Connected car communication by DLT technologies : mobility service implementation by adaptation of consortium blockchain consensus algorithms

Cyril Naves Samuel

Communications de la Voiture Connectée grâce aux Technologies DLT : Implémentation d'un Service de Mobilité par Adaptation des Algorithmes de Consensus de Blockchains de Consortium

Cyril Naves Samuel

Laboratoire d'Électronique, Antennes et Télécommunications (LEAT)

Présentée en vue de l'obtention du grade de Docteur en Informatique d'Université Côte d'Azur

Dirigée par : François Verdier, Professeur des universités, Université Côte d'Azur.

Co-encadrée par : Séverine Glock, Ingénieure, Groupe Renault

Soutenue le : 7 Décembre 2023

Président Du Jury :
Aurélien Francillon, Professeur des universités, EURECOM.

Devant le Jury, composé de :
Thi Mai Trang Nguyen, Professeur des universités, Université Sorbonne Paris Nord.
Cyrille Bertelle, Professeur des universités, Université - Le Havre Normandie.
Yves Roudier, Professeur des universités, Université Côte d'Azur.
Christine Hennebert, Ingénieure de Recherche, CEA LETI.
Séverine Glock, Ingénieure, Groupe Renault.
Patricia Guitton, Ingénieure, Groupe Renault.
François Verdier, Professeur des universités, Université Côte d'Azur.

Renault Group

Cifre

Jury

Communications de la voiture connectée grâce aux technologies DLT : Implémentation d'un service de mobilité par adaptation des algorithmes de consensus de blockchains de consortium

Connected Car Communication by DLT Technologies: Mobility Service Implementation by Adaptation of Consortium Blockchain Consensus Algorithms

Président du jury

- Aurelien Francillon, Professeur des universités, EURECOM.

Rapporteurs

- Thi Mai Trang Nguyen, Professeur des universités, Université Sorbonne Paris Nord.
- Cyrille Bertelle, Professeur des universités, Université - Le Havre Normandie.
- Yves Roudier, Professeur des universités, Université Côte d'Azur.

Examinateurs

- Christine Hennebert, Ingénieure de Recherche, Commissariat à l' Énergie Atomique et aux Énergies Alternatives.
- Séverine Glock, Ingénieure, Groupe Renault.

Invités

- Patricia Guitton, Ingénieure, Groupe Renault

Direction de These

- François Verdier, Professeur des universités, Université Côte d'Azur.

Abstract

Distributed Ledger Technology can be compared to a universe of galaxies that everyone can witness, that works in perfect synchronization through consensus decision, secures the life of biological species, exhibits transparency of scientific phenomena, and presents hidden potential. Distributed Ledger can bridge between friendly or competing parties to create a mutually beneficial and sustainable ecosystem for sharing data, monetary transactions, digital objects, and assets while guaranteeing transparency, security, and privacy.

Mobility solutions have been embracing distributed ledger decentralized solutions as they represent a perfect solution in the current age of multimodal transport-enabled smart cities. Be it decentralized energy grids, big data-hungry artificial intelligence platforms, autonomous vehicles, or drones they create an accountable and open society. This thesis presents two insights into decentralized mobility solutions around the theme of data certification and data monetisation.

The decentralized automotive-oriented service exploits the smart contract-empowered autonomous applications by the frameworks of Ethereum and Substrate. Each platform divulges contrasting features of consensus algorithms, smart contract construction, interoperability, external oracle communication, and data privacy. Our work evaluates these two solutions from a functional perspective and throughput performance accompanied by transaction finalization. It poses certain fascinating shortcomings around Byzantine Fault Tolerant (BFT) Algorithms of Clique, Istanbul BFT, Quorum BFT, and Practical BFT, which we understand and evaluate in our work.

We further ascertain these challenges in our conceived blockchain simulator for homogeneous and unbiased experimentation of BFT consensus algorithms. It exposes the challenges of scalability, communication complexity, failure tolerance issues, and protocol resilience. We treat these issues in our proposed consensus algorithm, Competing Utilitarian Byzantine Agreement (CUBA), along with its implicit variant, which leverages pipelining, quorum-based communication, network optimization, and utilitarian fairness to improve the outcome of the protocol. Benchmark results show better performance, security, and resilience than the BFT consensuses of PBFT, IBFT, and QBFT comparable to Clique. It regulates the honesty of the participants without any extrinsic factors of computation, assets, or trusted execution environment, achieving a truly democratic consortium byzantine agreement of competing participants.

Keywords
Distributed Ledger Technology, Blockchain, Byzantine Fault Tolerance, Consensus, Distributed Algorithms, Mobility

Résumé

La technologie de Blockchain peut être comparée à un univers de galaxies dont tout le monde peut être témoin, qui fonctionne en parfaite synchronisation grâce à une décision consensuelle, assure la vie des espèces biologiques, montre la transparence des phénomènes scientifiques et présente un potentiel caché. Le Blockchain peut servir de pont entre des parties amies ou concurrentes pour créer un écosystème durable et mutuellement bénéfique pour le partage de données, de transactions monétaires, d'objets numériques et d'actifs tout en garantissant la transparence, la sécurité et la protection de la vie privée.

Les solutions de mobilité ont adopté les solutions décentralisées à registres distribués, car elles représentent une solution parfaite à l'ère actuelle des villes intelligentes centrées sur le transport modal. Qu'il s'agisse de réseaux énergétiques décentralisés, de plateformes d'intelligence artificielle gourmandes en données, de véhicules autonomes, ou drones, ces solutions créent une société responsable et ouverte. Cette thèse présente deux points de vue sur les solutions de mobilité décentralisée autour du thème de la certification et de la monétisation des données.

Le service décentralisé axé sur l'automobile exploite les applications autonomes dotées de contrats intelligents établis sur les plateformes Ethereum et Substrate. Chaque plateforme présente des caractéristiques contrastées en matière d'algorithmes de consensus, de construction de contrats intelligents, d'interopérabilité, de communication avec des oracles externes et de confidentialité des données. Notre travail évalue ces deux solutions d'un point de vue fonctionnel et des performances de débit accompagnées de la finalisation des transactions. Il pose certaines lacunes fascinantes autour des algorithmes de tolérance aux fautes byzantines (BFT) de Clique, Istanbul BFT, Quorum BFT, et Practical BFT, que nous comprenons et évaluons dans notre travail.

Nous vérifions ces défis dans notre simulateur de blockchain conçu pour une expérimentation homogène et impartiale des algorithmes de consensus BFT. Il expose les défis de l'évolutivité, de la complexité de la communication, des problèmes de tolérance aux pannes et de la résilience du protocole. Nous traitons ces questions dans l'algorithme de consensus que nous proposons, Competing Utilitarian Byzantine Agreement (CUBA), ainsi que dans sa variante implicite, qui tire parti du pipelining, de la communication basée sur le quorum, de l'optimisation du réseau et de l'équité utilitaire pour améliorer le résultat du protocole. Les résultats de l'analyse comparative montrent que les performances, la sécurité et la résilience sont meilleures que les consensus BFT de PBFT, IBFT et QBFT comparables à Clique. Le protocole régule l'honnêteté des participants sans aucun facteur extrinsèque de calcul, d'actifs ou d'environnement d'exécution fiable, ce qui permet d'obtenir un accord byzantin de consortium véritablement démocratique entre participants concurrents.

Mots Clés
Technologie des registres distribués, Blockchain, Tolérance aux Fautes Byzantines, consensus, Algorithmes Distribués, Mobilité

Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.

– Marie Curie

Acknowledgements

When we give cheerfully and accept gratefully, everyone is blessed

– Maya Angelou

Gratitude is a simple word when I consider the amount of love and compassion I received from all those who have been part of my work.
Prof. Francois Verdier has been the architect of this thesis work from its planning until its finish. He has been more than a thesis director to me as he has never commanded; instead, he gave me courage, liberty, and a profound scientific inclination to accomplish this work. I am not sure If I would have a chance to meet someone as pleasant as him, and I offer my big thanks.
Another vital and dynamic person whom I have watched closely and inspired by her dedication to work and perfection is Severine Glock. She has guided me from the start of my internship until now, making me comfortable at Renault. She has been a remarkable mentor complementing my inadequacies in automotive business knowledge. I am incredibly grateful for her constant belief in me and her benevolence.
An ever-smiling and warm person is Patricia Guitton, who has been the bridge to transpose me from Renault to University and vice versa. She has always been enthusiastic about my work and trusted me greatly throughout this journey. I present my humble gratitude of appreciation and thanks to her.
I would like to extend my heartfelt gratitude to my respected thesis Jury: (Professors) Thi Mai Trang Nguyen, Cyrille Bertelle, Yves Roudier, Christine Hennebert, and Aurelien Francillon for having agreed to validate and review my work with patience and diligence.
Another important mentor during my internship was David Bercowitz, who guided me to Blockchain and Automotive and helped me mature as a thesis candidate. I am deeply obliged to him for his kindness and advice.
Feedback during any thesis work is vital, which I have regularly received from Christine Hennebert and Prof. Frederic Mallet. They have always been constructive with a blend of care and sympathy, for which I offer my thankful memories.
I would also like to place my profound sense of gratitude to my Renault managers Francois Amand, Frederic Turgis, Frederic Noraz, CTO-Katrin Matthes, and Engineers Gimeno Ignacio, Eric Marchand, Denis Delaveau, Julien Aknine, Laurent Thiry, Nicole Chalhoub, JP Larue, JP, Jean Yves Carre, Remi Laudebat, Pierre, Clement, Fred Joly, Cyril Bianconi, Marc Peresse, Sebastien Griffoul, Thierry Tambay, Alexandre Lewicki, Fabrice Olivero, Christophe Chauvin, Cedric Bondier, Laurent Bresciani, Cedric Vamour, Sally Alsayah, Gregory Bayle, Yasmine Assioua, Joelle Abou Faysal, Berkay Koksal and Raj Patel.
I am grateful to Laboratory Researchers Robert Staraj, Benoit Miramond, Alain Pegatoquet, Jean Yves Dauvignac, Laurent Rodriguez, Françoise Trucas, Lydie Nguyen, Sophie Gaffe, Laurent Brochier, Eric, Rahman, and colleagues Jonathan, Thomas, Edgar, Julien, Yann, Francesco, Walid, Andrea and Imourane.
Friends boost you always, and I was glad to have Luc Gerrits, Roland Kromes, Marino, Cristian, and Vanjul, who have been my buddies whom I have always relied upon.
Lastly, I am always grateful for my mother Juliet's unconditional love, and Buddy & Goofy.

Thesis Publications

Published papers may omit important steps and the memory of men of science, even the greatest, is sadly fallible.

– John Desmond Bernal

Journal Publications

- **Cyril Naves Samuel**, François Verdier, Severine Glock, and Patricia Guitton-Ouhamou, "Can Byzantine Fault Tolerant Consensus Agreement be Utilitarian, Scalable and Resilient? An Algorithm Proposition from a Consortium Blockchain Perspective: CUBA", ACM Distributed Ledger Journal Publication 2024. *Submitted*
- **Cyril Naves Samuel**, François Verdier, Severine Glock, and Patricia Guitton-Ouhamou, "A Fair and Inclusive Crowd-Sourced Automotive Data Harvesting and Monetisation Approach Using Substrate Hybrid Consensus Algorithms Blockchain", MDPI Future Internet Journal- Security in the Internet of Things (IoT) Publication 2024. *Submitted*

International Conferences

- **Cyril Naves Samuel**, Severine Glock, David Bercowitz, François Verdier and Patricia Guitton-Ouhamou, "Automotive Data Certification Problem: A View on Effective Blockchain Architectural Solutions," 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2020. *Awarded Best Research Paper and Best Paper Presentation*
- **Cyril Naves Samuel**, Severine Glock, François Verdier, and Patricia Guitton-Ouhamou, "Choice of Ethereum Clients for Private Blockchain: Assessment from Proof of Authority Perspective," 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia, 2021.
- Luc Gerrits, **Cyril Naves Samuel**, Roland Kromes, François Verdier, Severine Glock, and Patricia Guitton-Ouhamou. 2021. Experimental Scalability Study of Consortium Blockchains with BFT Consensus for IoT Automotive Use Case. In Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (SenSys '21). Association for Computing Machinery, New York, NY, USA, 492–498.
- **Cyril Naves Samuel**, François Verdier, Severine Glock, and Patricia Guitton-Ouhamou, "Virtuous Data Monetisation Cycle: A Hybrid Consensus Substrate Automotive Consortium Blockchain Solution" 2023 5th International Congress on Blockchain and Applications (BLOCKCHAIN'23), Guimarães, Portugal, 2023.
- **Cyril Naves Samuel**, François Verdier, Severine Glock, and Patricia Guitton-Ouhamou, "CUBA: An Evolutionary Consortium Oriented Distributed Ledger Byzantine Consensus Algorithm" 2023 20th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2023), Guimarães, Portugal, 2023.

National Conferences

- **Cyril Naves Samuel**, François Verdier, Severine Glock, Patricia Guitton-Ouhamou. Private Blockchain Assessment from Proof of Authority Perspective. 15ème Colloque National du GDR SOC2, Jun 2021, France.
- Luc Gerrits, **Cyril Naves Samuel**, François Verdier. IoT-Blockchain Based Ecosystem using Hyperledger Sawtooth. GDR SOC2, Jun 2022, Strasbourg, France.

Code Repository Publications

An algorithm must be seen to be believed.

– Donald Ervin Knuth

Thesis Public Code Repository Summary

- **Data Certification Implementation on Ethereum:**
  https://github.com/scyrilnaves/these-datacertification
- **Data Monetisation Implementation on Substrate:**
  https://github.com/scyrilnaves/these-datamonetisation
- **Blockchain Simulator Implementation of PBFT, IBFT, QBFT and Clique with Deployment:**
  https://github.com/scyrilnaves/these-blockchainconsensussimulator
- **Blockchain Simulator Implementation of CUBA with Deployment:**
  https://github.com/scyrilnaves/these-cuba
- **Results collation along with Rscript for plot generation:**
  https://github.com/scyrilnaves/these-resultscollated

Contents

List of Figures

List of Tables

Chapter 1

Introduction

All truths are easy to understand once they are discovered; the point is to discover them.

– Galileo Galilei

This chapter will inspire us to have a cursory overview of the concepts we will discuss and approach in the forthcoming research work chapters.

1.1 Decentralization as Reaction to Dystopia

Dystopia signifies an imaginary totalitarian state which was coined as an antonym to Utopia [1] by Thomas More in the 15th century. On the contrary 'Utopian' state, as the portrayed map in Figure 1.1, represents a perfect state with no private ownership and an egalitarian condition that is hard to imagine. It signifies an ideal solution to a surveillance state where everyone is subjected to suffering and injustice. This was further fictionalized in work by George Orwell in his work *1984* where the author of his age already coined the words 'thoughtcrime' and 'Big Brother' in his imaginary repressive state of Oceania. Another similar ideology fictional work is by Lewis, Michael in FlashBoys [2], where an organized group compromises a centralized stock exchange during high-frequency trading for performing preemptive transactions to gain an illegitimate monetary advantage. These works expose the fallacy of centralization or represent the challenging times we are currently in where the sovereign state or actors exercise their absolutism, as evidenced by the Snowden Whistleblowing Act or the Cambridge Analytica Scandal. Top companies like Google, Amazon, Facebook, and Microsoft, collectively termed as GAFAM or FAANG or 'internet gatekeepers,' have controlled the digital economy through their monopolistic activities, manipulation of public opinion, and disregard of people's privacy [3]. These issues have forced society to move towards a decentralized, democratic web termed Web3, which has given birth to a slew of solutions like Matrix (decentralized communication network), DTube (decentralized Youtube), and Akasha (decentralized social network), to name a few. This new movement has inspired the thesis to focus on a decentralized solution using a Distributed Ledger of Blockchain where participatory, remunerative, and liberal systems are constructed where people own, secure, monetize, and leverage the data they produce without any influence.

Figure 1.1: Thomas More's Utopia
[4]

1.2 Mobility Utopia

In work [5], the authors discuss the future of Mobility, where Mobility as a Service with community participation is the future utopia. It necessitates that each of the existing independent systems cooperate at the infrastructure and communication level for a particular transport zone. This aggregation will enable cities to be resilient and improve their social, economic, and innovative capabilities. A recent planning transformation proposed for the beautiful city of Paris is a *15-minute city* where proximity and sustainability are prioritized. Creating new mobility services enables the city to be sustainable, streamlined, and flexible. Other European cities of Milan, Amsterdam, and Copenhagen have imbibed this. It aims to offer a multimodal solution comprising electric bikes, scooters, and vehicle sharing, enabling a quick micro-mobility infrastructure. The decentralization community has understood these synergies well, which has created standards for a decentralized cooperative autonomous vehicle and infrastructure [6]. They have discussed the importance of autonomous systems where machines can interact among themselves to negotiate a transaction in an ecosystem. Blockchains have been explored as a core technology for Mobility as a service in several works [7],[8]. They consider blockchain to create a neutral, commonground infrastructure where the interest of each stakeholder in a consortium network is respected. Also, the possibility of shared Mobility is explored using blockchain [8], which offers a platform for the car-sharing and leasing process in a consortium setting of carsharing providers, leasing companies, and insurance providers. They assure the user and participant of security, privacy, authenticity, traceability, scalability, and interoperability, which centralized systems lack. Our thesis will further explore these impressions as we experiment with novel architectures around the current generation of mobility systems and overcome their challenges.

1.3 Sustainable Development Conundrum

Economies worldwide are on the resurgence after the COVID-19 pandemic has struck havoc, and they are embracing a green economy for sustainable and inclusive growth. This aligns with the initiative of the United Nations and World Bank *build back better* considering Sustainable Development Goal 13 to 'Take Urgent Action to combat climate change and its impacts.' Distributed Ledger Technology has proved significant as a building block in this sustainable green economic megapolis we envisage constructing. European Commission has recognized the blockchain-based solution for incentivizing actors to reduce carbon footprint, crowd-based climate financing, and supply chain solutions. Also, the use-cases of car-sharing solutions, electric charging, vehicle platoon management, and vehicle communication enabling a reduction in greenhouse gases have already adopted blockchain, which expresses the affirmed belief in it for a sustainable technology adoption [9, 10]. In our thesis, we explore creating a sustainable blockchain-based mobility solution and improvising the blockchain consensus algorithm, which is usually energy-consuming, like Proof of Work.

1.4 Distributed Ledger Chronicles

Distributed Ledger (DLT) is a technology that enforces a synchronized distributed data structure of transactions through consensus among a set of participants [11]. Distributed

Ledger Technology process involves transaction validation, verification, and constructing a unit data structure either in the form of blocks or vertices which is subjected to a consensus process. The consensus process is where several nodes or processes agree on the order, validity, and integrity of the data structure unit. There have been several contributions around how this distributed ledger and consensus is designed, which has led to a hierarchically organized structure of technologies that we explore here.

1.4.1 Taxonomy of Distributed Ledger

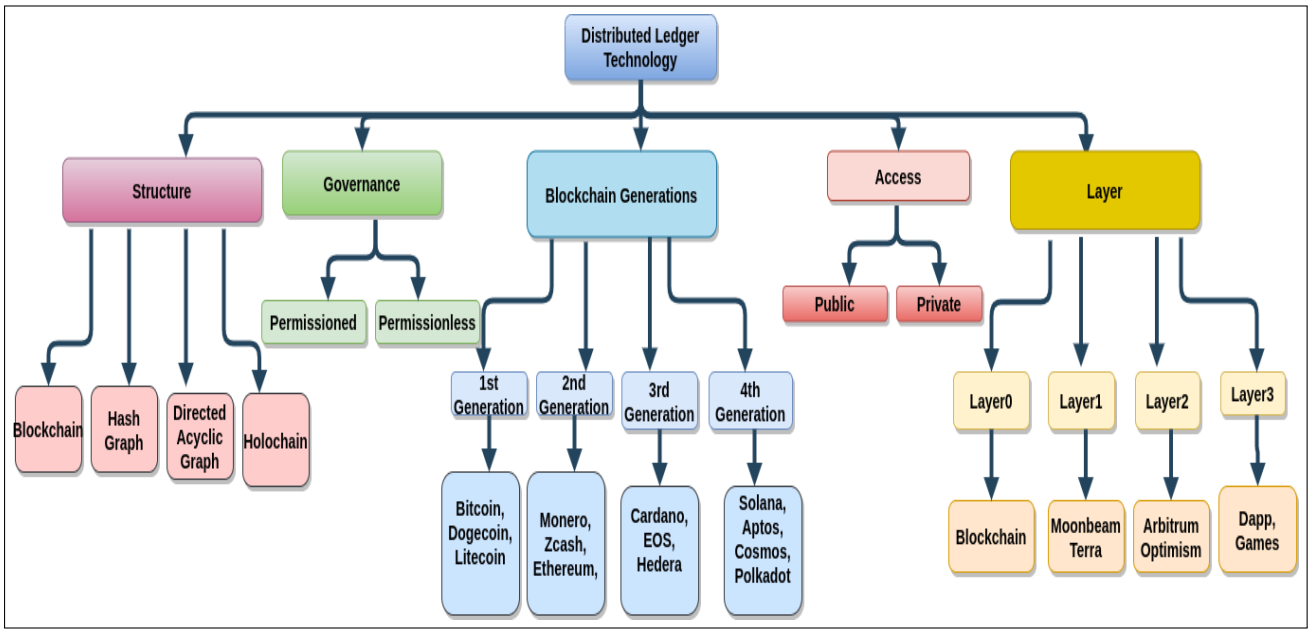

Figure 1.2: DLT Taxonomy and Classification

The taxonomy of Distributed Ledger is multifaceted as it can be studied from multiple levels of hierarchy as represented in Figure 1.2. It is explained as follows:

- **Governance:** Based on the type of organization and the openness of the network for its participants to join and leave, it is classified as follows:
  - **Permissioned:** Participants or Nodes are subjected to governance by a committee either for joining the network, authorization as consensus validators, controlling staking or monetary conditions in the network. E.g., Hyperledger Fabric, Quorum.
  - **Permissionless:** Network is completely democratized with anyone participating or organizing themselves in groups with rights to validate, consensus participation, and derive any monetary benefits. E.g., Ethereum, Bitcoin.
- **Access:** This is similar to governance but only constrained to viewing data, observing, and participating as a recognized account.
  - **Public:** All the accounts can join and leave at will with all the data obvious for audit.

  - **Private:** The account and data access is limited to a certain set of participants by consensus or simple delegation approval.

- **Structure:** Based on the data structure of a DLT, it can be classified as represented

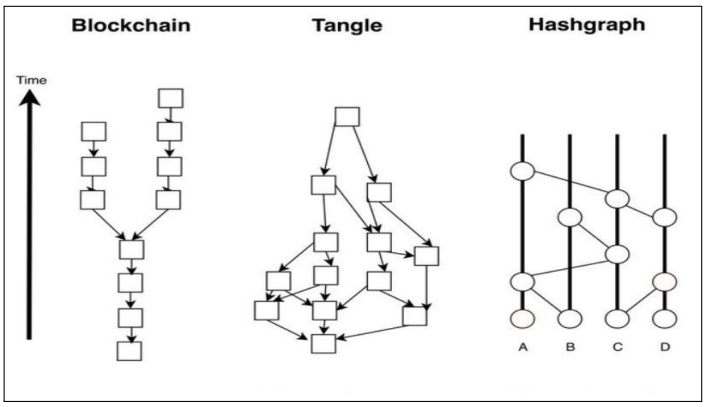

Figure 1.3: DLT Types: Blockchain, Directed Acyclic Graph & Hashgraph [12]

in Figure 1.3 and Figure 1.4 into the following:

  - **Blockchain:** Its structure is similar to a linked list of blocks containing transactions each. Each succeeding block is linked to its predecessor by its hash, establishing a chain.

  - **HashGraph:** Its data structure is composed of columns and vertices as represented in Figure 1.3 where a column represents the participant, and the transaction or events are represented as vertices [13].

  - **Directed Acyclic Graph:** Here, the DLT is a directed graph where each vertex represents a transaction. A vertex is linked to another vertex representing a validation relationship, and it features scalability and faster finalization than blockchains.

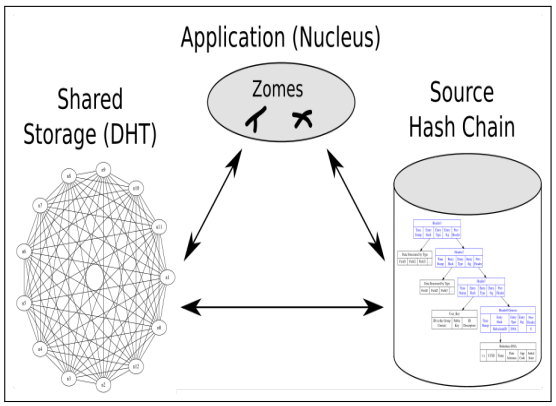

Figure 1.4: DLT Types: Holochain [14]

  - **Holochain** [14]: This type of DLT is similar to blockchain, but each node stores its own individual state instead of storing the common network state. It consists

of three components of Shared Storage, Source Hash Chain, and Application Nucleus (as in Figure 1.4). It is agent-centric, where each node or agent stores its own data in a hashed form to the source chain and simultaneously to the Distributed Hash Table shared storage. Then a distributed validation occurs on the data where every user agrees to the given validation rules.

- **Generation:** Based on the evolution of Distributed Ledger Technology [15] each major outcomes are chronicled as generations as follows:
  - **First Generation:** It signifies the launch of blockchain crypto solutions similar to Bitcoin, XRP, and Dogecoin. It involved the creation of decentralized cryptocurrencies as an alternative to fiat currencies centered around Proof of Work.
  - **Second Generation:** Improvising the first generation by including privacy and smart contracts like Ethereum, Dash, Monero, IOTA, and Zcash is present in the second generation. Many innovations centered around decentralized applications by the creation of Ethereum Virtual Machine and Solidity Language resulted from this.
  - **Third Generation:** There were several problems of scalability centered around smart contracts, which were solved by this generation with the founding of several projects such as Cardano, EOS, Hedera, and Polygon. They had proposed innovations around consensus mechanisms like Proof of Stake, Delegated Proof of Stake, or side chain solutions like Tron or Plasma Bridge.
  - **Fourth Generation:** Features of blockchain to have inter-operability and behave application-oriented were addressed in Solana, Aptos, Cosmos, and Polkadot Blockchains. They exhibit a higher throughput of 70-100000 transactions per second and enable the monolithic chains to communicate with other chains through Inter-Blockchain Communication Cosmos hubs and Polkadot Relay chains.
- **Layer:** To solve the scalability problems, the native blockchain is extended by creating several protocols and infrastructure. It is termed as layer solutions in the form of side-chain, state-channel, and rollups which are discussed as follows:
  - **Layer 0:** It is an infrastructure layer over the original blockchain that solves the interoperability, scalability, and application orientation. It is similar to inter-chain systems like Polkadot, Cosmos, and Avalanche.
  - **Layer 1:** It refers to the underlying or base protocol that can be built over layer 0 using Software development Kits like Terra built on Cosmos or Moonbeam built on Substrate Polkadot.
  - **Layer 2:** It is a scaling solution built over layer 1 to economize gas or processing complexity like Polygon, which comprises zkEVM based on Zero Knowledge-rollups. It also comprises optimistic rollups like Arbitrum One, Boba Network, and loopring.
  - **Layer 3:** This includes an application layer comprising DApps of games, wallets, and privacy based on Zero-Knowledge techniques.

In our thesis, we will focus more on native blockchain network solutions to have the flexibility of adding layered solutions later. We envisage a consortium setting and more primitive consensus decision optimization, as we always have the flexibility to build protocols on top of the solution we conceive.

1.5 Age of Autonomous Applications

Autonomous applications involving finance, automotive, health, or energy need enforced rules and regulations to complete transactions. It also requires an auditable record that is transparent, trustworthy, and resilient to fraud. Decentralized Apps and Decentralised Autonomous Organisation is a brainchild of these necessities agglomerated into one solution built on a blockchain network. It is made using a combination of blockchain (DLT) network, smart contracts, front-end tools, and crypto-currency wallets as well [16, 17]. It is classified as follows:

1.5.1 Decentralised Application (DAPP)

It is an application [18] which is built on distributed ledger and operates autonomously where the community participants control the decisions. The application enforces the condition using a cryptographic token which can be classified based on the underlying layer as blockchain layer 0, layer 1, and layer 2. Examples are Uniswap for trading crypto tokens, OpenSea for selling NFT, and Aave for crypto liquidity protocol.

1.5.2 Decentralised Autonomous Organization (DAO)

In the words of Vitalik Buterin, the founder of Ethereum, the subtle difference between a DAPP and DAO is the difference between decision and autonomous levels. It is termed as an *organization* which operates and takes decisions independently based on moral choices. Examples are Aragon for hosting projects and BitDao for decentralized finance. Our thesis focus is limited to DAPPS, where decisions are taken through consensus operations at the transaction level of smart contracts aided by the token representation of the object we aim to leverage through our architecture propositions.

