CSE3000 Weekly Progress Presentation

WEEK 1

Diana Micloiu

What have I done so far?


Adaptive Wide Area REplication
AWARE
automated and dynamic voting-weight tuning and partitioning scheme
↳ deterministic, self-optimization algorithm
geographically DS
↳ dynamically optimize consensus latency by self-reliably finding a fast configuration → latency gains
latency optimization scheme
resilient, adaptive Byzantine consensus
WHEAT + BFT-SMART
underlying weighting scheme
replication protocol

Covering Research Background

- Studied and took notes from the AWARE paper [1]
- Consolidated my research background by reading the following papers
  - PBFT: Practical Byzantine Fault Tolerance [2]
  - WHEAT [3]
  - State Machine Replication [4]

[1] Christian Berger et al. “AWARE: Adaptive wide-area replication for fast and resilient Byzantine consensus”. In: IEEE Transactions on Dependable and Secure Computing 19.3 (2020), pp. 1605–1620.
[2] Miguel Castro, Barbara Liskov, et al. “Practical byzantine fault tolerance”. In: OsDI. Vol. 99. 1999. 1999, pp. 173–186.
[3] João Sousa and Alysson 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.
[4] Fred B Schneider. “Implementing fault-tolerant services using the state machine approach: A tutorial”. In: ACM Computing Surveys (CSUR) 22.4 (1990), pp. 299–319.

Research Plan **(Draft Week 1)** for CSE3000 Research Project

Using Weighted Voting to Accelerate Blockchain Consensus

Diana Micloiu

April 23, 2024

Planning of the research project

Note that the "Project Meeting" is a weekly meeting together with the supervisor, responsible professor and peers to discuss project advancements and provide/receive feedback.

WEEK 1

1. Read the following papers:
   - PBFT: Practical Byzantine Fault Tolerance [3]
   - AWARE: Adaptive wide-area replication for fast and resilient Byzantine consensus [2]
   - WHEAT [1]
   - State Machine Replication [5]
   - HotStuff: BFT consensus with linearity and responsiveness [6]
   - DAMYSUS: streamlined BFT consensus leveraging trusted components [4]
2. During the *weekly Supervisor Meeting* discuss understanding of AWARE.
3. Analyse the research conducted in the area and on the topic of the **Research Question** and identify around 10 most relevant papers.
4. Scan the identified papers and choose 3-5 most relevant to base the research on and study them thoroughly.
5. Understand the background of the research and identify the limitations imposed by each of the existing papers.
6. Write the final Research Plan.

10 papers to support the research

AWARE: Adaptive Wide-Area Replication for Fast and Resilient Byzantine Consensus

Christian Berger, Hans P. Reiser, João Sousa, and Alysson Bessani

Abstract—With upcoming blockchain infrastructures, world-spanning Byzantine consensus is getting practical and necessary. In geographically distributed systems, the pace at which consensus is achieved is limited by the heterogeneous latencies of connections between replicas. If deployed on a wide-area network, consensus-based systems benefit from weighted replication, an approach that utilizes extra replicas and assigns higher voting weights to well-connected replicas. This approach enables more choice in quorum formation and replicas can leverage proportionally smaller quorums to advance, thus decreasing consensus latency. However, the system needs a solution to autonomously adjust to its environment if network conditions change or faults occur. We present Adaptive Wide-Area REplication (AWARE), a mechanism that improves the geographical scalability of consensus with nodes being widely spread across the world. Essentially, AWARE is an automated and dynamic voting-weight tuning and leader positioning scheme, which supports the emergence of fast quorums in the system. It employs a reliable self-monitoring process and provides a prediction model seeking to minimize the system's consensus latency. In experiments using several AWS EC2 regions, AWARE dynamically optimizes consensus latency by self-reliantly finding a fast configuration yielding latency gains observed by clients located across the globe.

Index Terms—adaptivness, weighted replication, consensus, geo-replication, Byzantine fault tolerance, self-optimization, blockchain

1 Introduction

State machine replication (SMR) is a classical approach for building resilient distributed systems. It achieves fault-tolerance by coordinating client interactions with independent server replicas [1]. Furthermore, SMR protocols typically use either some dynamically selected leader [2] or are fully decentralized [3]. In both cases, these protocols usually require communication steps involving a major subset of all nodes.

