the algorithm can be further tweaked to improve these results. By increasing the hyperparameter for step convergence, we can let the annealing process explore more, eventually reaching the state of a better-performing solution than the Best Assigned one.

However, the prediction model we developed in this research has one considerable limitation: *high computational complexity*. Due to the rigorous process of ensuring quorum system safety, the time it takes the *Simulated Annealing* algorithm to converge increases significantly with the number of faulty replicas the system can withstand. In this sense, Figure 9 showcases that running the prediction model for $f = 4$ took almost 40 times more than for the precedent value.

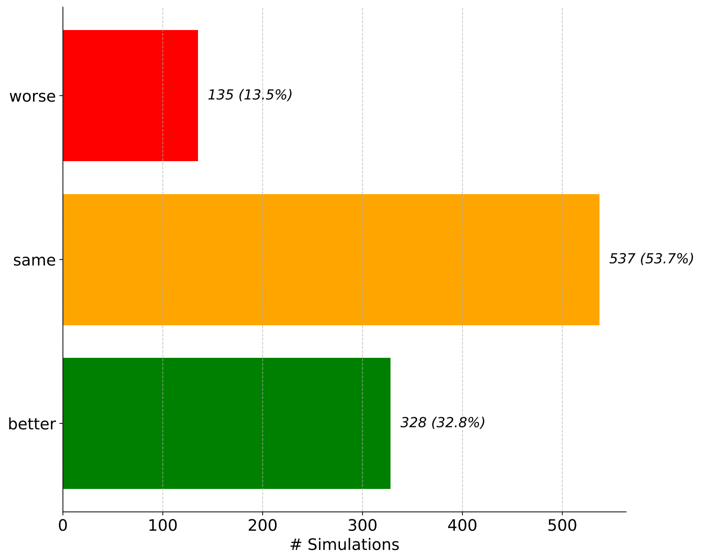

Figure 8: Latency performance comparison between Continuous and Best Assigned Weighted Hotstuff for 1000 *faulty scenario* simulations, $f = 1, \Delta = 1$, 10 views executed.

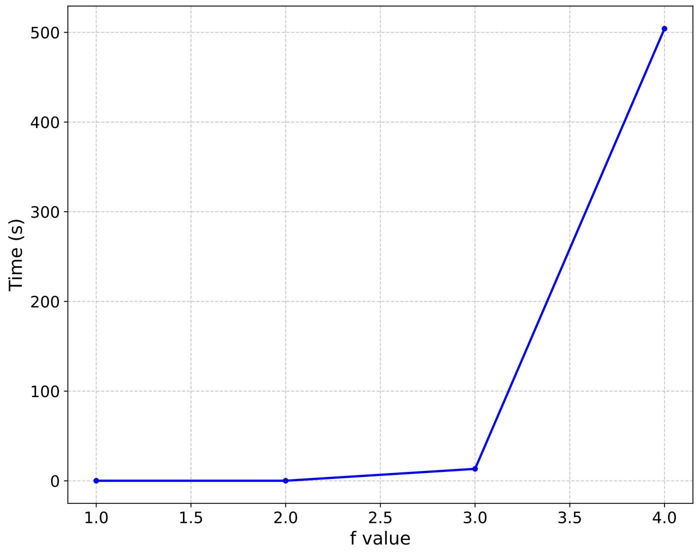

Figure 9: Simulated Annealing convergence time for Continuous Weighted Hotstuff for multiple f values, 1 view executed.