1.5.3 Smart Contracts

Smart Contract goes back to 1996 when Nick Szabo conceptualized it [19],[20]. Distributed Ledgers starting from the second generation, have created a massive economic space by creating a program that is verified, validated, and replicated on a global state machine. It establishes and enforces the necessary logic for the business system to run a blockchain network through opcode execution on a virtual machine that interfaces with the distributed ledger. All the transactions are interpreted using the virtual machine and verified using the blockchain. It has the following properties:

- **Composability:** It is the ability of a smart contract to interact with other smart contracts deployed in the network to inherit its functionality, data, and properties. It enables the design of modular, autonomous, and discoverable smart contracts.

- **Oracle:** These data feeds enable a decentralized blockchain smart contract to listen through an Application Programming Interface. The source for these oracles is mostly centralized, aggregating data like price feed, weather, or insurance data. For Example, Chainlink, Tellor, and Witnet.
- **MultiSignature:** Normal Smart Contracts require a single signature to execute a transaction, while multi-signature needs multiple participants to agree and sign a transaction. It is to avoid a single point of failure for enhanced security and governance consensus in a private network.
- **Diamond Pattern:** It enables a contract to be upgraded after deployment and organizes contract code and data. It modifies a contract to be multifunctional through facets as well as immutable and reusable.
- **Autonomous:** Massa blockchain [21] has enabled smart contracts to listen to external data, price feed, and execute autonomous functions like bots.
- **Confidentiality:** In this work [22], the confidentiality-preserving smart contract is combined with Trusted Execution Environments (TEE) to separate the consensus layer and execution layer. It enables high throughput and privacy for transactions, improving conventional blockchain systems.
- **Security:** It aims at securing the smart contract from Oracle manipulation where it could be affected by external data providers, reentrancy attack where exploiters call the function repeatedly before the end, frontrunning attack where malicious transactions are executed by bots to extract economic benefits or timestamp dependence is exploited for monetary gains to list a few.

Throughout this thesis, we build our mobility solutions utilizing smart contracts and evaluate them from a functional and performance perspective.

1.5.4 Privacy as a Commodity or Right?

Privacy, where an individual's data from a digital context point of view, is often regarded as a commodity to extract the maximum possible economic advantage. But with specific recent regulations like General Data Protection Regulation (GDPR) [23] and public sensitization of this issue, it has become necessary to consider it a right. With its assurance of data transparency and being publicly verifiable, blockchain is a labyrinth that Distributed Ledger organizations and enterprises should handle carefully. A spate of privacy-preserving solutions has been proposed in the community, which involves the following:

- **Compartmentalization of data:** Decentralized Applications can store personal data such as identifiers and metadata in a distributed database or file system such as InterPlanetary File Storage, Swarm Protocol, or Bigchain DB. Blockchain can store only the necessary data and its hashed versions to secure an individual's privacy.
- **Data Concealment:** Data can be hidden using encryption, hashing, access control techniques, or selective obfuscation to control data privacy.
- **Cryptographic Techniques:** In this mechanism, cryptography primitives are used

to efficiently guarantee privacy as well as correctness and legitimacy of the data explained as follows:

- **Commitments:** It is a non-interactive scheme composed of a *setup*, *commit* and *open* procedure. The setup phase gets an input security parameter, generating a second set of public parameters. The commit phase takes this set of public parameters, a message, and randomness to produce a final commitment. The *Open* procedure takes this commitment as input and gets the commitment. It satisfies two properties of *hiding*, which does not reveal any information, and *binding* guarantees the integrity.
- **Homomorphic Encryption :** This mechanism comprises *Keygen*, which generates keys, message encryption, and decryption. The advantage of this scheme is that the encrypted cipher text can be used to perform additional computation and verification without the necessity of a security key.
- **Zero Knowledge Proofs:** It is a technique that enables agreement between two parties for data in the form of verification proof. Here the proof can be shared with any transacting party without sharing data. It has three phases of *Setup*, which generates public parameters for a given input, *Prove* generates a statement, witness as well as proof, and *Verify* finally utilizes the public parameters, statement, and proof to check its validity.
- **Multiparty Computation:** It allows a certain set of unreliable parties to guarantee and agree over data without revealing the input data. It has two approaches called secret sharing and garbled circuit. It satisfies the correctness of the data shared, the privacy of the data, and the fairness of the process.

In this thesis, we experiment with decentralized mobility solutions, including certain privacy techniques of pseudonymization and cryptographic primitives. Privacy can introduce certain computational effort, which needs an efficient consensus aligning with the objective of the thesis.

1.6 Consensus as a Pace Maker

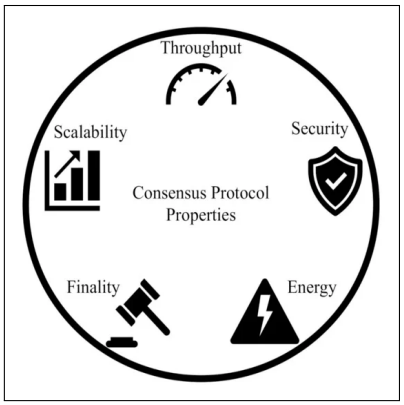

Figure 1.5: Consensus Properties[24]

Consensus in the case of distributed systems is considered 'tricky' [25] as by Fischer, Lynch, and Paterson Theorem [26] in the case of an asynchronous system with one faulty process where no consensus process can be developed to agree on a state of data. On the contrary, consensus is possible even in malicious processes for a synchronous system but is quite costly in processing complexity. This applies in the Distributed Ledger systems case, and the expected properties are represented in Figure 1.5. As a consequence, a consensus needs to have the following [27]:

1. High validation of transactions, construction of transactions into a block, and finalization of the block through a voting mechanism.
2. All the transaction interactions, as well as the data, should be integral, secured as well as tamper-proof.
3. With the consensus involving the message exchange and agreement between the participants, it should be able to make decisions faster or at least the same block validation rate as the members increase in the network.
4. Genealogy of Blockchain consensus starting from Proof of Work has an evident irony of explosive energy consumption[28]. Bitcoin and Ethereum have an energy consumption close to the entire country of Sweden and Romania, respectively. This has expedited the necessity for innovation to create a sustainable future, as signified by the recent report of the European Central Bank. Ethereum, due to this reason, has already changed its consensus to Proof of Stake which is a little more economical, sustainable, and scalable.

These factors testify that the consensus component is comparable to the pacemaker of a human heart system. It must be resistant to forks, avoid centralization issues, and be resilient to byzantine actors. The consensus protocol decides the validation, security, throughput, and scalability in a distributed ledger, which we discuss extensively in this thesis.

Having discussed the essential pillars of the blockchain and its significance, we elicit the broader goal of our thesis from a bird's view and the overall organization of the thesis journey as we embark on this chapter.

1.7 Objectives

Our thesis revolves around the four fundamental questions that act as our holy grail throughout our work as follows:

- **Can Distributed Ledger Technologies (DLT) resolve pertinent Mobility challenges linked to cross-organization synergy, transparency, integrity, incentivization, equity, fairness, security as well as privacy?**
  It aims to understand the current literature state of the art and nascent DLT technologies, which will help us track the evolution of technology from the age of distributed systems until the recent Distributed Acyclic Graph Chains. Also, the various mobility solutions based on DLT will be analyzed along with their advantages and disadvantages for designing our automotive use case. This sets the essential foundation for formulating our hypothesis, which will be explored in forthcoming steps.

- **Can we architect real automotive use-case solutions and, in turn, retrieve the fundamental obstacles that the technology needs to overcome from enterprise or consortium-wide adoption?**
  We define our automotive use cases formally with the network definition, business logic, interactions, value addition of data certification, and monetization. We propose a solution to create newer economic models and solve the issues present in existing works. The architecture is tested on an existing blockchain framework, and the performance is evaluated, especially from the Byzantine Fault Tolerance consensus perspective.
- **Can we identify, minimize, and solve the intrinsic problems present in the current Byzantine Fault Tolerant consensus algorithms?**
  Byzantine Fault Tolerance Consensus algorithms problems are analyzed from the heterogeneous blockchain frameworks to be communication complex, poor scalability, and fault-tolerant of benign and malicious failures. Using the developed simulator, these aspects are further tested homogeneously to avoid bias due to implementation, network, or database factors. The simulator helps to perform the consensus benchmark on a cloud infrastructure and compare them uniformly for further understanding.
- **How do we optimally balance these consensus problems and propose a solution that satisfies mobility solution constraints as well as maturing into real-life distributed ledger ecosystem?**
  The identified issues of BFT consensus are proposed to be solved in a novel developed consensus CUBA, and it is evaluated from theoretical, implementation, and security focal points. This consensus solution is identified to solve the problems and optimize the algorithm's message communication, security, scalability, and fault resilience.

1.8 Thesis Organisation

Our thesis work is organized as a Road Timeline (Figure 1.6), representing six major steps discussed across seven chapters.

- **Chapter 1**: This contains an introduction to the basic concepts of the necessity of decentralization, new generation mobility services, sustainability, distributed ledger evolution, smart contract features, autonomous applications of DAPP and DAO, privacy constraints as well as the importance of consensus algorithm.
- **Chapter 2**: State of Art covers the literature and existing works across data certification and mobility services. It also extensively discusses the foundations of distributed consensus and improvements of parallelization, Quorum-based protocols, Proof of Authority, Reputation as a measure in the consensus, and the Byzantine Altruistic Rational Model.
- **Chapter 3**: It covers Decentralized Mobility Services implementations involving our two central mobility use cases of data certification and monetization. It touches upon subjects of token engineering, Proof of Authority consensus of Clique, PBFT (Practical Byzantine Fault Tolerance), IBFT (Istanbul Byzantine Fault Tolerance), and

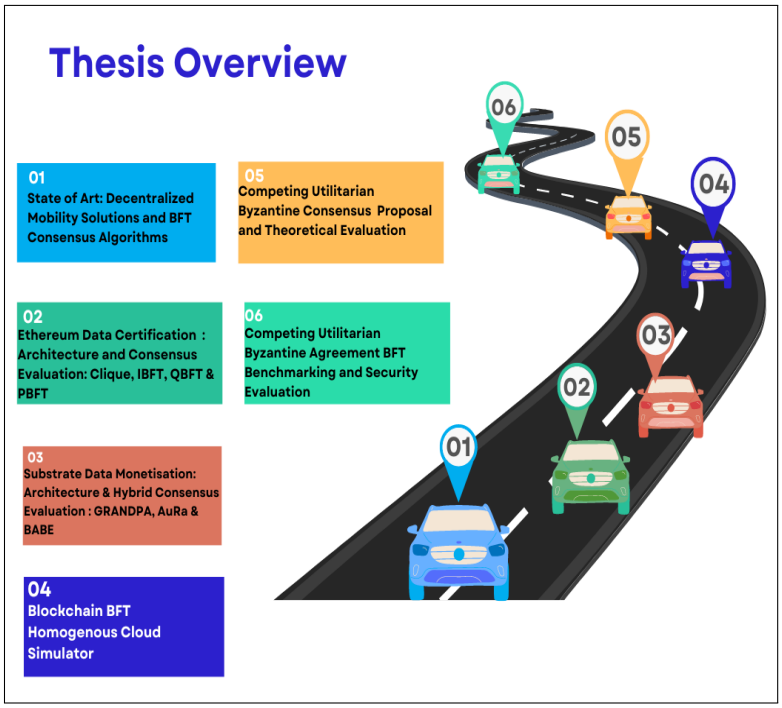

Figure 1.6: Thesis Overview

QBFT (Quorum Byzantine Fault Tolerance). Evaluation of the functionality and performance of the proposed architecture with a special investigation on consensus is carried out here.

- **Chapter 4**: This chapter introduces the need for creating a homogeneous blockchain simulator for evaluating the BFT algorithms. It details the design implementation and evaluation of the simulator. The simulator is deployed on a cloud infrastructure and tested for the classical blockchain consensus algorithms of Clique, IBFT, QBFT, and PBFT for unbiased analysis.
- **Chapter 5**: CUBA - An Evolutionary Consortium Distributed Ledger Byzantine Consensus Algorithm is proposed to minimize the drawbacks of Byzantine Fault Tolerant Algorithms from a consortium perspective which has the prominent features of pipelining, Intra-Quorum, Inter-Quorum, Utilitarian score measure, Panopticon Complement, Sisyphus Forgetting Coefficient, Swarm Utilitarian Classification and finally network optimization.
- **Chapter 6**: The proposed CUBA Consensus protocol is evaluated theoretically from Algorithm complexity, CAP Theorem, Blockchain Trilemma Analysis, Fairness, and adverse scenarios. It is subjected to further experimental evaluation to benchmark against existing consensus protocols, and security aspects of different scenarios are profoundly analyzed.
- **Chapter 7**: The conclusion and perspectives chapter resumes the findings across the entire breadth of our work and gives a final overview. It also offers future directions to enrich our work further and add value. We evaluate our thesis work from the strengths and opportunities perspective for further insight discussion.

- **Chapter 8**: The appendix contains interesting results, algorithms, and analysis which we displace here as we have a limited thesis presentation as well as to maintain the brevity of the thesis.

Chapter 2

State of Art

I have long felt that, because it was posed as a cute problem about philosophers seated around a table, Dijkstra's dining philosophers problem received much more attention than it deserves... The popularity of the dining philosophers problem taught me that the best way to attract attention to a problem is to present in terms of a story

– Lamport

This chapter traces the work in Blockchain-enabled Mobility Services for fully connected vehicles. In close accordance is the study of distributed consensus algorithms that have existed since the period of classical Distributed Systems until the present Distributed Ledger era.

2.1 Mobility Services

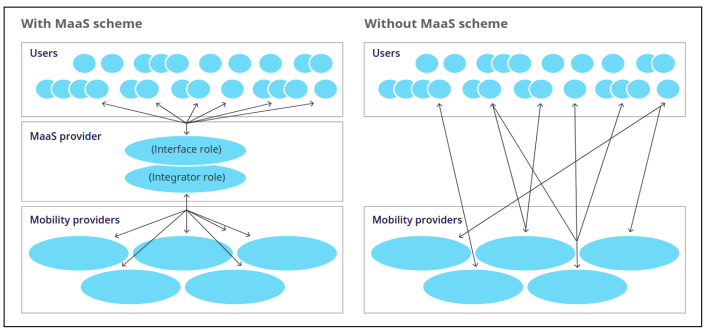

Figure 2.1: Relation with Mobility Stake Holders: With and Without MaaS Scheme [29]

A report by Bianchi Alves, Bianca and Wang, Winnie and Moody, Joanna and Waksberg Guerrini, Ana and Peralta Quiros, Tatiana and Velez, Jean Paul and Ochoa Sepulveda, Maria Catalina and Alonso Gonzalez, Maria Jesus of World Bank [29] on the adaptation of *Mobility-as-a-Service (MaaS) for Developing Cities* defines the concept of Mobility-as-a-Service as an integration of different modes of transport, including both informal modes like public transport, taxis, biking, cycling and formal methods like ride-sharing, micro-transit enabled with Information and Communication Technology (ICT) services. It also highlights the importance of MaaS for cities in low and middle-income countries that advance sustainable mobility and development goals. It demonstrates the policy objectives of Universal Access, Efficiency Safety, and Green Mobility under *The Global Roadmap of Action towards Sustainable Mobility*. The importance of the MaaS provider is also highlighted in this work as in Figure 2.1. MaaS provider is an intermediate layer between users like the public and the transportation or mobility providers, on the other hand. They have two beneficial critical roles:

1. Integrator Role among the Mobility service providers by creating an open platform for the mobility service provider participation.
2. Interfacing Role for users in creating a unified platform for accessing and managing trip information or booking and payment.

Apart from these two roles, the other benefits of having direct contact with the users are creating a sense of loyalty, analyzing comprehensive insights into users' preferences, needs, and willingness to pay for different modes, as well as influencing the demand through pricing and design strategies. The impediment in fully implementing the MaaS providers is that traditional mobility service providers are unwilling to forego their direct relationship with the users and instead rely on the MaaS provider for facilitation. This results in overcoming the tensions between Mobility Service Providers and MaaS Providers by creating a better value proposition than the existing [30].

Also, the study by de Wilde, Thijs [31] highlights the challenges in implementing MaaS which needs a multistakeholder cooperation of profit-based private and service-based public actors. The work highlights the risk of Brand Image, Operational Cost, Inclusion, Entry barriers, and Data Ownership in traditional MaaS systems. It pictures the blockchain as a facilitator for the MaaS platform as it promotes inclusivity, shared responsibility, incen-

tivization, and collective ownership of data. We will further analyze the existing implementations around the Mobility Services involving Distributed Ledger, and Blockchain, to understand their advantages and open questions.

2.1.1 Distributed Ledger Technology (DLT) Enabled Mobility Services

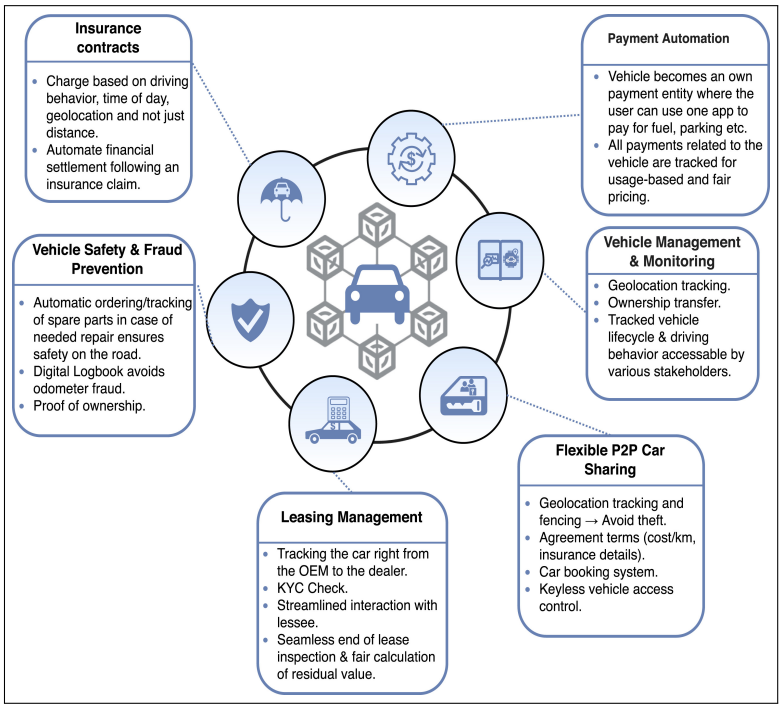

Figure 2.2: Conceptual Design of P2P Car Sharing and Leasing Platform [32]

In [32] by Auer, Sophia and Nagler, Sophia and Mazumdar, Somnath and Rao Mukkamala, Raghava, a high-level architecture for blockchain-IoT-based platforms for shared mobility, such as car-sharing and car-leasing, is presented. They demonstrate that an ecosystem of actors such as Original Equipment Manufacturers (OEM), ride-sharing providers, insurance providers, and public authorities can provide smart mobility via this platform. This work aims to realize the six use cases represented in Figure 2.2 of P2P car-sharing, leasing management, vehicle management and monitoring, payment processing or automation, vehicle safety, fraud prevention, and finally, insurance contracts. They also evaluate the platform's fundamental design principles of security, privacy, authenticity, reliability, scalability, and interoperability. Their work involves the creation of a prototype for a keyless vehicle access control for a car-sharing platform and a collection of Internet of Things (IoT) data from the vehicles streamlining car sharing and leasing using Hyperledger Fabric Blockchain.

A Cyber-Security risk assessment framework [33] is designed by Mallah, Ranwa Al and López, David and Farooq, Bilal for blockchain applications concerning Smart Mobility Datamarkets. They analyze the impact of each risk from monetary, privacy, integrity, and trust perspectives uncovering the security vulnerabilities in the ecosystem. Their study was

performed on a Blockchain framework for Smart Mobility Data-markets providing Distributed Mobility Information Management System implemented using Hyperledger Iroha Blockchain. They evaluate the system's risk using three steps: Actor-based risk analysis to extract the impact, Scenario-based risk analysis to extract the probability, and Combined Risk assessment to quantify the risk. They classify False Data Injection, Infrastructure Attacks of the Blockchain, and Denial of Service as unacceptable risks that must be mitigated with high priority.

Not limited to road or rail smart mobility, the work in [34] by de Oliveira, I. Romani and Matsumoto, T. and Neto, E. C. Pinto presents blockchain-based traffic management for advanced air mobility. The system enables distributed airspace allocation management and conflict resolution through smart contracts. Existing centralized system risk of system outages is avoided by peer-to-peer conflict resolution bringing more resilience to the ground communication infrastructure. However, the system does not consider the actors' uncooperative behavior or malicious data broadcast in the centralized system, which is a security concern for traffic management considering it to be high risk prone.

The backbone of Smart Mobility is the underlying communication system which has witnessed an unprecedented surge in data traffic, causing security, privacy, and trust challenges in [35] by Ayaz, Ferheen and Sheng, Zhengguo and Tian, Daxin and Nekovee, Maziar and Saeed, Nagham. Their work proposes integrating Federated Learning and blockchain, where the onboard Unit in each vehicle of the ecosystem can collect, process, and train a local prediction model. These models are then verified using a consensus mechanism through blockchain and then agglomerated to create a unified model for detecting traffic or road conditions, thereby rationalizing the network usage as well as decentralizing the system.

A blockchain-based MaaS system improving trust and transparency by avoiding individual commercial agreements with multiple stakeholders with agents is presented in [36] by Nguyen, Tri Hong and Partala, Juha and Pirttikangas, Susanna. The network of different modal transportation providers creates transparency and flexibility for the users accessing the system. The work evaluates more from a literature work and misses the protocol and the implementation details, which opens specific questions.

A new approach to mobility of shared ownership is proposed by MobiToken [37] by Gong, Shuangqing and Hossein Chinaei, Mohammad and Luo, Fengji and Hossein Rashidi, Taha, where an eMarket (Electronic Market) is constructed for commuters, mobility service vendors, and other stakeholders. The token's price varies according to the demand and the congestion of the mobility service. The solution is implemented using Raft [38] consensus algorithm and evaluated using multiple test scenarios. The design of eMarket using Hyperledger Fabric enables the creation, exchange, and utilization of MobiToken without double-spending. The liquidation of tokens being tied proportionally to the current traffic status improves the efficiency of the traffic management system and reduces the congestion of the traffic network system.

In the work of [39] López, David and Farooq, Bilal, a multi-layered Blockchain for the Smart Mobility data market addresses privacy, security, management, and scalability challenges in conventional centralized systems. They test the simulated data market in a 370-node

blockchain hosted on heterogeneous, geographically separated devices communicating on a physical network. The network comprises Individuals, Transport Agencies, Census Bureaus, Planning and Development Agencies, Universities, and Private Enterprises to pool their data and share it with each actor through a blockchain layer. The market is implemented using Hyperledger Indy and deployed on Amazon Elastic Compute Cloud with varied node sizes to test the scalability. The system is built on the principles of Data Ownership, Fine-Grained Access Control, Data Transparency, and auditability, ensuring privacy and giving users the power to control and profit from their transportation data. The system also ensures protection against Data Interception, Data Leaks, Unsolicited sharing of information, and Unsolicited Requests of Information.

2.1.2 Mobility Data Certification Services

In this sub-section, we discuss the works of DLT-enabled data certification solutions for mobility and other allied works.

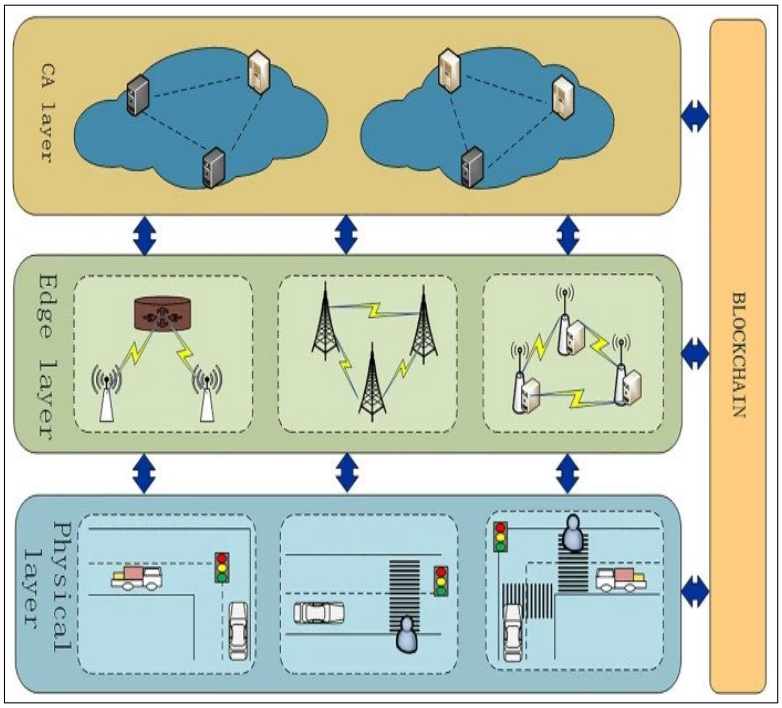

Figure 2.3: Overview of Internet of Vehicle Architecture [40]

A Blockchain Vehicle Certification Scheme in the Internet of Vehicles (IoV) using an identity verification scheme is designed in [40] by Wu, Junhua and Jin, Zhenyu and Li, Guangshun and Xu, Zhuqing and Fan, Cang and Zheng, Yuanwang. A blockchain-enabled framework is utilized to store private vehicle data to overcome the safety and reliability issues with the existing centralized systems. The resilience of the system against a single attack requiring the attacker to control half of the network's computing power is the key benefit of choosing this framework. The public entities, as illustrated in Figure 2.3, are partitioned into the Physical layer, Edge layer, and Certification Authority (CA) layer. The physical layer comprises connected vehicles, sensors, and pedestrians for receiving and submitting data. The

Edge Layer comprises the server, which manages data transactions forwarding the information from the physical layer to the CA layer, and finally, the CA layer, which provides the registration service of the IoV, ensuring the verification of vehicles. The vehicle certification process involves the Information Initialization phase, Internet of Vehicle Equipment Registration phase, Vehicle Identification, and Regional Consensus. The data shared by the vehicle is validated and then written to the blockchain using a Practical Byzantine Fault Tolerance Algorithm. Also, a Multi-vehicle cooperation algorithm for task scheduling among the idle vehicles in the ecosystem is proposed to utilize the shared vehicle resources. The system is evaluated across confidentiality, anonymity, and traceability parameters.

A systematic literature review [41] of blockchain in the automotive industry for autonomous vehicles, smart charging, smart manufacturing, automotive logistics, and counterfeiting is discussed in this work by Gîrbacia, F and Voinea, Gheorghe and Boboc, Razvan and Duguleană, M and Postelnicu, Cristian. The work discusses the problems faced in the case of autonomous vehicle roll-out, including proof of liability and forensic data storage which is solved by blockchain systems. Also, resilience against denial-of-service, Global Positioning System (GPS) spoofing, timing, or man-in-the-middle attacks by the blockchain solution is discussed. The rise of electric vehicle adoption backed by the European Commission legislation to end the sale of new Carbon dioxide emitting or ICE vehicles [42] has aggravated the problem of a limited number of charging stations. To solve this issue, blockchain as a solution is discussed. It motivates Electric Vehicle (EV) users to share their energy surplus through trading and managing charging infrastructure by developing a Peer-to-Peer (P2P) framework for private charging pile-sharing systems.

A blockchain-based middleware solution called Man4Ware [43] by Mohamed, Nader and Al-Jaroodi, Jameela brings in the advantage of smart manufacturing by the integration of Industry 4.0 concepts [44] [45]. The concepts are the Interconnection of machines via IoT, Information Transparency for data collection, Technical Assistance for assisting humans, and the last being Decentralized Decisions which allow the cyber-physical systems to make autonomous decisions from gathered data.

A Blockchain-aided vehicle certification through a secured E-Governance framework was proposed in [46]. The authors Das, Moonmoon and Azad, Rahat Uddin and Efat, Md. Iftekharul Alam discuss the importance of Digital Certification for Digital Transformation in every business infrastructure [47], ensuring security and traceability along with confidential communication. This framework was proposed to streamline vehicle verification by public officials and vehicle certification complexity enabling collaboration with all stakeholders in a distributed nature. The framework involves the government, vehicle owners, and other vendors for secured communication, data storage, and interoperability based on the blockchain through block binding and digital ring signature techniques. A framework prototype using Ethereum technology was developed with a pilot database to simulate actors' interaction and functionality securely. The system is evaluated by load testing and response time monitoring for robustness, scalability, and adaptability. They also understand the security implication of the framework based on **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and **E**levation of Privilege [48] *STRIDE* classification.