With the emergence of novel Byzantine fault-tolerant (BFT) blockchain infrastructures, BFT SMR protocols have been increasingly catching academic attention over the last few years. For example, the BFT-SMaRt [4] library has been employed as an ordering service [5] for the Hyper-

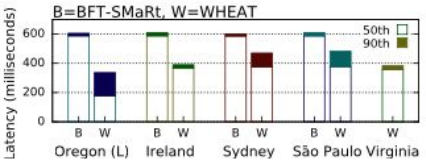

Figure 1: Latency gain using weighted replication in WHEAT and BFT-SMaRt [7], as measured by clients co-located with different replicas.

10 papers to support the research

State Machine Replication for the Masses with BFT-SMaRt

Alysson Bessani, João Sousa
Faculdade de Ciências, Universidade de Lisboa, Portugal

Eduardo E. P. Alchieri
Universidade de Brasilia, Brazil

***Abstract*—The last fifteen years have seen an impressive amount of work on protocols for Byzantine fault-tolerant (BFT) state machine replication (SMR). However, there is still a need for practical and reliable software libraries implementing this technique. BFT-SMaRt is an open-source Java-based library implementing robust BFT state machine replication. Some of the key features of this library that distinguishes it from similar works (e.g., PBFT and UpRight) are improved reliability, modularity as a first-class property, multicore-awareness, reconfiguration support and a flexible programming interface. When compared to other SMR libraries, BFT-SMaRt achieves better performance and is able to withstand a number of real-world faults that previous implementations cannot.**

I. Introduction

The last fifteen years have seen an impressive amount of

The main contribution of this paper is to fill a gap in the BFT literature by documenting the implementation of this kind of system, including protocols for state transfer and reconfiguration. Additionally, the paper presents an evaluation of BFT-SMaRt, comparing it with previous systems and shedding light on some performance tradeoffs related to tolerance of crashes vs. Byzantine faults.

The paper is organized as follows: §II and §III describe the design of BFT-SMaRt and its implementation, respectively. §IV presents alternative configurations for BFT-SMaRt. §V describes an evaluation of our system. §VI highlights some lessons learned during the development and maintenance of the system. Finally, §VII presents our concluding remarks.

10 papers to support the research

Separating the WHEAT from the Chaff: An Empirical Design for Geo-Replicated State Machines

João Sousa and Alysson Bessani
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

***Abstract*—State machine replication is a fundamental technique for implementing consistent fault-tolerant services. In the last years, several protocols have been proposed for improving the latency of this technique when the replicas are deployed in geographically-dispersed locations. In this work we evaluate some representative optimizations proposed in the literature by implementing them on an open-source state machine replication library and running the experiments in geographically-diverse PlanetLab nodes and Amazon EC2 regions. Interestingly, our results show that some optimizations widely used for improving the latency of geo-replicated state machines do not bring significant benefits, while others – not yet considered in this context – are very effective. Based on this evaluation, we propose WHEAT, a configurable crash and Byzantine fault-tolerant state machine replication library that uses the optimizations we observed as most effective in reducing SMR latency. WHEAT employs novel voting assignment schemes that, by using few additional spare replicas, enables the system to make progress without needing to access a majority of replicas. Our evaluation shows that a WHEAT system deployed in several Amazon EC2 regions presents a median latency up to 56% lower than a "normal" SMR protocol.**

mizations are actually effective in practice. This is aggravated by the fact that these evaluations tend to compare SMR protocols in an holistic manner and generally do not compare individual optimizations.

In this paper we present an extensive evaluation of several latency-related optimizations scattered across the literature (both for local data centers and geo-replication) using the same testbeds, methodology and codebase (see §III). More specifically, we selected a subset of optimizations for decreasing the latency of strongly-consistent geo-replicated systems, implemented them in the BFT-SMART replication library [6] (see §II) and deployed the experiments in the PlanetLab testbed and in the Amazon EC2 cloud. During the course of this evaluation, we obtained some unexpected results. The most notorious example is related with the use of multiple leaders – a widely accepted optimization used by several WAN-optimized protocols such as Mencius [23] and EPaxos [24]. Specifically, our results indicate that this optimization does not bring significant latency reduction just by itself; instead, we observed that using a fixed leader in a fast replica is a more

10 papers to support the research

Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial

FRED B. SCHNEIDER

Department of Computer Science, Cornell University, Ithaca, New York 14853

The state machine approach is a general method for implementing fault-tolerant services in distributed systems. This paper reviews the approach and describes protocols for two different failure models—Byzantine and fail stop. System reconfiguration techniques for removing faulty components and integrating repaired components are also discussed.

Categories and Subject Descriptors: C.2.4 **[Computer-Communication Networks]**: Distributed Systems—*network operating systems*; D.2.10 **[Software Engineering]**: Design—*methodologies*; D.4.5 **[Operating Systems]**: Reliability—*fault tolerance*; D.4.7 **[Operating Systems]**: Organization and Design—*interactive systems, real-time systems*

General Terms: Algorithms, Design, Reliability

Additional Key Words and Phrases: Client-server, distributed services, state machine approach

INTRODUCTION

Distributed software is often structured in terms of *clients* and *services*. Each service comprises one or more *servers* and exports *operations* that clients invoke by making *requests*. Although using a single, centralized, server is the simplest way to implement a service, the resulting service can only be as fault tolerant as the processor executing that server. If this level of fault tolerance is unacceptable, then multiple

service by replicating servers and coordinating client interactions with server replicas.[1] The approach also provides a framework for understanding and designing replication management protocols. Many protocols that involve replication of data or software—be it for masking failures or simply to facilitate cooperation without centralized control—can be derived using the state machine approach. Although few of these protocols actually were obtained in this manner, viewing them in terms of state

10 papers to support the research

HotStuff: BFT Consensus with Linearity and Responsiveness

Maofan Yin
Cornell University
VMware Research

Dahlia Malkhi
VMware Research

Michael K. Reiter
UNC-Chapel Hill
VMware Research

Guy Golan Gueta
VMware Research

Ittai Abraham
VMware Research

ABSTRACT

We present HotStuff, a leader-based Byzantine fault-tolerant replication protocol for the partially synchronous model. Once network communication becomes synchronous, HotStuff enables a correct leader to drive the protocol to consensus at the pace of actual (vs. maximum) network delay—a property called *responsiveness*—and with communication complexity that is linear in the number of replicas. To our knowledge, HotStuff is the first partially synchronous BFT replication protocol exhibiting these combined properties. Its simplicity enables it to be further pipelined and simplified into a practical, concise protocol for building large-scale replication services.

CCS CONCEPTS

• **Software and its engineering** → **Software fault tolerance**; • **Security and privacy** → **Distributed systems security**.

stabilization time (GST). In this model, $n \geq 3f + 1$ is required for non-faulty replicas to agree on the same commands in the same order (e.g., [12]) and progress can be ensured deterministically only after GST [27].

When BFT SMR protocols were originally conceived, a typical target system size was $n = 4$ or $n = 7$, deployed on a local-area network. However, the renewed interest in Byzantine fault-tolerance brought about by its application to blockchains now demands solutions that can scale to much larger n. In contrast to *permissionless* blockchains such as the one that supports Bitcoin, for example, so-called *permissioned* blockchains involve a fixed set of replicas that collectively maintain an ordered ledger of commands or, in other words, that support SMR. Despite their permissioned nature, numbers of replicas in the hundreds or even thousands are envisioned (e.g., [30, 42]). Additionally, their deployment to wide-area networks requires setting Δ to accommodate higher variability in communication delays.

10 papers to support the research

DAMYSUS: Streamlined BFT Consensus Leveraging Trusted Components

Jérémie Decouchant*
j.decouchant@tudelft.nl
TU Delft

David Kozhaya*
david.kozhaya@ch.abb.com
ABB Research

Vincent Rahli*
vincent.rahli@gmail.com
University of Birmingham

Jiangshan Yu*
J.Yu.Research@gmail.com
Monash University

Abstract

Recently, *streamlined* Byzantine Fault Tolerant (BFT) consensus protocols, such as HotStuff, have been proposed as a means to circumvent the inefficient view-changes of traditional BFT protocols, such as PBFT. Several works have detailed trusted components, and BFT protocols that leverage them to tolerate a minority of faulty nodes and use a reduced number of communication rounds. Inspired by these works we identify two basic trusted services, respectively called the Checker and Accumulator services, which can be leveraged by streamlined protocols. Based on these services, we design `Damysus`, a streamlined protocol that improves upon HotStuff's resilience and uses less communication rounds. In addition, we show how the Checker and Accumulator services can be adapted to develop `Chained-Damysus`, a *chained* version of `Damysus` where operations are pipelined for efficiency. We prove the correctness of `Damysus` and `Chained-Damysus`, and evaluate their performance showcasing their superiority compared to previous protocols.