The authors Chanson, Mathieu and Bogner, Andreas and Wortmann, Felix and Fleisch,

Elgar in [49] have proposed an Ethereum public blockchain solution using a database-blockchain approach to certify the vehicle odometer data and prevent its fraud or manipulation in used vehicles. After each trip, a vehicle transmits the following encrypted data to the NoSQL database: Global Positioning System (GPS) trip co-ordinates, timestamp of the trip, random nonce & odometer value; simultaneously, plain raw data is hashed and stored in the blockchain. During the time of issuing mileage value certification, cross-verification of the hash from the blockchain against the calculated hash from decrypted raw data in the database is done. This verification of the data from the blockchain hash against the data stored in traditional databases enables the certification process of the state of the data at a certain timestamp.

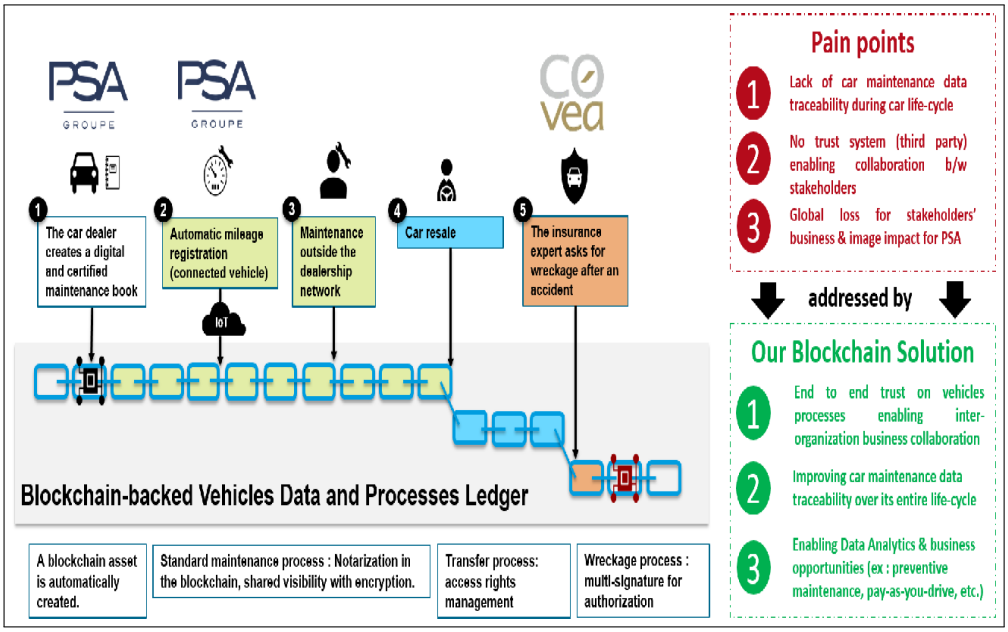

Figure 2.4: Digitizing Vehicles Life-cycle over a Consortium Blockchain: Pain Points and Benefits of the Framework [49]

The authors solve a slightly different use case in the following work, [50]. They proposed a blockchain-backed vehicle data and process ledger framework to manage the vehicle life-cycle history as represented in Figure 2.4. The framework records the new vehicle creation by a maintenance book, the mileage automatic registration, the vehicle's maintenance inside or outside the dealership network, and car resale, which utilizes the maintenance book information already recorded, avoiding fraud. The accident case is covered by allowing the insurance providers to access the vehicle data and evaluate the damages to award adequate compensations using verified data. This system allows collaboration between multiple stakeholders in the network at a consortium level using Quorum, a fork of Ethereum. Anonymous users are avoided as it is a permissioned blockchain network. However, their consideration to use a centralized database managing the total blockchain keys and network actors' passwords sacrifices decentralization. Also, functionalities of the shared and entity services are handled through a traditional web service-based system pattern which can be modeled inside the blockchain. Using an asymmetric key, the authors proposed a novel hybrid cryptographic protocol to solve data privacy issues. Distribution of the key in a multi-participant scenario can be a problem for production scalability.

A combination of distributed ledger technologies (DLT), such as Blockchain and InterPlanetary File Systems (IPFS), was presented in [51]. In this work, an accidentology use case where the vehicle sends data to the DLT Layer in case of a mishap is considered. An IoT authentication protocol is developed to identify the device and a Raspberry Pi 3B+ model where the SHA function of the standard CryptoPP C++ library is optimized for the ARM processor. The work was implemented on Hyperledger Sawtooth using the Practical Byzantine Fault Tolerance consensus algorithm and the modified Broadcom BCM2837 Model to reduce hash processing time. The system was further measured for transaction commit rate and latency in the Internet of Things (IoT) communication as a variation of the payload.

2.1.3 Mobility Data Monetization Services

In this sub-section, we explore the literature around data trading services using blockchain, which discusses data trading, and sharing data securely respecting the user's privacy concerns.

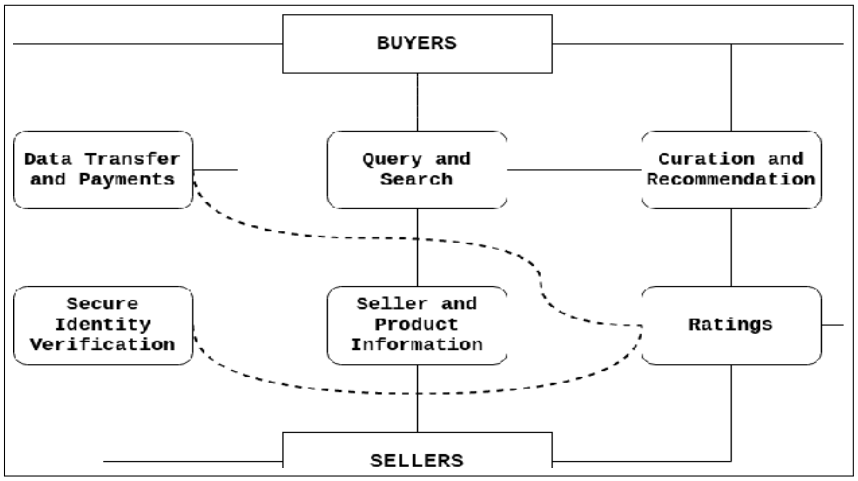

Figure 2.5: Decentralised Review system for Data Marketplaces [52]

A Decentralized Review System for Data Marketplaces [52] hosting urban quality of life data, vehicle data, and other IoT data is designed to replace the conventional centralized rating systems. The Decentralized Data Marketplace consists of the critical elements outlined in Figure 2.5 wherein the buyers and sellers of data interface through the marketplace for data transfer, buying, querying the data, reviewing the data, and providing ratings to the data. The system allocates pre-determined credible reviewers against a set of products in a randomized and double-blinded method to minimize collusion of like actors. Reviewers are incentivized through game theoretical modeling, where they identify the conditions for Nash Equilibrium policy for the reviewers. Nash Equilibrium policy aims at finding an optimal solution to a problem with competitive participants. The system is presented as a general framework and analyzed theoretically with no decentralized platform implementation. The presented work is simulated using NashPy Library, where they identify that as long as sufficient incentives increase the probability of a full review, it enhances the review quality. But there are still some pertinent questions unanswered in their work: 1) The Formation of a review committee when the platform is still new and the absence of reviewers' credentials, 2) The strategy to review a product continuously or at a determined

time-frequency 3) How can the seller's privacy be preserved behind the transaction process and 4) The scalability potential of the system with increasing products, reviewers, and sellers.

A credible data trading system for IoT data minimizing the risk of fraud and transactions is proposed in [53] by Meijers, James and Dharma Putra, Guntur and Kotsialou, Grammateia and Kanhere, Salil S. and Veneris, Andreas. The system permits the data producers and consumers to agree on data and settle the payment on the chain. A credit mechanism is developed to lower the fees incurred during the private Ethereum network implementation participation process. This system's objectives are Consumer Fairness, where customers do not pay for data not received; Producer Fairness ensuring the minimal risk of data loss; and Privacy, limiting data visibility at a minimal operational cost. The system is evaluated regarding gas consumption and incurred blockchain transactions for the trading scheme, as the idea is to minimize the transaction cost related to transaction gas complexity. They estimate around 35000 units of gas for fund deposit transactions and higher for receipt transactions as it needs data signature checking.

A consensus-based distributed auction scheme for data sharing is proposed in [54] for privacy preservation and avoiding malicious collusion of actors. The scheme allows the participants to group into clusters for privacy and then reach a consensus. The mechanism is further incentivized to share data without privacy leakages. Differential privacy, symmetric encryption, and zero-knowledge proofs are incorporated to design the auction mechanism for a trade-off between privacy preservation and social efficiency. The consensus algorithm is constructed where different kinds of witnesses are selected using anonymous verifiable random functions. It is performed without peer interactions and different parallel operations by varying witnesses to verify the proof and result, ensuring finalization. The evaluation of the system shows that the participants reach a consensus on the auction result with low computation and communication costs.

A novel proposition of data marketplace design that satisfies all desirable properties in any system of fairness, efficiency, security, privacy, and regulation adherence is proposed in [55]. They implement the FairSwap [56] to guarantee fairness where the participants agree on a value wherein a Proof of Misbehavior is generated to punish in case of wrongdoing. They rely on any generic encryption and hash function, including Zero-Knowledge Proofs, for transparency, security, and privacy. They mention using codified language, such as smart contracts, for the regulation aspect. To maintain efficiency, they suggest the usage of Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (SNARKs) or Merkle tree over boolean circuits to analyze encrypted data. The work is a more theoretical discussion rather than a concrete implementation with particular work on its challenges. The uncovered inconveniences are global fairness issues where they cannot guarantee fairness in a larger context if more participants of varying interests participate in the ecosystem. Another is the inefficiency of predicate checking in encrypted data for arbitrary logic. Their incentivization mechanism is not theoretically evaluated, and capturing practical constraints in the framework is complicated. Also, data duplication is quite a problem, and they suggest using convergent encryption without revealing any data information.

Since the blockchain system poses a problem of whether to be transparent and at the same time not reveal data ensuring privacy, the work in [57] by Koutsos, Vlasis and Papadopou-

los, Dimitrios and Chatzopoulos, Dimitris and Tarkoma, Sasu and Hui, Pan presents an answer to this confusion by designing a Privacy-Aware Data Marketplace. They try to ensure: data privacy, atomicity of payments, where no entity can deny paying for services, and output verifiability, where the data exchanged are verified before being sold to consumers. They propose a functional encryption scheme with an additional public parameter for data privacy and Zero-Knowledge Proof for output verifiability. The payment system is facilitated by an implementation over Ethereum where data generation, collection, and brokerage may happen sequentially and independently at any time. During the data generation phase, each producer creates data and further encrypts it into cipher text using the Multi-client functional encryption [58], an adaptive, secure Functional Encryption scheme. It enables multiple clients to produce cipher text individually and independently without any interaction, which is a crucial criterion. Zero-Knowledge Protocol ensures data brokerage and agreement of data correctness. The system does not allow for dynamic change of data generators. Also, they acknowledge that there is certainly an overhead with their Multi-client function encryption scheme, which needs to be improved.

A consortium Blockchain-based vehicle data marketplace, a data sharing scheme, and data-owner-attribute-based encryption are proposed in [59]. They target data confidentiality, integrity, and privacy and securely handle large-capacity and privacy-sensitive vehicle data by combining on-chain and off-chain methods. The metadata of the data exchanged is stored on Blockchain, and encrypted data is stored on off-chain storage. Also, the data owners can control their data by applying attributed-based encryption and owner-defined access control lists (ACL). The idea behind attributed-based encryption allows message-level granular control over the encrypted data. It is divided into fundamental policy and cipher text policy, where the algorithm distributes a random value according to a linear secret sharing scheme access matrix. The private keys are randomly assigned to avoid collision attacks. The encryption algorithm [60] uses three parties: private key generator, decryptor, and encryptor. According to the cipher policy, the private key generator is responsible for the secret key generation of the decrypter. The encryption scheme differs in the generation of secret keys by the encrypter instead of the key generator. This allows for achieving message-level-based fine-grained control, the key-escrow-free property, and straightforward message recovery for the encrypter, which are desirable in cloud storage. However, the proposition is not implemented and lacks concrete implementation evaluation and test results.

A Big Data Value Chain framework [61] is proposed by Faroukhi, Abou Zakaria and El Alaoui, Imane and Gahi, Youssef and Amine, Aouatif for data monetization to enable the processes in the organization to be entirely data-driven, support decision-making, and facilitate value co-creation. They propose the value chain framework phases starting from data generation, acquisition, and pre-processing, which involves filtration, extraction, transformation, validation, cleaning, fusion, reduction, linking, aggregation, denormalization, data storage, data analysis, data visualization, and then finally data exposition. Their idea has been to propose two data monetization models that can be integrated into the above phases of the value chain process: the Reduced Big Data Monetization Model and the Full Big Data Monetization Model. The former model is a reduced form to monetize data only for storage and visualization phases. In contrast, the latter is more generic and monetizes the whole value chain framework through data and insight sharing. This approach is evaluated us-

ing a simulation of the geolocation data of the trucking company, such as location, event, speed, and mileage. The value chain framework is implemented on Cloudera distribution based on open-source Apache Hadoop.

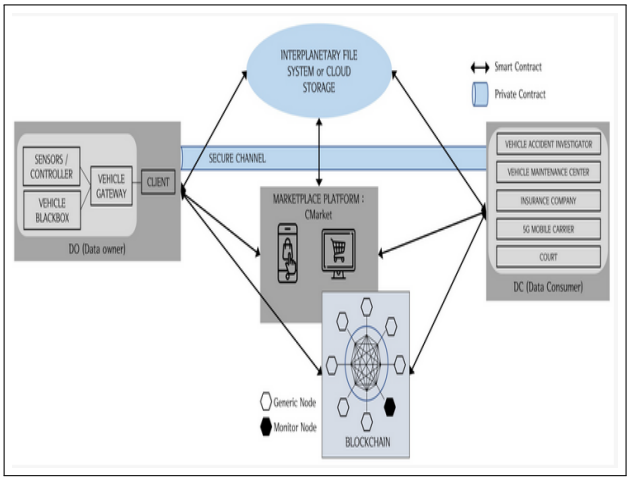

Figure 2.6: Blockchain Based Data Marketplace [62]

An IoT data marketplace on Blockchain facilitating transparent data exchange and access to consented data is modeled in [62]. The system is represented in Figure 2.6, which consists of a data owner (DO), data consumer (DC), blockchain network, marketplace platform, and external storage such as IPFS. The data owner provides the vehicle data, the consumer then uses the data, and the service providers act as an interface between the data owner and consumers. The implementation is made on the Ethereum blockchain and Swarm distributed storage platform for data off-chain storage. The IoT devices can communicate through an IoT Gateway device for registration to the Ethereum blockchain. Then it can store the encrypted data in the Swarm storage followed by pushing the metadata of IoT device type, Swarm URL (Uniform Resource Locator) onto the blockchain. The Artificial Intelligence (AI) Provider can query the blockchain for data, which notifies the IoT device to provide key access for the concerned data. Upon the response of the IoT device with the key, the AI provider can extract the data from the Swarm and decrypt it to use the data for prediction algorithms. However, the marketplace is not proposed for time-critical services as well as not privacy preserving nor General Data Protection Regulation [23] (GDPR)

2.2 Distributed Consensus

In this section, we discuss and understand specific relevant works behind the foundation of distributed consensus from the 1970s until recent times. This exercise is motivated to show the importance of each consensus and the scenarios of agreement and failures associated with each work.

2.2.1 Foundational Works

Distributed systems have been built since the 1970s [63] ranging from Advanced Research Projects Agency Network (ARPANET) [64] by the United States Department of Defense, the

Society for Worldwide Interbank Financial Telecommunication (SWIFT) protocol as well as the Software Implemented Fault Tolerance (SIFT) [65] project by National Aeronautics and Space Administration (NASA) for a Fault-Tolerant Aircraft Control system. Distributed systems have undergone extensive studies ranging from concurrency, failure recovery, and naming but are pretty hard to design. There have been the famous eight fallacies of distributed computing defined [66] which are: *1) the network is reliable 2) Latency is zero 3) Bandwidth is infinite 4) the network is secure 5) Topology doesn't change 6) there is one administrator 7) transport cost is zero and 8) the network is homogenous.* These mistaken reasonings are impossible as 1) systems are prone to hardware failure, 2) bandwidth is always minimal as even with new network technologies, applications are designed to be data hungry, 3) the increase of malware and ransomware attacks by 358% and 435% in 2020 [67] as noted by World Economic Forum supports that the above assumptions are indeed blunders to avoid.

The following subsections explain the seminal or classical foundations upon which the current distributed system exists. In all these explanations, we consider a system of **n** actors or replicas or nodes as in a blockchain system, out of which we have **f** malicious or failing participants.

2.2.1.1 Seminal Synchronous Solution: Byzantine Generals Problem

The first seminal work [68] in distributed systems was done on a formal request by NASA to build a robust avionics control system that needed formal guarantees as it was mission-critical. They had to prove the correctness of a cockpit control scheme composed of three computer systems that NASA had designed and their fault tolerance ability. This led to the identification of robustness and coordination problems in any replicated set of systems revealing the impossibility of a safe run for a mission-critical system. This led to two works [68, 69] where they identified a vulnerability as "a failed component may exhibit a type of behavior that is often overlooked –namely, sending conflicting information to different parts of the system." This fault model coincided with the works presented in [70], [71] coined as "The Two General Paradox," which led to the thinking by authors in [68] to personify a system of an avionic control system to an army of Byzantine Generals. The army has to succeed in coordinating an attack despite the traitors' malicious behavior representing the modern security flaws and failures that can occur in distributed systems.

The problem is represented in Figure 2.7 in which multiple army divisions are each headed by an individual general camp outside an enemy city. All the n generals in the army should either decide to attack a fort unanimously or retreat, abiding by the condition to communicate only by messenger. A general has the possibility of being a traitor as well. Each general must pass orders to the remaining N-1 generals in respecting the two Interactive Consistency conditions (IC) wherein the successor condition is derived from the previous condition. These conditions are:

- All loyal lieutenants obey the same order
- If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.

The solution to this scenario would have 3m+1 honest general tolerating, at most m, traitor generals, which are discussed as follows.

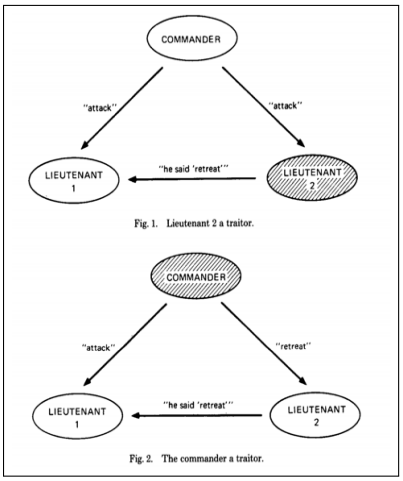

Figure 2.7: Byzantine Generals Problem [68]

2.2.1.1.1 Oral Message Solution A solution by plain oral message communication is proposed, which has three assumptions as follows:

1. A1 - Every message that is sent is delivered correctly.
2. A2 - The receiver of a message knows who sent it.
3. A3 - The absence of a message can be detected.

Assumptions A1, A2, and A3 prevent a traitor from interfering with the communication between two other generals or simply cannot block any message. RETREAT is the default order. The Oral Message Algorithms OM(m) for non-negative integers m are presented here Algorithm 1 for 0 traitors and Algorithm 2 for m traitors.
The Algorithm 2 assumes a functioning majority of the values v_i equal v otherwise, the

Algorithm 1: An algorithm OM(0) with no traitors, m=0
Data: $m = 0$
Result: ATTACK or RETREAT UNANIMOUSLY
1) The commander sends his value to every lieutenant.
2) Each lieutenant uses the value he receives from the commander or RETREAT if he receives no value.

value RETREAT. Otherwise, the median of v_i assumes they are from an ordered set.

Algorithm 2: An algorithm OM(m) with m traitors

Data: $m \geq 0$

Result: ATTACK or RETREAT UNANIMOUSLY

1) The commander sends his value to every lieutenant‘.

2) For each i, v_i be the value lieutenant i receives from the commander or RETREAT if he receives no value. Lieutenant i acts as the commander in Algorithm OM(m-1) to send the value v_i to each of the n-2 other lieutenants.

3) For each i, and each $j \neq i$, let v_j be the value Lieutenant i received from Lieutenant j in step (2) (recursively using Algorithm OM(m-1)) or else RETREAT if he received no such value. Lieutenant i uses the value majority(v_1,..,$v_{.n-1}$)

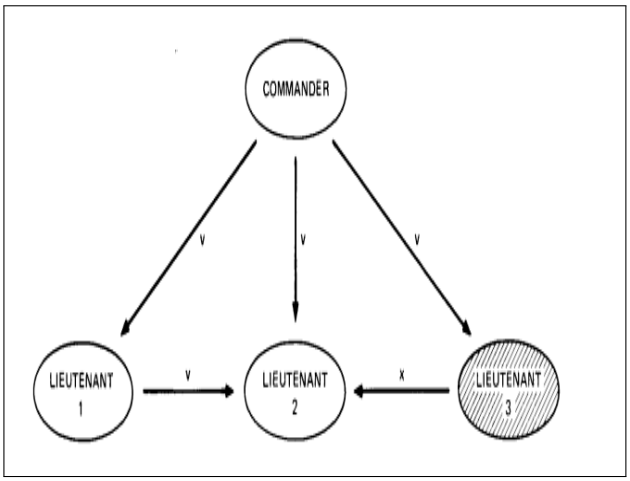

Figure 2.8: Byzantine Generals Problem: Algorithm OM(1); Lieutenant 3 a traitor [68]

In the case of one traitor, as in Figure 2.8, he can influence another lieutenant that the commander has ordered to retreat, but since the other two lieutenants are honest, a consensus of 2/3 is reached, ensuring that the attack is successful.

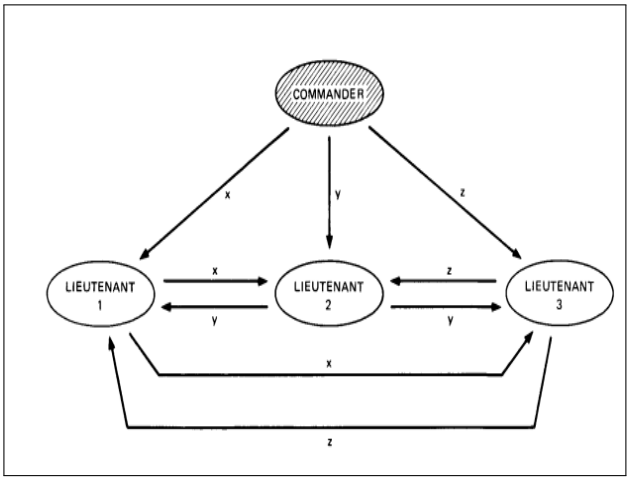

Figure 2.9: Byzantine Generals Problem: Algorithm OM(1); the commander a traitor [68]

In the case of Figure 2.9, the commander is a traitor and plots to fail the mission of attacking the fort. In this case, the commander sends a different message to each lieutenant, completely foiling the plan. Each lieutenant receives the battle plan from the commander, cross-communicates, and concludes that a treacherous message has been sent from the commander and, finally, RETREAT as it is the default if the message is obfuscated. This work is quite resilient to both the lieutenant and the commander's malicious behavior, paving the way for future works to be inspired by this allegory of an imaginary Byzantine empire. In conclusion, the synchronous system can be resilient for f byzantine actors if more than 2f honest nodes use digital signatures or more than 3f honest nodes for simple oral message communication.

2.2.1.2 Seminal Impossibility Theorem with Fault Process: Fischer, Lynch, and Paterson Theorem

This work in [26] answers the fundamental question of *In a fully asynchronous system, is there a deterministic consensus algorithm that can be safe, live, and fault-tolerant?*

The impossibility in this theorem states that the consensus cannot be achieved even within a malicious or faulty node. This theorem considers an asynchronous network setting where the message delay is finite but unbounded. In this case, the fault cannot be detected as there is a delay attribute for each message that can arrive within it, or it could have just crashed. Here the authors assume that the message channels are reliable and there are no byzantine actors in the system, meaning they can fail only by the crash but not due to malicious behavior. The theorem states that:

- No consensus protocol is correct despite one fault.
- There is a partially correct consensus protocol in which all non-faulty processes always reach a decision, provided no processes die during its execution and a strict majority of the processes are alive initially.

This enunciates that in the case of an asynchronous network model and even in the presence of one fault node, there can be no deterministic consensus protocol that satisfies the property of termination, agreement, and validity. Termination means the liveness property that all nodes that have not failed eventually decide on some value. The agreement is a safety property that builds on the liveness property to ensure that all the non-failed nodes decide on a common value. Validity signifies that if all non-faulty nodes have the same initial value, the final output should be the same upon agreement. Also, in this setting for the asynchronous system but with digital signature communication, considering f Byzantine or failure nodes, there should be n >3f honest nodes for the system to be safe.

2.2.1.3 Seminal Asynchronous Solution: Consensus in the presence of Partial Synchrony

Here the authors Dwork, Cynthia and Lynch, Nancy and Stockmeyer, Larry in [72] address the consensus for partial synchrony which lies between synchronous and asynchronous systems. The distinction between synchronous, asynchronous, and partial synchronous systems are:

- Synchronous systems have a known fixed upped bound Δ on time required for an inter-processor message communication and a known fixed upper bound Θ on the relative speeds of different processors.
- Whereas in asynchronous systems no fixed upper bounds Δ and Θ exists.
- In a partial synchronous system the fixed upper bounds Δ and Θ exist but are unknown.
- There exists another version of partial synchrony; the bounds are unknown but guaranteed to hold starting at some unknown time T. In other words, at Time T, the value of the bounds is clearly defined, but when it would occur is unknown.

Here the system model is based on the work of [26], where the communication system is modeled as message buffers. Each processor in the system can either perform a send or receive message of its protocol:

1. **Send(m,p_j)**: Places message m in p_j's buffer.
2. **Receive(p_i)**: Removes some (possibly empty) set S of messages from p_i's buffer and delivers the messages to p_i.

They study the problems of consensus for partial synchronous behavior in communication or processors with four types of failures for a processor p_i:

1. **Fail-stop faults:** Processor p_i executes correctly but can stop anytime without any recovery.
2. **Omission faults:** Faulty Processor p_i follows protocol correctly, but when Send(m,p_j) is executed by p_i, it might not place m in p_j's buffer and Receive(p_i) might cause only a subset of the delivered messages to be received by p_i.
3. **Authenticated Byzantine Faults:** Erroneous Messages signed with the sending processor's name that cannot be forged.
4. **Unauthenticated Byzantine Faults:** Erroneous Messages without signatures, but the sender's identity is known.

There are also four degrees of communication synchrony where the communication bound Δ varies:

1. Δ is known
2. Δ is unknown
3. Δ hold eventually
4. Δ holds sufficiently long

In this context, the correctness of a consensus protocol is defined by assumptions about processor and communication synchrony, a fault type F, a number N of processors, an integer t with $0 \leq t \leq N$, and a set C containing at least N-t processors and any run R:

1. **Consistency:** No two different processors decide differently.
2. **Termination:** There is an eventual decision.
3. **Strong Unanimity:** If all initial values are v and if any processor in C decides a value, then it decides v.

4. **Weak Unanimity:** If all initial values are v, if C contains all processors, and if any processor decides, it decides v.

This work discusses algorithms for partial synchronous communications and synchronous processors for different scenarios of faults like Fail-stop and Omission, Byzantine Faults with Authentication, and Byzantine Faults without Authentication with fewer processors but an exponential amount of communication. Their algorithms work in the two models of partially synchronous networks.

1. **Period of Synchrony model:** This considers that the consistency should hold regardless of the network delay and liveness only during sufficient long periods of synchrony.

2. **Unknown Delay model:** The delay Δ is not preconfigured. The idea is to develop a protocol for the unknown delay and respects the consistency and liveness property, but the confirmation delay T_{conf} may depend on the actual Δ.

These two models are interchangeable, meaning it is possible to construct one model from another, and the unknown delay model is used in theory. Still, the period of the synchrony model is practically considered while designing protocols.

2.2.1.4 Seminal Practical Solution: Practical Byzantine Fault Tolerance (PBFT)

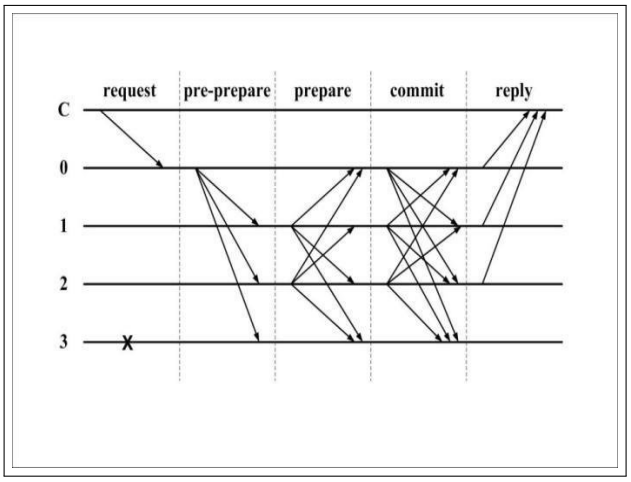

Figure 2.10: PBFT: Normal Case Operation [73]

The work of Castro and Liskov [73] is the first practical solution as it works in asynchronous environments like the Internet and proposes several optimizations for faster response times. The authors devised this algorithm for state machine replication to avoid the disruption in the availability of services by replicating or duplicating the states over multiple machines. The system considered here is asynchronous and can tolerate up to one-third of Byzantine actors.