CCS Concepts: • **Theory of computation → Distributed algorithms**.

Such systems comprise multiple software and hardware components that are bound to eventually fail, potentially causing system malfunction or unavailability. To this end, a distributed system relies on a consensus protocol to agree on actions critical to the system's correct operation.

In particular, Byzantine Fault Tolerant (BFT) consensus protocols allow systems to withstand arbitrary failures [1, 2, 3]. However, being able to tolerate arbitrary (Byzantine) component failures, induces a non-negligible system overhead, namely in the required number of nodes that should be present in the system as well as the communication complexity. For example, traditional BFT protocols, such as PBFT [4], typically require $3f+1$ nodes to tolerate up to f Byzantine faulty nodes. Moreover, albeit being efficient in the normal case, i.e., when the current leader is correct, traditional BFT protocols use complex "view-change" operations to replace possibly malicious (compromised) leaders and often require the nodes to "synchronize" and transfer their states.

In order to circumvent such complex view changes, recent "streamlined" BFT protocols [5][1] such as PiLi [6], PaLa [7], Tendermint [8], HotStuff [9], and Streamlet [10] rotate the leader on each command. Instead of performing a state trans-

10 papers to support the research

Streamlined Blockchains: A Simple and Elegant Approach

(A Tutorial and Survey) *

Elaine Shi
Cornell University runting@gmail.com

February 4, 2020

Abstract

A blockchain protocol (also called state machine replication) allows a set of nodes to agree on an ever-growing, linearly ordered log of transactions. The classical consensus literature suggests two approaches for constructing a blockchain protocol: 1) through composition of single-shot consensus instances often called Byzantine Agreement; and 2) through direct construction of a blockchain where there is no clear-cut boundary between single-shot consensus instances. While conceptually simple, the former approach precludes cross-instance optimizations in a practical implementation. This perhaps explains why the latter approach has gained more traction in practice: specifically, well-known protocols such as Paxos and PBFT all follow the direct-construction approach.

In this tutorial, we present a new paradigm called "streamlined blockchains" for directly constructing blockchain protocols. This paradigm enables a new family of protocols that are extremely simple and natural: every epoch, a proposer proposes a block extending from a notarized parent chain, and nodes vote if the proposal's parent chain is not too old. Whenever a block gains enough votes, it becomes *notarized*. Whenever a node observes a notarized chain with several blocks of consecutive epochs at the end, then the entire chain chopping off a few blocks at the end is *final*.

By varying the parameters highlighted in blue, we illustrate two variants for the partially synchronous and synchronous settings respectively. We present very simple proofs of consistency and liveness. We hope that this tutorial provides a compelling argument why this new family of protocols should be used in lieu of classical candidates (e.g., PBFT, Paxos, and their variants), both in practical implementation and for pedagogical purposes.

10 papers to support the research

STREAMLET: Textbook Streamlined Blockchains

Benjamin Y Chan[1] and Elaine Shi[2]

[1]Cornell University - byc@cs.cornell.edu
[2]CMU/Cornell - runting@gmail.com

September 12, 2020

Abstract

In the past five years or so, numerous blockchain projects have made tremendous progress towards improving permissioned consensus protocols (partly due to their promised applications in Proof-of-Stake cryptocurrencies). Although a significant leap has silently taken place in our understanding of consensus protocols, it is rather difficult to navigate this body of work, and knowledge of the new techniques appears scattered.

In this paper, we describe an extremely simple and natural paradigm called STREAMLET for constructing consensus protocols. Our protocols are inspired by the core techniques that have been uncovered in the past five years of work; but to the best of our knowledge our embodiment is simpler than ever before and we accomplish this by taking a "streamlining" idea to its full potential. We hope that our textbook constructions will help to decipher the past five years of work on consensus partly driven by the cryptocurrency community — in particular, how remarkably simple the new generation of consensus protocols has become in comparison with classical mainstream approaches such as PBFT and Paxos.