The network comprising n nodes moves through a series of views where each view is a state or interval wherein a node is elected as a primary or leader. The selection of a primary is based on the view number, and the ordering of the nodes is based on a modulus operation. The operation of the consensus algorithm is explained in Figure 2.10.

As in Figure 2.10, client C sends a request to Node 0, which receives a request and participates in the consensus. Since node 0 is selected as primary, it broadcasts the PRE-PREPARE message, the starting point for the consensus. The message format can be considered a tuple of <MessageType, View No, Message Digest, Sequence Number of Message>. The consensus is a three-phase protocol of PRE-PREPARE, PREPARE, and COMMIT:

1. **PRE-PREPARE:** The selected primary 0 node broadcasts PRE-PREPARE message to other replicas 1, 2, 3. The replicas verify the message's signature and view number v. Post verification, the replicas send PREPARE messages.
2. **PREPARE:** Each Node or replica receiving 2f+1 PREPARE messages, including its own, verifies each and then sends the COMMIT message.
3. **COMMIT:** Similar to PREPARE phase, each node waits for 2f+1 messages and verifies them. After that, each node commits them locally, ensuring a successful consensus.

Post that, the nodes reply to client C who confirms the consensus on receiving 2f+1 replies. Finally, the view-change phase is initiated, ensuring that the consensus is completed in a defined time interval, and post that, a new primary is elected. The intuition to initiate this process is that the earlier primary could have become faulty or unstable, rendering the system unavailable. PBFT algorithm lowers the systematic complexity of BFT from the exponential level to the polynomial level and achieves consistency but suffers complex communication complexity of $O(n^2)$ and low scalability.

A more recent study [74] has been performed by Nguyen, Thanh Son Lam and Jourjon, Guillaume and Potop-Butucaru, Maria and Thai, Kim Loan on PBFT applied to Hyperledger Fabric and Sawtooth blockchain [75] in the context of a private permissioned network. Their deployment and performance study of the PBFT network over a wide area extending across France and Germany revealed that high periods of network delay after a height of 100 blocks slow the consensus time for adding each block by more than 134 seconds. They have argued that although practical byzantine fault tolerance is fork-resistant but does not provide sufficient consistency guarantees to be deployed in critical live environments.

2.2.1.5 Seminal Fast Track BFT Solution: Zyzzyva

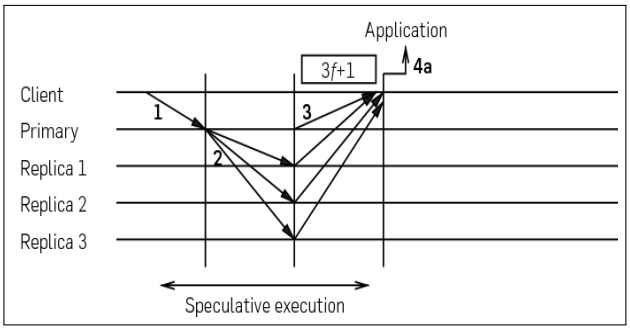

Figure 2.11: Zyzzyva: Protocol communication pattern within a view for gracious execution [76]

A more fast-track consensus that is optimistic in a sense was presented in [76]. This work

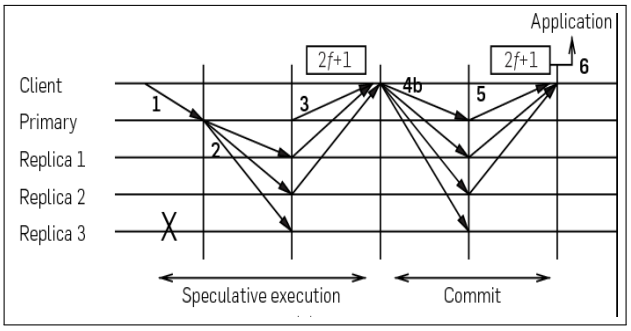

Figure 2.12: Zyzzyva: Protocol communication pattern within a view for faulty replicas [76]

belongs to the family of protocols that aim to improve the performance of PBFT protocols. It belongs to the optimistic branch of protocols like [77],[78, 79, 80, 81]. This work considers a fast and optimistic case of no network or system failures and has a fallback protocol with strong liveness guarantees. The reason for being optimistic is that there is a strong assumption of transaction genuineness and order [82] as there is no ordering and consensus initiated by the primary but a single phase protocol of message execution by the primary is accepted by the replicas. But in case of failures, it follows a three-phase complexity. On receiving a request from a primary or leader within the replicas, as in Figure 2.11, the other replicas execute the request on a valid PRE-PREPARE message. Then all the replicas, including the primary reply to clients for the final commit until the 3f+1 majority is reached. This execution, called the gracious execution as in Figure 2.11 follows the steps of:

1. Client sends a request to the primary.
2. Primary receives the request, assigns sequence number, and forwards ordered request to replicas.
3. Replica receives ordered requests, speculatively executes them, and responds to the client.
4. Client gathers speculative responses.

4A. Client receives 3f+1 matching responses and completes the request without faults.

Compared to the PBFT, the PREPARE and COMMIT Phases are reduced to a single phase. So the message communication complexity is O(n). But if in case of faulty replicas, the protocol needs to execute two more phases as in Figure 2.12. In this faulty case, after the client receives less than 3f+1 messages, it waits for a certain time and sends a commit message to the replicas. The replicas reply to the client and are finalized if at least 2f+1 replica messages are received.

Execution of additional steps for fault cases are:

4B. Client receives between 2f + 1 and 3f matching responses, assembles a commit certificate, and transmits the commit certificate to the replicas.

4B.1. Replica receives a commit message from a client containing a commit certificate and acknowledges with a local-commit message.

4B.2. Client receives local-commit messages from 2f + 1 replicas and completes the request.

4C. Client receives fewer than 2f + 1 matching SPECRESPONSE messages and resends its request to all replicas, which forwards the request to the primary ensuring the request is assigned a sequence number and eventually executed.

4D. The client receives responses indicating inconsistent ordering by the primary and sends proof of misbehavior to the replicas, which initiates a view change to oust the faulty primary.

This protocol is speculatively faster than PBFT considering the gracious execution case but suffers from certain problems, as discussed in [77]. They highlight the problem of safety where in a gracious execution phase, a byzantine primary can equivocate for different values and manipulate the consensus to decide a malicious value. They further in their work [77] conclude that none of the fast Byzantine agreement works or optimistic Byzantine agreement can simultaneously address the problems of:

1. Optimal step-complexity.
2. Optimal resilience.
3. Safety against failure of less than a third of the system.
4. Progress during periods of partial synchrony.

2.2.1.6 Seminal Linear Communication and Simple Solution: HotStuff

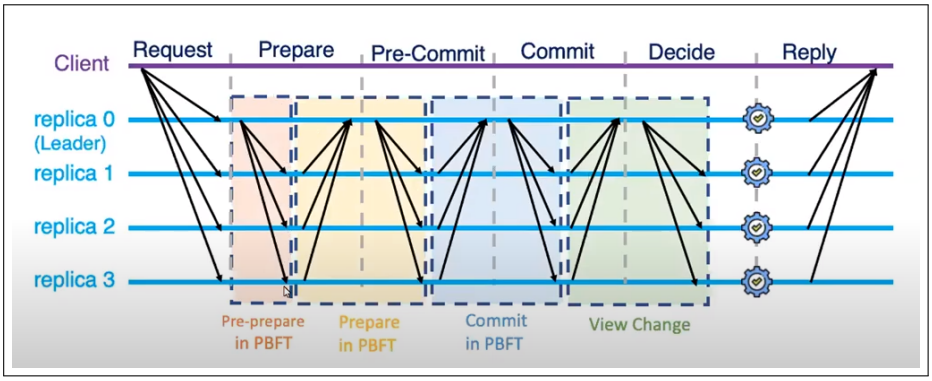

Figure 2.13: Hot Stuff Protocol[83]

A more recent protocol that works in partial synchrony [83] was proposed to be used in Libra, a Facebook project, before it was shelved later [84]. It has a linear communication in both cases of normal phase and leader selection phase, unlike Zyzzyva [76], which has O(n) and $O(n^2)$ communication complexity. To reduce the complexity, unlike PBFT and Zyzzyva [73, 76, 85], the protocol tries to reduce the communication messages by replying only to a chosen leader instead of inter-peer communication messages like PBFT. The protocol increases the phases compared to PBFT but simultaneously reduces the communication message, which is a trade-off we must consider.

The protocol has the phases of Request, Prepare, Pre-Commit, Commit, Decide, and Reply as in Figure 2.13[82, 85] where the communication is an N to a leader instead of an N to N as in PBFT. The protocol works in a succession of views, each having a chosen leader. The leader is rotated after every round of consensus, which is achieved by the inbuilt view

change protocol, which is O(n) compared to $O(n^3)$. The leader uses the (k,n)-threshold signature schema to prove the validity of received messages from other replicas.

The protocol works as follows in Figure 2.13, pictured analogous to PBFT but with more phases under reduced communication messages. The message consists of a command to execute by the replica, metadata, and a parent link. The protocol guarantees liveness if the messages arrive with a Δ time, and it always satisfies safety. It works as follows:

- The Client sends a Request message to the N nodes, and an elected Leader then forwards a Prepare message to the other replicas. The leader waits for new-view messages from n-f replicas to prepare a Quorum Certificate with the highest view.
- The leader then broadcasts the new proposal to all the replicas with the Highest Quorum Certificate. A quorum Certificate (QC) is a collection of votes from the Quorum of (n-f) replicas. The replicas acknowledge the leader with Prepare message by a partial signature if they accept it.
- Leader, upon receiving (n-f) PREPARE votes, combines them in a quorum certificate and broadcasts it in a PRE-COMMIT message. The replicas respond with votes for the message.
- Leader, upon receiving (n-f) PRE-COMMIT votes, combines them in QC and broadcasts them in a COMMIT message. The replicas reply with votes to the Commit message.
- Finally, the leader receives (n-f) COMMIT votes, which are, in turn, combined in QC. It then broadcasts it in a DECIDE message, which is finally executed by the replica. The replicas increment the view number, and the next view is started choosing a new leader.

To simplify further, that is, to reduce the number of messages and pipelining of decisions, a chained version of this protocol is presented in Figure 2.14, which is similar to the CASPER protocol of Ethereum [86]. A new view is proposed for each Prepare message resulting in the simultaneous processing of different views.

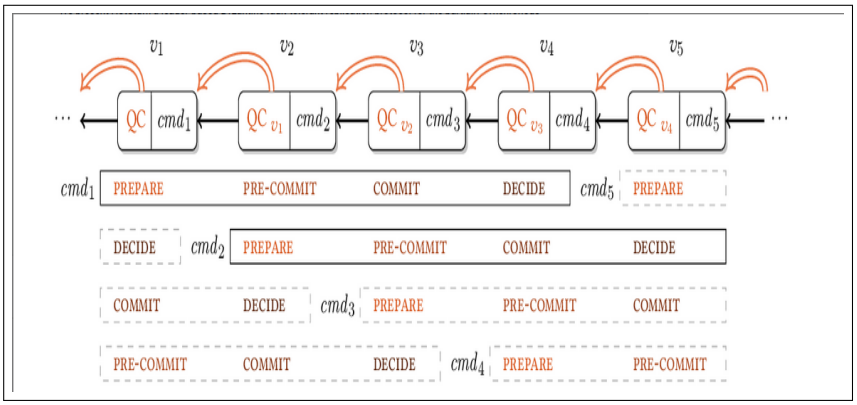

Figure 2.14: Hot Stuff Protocol: Chained Version[83]

As in Figure 2.14, for view v_1 a leader proposes command cmd_1, then for view v_2 a new leader proposes cmd_2 piggybacking the messages for view v_1 and view v_3, a new leader

proposes cmd_3 piggybacking the previous two views v_2, v_3. A single quorum certificate is expected for a successful transition in all these phases. Thus, combining the different views and further processing them, the chained version improves the performance compared to the normal case of HotStuff protocol.

2.2.1.7 Seminal Fast Track BFT Solution: Tendermint

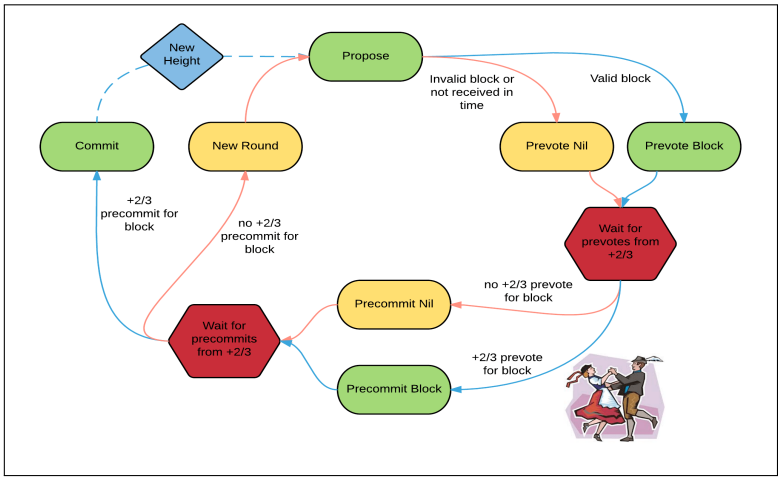

Figure 2.15: Tendermint Protocol [87]

Tendermint [87] is a weakly or partially synchronous protocol with a dynamic committee based on Proof of Stake for its selection. It favors consistency over availability, meaning the system can be prone to liveness issues. In case of benign failures or malicious attempts, it has an additional stake-slashing layer to guarantee the liveness and finality of the chain. Cosmos blockchain uses this protocol for its consensus and finality mechanism. It is an interoperable blockchain communicating with other blockchains through inter-blockchain communication (IBC) protocol, a networking layer for blockchains.

This protocol as illustrated in Figure 2.15, works as follows:

- At first, a committee among the validators is chosen based on the amount of stake in the chain. Then among the validators in the committee, a proposer or block creator is selected based on round robin and proportional stake in the network.
- Then the chosen proposer proposes a block that can either be attributed to a Prevote-Block or Prevote-Nil by the other validators if they reject or accept the block. A 2/3 validator's voting threshold is needed for the block to progress. If a validator doesn't respond, the stake is slashed, as it can be due to network failures or other malicious attempts.
- On receiving 2/3 of the validator's majority votes of Prevote-Block, it progresses to the next phase of Precommit, which expects the Precommit-Block or Precommit-Nil if valid or not.
- Then, in the final phase, the proposer receives the 2/3 Precommit-Block votes and Commits the block. All these three phases have a Δ timeout interval until they wait for the votes from the validators as it is partially synchronous.

This protocol is a more practical solution applicable to the real-world blockchain but, however, suffers from fairness issues in forming a committee based on the stakes of the individual validators as discussed in [88], which is a drawback in general for Proof of Stake protocols.

2.2.1.8 Seminal Safe Proof of Stake Solution: GASPER

This protocol [89] is used in Ethereum 2.0, a Proof of Stake protocol. GASPER is a combination of two components that works along with Proof of Stake consensus, which is: 1) Casper Friendly Finality Gadget (FFG) and 2) Latest Message Driven Greediest Heaviest Observed SubTree (LMD GHOST). The authors define FFG as an algorithm that marks certain blocks in a blockchain as finalized so that participants with partial information can be fully confident that the blocks are indeed canonical. It is termed a gadget to signify a module that adds to the proof of stake protocol to provide finality. The second component, LMD GHOST, is a fork-choice rule where validators attest blocks they find valid.

In a proof of stake protocol, the validator's vote is quantified in proportion to their stake in the network. This migration by the Ethereum community from erstwhile Proof of Work to Proof of Stake has multiple good reasons:

- Better Energy Efficiency as less energy is spent on calculation.
- More participants can join the network as the hardware requirement is avoided.
- Penalty mechanism for Byzantine behavior in the form of stake slashing helps in the network's resiliency.

In this protocol, the validators deposit a certain amount of tokens or crypto money, which is Ether in the case of Ethereum, into a deposit contract and then join a queue to activate its participation. Then the validators receive the blocks and then attest to them if they are valid. Temporal division in this protocol consists of a slot which is 12 seconds, and 32 slots amount to an epoch. For every slot among a committee of validators, a validator is chosen randomly every slot to become a block proposer. The proposer creates the block, which is then validated by the voting mechanism in the committee.

2.2.1.8.1 CASPER Friendly Finality Gadget: It is a mechanism that ensures the validated blocks are not reverted unless there is a consensus failure leading to the destruction of the staked tokens. The two procedures for a block to be finalized are:

- **Justification:** It is justified when a validated block is voted by $2/3^{rd}$ of staked validators.
- **Finalization:** When the previous justified block is added in the next height with another justified block, it is said to be finalized, enabling the block to be added to the canonical chain.

This two-step process of block finalization happens every epoch until they are known as checkpoints. The condition of $2/3^{rd}$ votes for a canonical block makes it difficult to break unless the staked ether is destroyed or manipulated. Additional security against Double Voting, where a validator equivocates his votes, is by stake-slashing condition punishment. In [90], they note that an attacker requires to lose $10 billion Ether which makes it eco-

nomically disincentive.

Economic Model: In [91], Vitalik Buterin, one of the co-founders of Ethereum remarks in his philosophy behind the Proof of Stake(PoS) that, the PoS is not "security comes from burning energy" but rather "security comes from putting up economic value-at-loss." In line with this, the validators get slightly rewarded for proposing and validating blocks honestly. But if there is dishonesty either passively by being offline or failing to respond or actively by being malicious, there is an economic loss. The loss for a passive activity is minimal, but an active one attracts deep slashing of the stake. The protocol also provides "plausible liveness" or reasonable liveness by reacting when a chain fails to finalize for more than four epochs; the validators who have not participated will be slashed off their stake. So there is a simultaneous incentivization and disincentivization of economic behavior for the validators to secure the network.

2.2.1.8.2 Latest Message-Driven Greedy Heaviest Observed Sub-Tree (LMD-GHOST):

Normal fork resolution mechanism called GHOST was previously proposed as part of CASPER in [92] had the mechanism of choosing the chain (fork-choice algorithm) that has the greatest height. GHOST was defined for Bitcoin in [93] as a construction and re-organization mechanism for blockchain which secures Bitcoin against double-spend attacks. Fork here means an alternative chain that exists in addition to a canonical or true chain, which causes divergence into two potential block paths. The fork-choice algorithm is used to identify the head of the blockchain, which normally has more blocks or difficulty weightage, signifying the greater computational effort.

But in GASPER, a more robust algorithm called LMD-GHOST selects the fork with more attestations or votes. Also, in case of duplicate or multiple messages sent from a validator for the fork-choice process, the latest message is considered.

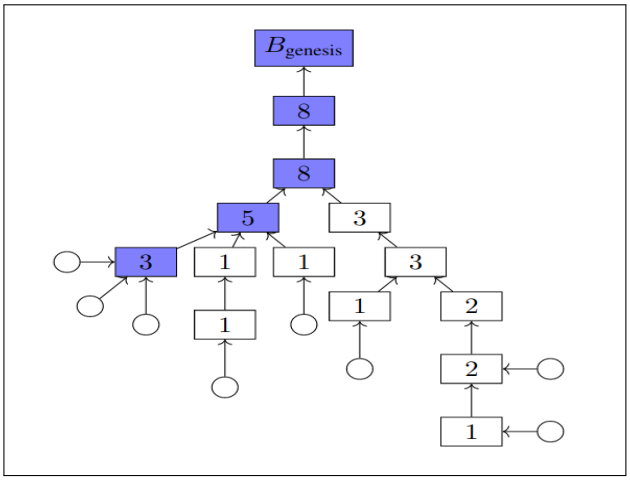

Figure 2.16: LMD GHOST [90]

As illustrated in the [90] Figure 2.16, we decide the leaf block of the blue chain to be the latest block, as we apply the heuristic of the weights of each subtree and compare them. The attestations are marked as circles, and the blue chain has the highest attestation. The weight of the stake signifies the number in each block, and the attestations are all assumed

to be a weight of 1. By comparing the chains from the left, the blue chain has a total weight of 27, whereas others have a lesser weight at the same height as the Genesis Block. So this Proof of Stake protocol with an added gadget of Finality and LMD-GHOST proves safety, plausible liveness, and consistency which accounts for both stakes weights and attestations. However, the authors argue in [90] that their version of the Proof of Stake protocol is not better than others. Still, if safety is preferred over liveness, they recommend Tendermint as previously discussed in SubSection: 2.2.1.7 or HotStuff as there are arbitrary network delays discussed in SubSection: 2.2.1.6.

2.2.2 Relevant Work

This section discusses more relevant works in Byzantine Fault Tolerance Consensus algorithms, focusing on improving the throughput, parallelization of consensus, and Quorum-based consensus for consortium networks. We discuss apart from the Proof of Stake protocols like Tendermint, Hotstuff, or GASPER, which stakes the real economic value as discussed earlier. We highlight the works on abstract economic values like Authority, Reputation, and Altruism, which can be an alternate study to focus on as it is less penalizing. Still, there is a social cost, and it is more democratic because there are no entry economic barriers of huge monetary stakes, which can be interesting. We avoid Proof of Work in our analysis as it is more calculation intensive, probabilistic finalization of blocks, and has huge energy costs, which are unsuitable in our case as our focus is more on consortium with closely verified participants.

2.2.2.1 Parallelization

In this section, we focus on works dedicated to improving the scalability issues that arise in classical single leader BFT protocols such as PBFT [73], where communication and protocol phases are costly in terms of computation. This reliance on a single leader for directing the consensus has been dealt with earlier in the Hotstuff Algorithm by improving the communication from N-to-N validators to One-to-N validators explained in section 2.2.1.6. There are various protocols following this idea of parallelization, like in [94],[95, 96, 97, 98, 99, 100], but we select a few recent works to elucidate here.

2.2.2.1.1 Mir-BFT In the work of [94], the authors Stathakopoulou, Chrysoula and David, Tudor and Pavlovic, Matej and Vukolić, Marko define a more robust and high throughput protocol called Mir-BFT, a multi-leader protocol. They allow multiple leaders to propose batches of blocks independently and concurrently. The intuition behind these parallel leaders is to distribute the CPU and bandwidth load more uniformly. At the same time, they also handle the duplication of requests, which is a side effect of parallel leaders by a modular arithmetic operation. This prevents two types of attacks which one should consider in multiple leader consensus protocols [94]:

- **Request Censoring attack** in which a leader can drop or delay a request.
- **Request Duplication attack** where a request can be executed multiple times.

The protocol that transitions for each epoch period works as follows in Figures 2.17,2.18:

- For an epoch, primary and multiple leaders are selected, which can be proposed in parallel.
- Then each leader, including the primary, is assigned a bucket as in Figure 2.17 where each bucket has a certain hash space range to accept into its queue. If a request's hash falls into a particular hash space, it is accepted by the particular leader of a bucket. This mechanism avoids request duplication; each leader proposes the request in their buckets.
- Then, after the requests are filled in the hash space buckets, a primary initiates epoch and assigns a sequence number to leaders. The primary proposes the first batch in the epoch, and leaders take turns proposing the other batches sequentially as in Figure 2.18. Each leader proposes a batch with a PRE-PREPARE message with a sequence number to its set of 2F+1 replicas. The replicas other than its own are called observers, verifying the message of its batch number, which should be unique. This phase is then followed by the PREPARE and COMMIT message in parallel with multiple batches of blocks to finalize the blocks.

So this protocol, with the proposition of multiple batches and the request duplication handling, shows a higher throughput of 60000 transactions per second as claimed in the Go language implementation with 100 nodes and 1 Gbps WAN setup, which is quite promising in the BFT scenarios.

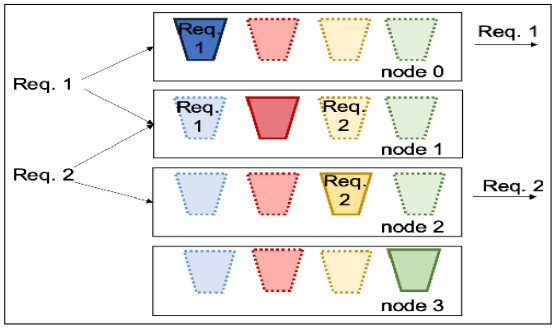

Figure 2.17: Sorting of Request into Buckets [94]

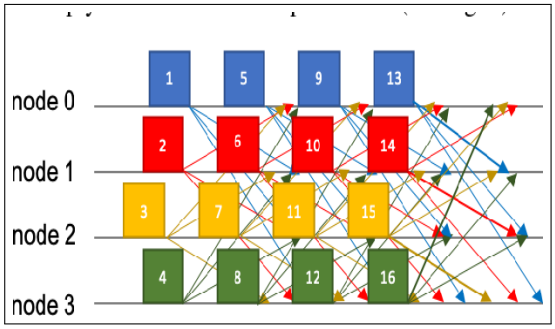

Figure 2.18: Balancing of proposal load among the 4 nodes for messages in an epoch [94]

2.2.2.2 Quorum Based Protocols

In this section, we discuss the most relevant quorum-based consensus protocols that require an effective subset S of participants from a universal set of N Nodes. These types of protocols need a permission setup of the network where the nodes can join the network through access to smart contracts, stake deposition, or other verification mechanisms. Authentication mechanism through public key infrastructure is enforced to avoid security violations like Sybil attacks or to identify malicious behavior. In the analysis by [101]Rebello, Gabriel Antonio F. and Camilo, Gustavo F. and Guimarães, Lucas C. B. and de Souza, Lucas Airam C. and Duarte, Otto Carlos M. B., they classify the quorum-based protocols into three classes:

- Practical Byzantine Fault Tolerant Protocol and its derivatives.
- Federated Byzantine Fault Tolerant Protocols.
- Delegated Byzantine Fault Tolerant Protocols.

2.2.2.2.1 Practical Byzantine Fault Tolerant Protocol They note the main limitation of PBFT protocol already discussed in Section 2.2.1.4 as scalability concerning the number of consensus participants. To tolerate malicious behavior, the participants require $O(n^2)$ messages. The other limitation noted is the inability to add and remove the participants at runtime in this protocol. Alternative protocols in PBFT have been proposed by [102] Aublin, Pierre-Louis and Mokhtar, Sonia Ben and Quéma, Vivien to spin or change the leader every round, avoiding centralization leading to malicious action and compromising the liveness. Further, another alternative in PBFT is Redundant Byzantine Fault Tolerant Protocol by [103] Chase, Brad and MacBrough, Ethan, which is robust in the sense of identifying the leader who is malicious when a certain round exceeds a time threshold and a new leader is elected to replace the latter.

2.2.2.2.2 Federated Byzantine Fault Tolerant Protocol: Next classification by the authors in [101] is Federated Byzantine Agreement, where they partition the quorum into slices managed by a Byzantine General, reducing the communication complexity. Here the participant in the network can choose the quorum slice to trust, which makes a malicious participant with duplicate identities very difficult to join the network and participate in the quorum. They have discussed two prominent protocols of this genre: 1) Ripple or XRP Ledger Consensus Protocol and 2) Stellar Consensus Protocol.

In Ripple or XRP Ledger Consensus Protocol by [103] Chase, Brad and MacBrough, Ethan, they have a low latency compared to other BFT protocols as they introduce the concept of subnets within the totality of the network. The protocol has a Unique Node List (UNL), which acts as an Access Control List (ACL) to identify the eligible and reliable nodes. It operates in two phases:

1. **Consensus Phase:** In this phase, each validator proposes a set of transactions and simultaneously validates the proposal of other peer validators identified in the UNL. The required majority for this phase is 80%
2. **Validation Phase:** Each validator calculates the block using the previously collected transaction proposals and broadcasts the block's hash in the network. When the

required majority of 80% votes is achieved for the block, it is considered finalized.

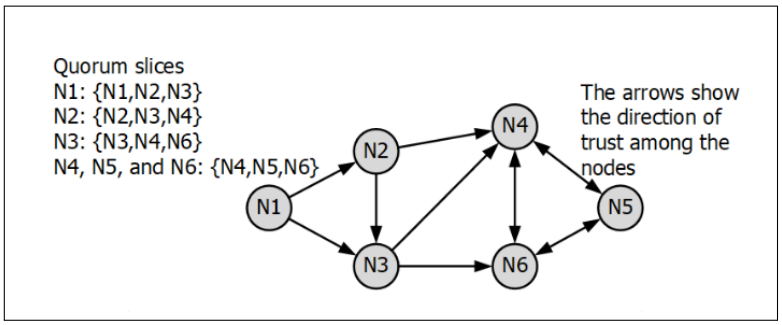

Figure 2.19: Stellar Consensus Protocol [104]

The next discussion is on Stellar Consensus Protocol by [104] Kim, Minjeong and Kwon, Yujin and Kim, Yongdae has the same low latency as XRP protocol but has flexible trust and a Federated Byzantine Agreement model. In this, the protocol operates with an inter-quorum-slice message passing where a single quorum is split into slices. The consensus within these slices reduces the message complexity. It operates in two phases:

1. **Nomination:** Participants generate a set of transactions during this phase, and their validation is complete as soon as a slice of the quorum is complete, as in Figure 2.19. Here N1, N2 ...N6 are individual participants, and each participant is mapped to a Quorum slice. N1 belongs to the quorum slice comprising N1, N2, N3, and the other slices are represented in Figure 2.19. A complete consensus at the quorum level is obtained once participant N traverses all the slices and returns to its position. As each quorum slice validates a certain set of transactions, all these are merged deterministically into a single set of transactions forming a block.
2. **Balloting:** Participants reach a consensus on the block between multiple quorum slices based on federated voting. Here multiple ballots are created for the set of transactions to vote on from the block and are voted on until an agreement is reached.

2.2.2.2.3 Delegated Byzantine Fault Tolerant Protocols Next classification by the authors in [101] Rebello, Gabriel Antonio F. and Camilo, Gustavo F. and Guimarães, Lucas C. B. and de Souza, Lucas Airam C. and Duarte, Otto Carlos M. B. is Delegated Byzantine Fault Tolerant Protocol which is used in Neo Blockchain. It works through proxy voting, where the holder of the tokens, by voting, supports a consensus node. The protocol consists of three types of participants: 1) Clients who handle the send and receive of requests 2) Speaker who perform the consensus and is selected by the delegates 3) Delegates also perform the consensus and help in choosing nodes. Then the set of consensus nodes undergoes a PBFT style of consensus [105], which works as follows as in Figure 2.20:

1. Speaker broadcasts the Prepare Request message to the network's nodes.
2. Delegates broadcast the Prepare Response message to the network
3. Then, the Speaker and Delegates broadcast the message until the block is confirmed after reaching the threshold.

The final classification is the Byzantine Fault Tolerant and Delegated Proof of Stake Pro-

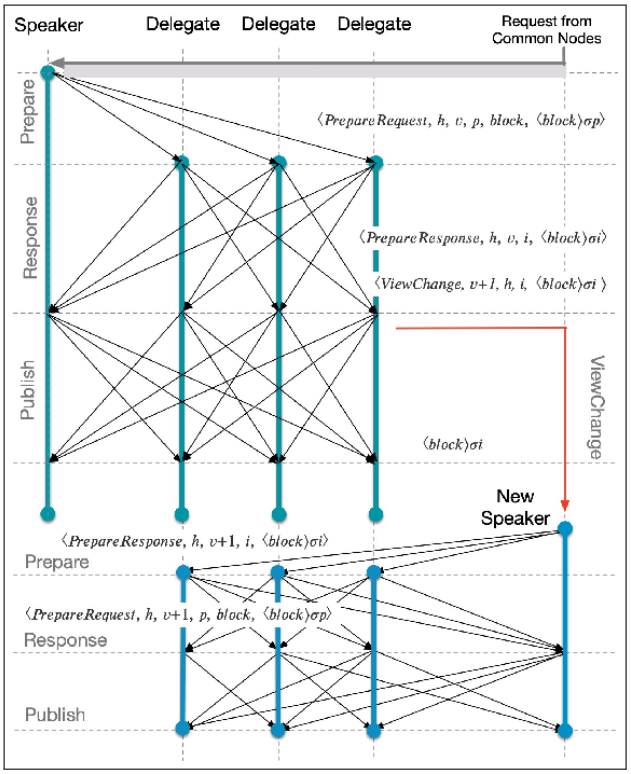

Figure 2.20: Delegated Byzantine Fault Tolerance Protocol [101]

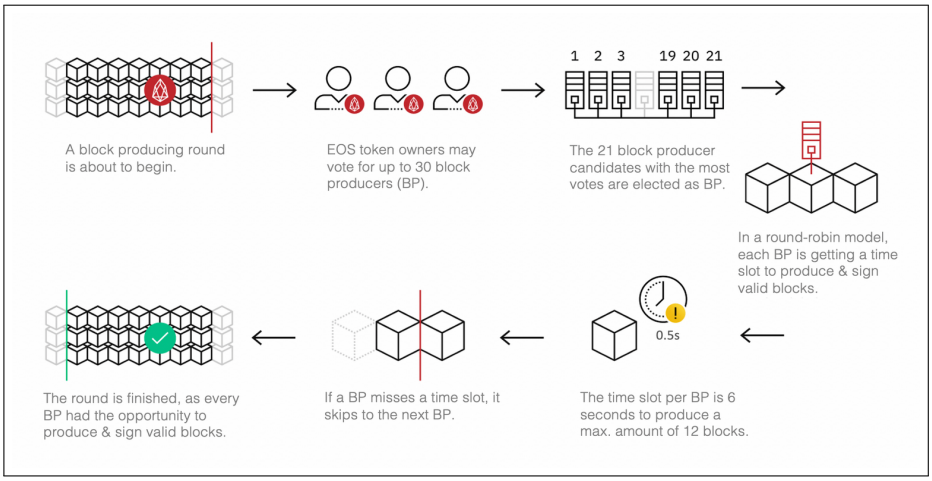

Figure 2.21: EOS.IO Consensus Protocol [106]

tocol, developed in the EOSIO protocol. The protocol works as represented in Figure 2.21 with two phases:

1. **Election of Delegates:** In this phase, the users who hold tokens elect the block producers through voting. Each token holder can vote for at most 30 block producers who act as delegates.
2. **Production of Blocks:** Then, the delegates validate the transaction for a certain time period. Then among the delegates, a block producer produces the block, which is agreed upon until 2/3rd of the delegates confirm through signed messages.

In [101], Rebello, Gabriel Antonio F. and Camilo, Gustavo F. and Guimarães, Lucas C. B. and de Souza, Lucas Airam C. and Duarte, Otto Carlos M. B. identify the vulnerabilities in this protocol:

- The centralization of 21 delegates elected by voting undermines the decentralization aspects.
- The voting process in this protocol is weighed proportionately to the held assets, which can lead to collusion by the high token holders already identified by Zhao, Yijing and Liu, Jieli and Han, Qing and Zheng, Weilin and Wu, Jiajing [107] through the formation of voting gangs.
- After the election, the delegates have the same power regardless of votes received, which can incentivize the token holders to collude and elect delegates with the least votes.

More exhaustive work on a committee or quorum-based proof of stake (PoS) Protocol has been done by [108]Zhao, Yijing and Liu, Jieli and Han, Qing and Zheng, Weilin and Wu, Jiajing. Here they identify that a committee-based PoS helps maintain a more ordered process in which a committee of stakeholders valued with their stakes can produce blocks. Based on this committee-based PoS idea are the well-known protocols of Chain of Activity [109], Ouroboros [110], Ouroboros Praos [111], and Snow White [112].

In **Chain of Activity** by Bentov, Iddo and Gabizon, Ariel and Mizrahi, Alex [109] works in two phases: 1) Committee Selection and 2) Block Generation. Based on participants' current blockchain state and stake information in the committee selection phase, a Multi-Party Computation(MPC) is performed to select the block producer and its sequence for block generation. MPC [113] is a subject within cryptography that allows the participants to solve a function over their individual inputs without sharing with others. Then a block sequence is generated using the Follow the Satoshi (FTS) algorithm.

Follow the Satshi (FTS) [114] is an algorithm for staking. It works in two phases: 1) First Phase involves the randomized selection of tokens in the network by a pseudo-random function. 2) The second phase gets the account (owner) of the particular associated token. Since the input of the random function is a sequence of seeds, the algorithm likewise furnishes multiple output accounts, among which an account is elected based on the proportion of tokens held by the node.

MPC computation from the previous involves this FTS algorithm to generate the block generation sequence. The participants use this sequence to know their turn and then propose

a block according to it.

In the next work, we discuss **Ouroboros** protocol by Kiayias, Aggelos and Russell, Alexander and David, Bernardo and Oliynykov, Roman [110], which is used in the blockchain Cardano. The protocol considers time periods in epochs which in turn are divided into fixed slots. A slot requires a leader to be elected among the electors. Eligible electors are those with enough stakes for a particular epoch.

The election process of a leader for a slot involves an MPC procedure, more specifically known as Publicly Verifiable Secret Sharing (PVSS).

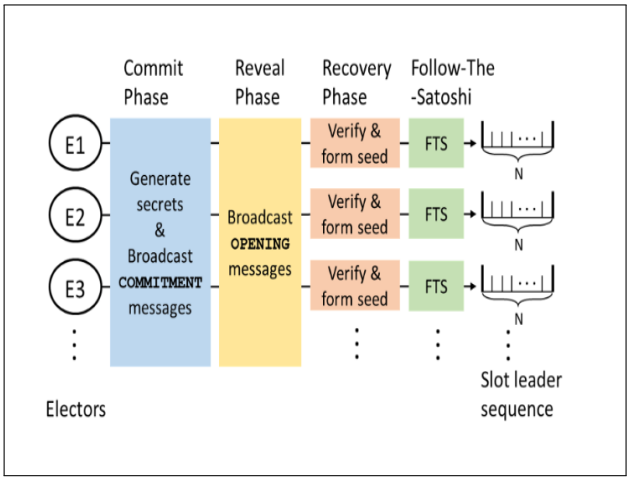

Figure 2.22: Publicly Verifiable Secret Sharing [110]

PVSS is represented in [115] Figure 2.22, which has three phases of Commit, Reveal, and Recovery followed by the FTS algorithm previously discussed. Here we note that the electors E_1, E_2... E_n generates a secret and broadcasts them via a COMMITMENT message. Then in the next phase, they send an OPENING message. Finally, they use both the COMMITMENT and OPENING messages to form a seed verified to be uniform across the network. The network is assumed to be synchronous, and then the seed is used to calculate the leaders for a predefined sequence during the epoch by the FTS algorithm.

Another committee-based PoS, which is a security improvement over Ouroboros [110] previously discussed, is **Ouroboros Praos** [111] by David, Bernardo and Gaži, Peter and Kiayias, Aggelos and Russell, Alexander. The strong considerations of synchronous networks and the public exposition of elected leaders for a slot beforehand which can lead to targeted attacks, are the major issues of Ouroboros that are solved in this protocol.

This protocol operates in partially synchronous networks assuming a Global Stabilisation Time (GST) of Δ. They [111] David, Bernardo and Gaži, Peter and Kiayias, Aggelos and Russell, Alexander modify the protocol by introducing empty slots in a particular epoch for re-synchronization purposes and multiple leaders per slot to handle liveness issues in case of failures or delays. The next concern regarding the knowledge of slot leaders beforehand, which opens targeted security attacks, is solved by adding a Verifiable Random Function (VRF). This function, upon solving, reveals the slot sequence only to the elected leaders, and the VRF Proof is produced by the leader for verification, solving the second security

issue as well.

The final work discussed in committee-based PoS Summary Work by Xiao, Yang and Zhang, Ning and Lou, Wenjing and Hou, Y. Thomas is Snow White [112] by Daian, Phil and Pass, Rafael and Shi, Elaine. The protocol is designed to work asynchronously and is resilient to network delays as it borrows some of its features from the Sleepy Model of Consensus [116] by Pass, Rafael and Shi, Elaine. This Sleepy Model assumes high network delays. They assume honest participation of the actors, which uses a Public-Key-Infrastructure and a common random string to build the protocol.

Snow White [112] uses an MPC procedure to decide a block proposal committee who are provided with an eligibility pass individually. As in the other previous committee-based PoS protocols, the input for the calculation is the stakeholder information. This protocol is resilient under benign network delays and uses a checkpointing scheme to finalize blocks already committed.

The common drawback associated with committee-based PoS identified in [115] is the committee's size, which can cause communication overhead to increase. Also, the network synchrony assumption for these protocols of a Δ GST and increasing committee members can arbitrarily delay the block finalization process. The example of Cardano Blockchain, which minimizes the participation size of the committee to 50, proves quite efficient for the protocol to respect liveness, termination, and validity.

2.2.2.3 Proof of Authority

This set of protocols belongs to the permissioned blockchain wherein the nodes' identities are well-defined and traceable. These are purely based on identity as a resource and are extremely relevant for consortium or federated blockchains. At first, we discuss the classical algorithm variants of Proof of Authority and then the analysis based on this protocol.

2.2.2.3.1 Variants of Proof of Authority Proof of Authority (PoA) is based on the identities of nodes called authorities. Authorities are assumed to be honest since they are a closed network of well-known participants who always follow the protocol. The required threshold for safety if we consider N Nodes is N/2+1. Block proposal is fair, with each identity or node being given a chance. The different variants of this family of algorithms will be discussed further as follows.

Clique : Clique [117] is a Proof of Authority Protocol proposed and implemented in Go-Ethereum or Geth. This protocol was conceived to develop an Ethereum Virtual Machine based on a Testnet called Rinkeby. This consensus of faster block proposal without any major computational effort like PoW facilitates the test of the Ethereum Smart Contract before deploying on a live network.

The algorithm consists of a set of authorities or proposers working in epochs where each epoch consists of a certain block height or limit of blocks proposed. The protocol has a single phase of BLOCK_PROPOSAL. All the block proposition happens for a pre-defined block interval that can be configured. Then for a particular block height, a proposer is

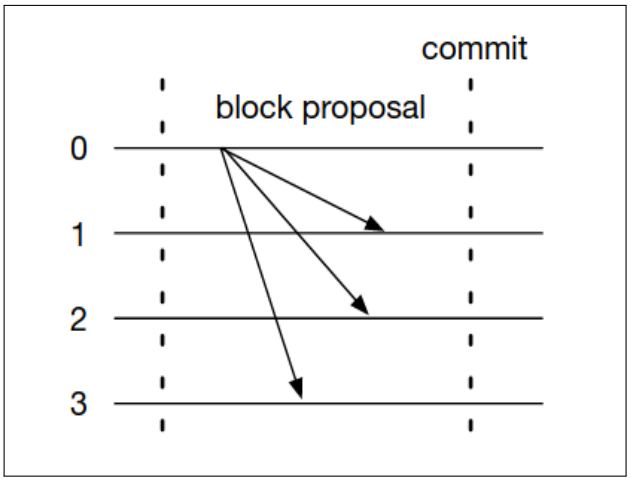

Figure 2.23: Clique Consensus [117]

selected based on a modular arithmetic operation among the authorities if it has not been signed recently. The block proposition can happen either in two ways:

1. **In-order:** When the proposer of the block is legitimately selected, then he proposes a block with weight 2 called in-order sealing.
2. **Out-of-order:** When the legitimate proposer does not propose the block, then any other block proposer can propose with a weight of 1, called as out-of-order.

As in Figure 2.23, a single block proposer is selected who proposes an in-order or out-of-order block containing the transactions. The protocol, although lightweight and simple in terms of calculation, poses a problem in the finalization of the blockchain as it is prone to many chain forks, as noted in [118]. This leads to another complementary protocol, Greediest Heaviest Observed Sub Tree (GHOST), to be applied [93]. The GHOST protocol calculates the cumulative weight of the chain based on the individual weight of each block, as they can be of weight 1 or 2. The protocol arbitrarily chooses a chain with a heavier cumulative weight or one with more in-order sealed blocks.

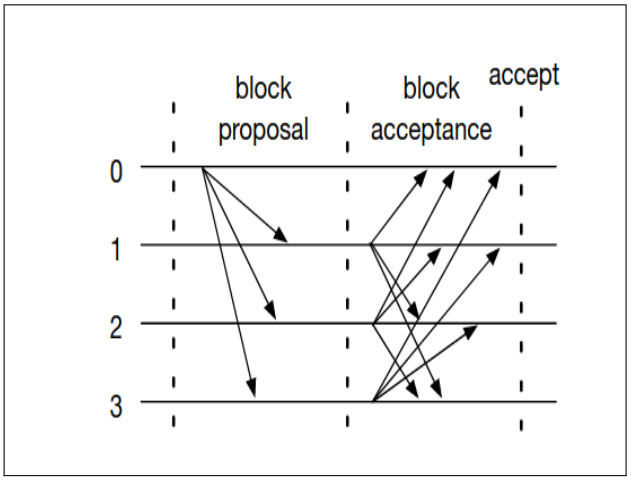

Figure 2.24: Authority Round (Aura) Consensus [117]

Authority Round (Aura) This variant of PoA is implemented [117] in Parity, an Ethereum-based binary and Substrate Blockchain platform. The network assumed here is synchronous within a UNIX time t, with two phases of consensus: BLOCK_PROPOSAL and BLOCK_ACCEPTANCE. The block proposition happens with a pre-defined step_duration [117]. Similarly to Clique, a modular arithmetic operation over the total proposers is performed to identify the proposer. Still, the condition of avoiding recently signed proposers is not used here.

The selected leader proposes a block and broadcasts to the other authorities as in Figure 2.24. On receiving the block proposal from the leader, the other authorities relay this proposal to others for verification which is quite a heavy phase of message communication. The block is accepted and added to the chain if it is valid. If it is invalid, the leader is voted out for malicious behavior by the remaining authorities: 1) No block proposal for a round, 2) Proposed conflicting blocks. This algorithm is also subjected to forks similar to Clique, as time drift can happen, and a perfectly synchronous network assumption is strong to consider.

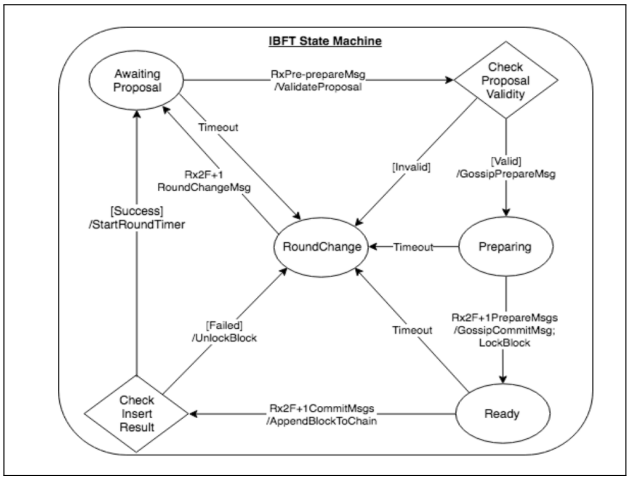

Figure 2.25: Istanbul Byzantine Fault Tolerance (IBFT) Consensus [119]

Istanbul Byzantine Fault Tolerance (IBFT) This protocol [119] inspired by Castro, Miguel and Liskov, Barbara [73] with modifications is implemented in Ethereum Enterprise Blockchains: GoQuorum and Besu. The necessity of the client sending transaction is replaced by Peer-to-Peer validators exchanging messages and reprising the same 3 Phases of consensus: PRE-PREPARE, PREPARE and COMMIT. Assuming the network has F faulty nodes in an N validator network, it needs 3F+1 honest or benign actors.

The algorithm which operates in rounds, works as follows represented in Figure 2.25:

- At the start of the round, a proposer is chosen from the set of validators in a round-robin fashion. The proposer proposes a new block and broadcasts it with a PRE-PREPARE message.
- The other validators, upon receiving the PRE-PREPARE message, then validate the block and broadcast PREPARE message, ensuring that they work on the same block sequence and round.

- Upon receiving 2f+1 PREPARE messages by each validator, a COMMIT message is broadcasted informing the other peers the block proposal is accepted.
- On reaching 2f+1 COMMIT messages, the block is added to the blockchain.
- In case of any network failure or invalid block, a ROUND CHANGE is initiated, and a new proposer is selected, ensuring liveness.

The advantage of this protocol resembling close to PBFT [73] is that there are no forks, unlike Clique and Aura, and the blocks are finalized.

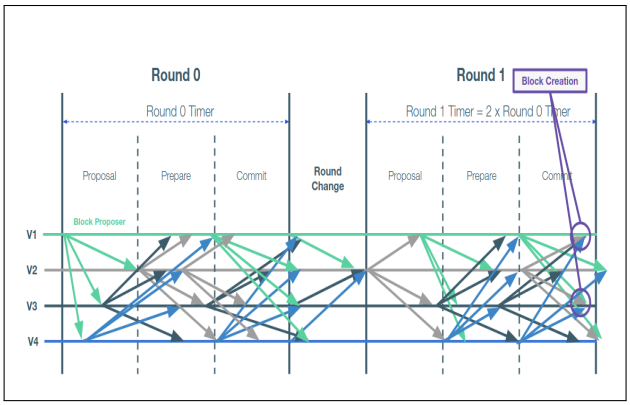

Figure 2.26: Quorum Byzantine Fault Tolerance (QBFT) Consensus [120],[121]

Quorum Byzantine Fault Tolerance (QBFT) Quorum Byzantine Fault Tolerance Protocol is an algorithm whose core design is from [121] by Henrique Moniz. It is implemented by the Enterprise Blockchains of Besu by Hyperledger and GoQuorum by Consensys, which works with a verified set of participants. The network assumption made here is eventually synchronous with a convergence within Global Stabilisation Time which is finite but unknown. The algorithm is non-randomized, with a set of N validator nodes allowed to propose and verify blocks. The protocol can tolerate up to f faulty nodes where $N \geq 3f+1$ The algorithm [120] works as represented in Figure 2.26:

- The protocol works in rounds with a time period frequency for block creation. Each round consists of three phases: PROPOSE, PREPARE, and COMMIT.
- A proposer is chosen in round round-robin pattern among the set of validators and can broadcast blocks to the remaining validators.
- The peer validators, on receiving the PROPOSE, then broadcast the PREPARE message to all the validators. At a threshold of receiving 2N/3 votes, it passes to the COMMIT phase ensuring the whole network has agreed on the block height and validity.
- If 2N/3 COMMIT votes are received, the block is finalized and added to the chain.
- Then, a ROUND CHANGE phase is initiated to select the new validator eligible to propose the next block.

2.2.2.4 Reputation as an Asset

In this section, we analyze an alternate economic value system that can be used as a determinant or a criterion to enforce honesty or regulation inside an artificial society such as blockchain ecosystems. Proof-of-Stake blockchain has seen astronomical growth with close to 245 coins or assets, and Delegated Proof-of-Stake has around 31 assets. These systems have seen at least an average of 5 Initial Coin Offerings (ICO) per day, which is surprising. The Proof of Stake (POS) systems, even though are on the sunrise period they, have some drawbacks as highlighted in [115] as below:

- **Costless simulation:** This means that instead of using the classical systems like Proof of Work which are derivatives of works to handle Denial of Service Attacks like [122] Pricing via Processing or Combatting Junk Mail via Hashcash - A Denial of Service for enforcing the honesty in the system. This technique involves intensive computation cost, but the PoS systems, on the other hand, use the stake and MultiParty Computation (MPC)-based randomization mechanism to select the block proposer, which does not involve any real cost. This convenience of costless security attracts the following vulnerabilities, which will be discussed as follows.
  - **Nothing-at-stake:** A major problem is double-spending attacks in blockchain wherein a user with a 50% majority of either calculation or economic resources in the network can create conflicting chain forks and make the desired fork finalize, benefiting economically. This concern appears graver in the case of PoS chains where a fork is created, the other nodes cannot decide which fork will become canonical, and they will tend to multi-bet. Multi-bet is the process of voting on both chain forks, risking the honest chain's resolution impossible. This can be solved by penalizing or the slashing process of stakes when someone multi-bets, but this can be overcome if there is a significant majority of stakes.
  - **Posterior corruption:** Another striking costless attack is posterior corruption or bribing attack [114]. Here certain nodes can be convinced to collude and attest a malicious chain fork which can be attractive as the participant will be rewarded. This is possible for two reasons: 1) Publicly available data of nodes allows to target nodes with low stakes. 2) Fixed Asymmetric Encryption Keys throughout the lifetime of the participation by the nodes, which can allow this type of collusion. A possible solution adopted is through 1) Key Evolving Cryptography adopted in Ouroboros Praos [111] by David, Bernardo and Gaži, Peter and Kiayias, Aggelos and Russell, Alexander as it protects from forging of the past signature using future keys 2) Another protection adopted by Snow White Protocol [112] and Casper FFG [123] is through checkpointing where a ledger can be finalized to avoid corruption attacks.
  - **Long-range attack:** This attack consists of forking a malicious chain and growing it faster to make it valid [124]. In the case of an incentivized network with rewards, this chain can claim the reward by producing a much longer chain faster. In the case of the PoS network, where the rewards are absent, the attack is named a stake-bleeding attack [125] by claiming the transaction fees.
  - **Stake-grinding attack:** The PoS uses pseudo-randomness to select the block proposer instead of calculation-intensive hashing of Proof of Work. The latter

uses an MPC with staking history as public input. The attacker in this scenario manipulates the history to bias the randomness in their favor, as noted in Peercoin, a PoS blockchain.

- **Centralization risk:** The centralization risks with PoW and PoS are likely the same. However, different quantitative computation and economic power factors can lead to the famous Pareto Principle [126]. This principle states that "for many outcomes, roughly 80% of consequences come from 20% of causes, which means in this case that the more powerful in computation or stakes can have more say in the network, which defeats the central idea of decentralization in the blockchain.

So the inconveniences with PoW and PoS systems are leading to alternate consensus protocols such as:

- **Proof of Elapsed Time:** This consensus protocol [127] is used in a permissioned network where the participants are given a fair chance to be a winner. Each node or participant has a different timer, and they wait for a random time. The first participant to finish waiting then proposes a block. All these executions happen in a trusted execution environment which Intel has proposed. The advantage of the protocol is energy efficiency, but this requires specialized hardware, which is a constraint for this protocol
- **Proof of Activity:** This protocol [114] combines PoW and PoS, which operates in two phases. In the first phase, the Miners or Nodes participate in puzzle-solving and earn rewards. The block's header mined is used as an input for pseudo-random selection of validators for signing the block. The probability for a signer to be elected for validation is based on the stakes held by the nodes. These protocols, however, suffer from the same PoW and PoS disadvantages discussed earlier.
- **Proof of Capacity:** This protocol implemented in blockchain Signum uses Storage space instead of calculation-intensive Proof of Work (PoW). In this protocol, each Node uses a Hard Disk Drive (HDD) to generate data called plots necessary to generate blocks. Then they join a pool and use their forged blocks to decide if the value found respects a constraint, as in PoW. During this process, they remain idle and use pre-calculated data to perform this procedure. This protocol is similar to PoW but a little lighter as computational energy is lost. However, disk space is cheaper than Application-specific-integrated-circuit (ASIC) miners in PoW.
- **Proof of Location:** In this protocol, a device's location coordinates are used to establish trust in the protocol. This protocol overcomes the conventional location services, which are unsecured and hard to rely upon. Instead, this protocol uses a permissionless and autonomous radio network that offers secure location services through time synchronization. This protocol has been implemented by Foamspace blockchain, which has deployed a secured network in the Brooklyn Navy Yard Zone. It consists of specialized hardware called Zone Anchor or Nodes, which has the components of a PCB stack, battery, LTE Modem, and two antennas. These zone anchors are used to form a decentralized location network. This protocol has advantages such as being open, trustless, and verified. But relies more on location and hardware specification, making it less widely adopted.

- **Proof of Space:** A dedicated memory or disk space is allocated to the node whose voting rights in the network are equivalent to the allocated storage space. The hash function of SHA256 is used along with ChaCha8 [128] and BLAKE3. This protocol is used by the blockchain Chia, a more sustainable and secure blockchain. The disadvantage of this protocol is the same as Proof of Capacity which makes the disk space cheaper but a not-so-straightforward consensus with hardware requirements.

These protocols presented along with earlier protocols require an extrinsic factor of calculation such as power, memory space in the form of HDD or RAM, location-enabled through specialized devices, or the stake as an economic power, which is not straightforward or less democratic with specific entry requirements in the network. This has led to certain protocols using intrinsic values like, for example, Reputation.

As per Merriam-Webster, Reputation is defined as "overall quality as seen or judged by people in general." Warren Buffett says, "It takes 20 years to build a reputation and five minutes to ruin it. If you think about that, you'll do things differently," which describes, in a nutshell, the core principle of Reputation-based consensus. Proof of Reputation and Proof of Authority are analogous, as the accounts known as validators are held accountable here. This is quite suitable for consortiums where accounts are traceable among the consortium members. This model uses Reputation, which can be diminished or incremented based on their actions in the network affecting their brand as a consequence of their behavior.

The following section discusses some works that use this abstract and subjective quality of Reputation in blockchain consensus.