10 papers to support the research

Fast-HotStuff: A Fast and Robust BFT Protocol for Blockchains

Mohammad M. Jalalzai*†, Jianyu Niu*†, Chen Feng*† and Fangyu Gai*†
*School of Engineering, The University of British Columbia, Kelowna, Canada
†Blockchain@UBC, The University of British Columbia, Vancouver, Canada
{m.jalalzai, jianyu.niu, chen.feng, fangyu.gai}@ubc.ca

***Abstract*—The HotStuff protocol is a recent breakthrough in Byzantine Fault Tolerant (BFT) consensus that enjoys both responsiveness and linear view change by creatively adding a round to classic two-round BFT protocols like PBFT. Despite its great advantages, HotStuff has a few limitations. First, the additional round of communication during normal cases results in higher latency. Second, HotStuff is vulnerable to certain performance attacks, which can significantly deteriorate its throughput and latency. To address these limitations, we propose a new two-round BFT protocol called Fast-HotStuff, which enjoys responsiveness and efficient view change that is comparable to the linear view-change in terms of performance. Our Fast-HotStuff has lower latency and is more robust against the performance attacks that HotStuff is susceptible to.**

***Index Terms*—BFT, Blockchain, Consensus, Latency, Performance, Security.**

and there is a pessimistic bound Δ on the network delay. Hence, the protocol has $O(\Delta)$ latency (instead of the actual network latency), therefore lacking responsiveness.

The HotStuff protocol is the first to achieve both linear view change[3] and responsiveness, solving a decades-long open problem in BFT consensus. Linear view change enables fast leader rotation while responsiveness drives the protocol to consensus at the speed of wire $O(\delta)$, where δ is the actual network latency. Both are desirable properties in the blockchain space. In BFT protocols, a decision is made after going through several phases, and each phase usually takes one round of communication before moving to the next. HotStuff introduces a chained structure borrowed from blockchain to pipeline all the phases into a unifying propose-vote[4] pattern which significantly

10 papers to support the research

Liveness Checking of the HotStuff Protocol Family

Jérémie Decouchant, Burcu Kulahcioglu Ozkan, Yanzhuo Zhou
Department of Software Technology
Delft University of Technology
Delft, the Netherlands
J.Decouchant@tudelft.nl, B.Ozkan@tudelft.nl, Y.Zhou-21@student.tudelft.nl

***Abstract*—Byzantine consensus protocols aim at maintaining safety guarantees under any network synchrony model and at providing liveness in partially or fully synchronous networks. However, several Byzantine consensus protocols have been shown to violate liveness properties under certain scenarios. Existing testing methods for checking the liveness of consensus protocols check for time-bounded liveness violations, which generate a large number of false positives. In this work, for the first time, we check the liveness of Byzantine consensus protocols using the temperature and lasso detection methods, which require the definition of ad-hoc system state abstractions. We focus on the HotStuff protocol family that has been recently developed for blockchain consensus. In this family, the HotStuff protocol is both safe and live under the partial synchrony assumption, while the 2-Phase Hotstuff and Sync HotStuff protocols are known to violate liveness in subtle fault scenarios. We implemented our liveness checking methods on top of the Twins automated unit test generator to test the HotStuff protocol family. Our results indicate that our methods successfully detect all known liveness violations and produce fewer false positives than the traditional time-bounded liveness checks.**

***Index Terms*—Byzantine consensus, Hotstuff protocols, Liveness checking, Lasso detection, Testing**

Previous testing works on consensus have mostly focused on analyzing crash-tolerant protocols. For example, Jepsen testing tool [3] simulates network partitions for distributed systems, and it has detected several violations in the consensus systems [4]–[6]. Twins [7] is one of the few testing systems that have been specifically designed to test the safety of BFT consensus protocols under Byzantine scenarios. Twins can detect safety violations using scenarios that involve only a few communication rounds. However, most existing testing systems do not check for liveness violations, which require the generation of infinite executions.

A common approach to finding liveness violations is to check for *bounded* liveness, i.e., checking whether the properties are satisfied within a bounded amount of time[8], [9]. To do so, the programmer sets some bounds for an event to happen and reports the executions that exceed the specified thresholds. In the case of consensus protocols, correct processes should accept the same value within a certain delay or within a given number of execution steps. However, it is difficult for developers to correctly estimate adequate bound values, in particular in real-world production-level consensus

10 papers to support the research

Concordia: A Streamlined Consensus Protocol for Blockchain Networks

CARLOS SANTIAGO, SHUYANG REN, CHOONHWA LEE, AND MINSOO RYU
Department of Computer Science, Hanyang University, Seoul 04763, South Korea

Corresponding author: Choonhwa Lee (lee@hanyang.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant 2020R1A2B5B01001758, and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korean Government (MSIT) (A Delegated Byzantine-Tolerant Consensus Algorithm for Blockchain Scalability) under Grant 2019-0-00458.