In the work by Swamynathan, Gayatri and Almeroth, Kevin and Zhao, Ben [130], they attempt to build a reputation system for distributed infrastructures. They define Reputation as a statistical estimate of a user's trustworthiness computed from feedback given by previous transaction partners to the user. As in Figure 2.27, a service Requester R uses the reputation profile of provider P to determine whether to transact with P. Rating schemes are binary ratings like 0 indicating bad, 1 indicating good or subjective ratings like Very Good, Good, Ok, Bad and Very Bad. They outline the process of the reputation system

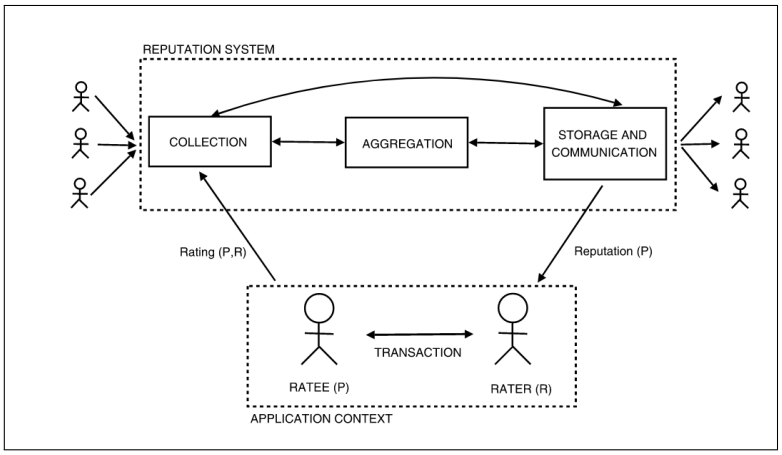

Figure 2.27: Reputation System [130]

through the process of:

1. Collection: Ratings are generated by users undertaking transactions.
2. Aggregation: The user's reputation from various sources is aggregated to form a profile.
3. Storage and Communication of Reputation data: Constructed Reputation profiles must be stored efficiently and securely without tampering.

They propose the solution using the Lorenz curve and Gini Coefficient to address the reasons for erroneous and misleading values produced by the reputation system: user collusion and short-lived online identities.

The consensus protocol by Oladotun Aluko and Anton Kolonin [131] uses the interaction between nodes in the network to determine the Reputation associated with each node. The Reputation is used to determine the nodes which can perform the consensus and calculation of new reputation values. Their reputation-based protocol is based on the following principles:

- Reputation is calculated recursively and progressively after each block proposition.
- Reputation calculated recently has more priority than the earlier calculated values.
- All the reputation values are visible to all the community members.

They discuss the work of RepuCoin [132] by Yu, Jiangshan and Kozhaya, David and Decouchant, Jeremie and Esteves-Verissimo, Paulo where the authors discuss adding a hybrid reputation mechanism over Proof of Work to subvert the 51% attack. The computation mechanism, along with a reputation score, is taken into account for finalizing a block. But in this work [131] by Oladotun Aluko and Anton Kolonin, they use a simple protocol concentrated only on Proof of Reputation without any hybrid nature. This protocol works in a strong synchronization setting but considers benign failures through network delays. The rating for a given transaction is given by node i to j, where i is considered the rate provider, and j is the recipient. The total number of nodes in the network is N with at-most f faulty nodes where 3f+1 $\leq$ N. They consider each node P_i, where i $\in$ N. The transactions in the network are denoted as

$$\text{Transaction} = E_{sk_i}, pk_j, r$$

where E_{sk_i} is the encrypted message of i, pk_j is the public key of node j being the recipient, and r is the rating between 0 and 1. To be selected for participation in the consensus group, the members belonging to high reputation values which is 50% of the total cumulative value of the entire network.

Then a leader is randomly selected from the consensus group members from high-ranking nodes. Leader performs the following functions:

1. Propose a block using the pending transactions, then calculate the new reputation value for all nodes based on each transaction's reputation values.
2. Broadcast the commit message to the consensus group.

The commit vote by each node is assigned a weight based on the Reputation of the nodes. A minimum threshold of 2/3 of the total weight in the consensus group needs to be achieved for successful consensus. The reputation calculation by the leader considers not only the current transaction's Reputation but also the earlier Reputation applied to a forgetting co-

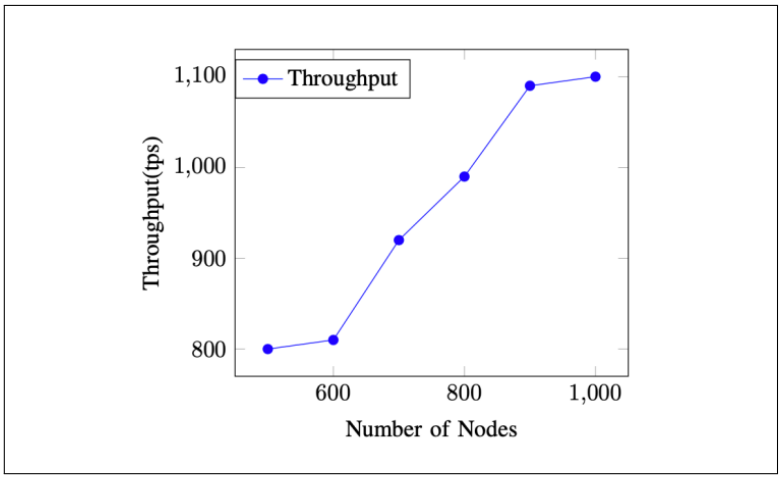

Figure 2.28: Throughput of the Network [131]

efficient prioritizing more current values.

In the implementation results as in Figure 2.28, they notice an increasing throughput with increasing nodes due to a single leader. Still, the communication complexity proposed in the protocol appears counter-intuitive to the results as it can diminish the throughput performance. Although it uses group-based communication, the number of members in the group is not fixed. It is a percentage of members who are selected based on reputation distribution in the network.

In the following work by Kleinrock, Leonard and Ostrovsky, Rafail and Zikas, Vassilis [133], a Proof of Reputation protocol with a Nakomoto or Proof of Stake fallback is proposed. They design a reputation system where each node or party P_i has an associated reputation R_i from [0,1]. Their understanding of Reputation is defined as the probability of a node behaving honestly. The Reputation considered is uncorrelated, where a change in one's Reputation doesn't change others. The protocol works as follows:

1. Each block is proposed and voted in rounds. Each round comprises a small committee sampled C_{BA}, called endorsers, and a further sub-committee is sampled called proposers. This sampling is called a Reputation-based Lottery which will be discussed later.
2. Proposers broadcast the transaction to endorsers.
3. Endorsers sign the received transaction and relay it to other endorsers.
4. When a threshold of $|C_{BA}|/2$ that is half the endorser committee size is reached, it is diffused to the other type of participants.

The communication complexity is reduced since the communication is committee-based or quorum-based. The protocol assumption is a synchronous network.

Reputation-based Lottery: Considering each node or Party p_i has a reputation R_i then a small committee is selected using an algorithm for reputation systems by Asharov, Gilad and Lindell, Yehuda and Zarosim, Hila [134]. This algorithm sorts the participants in decreasing reputations and selects the most reputable ones. For the system to be fair by giving

a fair chance to everyone, they divide the set of nodes into 4 Tiers T_1, T_2, T_3 and T_4:

- T_1 where Reputation $R_i > 0.75$
- T_2 where Reputation $0.5 < R_i \leq 0.75$
- T_3 where Reputation $0.25 < R_i \leq 0.5$
- T_4 where Reputation $0 < R_i \leq 0.25$

The selection from the tier is based on the idea that a node from T_i is more likely to be chosen than a node from T_{i+1} and also in more numbers. Meanwhile, the parties from the same tier have equal chances of selection. To circumvent the issues of a node behaving maliciously upon reaching a high reputation, they propose a fallback hybrid consensus of Proof of Reputation (PoR) and Proof of Stake(PoS) where the digest from the PoR is posted in PoS and resolved if any discrepancy.

Another Proof of Reputation-based consensus protocol variant is proposed by Gai, Fangyu and Wang, Baosheng and Deng, Wenping and Peng, Wei [135]. They assume the protocol to work in a permissioned blockchain with well-identified participants. The broadcast channel is secure, avoiding any man-in-the-middle attack. They identify several attacks common in reputation-based protocols:

- **Bad-mouthing attack:** This attack aims to give bad recommendations or mislead while improving one's trustworthiness.
- **Replay attack:** It reuses or replays the transactions that profit oneself or a group.
- **On-off attack:** It is a kind of stealth attack where they behave good and bad alternatively to avoid suspicion.
- **Sybil attack:** This attack identified by Anderson, Ross J. [63] is aimed at creating multiple identifications (id) where a node can switch to a new id if its reputation is marked as low.

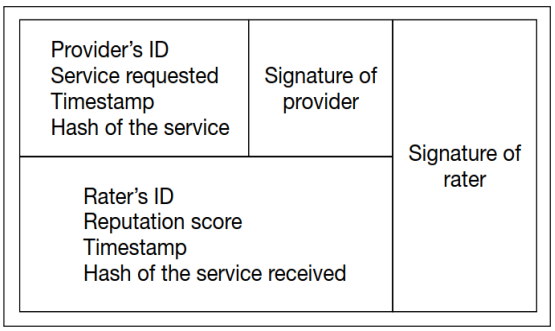

Figure 2.29: Proof of Reputation: Transaction Format [135]

The protocol works by the following steps:

1. When a participant requests a service, the service provider replies with a signed rating message by another participant called a rater.
2. Each transaction is composed of the payload as in Figure 2.29, where there is a signature of the rater, provider of service, the rating message of the service payload, and the signed message of the service.

3. After the rating calculation, this transaction is broadcasted to the entire network, relaying the information and cross-verification by others.
4. Reputation scores are represented as $x_{ij} \in 0,1$ where 1 means satisfied and 0 otherwise or a real value $x_{ij} \in [0,1]$.

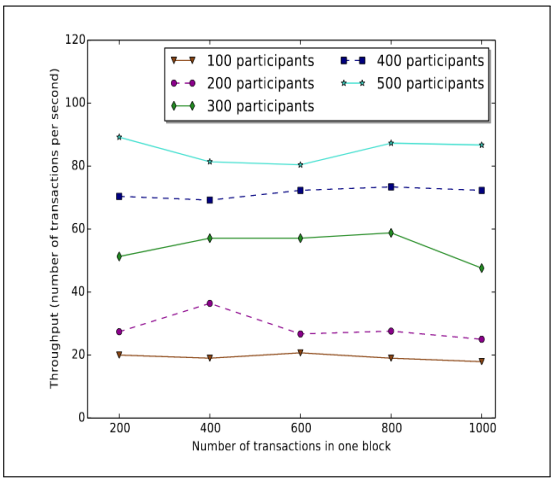

Figure 2.30: Proof of Reputation: Throughput with different block sizes and participants [135]

They devise algorithmic filters to test the message's authenticity, self-rating attacks, or bad-mouthing attacks to prevent malicious action. The set of verified transactions is used to evaluate a node's trustworthiness. Then based on that score, a ranking list is generated, and the participant with the highest trust is allowed to propose a block. The implementation results show that the throughput, as in Figure 2.30, increases with more participants as the authors claim the production of more messages leads to faster consensus. But on the other hand, it may be a bit contradicting as more messages can lead to more communication overhead. Each set of normal service transactions produces a rating transaction that needs to be broadcasted to the whole network. Also, choosing a single leader to propose a block affects the liveness in case of failure, and no fallbacks are presented, which might be a drawback. However, transaction filters like signature verification are an excellent technique to avoid the earlier listed attacks but can introduce certain computational overhead.

2.2.2.5 Byzantine Altruistic Rational (BAR) Fault Tolerance

We have discussed protocols that operate in artificial societies or blockchains with single administration or commonly motivated nodes. But if the network is deployed across multi-administrative behavior, then each set of participants has a reason to deviate from the expected behavior. The Byzantine Generals' problem can be applied in the case of a single administration, but multi-administration needs a new model. The Byzantine Model is too general and cannot be applied in multiple-administrative domain deployment or multi-chain or Interoperable chains as in a blockchain context. In the BAR model, Byzantine and Rational failure are considered, where the Rational understanding is a self-interest behavior from a particular node. We discuss certain BAR model consensus from a set of contributions. [136, 137, 138, 139].

A Byzantine Altruistic Rational Modeled Fault Tolerance system for cooperative services spanning Multiple Administrative Domains (MAD) as in Internet was developed by [138] Aiyer, Amitanand S. and Alvisi, Lorenzo and Clement, Allen and Dahlin, Mike and Martin, Jean-Philippe and Porth, Carl. They propose an approach for tolerating Byzantine actors as in the classical BFT protocols and rational actors who are selfish nodes aimed at self-improvement or benefiting oneself.

2.2.2.5.1 BAR Model: The Model has three sets of participants which are:

- **Altruistic:** They follow the suggested protocol exactly or behave correctly.
- **Rational:** Nodes are self-interested and maximize following a utility function. Utility function considers a node's costs like computational cycles, storage, network bandwidth, the overhead associated with message communication, power consumption, or financial sanctions.
- **Byzantine:** These nodes show arbitrariness as they do not follow the protocol or may malfunction due to hardware or network failure.

The BAR model aims to guarantee safety and liveness as in Byzantine Fault Tolerance for "rational and altruistic nodes" contrary to "correct nodes," which is a broader definition. They classify two types of protocols for the BAR model:

- **Incentive-Compatible Byzantine Fault Tolerant Protocols (IC-BFT):** It guarantees the safety and liveness properties and incentivizes rational nodes to ensure they follow the protocol.
- **Byzantine Altruistic Rational Tolerant Protocols (BART):** This is devoid of incentivization but guarantees safety and liveness even in the presence of rational (selfish) actors. IC-BFT protocols are a subset of BART protocols.

For liveness, under the BAR model, the system makes additional assumptions like:

- Incentivisation for the nodes to be synchronized as possible and penance or slashing scheme if not.
- Second is node A sends a non-Byzantine message to node B at time t, then receives the message at time t + *max_response_time*.

Rational nodes, in particular, have four further technical assumptions:

- Rational nodes receive long-term benefits from the protocol.
- They are conservative when computing the impact of Byzantine nodes.
- If a protocol follows a Nash Equilibrium, rational nodes will abide by it.
- Rational nodes do not collude, but if they collude, they are classified as Byzantine.

A three-level architecture for Byzantine Altruistic Rational Tolerant (BART) protocols is proposed in Figure 2.31. In Level 1, the primitives level provides incentive compatible - BFT versions of reliable distributed services like Terminating Reliable Broadcast (TRB) [68] as in Figure 2.32. In TRB, only the sender proposes a value for a particular instance. Then it can be accepted by the non-Byzantine nodes or rejected by obtaining a default value. Level 2 work assignment, as in Figure 2.31, allows the building of a system where work is assigned to

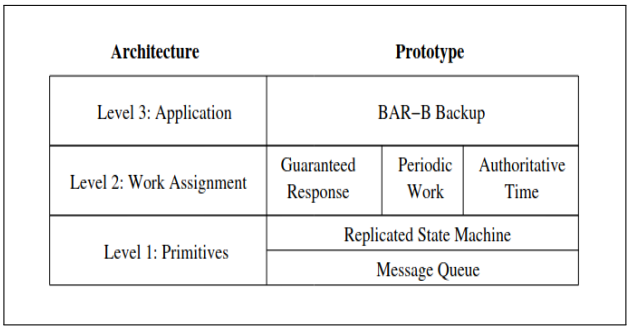

Figure 2.31: BAR System Architecture [138]

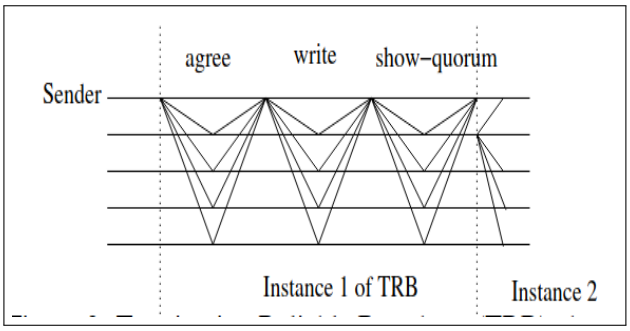

Figure 2.32: Terminating Reliable Broadcast Phases [138]

nodes instead of executed by all nodes. A guaranteed response protocol generates verifiable proof that a node failed to respond to a request. In Level 3, the application level implements the desired service using the levels underneath.

Accountability is mentioned in BART services by establishing a strong identity of nodes and restricted membership to enforce meaningful incentives. Proof of Misbehaviour to detect malicious activity is also put in place in the system.

The authors in [138] use the above BAR model along with PBFT consensus algorithm [73] to design a replicated state machine called BART Asynchronous Replicated State Machine (BART RSM). The BART RSM is based on four guiding principles:

- Ensuring long-term benefit to participants: Rational nodes must be incentivized to guarantee honesty and due diligence by them. So the leadership role is rotated to all participants to ensure every node has an equal chance.
- Limit non-determinism: Non-determinism can open the participant to multiple behaviors, such as normal behavior, abnormal behavior, or procrastination which needs to be minimized to achieve optimal determinism. So choosing a Terminating Reliable Broadcast instead of a plain consensus reduces the possible outcomes and gives rational nodes fewer choices.
- Mitigate the effects of residual non-determinism: Since non-determinism needs to be minimized, they propose two techniques: 1) *Cost Balancing* where a rational node would be indirectly motivated to follow the protocol by making the other choices to be costly in term of utility. 2) *Encouraging Timeliness*: Since the system operates in an asynchronous network, the peers of a node can judge whether the node's behavior is

early, on time, or late and then recommend sanctions or punishments.

- Enforce predictable communication patterns: The nodes are encouraged to participate in all the protocol steps and not selectively participate for self-profiting.

So this work presented the BAR approach to Multi-Administrative Domain systems, particularly for a replicated state machine.

A new Byzantine Altruistic Rational Gossip Model is proposed in [136] by H. Li and A. Clement and E. Wong and J. Napper and I. Roy and L. Alvisi and M. Dahlin. The traditional gossip model data is disseminated with random peers, which is non-deterministic. Still, in this proposed model, a pseudo-random node is selected, which is unpredictable and achieves rapid convergence as the latter. The reason for defining this model is to be robust against byzantine nodes or colluding nodes, which may not be the case in traditional models. The model here assumes a Peer-to-Peer network where a video stream is telecasted using this network. The peers are of three types: 1) Altruistic, 2) Rational, and 3) Byzantine Nodes. The broadcaster of this network is considered to be altruistic, meaning it follows the specified protocol at all costs. On the other hand, Byzantine nodes do not gossip about the information and discard it all the time. Non-Byzantine, unlike altruistic, does not deliver at all times but only within a particular period after which it considers the message expired. Finally, rational are the ones that maximize their utility. The utility here means the costs and benefits. The costs incurred in this case are the communication costs versus the ability to play the live stream. The protocol works in the following manner:

1. The model works in a sequence of rounds r where a node delivers the pending message from round r-1. It then concurrently waits for new updates from the broadcaster while exchanging the current non-expired information and is randomly called a Balanced Exchange. Also, it performs an Optimistic Push Protocol of messages with other nodes randomly.
2. In a message exchange, a sender S selects a receiver R. Both exchange their history of messages, update messages, and then exchange keys for accessing the swapped messages. However, a balanced exchange differs from an optimistic push in information exchange. A balanced exchange of messages happens when it sends and receives messages from peers in equal terms or the equivalent number of updates. However, the optimistic push protocol works more to help peers who have expired messages and need to update their current state.

The BAR gossip follows two principles of restricted choice and reward at the end to make it robust against rational or selfish behavior. They *determine choice* within the two protocols of balanced exchange and optimistic push. Restricted choice brings safety properties, which means that if a rational node decides to participate in an exchange, it sends only messages prescribed by the protocol and receives a reward after the final key exchange for data access. Making the reward at the end of the key exchange ensures that the rational node completes the agreed message exchange correctly.

The probable selfish behavior of the rational node is in selecting peers to communicate with. In this, the BAR model uses a Pseudo Random Number Generator (PRNG) to choose the random peer as in traditional gossip protocol but has control over the seed or initial value of the PRNG. Controlling the PRNG with a seed is applied to the Balanced Exchange

and Optimistic Push protocol. The seed value for sender S is <r, BAL>$_s$ where r is the current round, and BAL is the message the sender signs. This way of controlling ensures that no node can maximize one's benefit during peer selection. Finally, they evaluate the protocol and show that it is robust as long as not more than 20% of nodes are Byzantine, and no more than 40% of nodes collude.

2.3 Conclusion

In this section, we discussed the current state of mobility services and their part in enabling the next generation of Mobility as a service where the different modes of transport providers are integrated through a single unified platform allowing data exchange, trip information, and multi-stakeholder participation. The intervention of blockchain on various MaaS mobility architectures was analyzed for car-sharing, data-market place, and air mobility traffic management. The idea of tokenization to regulate vehicle mobility was also discussed.

Further, the certification of mobility data, along with its provenance respecting the privacy and security concerns in the age of GDPR, was discussed. Data Trading services encompassing the production, storage, review, and monetizing of the data through blockchain were explored, handling the system's security and privacy. The system's fairness gave the actors an equilibrated reward and participation. The challenge of the blockchain-enabled marketplace in handling privacy is also discussed, where the Zero Knowledge proof, attributed-based encryption schemes, and functional encryption schemes are invoked to handle the data privacy concern.

Next, some classical works on the distributed system start from the Byzantine Generals problem, Fischer, Lynch, and Paterson Impossibility Theorem, where a consensus cannot be reached in case of even one faulty process. Building on these works, we discuss the most practical approaches to consensus: Practical Byzantine Fault Tolerance, Zyzzyva, a faster variant of BFT consensus, Hotstuff, a Linear and Simpler BFT consensus, and then a fast track BFT protocol called Tendermint. The more recent works in the Proof of Stake consensus family are analyzed from GASPER, a combination of CASPER as a friendly gadget, and LMD-GHOST for fork resolution.

Innovation in consensus through parallelization and reducing consensus participation by quorum formulation are also dissected for further in-depth understanding. As a consortium-based network with limited participants is more focussed in this work, we examine the Proof of Authority and reputation-based protocols where the abstract participants' Reputation is given weightage instead of the monetary aspect. Further extending the approach of consortium-based participants with a different profile of participants being Byzantine Altruistic or Rational is studied through the BAR approach, and specific architectures are designed using an incentivization mechanism to ensure the consensus.

Chapter 3

Decentralized Mobility Services

Negative results are just what I want. They're just as valuable to me as positive results. I can never find the thing that does the job best until I find the ones that don't.

– Thomas Edison

This chapter conceptualizes mobility services using enterprise blockchain frameworks to test their applicability, performance, and benefits. Through this study, we focalize the state-of-the-art BFT consensus algorithms and test their actual implementation scalability. We measure the implementation's readiness and challenges for industrial adoption, especially from a consortium setting. We test two prominent mobility use cases with varied blockchain frameworks and consensus algorithms implied with Group Renault's ambition to adopt blockchain as part of its digital transformation project [140, 141].

3.1 Conception of Decentralized Mobility Service Architecture

Decentralized Mobility has been a long-drawn concept that has spiked recently with more automotive and other transport partners embracing it to make mobility sustainable, environment-friendly, beneficial, and secure. In the wake of the recent trend of the "Software-Defined Vehicle (SDV) Approach," [142] software is used to model the behavior of the vehicle hardware. The increasing complexity of vehicle infotainment systems (IVI) and advanced driver assistant systems (ADAS) has enormous potential to convolute towards blockchain or distributed ledger. It helps gain data privacy, data sovereignty, and monetization of data and services. It enables data standard compliance with European Telecommunications Standards Institute (ETSI), Mobi, regulations like General Data Regulation and Protection (GDPR), ISO/TC 307 Blockchain standard [143] and Health Insurance Portability and Accountability Act (HIPAA). This has been the reason for automakers like Stellantis to adopt a Responsible Sourcing Blockchain Network for tracing the supply chain of cobalt, mica, and other minerals as part of its sourcing. It is part of its initiative to trace the material's provenance and help in the responsible and ethical procurement of raw materials. Not far behind is Volkswagen Group [144, 145, 146], which has exploited blockchain for Vehicle-to-Vehicle communication, Supply Chain transparency for sustainable mobility or Jaguar Land Rover for sourcing its leather supply chain based on blockchain Traceability-as-a-Service.

In the following sections, we define the use-case problem identified by Renault Group and evaluate it from the applicability perspective through the lens of the consensus algorithm blockchain component.

3.1.1 Token Engineering

This section explains the concept of Tokenized economy, which is applied in the arts, finance, or currency basket and can be applied even in the automotive context to represent any unique asset commonly called Non-Fungible Tokens or Utility Tokens. These tokens may be used for incentivization, value exchange, or utility exchange identified in a decentralized platform as an Ethereum Request for Commons (ERC) standard. ERC specifies the technical requirements for Ethereum improvements by a task force similar to Internet Engineering Task Force. Tokens represent an individual state in a decentralized crypto-economic system using smart contracts. These can be possessed by any autonomous actor or human in the form of wallets secured by public key cryptographic systems. Tokens are accompanied by a legal contract while engineering to make peer-to-peer settlement regulatory compliant. We discuss the token standards which are predominantly used in the crypto-economic systems as below:

- **ERC-20:** This standard details the construction of API (Application Programming Interface) Tokens, which have an equivalent value representing any asset, virtual game points, fiat, or stable currency. It outlines the construct of token creation, transfer, token balance management, token validation, and its re-usability from one decentralized network to another.
- **ERC-721:** This interface deals with creating Non-Fungible tokens (NFTs), which can

be owned, transferred, and auctioned by the participants in a decentralized network. This can represent digital, physical, or virtual assets that inherit the ERC-20 functionalities but for tracking and managing unique or distinct assets.

- **ERC-777:** This specification is similar to ERC-20 discussed earlier but enabling the feature of Hooks. Hooks are callbacks or functions executed when tokens are sent and received from a particular account. This requirement of implementing hooks during the tokens' lifecycle ensures that tokens are not lost or stuck in the contract. This specification is backward compatible with ERC-20 enabling the interoperability between ERC-20 and ERC-777 tokens.
- **ERC-1155:** A standard for maintaining multiple token types like fungible (mutually inter-changeable token), non-fungible, or semi-fungible is proposed here. This is a hybrid standard deriving the characters of both ERC-20 and ERC-721, enabled with Hooks for comprehensive functionality.
- **ERC-4626:** This Smart Contract token standard can enable yields or profits to be accrued by staking or holding ERC-20 tokens. These tokens are designed for lending purposes, aggregators, and interest-bearing tokens. Further, this contract standard can be extended to implement the transfer of token shares through multi-signatures enlisted in ERC-2612.

3.1.2 Data Certification Service

The automotive industry has made great strides in terms of connectivity in a car. This has transformed it from a mechanical machine to a connected device on wheels, exchanging data with any outside world entity. Mobility Open Blockchain Initiative (MOBI), a consortium of major automotive industries like BMW, Ford, GM, Renault Group, and Honda, has envisaged creating a Vehicle Identity Standard (VID) for building blockchain-based mobility solutions. VID helps the vehicle be associated with all the autonomous actions in the blockchain ledger, enabling transparency, certification, transportation services, traffic tracking, and fleet maintenance throughout its life cycle. These new mobility standards require the mobility data shared to be immutable, reliable, and traceable, which can be attained by using blockchain technology. Additional privacy, security, and accountability concerns make blockchain a rational choice by the industry [147].

In this work, we propose a consortium blockchain architecture to solve the relevant use case of odometer fraud in the automotive industry. We designed the architecture from two approaches: a Data-Registry and Non-Fungible Token (NFT) based vehicle data certification, followed by its qualitative and quantitative evaluation. This work concentrates on data certification methodology, its implementation, and evaluation, including privacy and security. We don't consider the interoperability concern of connecting the blockchain solution with Layer 2 Solutions, Distributed Databases, or InterPlanetary File Storage, as we can extend the present architecture later. This is essential for improving data exchange and porting in the industry [148]. Still, we ignore them as we focus more on the blockchain consensus and its related study. We formulate our solution to analyze the performance of BFT family consensus algorithms of Clique, PBFT, IBFT, and QBFT. These algorithms are discussed in detail in Chapter State of Art in section 2.2.

3.1.2.1 Use-Case Definition

In this Data Certification use-case, we propose a consortium blockchain of actors like original equipment manufacturers (example: Renault, PSA, Ford, etc.), car resellers (example: Argus), government agencies, and other players like insurance providers responsible for the network creation. The reason for choosing consortium blockchain is that it offers the advantage of faster consensus among verified participants, the privacy of the data stored, and also no transaction costs, unlike in Ethereum or Bitcoin, while maintaining the decentralized nature and transparency similar to public blockchains [149].

We work to satisfy the broader automotive industry goals of car data certification. We propose a generic architecture focusing on odometer data that can be extended to multiple car data types. For example, it can be the state of health of a car battery, mobility services such as virtual keys, location services, or traffic data. The preferred approach of our study is a consortium blockchain solution, as it can enable original equipment manufacturers (OEM), government agencies, or car resellers to participate in odometer data certification. We started with the Data-Registry approach and moved to the Non-fungible Token (NFT) approach after an evaluation phase decision as in Table

Property	**Data-Registry**	**Non-Fungible Token**
Blockchain(BC)	Ethereum	Ethereum
BC Consensus	BFT Family	BFT Family
Internal Smart Contract Consensus	No	Vote based
Database used	Yes	No
Vehicle as actor	No	Yes
ECU embedded client	Yes	Yes
Key distribution	Yes	No
Smart Contracts Usage	Hash storage	Permission, Token economy and Internal consensus
BC data type	Hash	Token based
Validation	Off-chain	Off & On-Chain
Permission & Role based	No	Yes
Node type	Full	Full & light

Table 3.1: Comparison of Data-Registry & NFT approach

3.1.2.2 Architecture Solution

The architecture of the two proposed different systems Data-Registry & NFT approach, along with the considered design decisions common to both systems, will be discussed in the following sub-sections.

To evaluate the choice of blockchain technology, we initially constructed a proof of con-

cept solution based on Ethereum [150], a public blockchain configured for a private environment, and Multichain [151], a private enterprise blockchain solution. It is discussed as follows:

- **Ethereum** [150]: It is one of the most popular public blockchain systems after Bitcoin. Its suitability for our use-case is due to several reasons: It is a public blockchain, but it is flexible to deploy a private network. Also, smart contracts can be deployed in this blockchain, allowing us to build a decentralized system without third-party interference. Vehicles are hosted as light nodes along with OEM admin full nodes to avoid the necessity of having an extensive network of full nodes causing scalability & data costs. It is based on the Proof of Authority consensus.
- **Multichain** [151]: It is an enterprise blockchain, a fork of Bitcoin binary development. It has smart filters to implement certain logic transactions in the blockchain network. The network participants configured as admin finalize the block consensus using a Randomised Proof of Work. Here for each round, a validator is selected in a randomized mode and allotted to construct a block by calculating a proof of work hash. It was analyzed during our proof of concept realization to be computing expensive, and we did not proceed forward for the implementation stage. Our proof of concept implementation, along with the deployment scripts and smart contract filters, are available in our GitHub repository: https://github.com/scyrilnaves/these-datacertification/tree/main/datacertification/multichain

We shortlist Ethereum Proof of Authority (PoA) BFT Type consensus for our blockchain system instead of the conventional Proof of Work available with Ethereum or Multichain for multiple reasons:

- Consensus process is lighter and faster regarding computation, speed, and energy.
- We deal with verified participants in a private network.
- Rate of block processing can be controlled in PoA in terms of the block period.

3.1.2.2.1 Data Registry Approach In the following subsections, we explain the Data Registry architecture's components and their communication along with illustration in Figure 3.1.

Blockchain Network Component The consortium network is private and permissionless. It follows the PoA consensus with all the nodes as sealers. Sealers are nodes that can validate the transactions and create blocks. Our smart contract, termed as *"Contextual Data Hash Registry Smart Contract"*, is deployed to the blockchain during its initialization. It stores the calculated hash of the contextual data in a map data structure with the key as the hashed value of the vehicle id, and the value is the contextual data hash as follows: Map <Hash(VehicleId), Hash(contextual data)>

Contextual Database Component This database stores the encrypted vehicle contextual data after each trip. The contextual data schema comprises vehicle id, timestamp, nonce, GPS coordinates & odometer data. Data is encrypted using the vehicle's private key

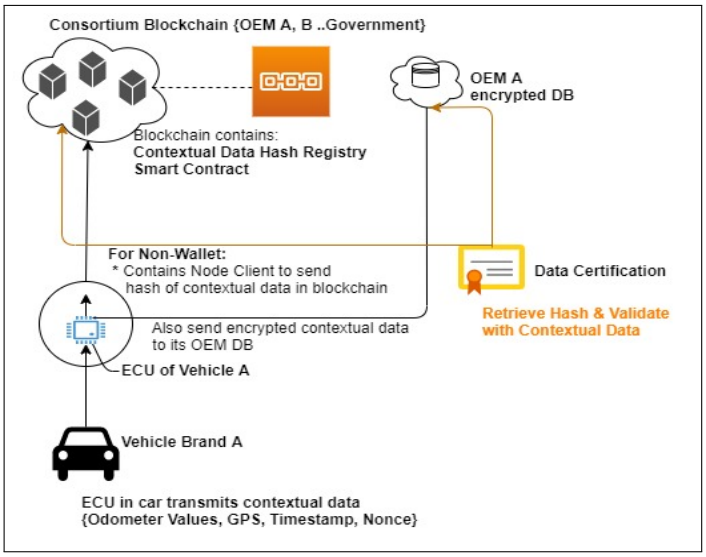

Figure 3.1: Data-Registry Data Certification Architecture

and then with the manufacturer's public key to limit the read access only with it, mathematically represented as:

$$First_{Enc} = RSA_{Encrypt}(RawData, PrivKey_{car})$$
$$Second_{Enc} = RSA_{Encrypt}(First_{Enc}, PubKey_{OEM})$$

Key Vault Component Each OEM maintains its key vault. It stores all its vehicle's public keys as well as its private key in the vault. It uses the vehicle's public and private keys to decrypt its encrypted data.

Vehicle Component Vehicle component consists of a client that transmits data in two steps via the application programming interface (API) calls. At first, the web service passes encrypted contextual data to the contextual database. Secondly, by remote procedure call (RPC), the hashed contextual data is stored in the smart contract within the blockchain.

Certification Component The certification component for each vehicle is limited to each OEM as it holds the vehicle's public key and private key to decrypt the data. It fetches the latest hash value from the <key,value> pair of the map in the smart contract and verifies the hash. Next, the contextual data with the GPS coordinates and the odometer value is verified by contextual data validation process.

During contextual data validation, when a vehicle finishes its trip from Point A to Point B, it traverses a certain distance termed as odometer kilometer. We cross-verify the calculated distance from the recorded GPS coordinates with another data source to validate the data and test its reliability. After these two validation checks, the mileage certification is issued.

3.1.2.2.2 Non-Fungible Tokenised Approach The architecture of the NFT odometer-token economy will be explained in the following as highlighted in Figure 3.2.

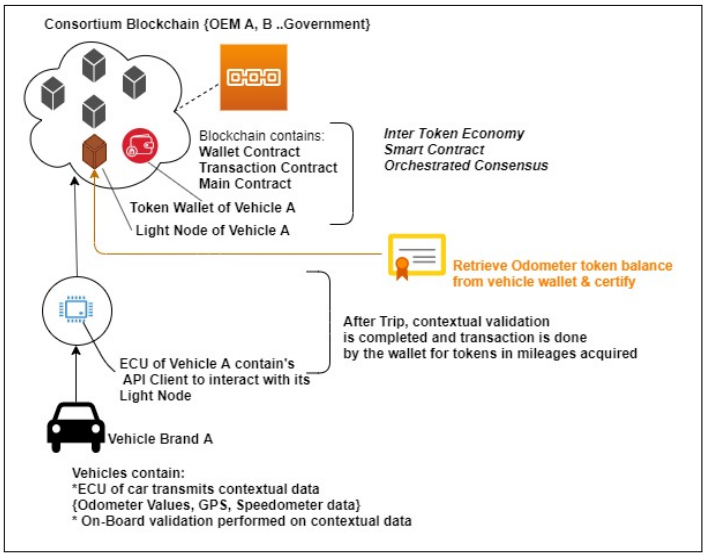

Figure 3.2: Non-Fungible Token Data Certification Architecture

NFT Based Token Economy Network We use the same *Ethereum* PoA consensus for the network but transform it from permissionless into a permission-based hierarchical mileage token economy. We induct a vehicle node as a light sync node in the network. Light sync nodes are not embedded in a vehicle's ECU but are hosted along with other full admin nodes. The reasons for not embedding the Blockchain node in the vehicle's ECU are that Vehicles cannot afford the huge data storage, network connection, and computing power necessary even for a light node. So, a hybrid of light vehicles and full admin nodes is inducted into the network.

Transformation of this network to an odometer economy needs the enforcement of rules for permission & role grant, consensus & workflow, token creation & transfer, which is accomplished by the smart contract. After acceptance by consortium members, deployment of wallet, transaction, and main smart contract in succession creates the desired token economy. In the following, we explain each contract and its significance.

Wallet Contract: In this contract, we enforce *permission* and *token-economy* functions along with their data storage. It deals with permission handling of admin or non-admin nodes, token-economy operations like the initial token generation in the network, and the criteria to distribute tokens to a non-admin node.

Transaction Contract: Transaction contract handles the consensus termed as *Inter-network smart contract orchestrated consensus*. This contract handles the voting for each operation of 'permission' or 'token transfer' and fulfills it after the voting majority is obtained. At the blockchain network level, we already have Proof of Authority consensus with only the admins (consortium admins) as sealers. But here we explain consensus with *'respect to token economy'* realized through the smart contract.

During the network deployment phase, the admin and non-admin nodes are differentiated based on their blockchain public key address stored in the smart contract. After this initial-

ization process, based on consensus, the following interactions are constrained in a token economy: Permission to join or leave, Elevation of the role, Token request & reception, and Destruction of the economy.

Any of the above-listed "message-request" can be initiated based on role-based permission. Admin role can add or remove non-admin or admin nodes' participation in the token economy, create and transfer tokens for its trade-in mileage among wallets, and vote in the internal consensus for each "message request." Non-Admin role functions are limited to request tokens in terms of mileage completed. Each "message request" is fulfilled when most votes are received. Vote majority is calculated for each "message request" depending on its priority & agreed consensus percentage during the deployment phase. This percentage can range from 51% (approval of token request) to 100% (addition of new participant). This calculation is dynamic as we cannot expect all the admin nodes (100%) in the token economy to vote for each "message-request" considering 1000 - 10000 message-requests/min by different actors in a production case. Dynamic vote majority calculation is represented mathematically as:

$$V(O) = T * A(O)/100$$

where O can be operation type (permission/token related); V is the voting majority for operation; A is the agreed consensus percentage needed for an operation; T is the total number of admin nodes.

Main Contract: This contract is an abstraction layer that inherits the previous two contracts and orchestrates between them. It handles all the vehicle's "message-request" and redirects to the appropriate wallet or transaction smart contract for its processing.

Economy Orchestration After economy creation, a vehicle is inducted into the system by an OEM granting permission with the main contract. It is then subjected to a consensus from other OEMs. After the required vote majority is obtained, the transaction contract forwards the request to the wallet contract. It is then finalized by initializing the vehicle with a '0' odometer token balance in its wallet and attributing it non-admin role.

Then post the trip completion by a vehicle, its ECU client performs the contextual validation and transacts with the main contract for token requests in mileage equivalency. After consensus, the transaction contract deposits the tokens in the vehicle's wallet. The token balance of the vehicle, which increases periodically after each trip, can be known by any non-admin / admin node signifying the certified odometer mileage.

3.1.2.3 Accidentology Usecase on PBFT Consensus

In this work, we implement another varied but similar data certification use case of Renault, an accidentology scenario where the blockchain network is defined as a consortium network of partners involved in an accident claim and alert process. Our node participants (i.e., validators) are Renault - Original Equipment Manufacturer (OEM), Accident Insurance Providers, Medical Services, State Actors for legal and dispute resolution, Police, and other emergency services. The novel concept of this use case is to enable the blockchain as a distributed ledger that stores the accident's transactions. The accident's transaction details are

the latest vehicle speed, vehicle condition, radar information, driver condition, etc. These are contextual data stored in a database maintained by the consortium partners. The hash of this data is stored as proof of existence on the blockchain. We implement this architecture in Hyperledger Sawtooth based on PBFT consensus. This use case has been developed as part of the Smart IoT for Mobility project [152] initiated by Prof. Francois Verdier in 2017, funded by the French National Research Agency. It is a collaborative project between Research Groups of LEAT Electrical Engineers, INRIA Computer Science Engineers, GREDEG Economists, Jurists, and companies like Renault Group and SYMAG group. The motivation of this project is to create a blockchain-based mobility ecosystem ensuring a smart, secure, and sustainable next-generation transport solution.

3.1.2.3.1 Significance of Ethereum Consensus In Ethereum, the client binaries chosen for constructing the private consortium networks are Geth v1.10.7 for Clique and Besu v21.7.2 for IBFT and QBFT consensus algorithms, respectively discussed in detail earlier in Chapter State of Art in section 2.2.

Ethereum Geth (Clique) Ethereum, one of the most prominent public blockchains, has a mechanism to build private networks with a proof-of-authority (PoA) algorithm named Clique implemented in the Geth client. In the algorithm, the creation of a new block is restricted to a fixed set of n nodes called sealers, in which a maximum of $f < n/2$ can be faulty nodes or *Byzantine*. Every sealer can seal a block at a fixed time, but it has to wait until it is not sealed recently for $(n/2) + 1$ blocks until its last block. If a designated sealer signs a block for a particular sealing round, it is termed in-order sealing. On the other hand, if the designated sealer is subject to byzantine conditions and cannot seal a block, any other sealer may propose a block after waiting for the block period, termed as out-of-order sealing. Out-of-order sealing can frequently occur in case of short block periods and significant network time delays between nodes. If the conditions mentioned above are met, it can eventually cause more forks in the network, which is an issue in Clique and lacks chain finalization compared to other consensus algorithms.

Ethereum Hyperledger Besu (IBFT) Hyperledger Foundation has created Besu, a client who have implemented the IBFT [119] consensus algorithm, which has immediate chain finality. Creating a single block at a particular height avoids the problem of the forks. Also, the need for $n/3$ majority for completing consensus makes a forked chain less probable. In this algorithm, out of a set of n validators, an arbitrary node is selected to be a block *proposer*. It is accepted if the other validators validate the proposer as a block creator. A block-locking mechanism is introduced to avoid creating multiple blocks when a supermajority of validators accept the block proposition. Then a new round change is proposed with a new validator for the next block creation.

Ethereum Hyperledger Besu (QBFT) QBFT algorithm was created to avoid safety and liveness issues such as [153] [154] in IBFT protocol where two valid nodes can lock different blocks at the same height. This can be attributed to transmission delay between the nodes, and there is no provision to unlock the blocks in the consensus algorithm. Due to these drawbacks, the Quorum blockchain has developed a variant of IBFT, termed QBFT algorithm [154]. For each round, a block proposal phase broadcasts a pre-prepare message to

the rest of the validators. Other validators on receiving the pre-prepare message broadcast a prepare message. On receiving a pre-prepare message, it then sends a commit message. The majority for each state (pre-prepare, prepare, and commit) is thus $2N/3$. Finally, the proposer inserts a new block into the blockchain after $2N/3$ commit messages. Then the next round is invoked after $2N/3$ round changes. Considering the above four BFT consensus algorithms, we compare them across desirable properties in a distributed system as in table 3.2.

Property	**Clique**	**QBFT**	**IBFT**	**PBFT**
Binary	Geth	Besu	Besu	Sawtooth
Chain-Finality	Probabilistic	Deterministic	Deterministic	Deterministic
Forks	Yes	No	No	No
Block-Locking	No	No	Yes	No
Chain-Reorganisations	Yes	No	No	No
Liveness	Upto $N/3$ failures	Upto $N/3$ failures	Upto $N/3$ failures and Prone to deadlock	Upto $N/3$ failures
Throughput	High	< Clique	< QBFT	< QBFT

Table 3.2: Comparison of BFT Consensus Consortium Blockchains

3.1.2.3.2 Significance of Hyperledger Sawtooth PBFT Consensus This blockchain framework, created by the Linux Foundation Hyperledger, developed Sawtooth as a modular and enterprise-focused blockchain. Sawtooth blockchain consists of a validator, the core of the blockchain peer, one or multiple transaction processors handling transaction business logic, and a REST API providing convenient HTTP communication with the peers. Moreover, Sawtooth supports two main consensus algorithms, Proof-of-Elapsed-Time (PoET) and Practical-Byzantine-Fault-Tolerance (PBFT).

Practical Byzantine Fault Tolerance (PBFT) Practical Byzantine Fault Tolerance consensus is a voting-based algorithm whose implementation in Sawtooth is based on the work of Barbara Liskov and Miguel Castro [155]. PBFT properties are Byzantine fault-tolerant, non-forking, leader-based, and communication-intensive.

3.1.2.4 Evaluation

In this section, we evaluate our architecture from a functional perspective, theoretically comparing the two architectures of Data-Registry and NFT and performing a real experimental evaluation in a cloud environment.

3.1.2.4.1 Functional Evaluation We functionally evaluate our two architectures across different properties to evaluate a decentralized blockchain architecture and a general distributed system as represented in Figure 3.3. The properties considered are:

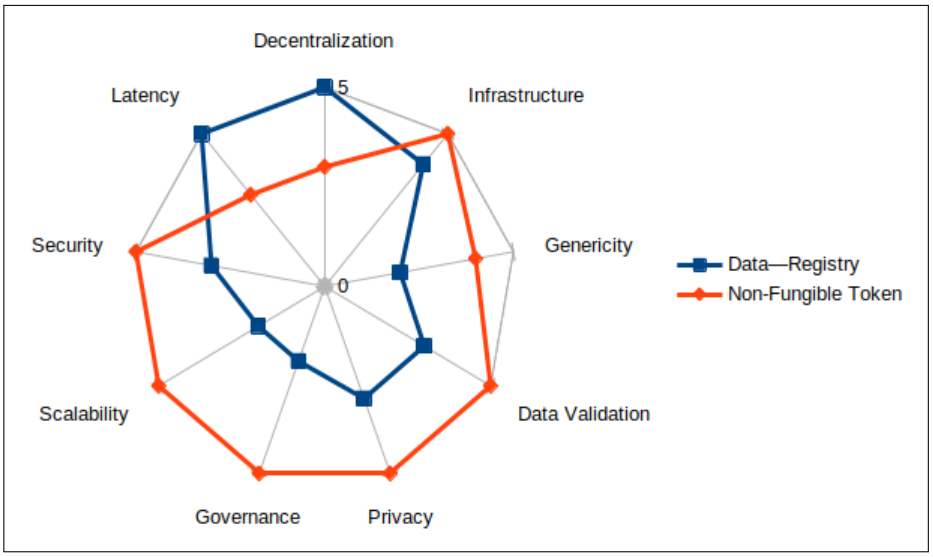

Figure 3.3: Functional Evaluation of Data Certification Architecture

1. **Decentralization:** This measures nodes' participation with equal or rationed privileges in a blockchain, making it entirely or partially decentralized.
2. **Latency:** Transaction emitted by each vehicle, depending on the architecture, needs to be validated, encrypted & decrypted, consensus processed, and then finalized, which needs to be measured on time consumption.
3. **Security:** Authentication and Authorization levels are fundamental criteria for evaluating a distributed system and, more importantly, in a private decentralized network.
4. **Scalability:** Since decentralized systems are more consensus-oriented, adding more participants invariably adds more consistency and processing delay of the transactions, which must be considered.
5. **Governance:** Consortium-based private network naturally invites a governance structure designed to control the network as it is limited to a set of valid participants. Both architectures have a varied level of governance which affects the control and audit in case of adverse issues.
6. **Privacy:** Data distribution needs to be controlled and checked since every vehicle owner's data can be subject to any compromise or can be used for tracking. Both architectures address this issue where the wallet does it entirely, but in non-wallet, there is a potential threat with data storage in traditional databases.
7. **Validation:** Data that enters the system must be genuine and source-tracked to check its credibility. In wallet architecture, data is transformed into a token value which needs to be earned by each vehicle after a consensus opinion. Also, we ensure the cross-comparison of GPS and odometer data sources in both architectures.
8. **Genericity:** This is key to any architecture design as it must satisfy data certification without significant modifications. Wallet architecture is more generic than non-wallet, with the ability to certify any numerical data representation.

9. **Infrastructure concern:** Minimal infrastructure benefits cost as decentralized systems aim to share the hardware and profit mutually. Non-wallet architecture carries more components like database and key storage than a wallet, which can incur maintenance and deployment costs.

3.1.2.4.2 Experimental Evaluation Performance measurements for our Sawtooth - PBFT, Ethereum - Clique, IBFT, and QBFT are tested in the TAS Group cloud infrastructure based in Sophia Antipolis, France. The underlying infrastructure hosts a Kubernetes cluster, enabling us to create Kubernetes pods, each hosting a blockchain node. We monitor the test performance using telemetry from the blockchain node on the Grafana dashboard and MongoDB for test log maintenance.
We assume here a partially asynchronous network as there is a minute delay counting the cloud virtualization, Kubernetes inter-pod routing, node computation load, and peer-to-peer factor. We do not consider any threat model to be included in our test and assume a mutually beneficial consortium. We vary the input transaction rate against the number of nodes participating in the consensus. Here, we consider the output to be the transactions validated and ordered successfully after the consensus. Code Implementation along with smart contracts, cloud deployment of both Terraform and Kubernetes, transaction clients, Ethereum network configuration files, and test results are released publicly in the GitHub repository: https://github.com/scyrilnaves/these-datacertification

Ethereum The Ethereum implementation of the use case is designed as three layers. It is deployed on the TAS cloud network as in Figure 3.4. They are:

1. **Consortium Network Layer:** Each consortium participant is a sealer in the blockchain private network. These sealers are considered validators who participate in consensus to validate transactions and create blocks. A boot node is installed to connect the other nodes in the blockchain peer-to-peer network. The network enables the smart contract to be deployed via the Application programming interface. The private network is configured to have a nominal gas price as we focus only on the enterprise use case rather than cryptocurrency transactions.

2. **Smart Contract:** Smart Contract deployed on the Ethereum network executes on the Ethereum Virtual Machine of each node. The smart contract is built using solidity.

3. **Client Implementation:** Next, a client based out of the web3 library is built to communicate with the blockchain network. This client will be embedded in the vehicle's Electronic Control Unit, sending the smart contract transaction to the network via WebSocket API.

In an Ethereum context, the main factors that affect the processing apart from the consensus mechanism for a PoA private chain are:

1. **Block Size or Block Gas Limit:** Transactions are collated into a block up to a fixed size. Ethereum is measured by setting a block gas limit, as each transaction is measured not in terms of memory size but in terms of gas, as explained in the earlier section 3.1.2.3.1. A higher block gas limit can accommodate more transactions in a block, directly affecting the execution and validation time.

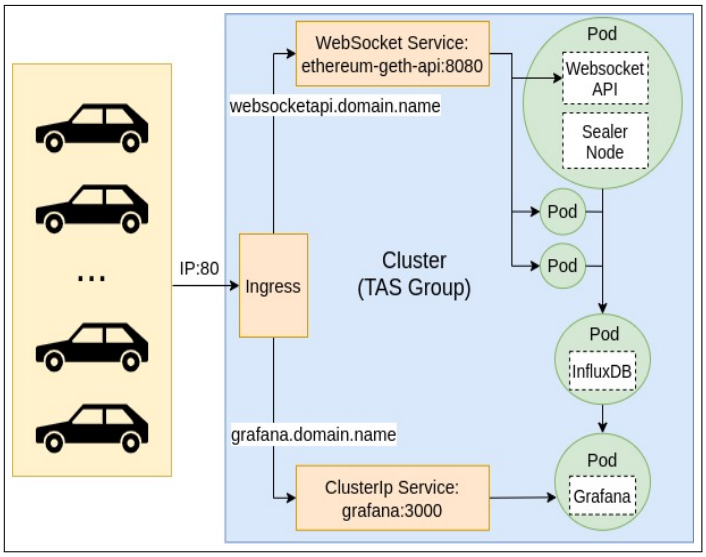

Figure 3.4: Ethereum network cloud deployment

2. **Transaction Type:** Since a transaction is calculated in terms of gas, the higher the complexity of the transaction, the higher the gas for it. For example, adding two numbers can attract lower gas than multiplication operations. The complexity delays the execution speed of transactions on the EVM.
3. **Block Period:** Since all the PoA family algorithms seal a block every fixed time interval, higher block periods can increase the latency in throughput. Lower block periods can increase the throughput of transactions but can affect the consistency of the chain leading to more forks, as we will see during our evaluation. A balanced and short block period is also essential for the network to accommodate the consensus time from validators.

In addition to the above conventional Ethereum-based factors, another factor to note is the creation of forks. Forks due to consensus issues can impact the throughput as transactions will likely get attached to an invalid fork chain and discarded eventually. We explain this issue in section 3.1.2.5, which frequently occurs in the Clique algorithm.

Hyperledger Sawtooth In previous research, [156] showed the performance limits of Hyperledger Sawtooth in the same car accident context as our work. The experiment setup uses a cloud infrastructure, and a distant client sends transactions (i.e., input transaction per second, TPS) according to the use case as in Figure 3.5. The input TPS and different implementation configurations demonstrate the software and PBFT consensus limits. On the other hand, Finalized Transaction Per Second or Throughput (TPS) is the number of transactions that can be validated, executed, finalized, and confirmed in a network after consensus.

3.1.2.4.3 Performance Study Format for the test performed in each blockchain varies a bit since the implementation of Sawtooth-PBFT, Ethereum-Clique, IBFT, and QBFT have completely different organizational and design structures. But the fundamental idea is to

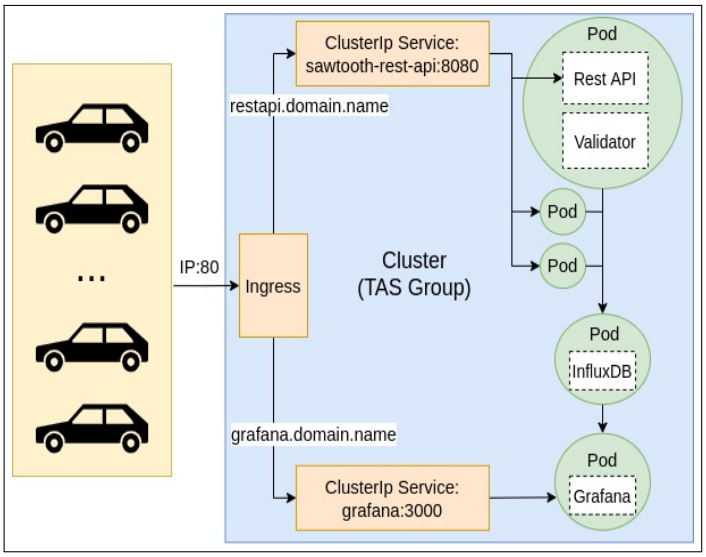

Figure 3.5: Hyperledger Sawtooth network cloud deployment

test the blockchain's performances by varying the number of nodes, also named *scalability*. Performances are also tested by incrementally changing the input TPS (Transaction-Per-Second) submitted to the blockchain node and pushing the maximum threshold it can achieve. Our developed test suite for this work is available online at [157] for more information about code, configuration, and further contributions.

Sawtooth performance measurement A previous in-depth study of Hyperledger Sawtooth revealed a substantial limitation of the blockchain transaction processing [156]. This study demonstrates the same use case and smart contract business logic. The simulation consists of sending car crashes using IoT devices.

Figure 3.6 shows the transaction processing speed of the blockchain by varying the input transaction per second. We see a maximum of 25 finalized transactions per second using the 4 nodes configuration. When the number of nodes increases, the transaction processing speed decreases to 13 finalized transactions per second.

The study on Sawtooth also discusses the possible factors reducing the transaction speed. The consensus and the software are the main factors reducing throughput. The results in Figure 3.6 show the consensus network limitation with the number of node validators. The software limitation is due to single-threaded and intrinsic language (Python) performance issues.

Ethereum performance measurement In this section, we discuss the performance of Ethereum transaction processing along with the Clique consensus algorithm.