ABSTRACT In this paper, we present a novel Byzantine fault-tolerant consensus protocol for sharded blockchain networks that does not rely on expensive leader-driven communication. The proposed protocol selects a single block proposer at a time and uses threshold signatures as a voting mechanism to confirm the validity of the proposed block. By using a gossip-like communication scheme, each node can collect and recover the group signature within O(log$N$) steps. With only one block proposer per consensus round, there is no possibility of conflicting blocks and resultant forks. Therefore, our consensus protocol requires only one round of one-way communication to achieve finality for each block. Our protocol guarantees safety and liveness while tolerating up to f faulty participants among $2f + 1$ nodes. Our performance study shows that the proposed protocol enables hundreds of nodes to participate in the agreement process, and can finalize large blocks in approximately 10 seconds.

10 papers to support the research

Pipelet: Practical Streamlined Blockchain Protocol

Vivek Karihaloo
Ruchi Shah
Panruo Wu
University of Houston

Aron Laszka
Pennsylvania State University

ABSTRACT

Fueled by the growing popularity of proof-of-stake blockchains, there has been increasing interest and progress in permissioned consensus protocols, which could provide a simpler alternative to existing protocols, such as Paxos and PBFT. In particular, the recently proposed Streamlet protocol provides a surprisingly simple and streamlined consensus approach, which crystallizes years of research in simplifying and improving classical consensus protocols. While the simplicity of Streamlet is a major accomplishment, the protocol lacks certain practical features, such as supporting a stable block proposer, and it makes strong assumptions, such as synchronized clocks and the implicit echoing of all messages. Most importantly, it requires sending $O(N^3)$ messages per block in a network of N nodes, which poses a significant challenge to its application in larger networks. To address these limitations, we introduce Pipelet, a practical streamlined consensus protocol. Pipelet employs the same block-finalization rule as Streamlet, but attains state-of-the-art performance in terms of communication complexity and provides features that are crucial for practical applications, such as clock synchronization and stable block proposers. At the same time, Pipelet retains the simplicity of Streamlet, which presents significant practical advantages, such as ease of implementation and verification.

agreement among honest participants despite the presence of malicious ones.

Distributed consensus can be viewed as a replicated state machine [27], where a deterministic state machine is replicated across a set of processes but functions as one. The input of the state machine is called transaction, which causes a transition of states. A transaction [17] is an atomic operation that either completes or does not occur at all. The responsibility of a consensus protocol is to order the transactions such that the resulting transaction log of every process is the same.

Despite the great success of the Bitcoin network, there are major concerns, including high energy consumption and environmental impact, slow transaction confirmation, and low throughput. More modern blockchain technology often builds on Proof-of-Stake [11, 19] instead of Proof-of-Work to be energy efficient, which often relies on a permissioned committee of validators [15]. Such approaches can be described as "blockchainized" classic consensus algorithms, such as Tendermint [4, 5], which adopts classic consensus algorithm PBFT [7]. This approach typically involves two rounds of voting to confirm a block: a prepare round and a commit round. An interesting recent idea is to pipeline the voting, that is, to piggyback the second round (commit) of the block on the first round (prepare) of the next block. HotStuff [30], PaLa [9], and Casper FFG [6] are such pipelined protocols.

Understanding of the Research Question

Original formulation: **How can weighted voting improve the performance of streamlined algorithms (2nd gen) [6]?**

Interpretation:

- AWARE is using WHEAT and BFT-SMaRt to improve performance on geo-distributed settings
- The research question should validate if a similar approach of weighting voting can be applied to streamlined algorithms such as Hotstuff (or DAMYSUS which is an improvement on top of it) to improve their performance in geo-distributed settings.
- That is use the idea of weighting and possibly adding more replicas, together with Hotstuff to enable optimised leader allocation and redistributing of weights such that latency optimization is achieved

What is next?

Goals for the week onwards

1. **FINISH** tackling the Reading Material **-** Identify the main papers to guide the research upon and study them thoroughly.
2. Complete the Final Research Plan and submit it.
3. Identify limitations of the current research in the area.
4. Shape the idea of improvement, namely how exactly can weighted voting can be used to leverage latency optimization.

Question round