In Ethereum, we perform the test for accident transactions totaling 30000 transactions for each iteration in 3 iterations and measure the average throughput results. For Clique, as in Figure 3.7, we see that the output TPS constantly increases for five nodes until it becomes a plateau at input TPS of 2000, outputting 1500 finalized TPS due to Ethereum EVM (Ethereum Virtual Machine) computation overlead and a minute scalability factor. Minute scalability

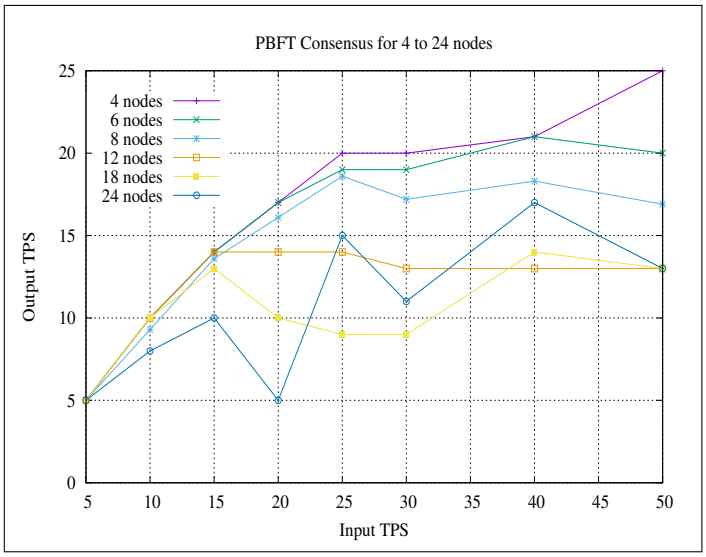

Figure 3.6: Hyperledger Sawtooth PBFT Consensus Performance

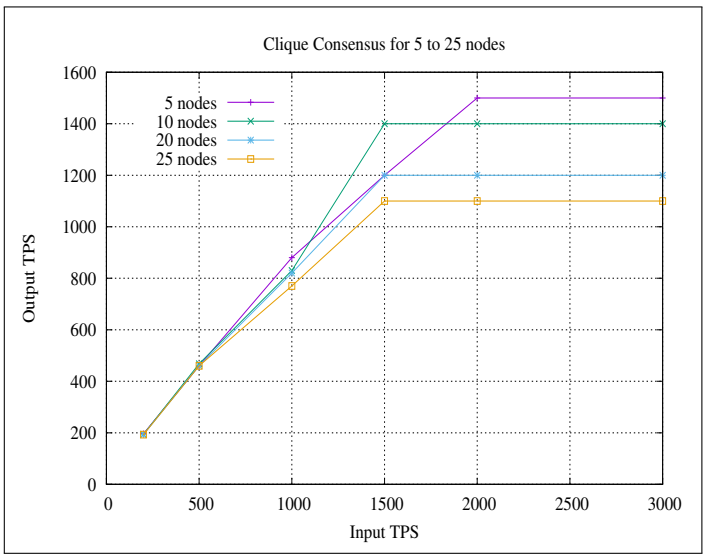

Figure 3.7: Ethereum Clique Consensus Performance

here signifies that the consensus proportion is less at N/2 and the phase is less compared to other consensus algorithms IBFT, QBFT where we see a significant drop with the number of nodes as in Figures 3.8 and 3.9.

Compared to the work as in [158], where the test was performed on the Microsoft Azure network, we noticed a similar performance for Clique and IBFT. Still, the test format varies with no variation across input TPS. Also, the number of clients for IBFT is a single instance that is multi-threaded. Still, in the latter, it was multiple instances that might improve the API processing of transactions but not significantly as the transactions are stocked in the queue immediately before validation and consensus.

In Clique, we can see the saturation arriving quickly at an input TPS (Transaction Per Second)) of 1500 for more nodes as it has less message overhead. In IBFT and QBFT, we see

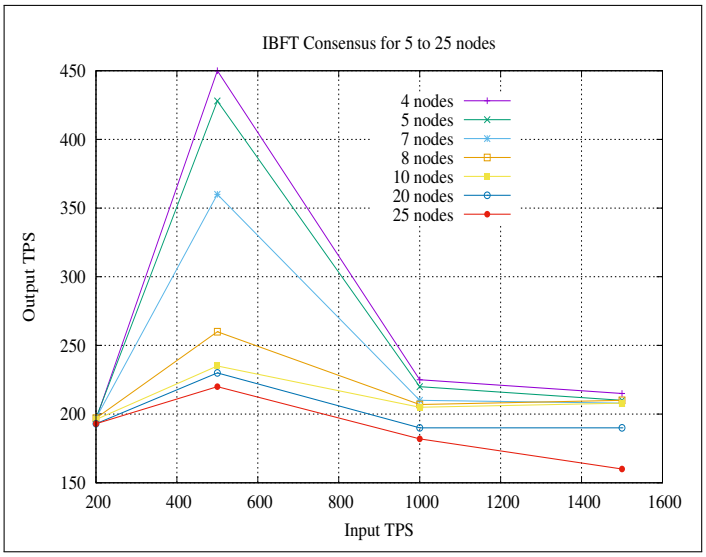

Figure 3.8: Ethereum IBFT Consensus Performance

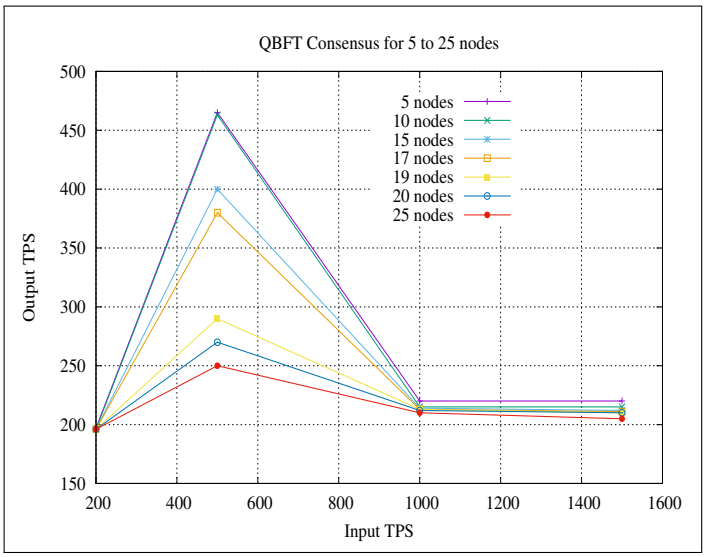

Figure 3.9: Ethereum QBFT Consensus Performance

a gradient increase up to an input of 500 TPS outputting 460 average finalized TPS. At this step, it has a hash calculation load of the Bouncy Castle Library, which it uses in its Java implementation. Aside from the computational hash overload, we see a drop as message overload increases. Still, in IBFT, it is more significant as we have block-locking latency in the consensus to consider here. We see a substantial drop for an input rate of 1000 TPS as the consensus and implementation reasons discussed earlier cause it.

In Clique, we notice frequent forks and chain reorganization, which affects the performance and stability of the chain. On the other hand, IBFT and QBFT have no forks in the chain, which ensures the finalization of the chain. In IBFT, we encounter a stalling issue due to the block locking mechanism. Block locking for each consensus round is implemented in IBFT to improve consistency, but instead, it affects the liveness of the network. In QBFT,

we did not notice any stalling of the network or the presence of forks in the chain.

3.1.2.5 Issues and Root Cause Analysis

In this section, we discuss in detail the bottleneck and stability issues and point out the root causes, which we reported to the community for their attention [159] [160] [161].

3.1.2.5.1 Theoretical Ideal case We can theoretically calculate the throughput by considering the ideal case according to the Ethereum whitepaper [32] a synchronous network with N participants. For example, consider an Ethereum network with a block gas limit of 572235000 and a block period of 2 seconds, and we submit a simple Addition transaction incurring 38149 gas cost. Then the platform can include 15000 transactions in each block. Then theoretically, we should achieve a throughput by:

$$[BlockGasLimit/TxGas]/BlockPeriod \tag{3.1}$$

$$[572235000/38149]/2 \tag{3.2}$$

Ideally, it should be 7200 transactions per second by calculation, but we get a much lesser

	Issue	Client	Functional Level	Rootcause
1	Higher Transaction Rate stalls EVM	**Geth**	CodeBitMap	LRU Cache needs to be improved for frequent calls
2	Higher Block Gas Limit / Complex Transaction slows throughput	**Geth**	EIP 155 Signer	Recovering Public key from Signature
3	Short Block Period Inconsistency	**Geth**	Fork in Clique Consensus	Less Time between peer update and block creation
4	Fork Deadlock	**Geth**	Fork Deadlock unresolved in Clique Consensus	Clique needs to be modified at difficulty calculation
7	Transaction API Failure	**Besu**	HTTP Timeout	Client encounters JVM out of memory error
8	Lesser throughput than Geth	**Besu**	Bouncy Castle library on Keccak Digest	Keccak Hash slows down

Table 3.3: Issues observed in client and root cause

throughput in all the clients, which we discuss below. In table 3.3, we list the various issues we encountered with our clients and our analysis of each. We debug each issue using profiling tools, pprof for Geth implemented in Go language, and then Besu using Java Flight Recorder.

1. **Geth Issues:** In Geth, when many transactions were received, the EVM layer witnessed frequent smart contract calls to be executed. This execution of repeated smart contract transactions was slowed due to ineffective LRU caching. When we reported this to the Go-Ethereum community in Geth V1.9.12, it was subsequently fixed in V1.9.18. The next issue was a drop in throughput observed using pprof for higher than 6000 block gas limit and triple map smart contract transactions which were caused due to the Ecrecover function of recovering a public key from an EIP155 signature needing improvement. Also, in the case of a low block period of 1 second, we noticed a higher number of forks in the chain which affected the throughput, attributed to the short duration between the synchronization operation of receiving transactions and the block creation process.

2. **Besu Issues:** Besu has a similar concern like Parity with lesser throughput and API failure noted throughout the test. In Besu, Java Virtual Machine out-of-memory error occurs as many processes are stuck on a Keccak hashing function. By noticing the flight recorder profile, we observed the usage of a library from BouncyCastle for Keccak hash which needs to be improved for better efficiency in throughput.

3. **Clique Consensus Fork Issues in Geth:** We noticed network fork issues and chain re-organizations during our previous tests. In the case of Clique, the forks are resolved by following the heaviest difficulty chain, but some remain unresolved. A node must choose between the chains of the greatest difficulty summed weight during the fork. But consider a chain has a fork of chain A and B at block number N with a total difficulty till N to be X. Chain A produces two consecutive blocks at height N+1, N+2 with difficulty 1 and 2. Similarly, Chain B produces two consecutive blocks at heights N+1 and N+2 with difficulties 2 and 1. A node not part of any fork at height N cannot decide between the two forks, resulting in a deadlock. This is one of the Clique issues that needs to be solved by modification of fork resolution through dynamic difficulty value other than 1 or 2 [161].

CAP	Clique	IBFT	QBFT	PBFT
Consistency	Low	High	Medium	High
Availability	High	Medium	Medium	Low
Partition-Tolerance	High	Medium	Medium	Medium

Table 3.4: CAP Analysis of BFT Consensus Algorithms

3.1.2.6 Conclusion

Our analysis of the BFT family of consensus blockchains from the above results can be based on three perspectives: 1) Consistency Availability and Partition Tolerance (CAP), 2) Performance 3) Applicability to Industry. We apply the CAP Theorem perspective as in [117] to our above blockchain implementations. We consider the consistency to be fork affinity, ordering, and replicated transactions and messages. Availability means the blockchain can respond and accept a valid transaction to be added to the chain. Partition tolerance is when

the network partition or peering problem doesn't impede the system, and it can recover from it.

We notice, as listed in table 3.4, that the Clique suffers from consistency issues. Different sealers can have different network views and create a fork, eventually leading to an unresolved fork affecting partition tolerance. The chain may progress at N/2 participants consensus, but it would not be easy to finalize as we cannot decide on the ephemeral chain. IBFT, PBFT, and QBFT favor consistency and partition tolerance. As it is subject to multiple phases at each block consensus, a fork is not created, reducing availability. Since it waits for N/3 participant's response at each round, it has to wait if a network or node fails. The performance of BFT Blockchain in our results considers many factors, such as:

1. Number of Nodes
2. Message Communication
3. Leader Selection Phase
4. Consensus Phase Count
5. Implementation Bottleneck

The number of nodes augments the message communication overload. Also, the leader, selection, and the number of consensus phases in each round are vital factors. The implementation bottlenecks like EVM processing for Geth, Sawtooth Transaction Processor, and Besu Bouncy Castle Hash calculation also count for the drop in transaction processing. So Clique performs better in this case as it has a predefined leader operation in a round-robin mode, with fewer phases in each consensus and a minor EVM implementation problem. IBFT and QBFT have an average impact on all the considered factors, with improvement needed in hash calculation. Sawtooth PBFT has a massive drawback in the implementation part compared to others, and its consensus has more phases and communication.

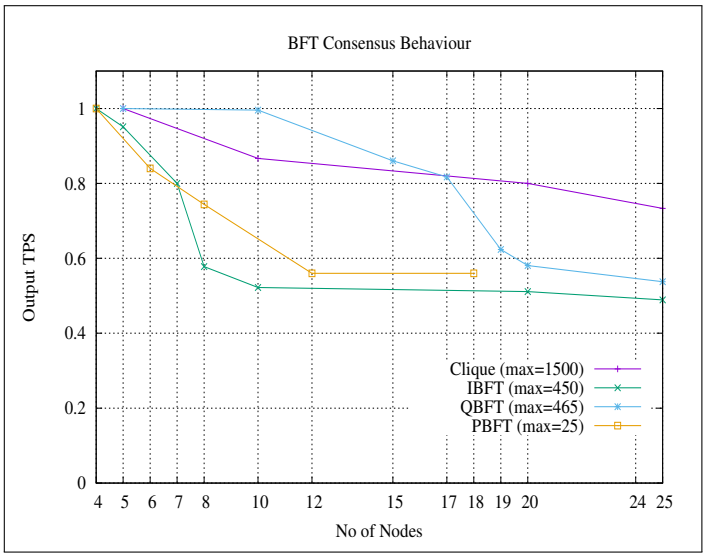

Figure 3.10: Normalised Output TPS Behaviour of BFT Consensuses

In our figure 3.10, we plot the best normalized over the maximum output TPS performance of each variant of BFT blockchain at optimal input TPS to exclude the implementation overload. We consider the Output TPS coefficient of 1 as the best TPS performance for each blockchain and 0 as the lowest TPS. Output TPS Coefficient shows the performance variation as the nodes increase in the consensus process and their behavior. Globally, in the BFT consensus family, we note that the best performances are obtained using 4 to 6 nodes. Except for QBFT, the descent starts to arrive slowly at 12 nodes, but compared to Clique, the processed transaction throughput is less. In this range of nodes, the transaction processing performances of the entire blockchain are best. It lies in the range desired for the number of validator participants for an accident use case. After we surpass 10 nodes, all the BFT consensus blockchains decrease in performance until they reach a stable rate of output TPS. This can be attributed to the fundamental reason behind all the BFT algorithms of communication overhead to prevent byzantine faults. Even though each BFT consensus algorithm varies in the number of phases and leader election, they have the same behavior. We consider each blockchain consensus its best performance with optimum participants and extrapolate the transaction supportable by the network for an entire day; we get the table 3.5. It shows the applicability check of the consensus to the automotive use case in general and the accident use case.

Applicability	**Clique**	**IBFT**	**QBFT**	**PBFT**
TPS per day	129600000	38880000	39744000	2160000
Ideal for Automotive Use-Case	Medium	Medium	High	Low

Table 3.5: BFT Blockchain Applicability transactions

As in the above table, we can conclude that Clique has high performance but suffers from consistency issues. QBFT has an average performance with better scalability, and PBFT has less performance but more consistency. Based on these results and conclusions, BFT consensus is suitable for this automotive use case, with some improvements needed. Still, it depends on the design choice factor to consider CAP, performance, or applicability properties. Further discussion on this data certification BFT consensus analysis is covered thematically under Ethereum Client Testing Architecture, BFT consensus experimental performance, and behavior analysis in the Appendix Chapter, in Section 8.1.

3.1.3 Data Monetisation Service

"Data is the new oil" or "Information is the oil of the 21st century, and analytics is the combustion engine" are breakthrough analogical phrase notions adopted by various organizations like Apple, Amazon, Facebook, Alphabet or automotive innovators like Waymo, Tesla, Renault as they embark the digital age of autonomous driving (ADAS). ADAS systems learn and predict using data generated from a fleet of vehicles where each autonomous vehicle can generate up to 300 Terabytes of data per year comprising of different sensors like Radar, Light Detection and Ranging (LIDAR), Camera, Ultrasonic, Vehicle motion, Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU).

These positive connotations require a certain prudence as data needs the five V's of Volume, Velocity, Variety, Veracity, and Value and the three A's of Analytics, Algorithms, and Applications. These have been well argued, as with the explosion of data by vehicles, computing devices, or the Internet of Things, the data needs to have practice. These data practices or management needed are:

- **Data Provenance:** Knowledge of the origin of data to generate insights and void bias.
- **Data Privacy:** Right to access, rectification, erasure, processing, and portability guaranteed by General Data Protection and Regulation (GDPR) must be met while handling data.
- **Data Protection:** Securing the data is essential as it comes at a high cost if availability is not maintained as earlier; even Toy Story Movie Franchise had the risk of becoming obsolete when a technician accidentally deleted it.
- **Data Preparation:** It is necessary to clean insight extraction to make it AI usable as data quality needs to be augmented.

To ensure the properties mentioned above and practices, the data needs to be incentivized and adopt a decentralized technology like Blockchain where multiple stakeholders like data creators, handlers, exploiters, and aggregators are involved. In our use case, we can consider the data-creator to be vehicles or vehicle proprietors, data handlers to be Original Equipment Manufacturers like Renault, data exploiters like Autonomous Driving Solution Providers like Waymo, Wejo, Momenta, or Oxbotica, as well as mobility data aggregators like DIMO a blockchain-based solution, Otonomo, Carscan.

Mckinsey has analyzed incentivization through data monetization, which states it is a differentiator and is still nascent. They also explore that with the available data, it is necessary to engage with other partners to create new business ecosystems, and it is essential to dissolve sectoral borders. This is one of the business models we would approach in our architecture to create an ecosystem of related businesses. Data Monetization defined by *Gartner* as "using data for quantifiable economic benefit" can be one of the following strategies as highlighted in [162] along with the adopted organizations:

- **Asset Sale:** Sale of Direct Data by Strava, Verizon Wireless.
- **Business Process Improvement:** Value is extracted from data for optimization of one's business process by Lufthansa, ThyssenKrupp, and Deutsche Bank.
- **Product / Service Innovation:** Offering new business services or processes based on data by IBM, Rolls Royce.
- **Product / Service Optimization:** Optimize existing service based on data by Ford, Zara, and Pirelli.
- **Data Insights Sale:** Selling derived knowledge by analytics, visualization by Olery, Sendify, and DealAngel.
- **Contextualization:** Adding additional data over the existing for economic benefits

by Staples and Walmart.

- **Individualization:** Customer data is used to customize the product offering and preferences, enhancing the value proposition by eBay, Daimler, or Netflix.

In our use case, we would adopt data bartering indirectly as the strategy process between the different ecosystem actors. It is a simplified approach, not considering the capabilities of Big Data or Artificial Intelligence Services, which can be very well included in future scenarios for enhanced returns. We concentrate on an extensible, generic architecture that can be recast and shaped vertically, pipelining other technological services or products. Some challenges in Data Monetisation are final data usage policy, legal liability, cross-border data trade, security, and privacy challenges as uncovered in [163], which we will also analyze in our architecture.

3.1.3.1 Decentralised Data Monetisation Solutions

In this section, we compare and contrast two decentralized Mobility Data solutions already in production with a modest market share thriving in an ecosystem of partners for offering "Return on Data" shared with these networks.

- **Digital Infrastructure for Moving Objects (DIMO)**: It is a data-driven decentralized public IoT platform that requires the vehicle owners to install an AutoPi telematics unit in their vehicle, which then submits the data at frequent intervals to the cloud infrastructure. Its infrastructure comprises Polygon Blockchain, Inter Planetary File Storage (IPFS) decentralized storage, Helium decentralized LoRaWAN wireless network, and other additional protocols. Users can submit their data and can gain DIMO tokens as rewards. The platform then utilizes the collected data for getting recommendations on preventive vehicle maintenance, service history, and other sensor records. The shared data is then sold to aggregators who reward and incentivize the DIMO network and vehicle owners. The compromise we notice with this system is that the data lifecycle is a question regarding its procuration, data destruction, data obfuscation by removing sensitive details, and the overhead of installing a device for protocol communication. Also, a conflict of interest may arise with the OEMs initially managing the generation of data, and its exploitation is not reaped by them, posing a risk and a lost business opportunity. It was envisaged to solve the problems of centralization and as an alternative to Otonomo and Wejo, as well as privacy concerns that need more understanding as the data is ported to centralized actors in the ecosystem.
- **Ocean Protocol**: Ocean Protocol is a data marketplace built on parity Ethereum Proof of Authority blockchain protocol. It is a decentralized data exchange where individuals or enterprises share the data shared as ERC721 data NFT tokens by the Ocean smart contracts. Then the ERC20 data tokens are generated for data service to access the published data for a dynamic or fixed price. The existing use cases generated are for connected living, where the elderly patient's data is shared for customized insurance and medical assistance. Also, the health data for heart condition patients is shared with Roche Diagnostics from a CoaguChek IN Range device to the ocean protocol enabling data discovery to other third-party partners offering med-

ical services. Also, a data NFT can be marked as purgatory or unusable if there is an issue with data privacy, quality, or copyright infringement. To preserve the confidentiality of the data, the marketplace offers the compute-to-data where instead of the data transfer, the buyer of tokens can get the pre-computed trained data model to be used, and the raw data stays in the storage of the data marketplace. This protocol is quite comprehensive with no additional hardware requirements, but there is a problem with the data certification, as data shared can be tracked for its provenance. Still, its quality and genuineness are pointers to be considered by them as anyone can share data and earn tokens without pre-emptive checks.

Our data-monetization architecture would solve the above-mentioned concerns regarding data provenance and certification using a streamlined data flow with no additional hardware required. It will also ensure the privacy of the data within a consortium blockchain of agreed and verified partners with mutual benefit for everyone in the network.

3.1.3.2 Decentralised Mobility Data Standards

This section explores the standards around the blockchain data market by different working groups of consortiums and private and technical organizations.

3.1.3.2.1 Mobility Open Blockchain Initiative (MOBI): It is a global smart mobility consortium [164] of mobility providers, technology companies, governments, and NGOs. The consortium has working groups for Vehicle Identity, Usage Based Insurance, Electric Vehicle Grid Integration, Supply Chain, Finance, Securitization, and Smart Contracts. They define the Connected Mobility Data Marketplace (CMDM) standards for Vehicle-to-X (Everything including cloud, infrastructure, or vehicle). It enables vehicle or fleet owners to monetize the data by selling to third-party providers like road conditions or algorithm developers, ensuring privacy and security. They aim to create a standard where infrastructure like roads, bridges, vehicles, and other third-party intelligent transport providers interconnect seamlessly. As data sets by an individual organization are incompatible with another, this standard creates the interoperability for scalable data sharing ensuring efficient monetization.

3.1.3.2.2 European Telecommunications Standards Institute (ETSI) It is a non-profit standardization organization that has published a specification for Permissioned Distributed Ledger in supporting distributed data management. It defines the specification for data discovery, collection, storage, sharing, and computation. It ensures the following requirements: decentralization, trust, incentivization, data provenance, data privacy, integrity, control, sovereignty, and data management automation, ensuring GDPR compliance.

3.1.3.2.3 International Standards Organisation / TC 307 This standard [143] is for blockchain and electronic distributed ledger systems and application, interoperability, and data exchange between users. It prescribes the data flow model for DLT use cases and decentralized identity standards. It also enlists smart contract security good practices and a data interoperability framework.

3.1.3.2.4 European Union Blockchain Observatory & Forum In this report [165], the relationship between blockchain and the automotive sector is discussed for communication and data transaction security. It speaks about the supply chain for components and the introduction of new services which can be built upon. They discuss the necessity of data privacy through zero-knowledge proof, privacy-preserving transaction settlement, and data interoperability.

3.1.3.3 Significance of Data Monetisation Architecture

In line with the above concerns of standards and previous data monetization solutions, we analyze here the necessity for building a new architecture, especially from the mobility context, as follows:

- Vehicle OEMs are hit by the wave of Industry 4.0 with the advent of Digital Twin, where all the configuration and data of a vehicle are stored in the cloud. In close accordance is the launch of Software Defined Vehicle (SDV) [166] allowing to "centralize data with other components of ADAS, bodywork, chassis, and telematics in a Physical computer unit." Also, all the customization and upgrades are made over the FOTA (Firmware Over-The-Air), which is simplified by at most two High-Performance Computers (HPC) for SDV in a vehicle. All these Vehicle-to-Cloud communications, as well as the data that is shared, create the necessity of moving towards a decentralized data monetization where each participant in the ecosystem of OEM like Renault, Component Manufacturers like Qualcomm, Cloud Service providers like Google, and the vehicle owner can own each a piece of the pie (service built on top of data and rewards) offered by the data ensuring mutual benefit.
- Data which the vehicle owner engenders has to create a virtuous cycle where each one starting from the sensor data, can be used by the equipment OEM for improvisation, Vehicle OEM can enable the provisioning of data, a vehicle owner can get service benefits and up-gradation from the equipment OEM. All these actors will create a virtuous cycle of fidelity and continuous improvement complemented by data monetization.
- We design our architecture from a consortium point of view with benefits shared with each actor and guarding data privacy, provenance, certification, accountability, and validity with the closed set of participants, which is needed for an automotive ecosystem.

3.1.3.4 Next Generation Distributed Ledger

Our architecture implementation consideration aims to solve the previously encountered problems in our previous works [158, 167, 118] as well as previous discussions as follows:

- **Consensus Scalability:** As we had noticed in our previous works related to Byzantine Fault Tolerance (BFT) blockchains except Clique and QBFT, other protocols' performance was affected as nodes were increased even from a consortium setting of limited participants. Also, in Clique, the finalization of transactions was a questionable aspect, and from an automotive context, we need finalized transactions as security

cannot be compromised.

- **Interoperability:** Blockchain solutions we had earlier designed were not interoperable with other blockchains limiting their communication to receive external information and create new synergies and services, which is needed from the current Web3 view standpoint.
- **Embedded Smart Contract:** Ethereum smart contract, which we had earlier used to create ERC20 and ERC721 tokens, used an EVM (Ethereum Virtual Machine). It uses EVM for the interpretation, execution, and finalization of smart contract transactions, which adds additional processing bottlenecks apart from the transaction consensus and processing. But we envisage building the smart contract and the blockchain node binary to avoid separate execution in an EVM for faster and more secure processing.
- **Secured Oracle and Off-Chain Communication:** Ethereum-based smart contracts need a Third-party decentralized Ethereum-based Oracle for any external communication or knowledge feed to the smart contract decision logic. These are Chainlink, Band protocol, Pyth Network, and others, which are third-party networks to be depended upon, which defeats the purpose of a decentralized solution. So we need an OffChain worker solution that can connect seamlessly to our internal and external Application Programming Interfaces (APIs) or blockchain networks in the form of Oracles, which we solve in our architecture.
- **Mutual and Divisible Monetisation:** Data that needs to be monetized has to be beneficial and sustainable by attributing rewards for everyone in the ecosystem and creating new services and enhanced customer experience, value addition, and product evolution.
- **Cycle of Data Certification and Provenance:** Data that is shared has to be collated from multiple actors or devices, processed, cleaned, obfuscated for privacy concerns, and finally submitted via an API or data pool even for a simple raw data monetization without any computation. These successive stages must be acknowledged from their nascent stage until the end stage by certifying at each step as a Signature or Hash and submitting to the decentralized protocol, ensuring a data certification and provenance cycle.
- **Privacy By Design:** Our architecture is designed to follow the Article 25 of GDPR principles of 'Privacy By Design' rather than 'Privacy By Default' [168, 169]. In our earlier work on data certification, we followed Privacy by Default through pseudonymization. In this work, we ensure that during data monetization we follow Privacy by Design where the data can be read, accessed, and verified only by the necessary participants without being public to anyone in the ecosystem.

3.1.3.4.1 Blockchain Framework by Design: Substrate In our earlier works, we made the architectural decision of Ethereum Blockchain, a second-generation blockchain comprised of smart contracts to build a distributed application. Meanwhile, Bitcoin is considered the first generation of blockchain as it was focused on the crypto-money transaction with no additional mechanism to create an application or service [170]. But for this design, we choose Substrate [171], which is a third-generation blockchain like Cosmos, Polkadot,

Cardano, or Avalanche. But to be precise, it is not a precompiled blockchain node but a 'modular, extensible framework' that we can utilize to create our custom blockchain node and generate its binary. It can be made interoperable with other chains and also scalable by using the feature of para-chains. Further smart contracts are directly built onto the blockchain node instead of external deployment on EVM. It is built using Rust language and has a core library component called Framework for Runtime Aggregation of Modularized Entities (FRAME), which can be used to inherit or build custom blockchain components on top of it. Also, there is a separation of responsibility design by distinguishing node usual activities of cryptography, network or consensus and custom logics of business conditions, external communications like Oracles and other interfaces. All the signed transactions executed in the smart contract pallets can be invariably called Extrinsic in Substrate terminology. Many enterprise networks like Chainx utilize the Substrate framework based on BABE and GRANDPA consensus, Aleph based on custom BFT algorithm, and Acala, Plasm, Edgware, Moonbeam, and Darwinia based on either AuRa or Nominated Proof of Stake customized with GRANDPA consensus algorithms on public networks.

3.1.3.4.2 Hybrid Consensus In this section, we explain the novel concept of Hybrid Consensus proposed in Substrate, which is comprised of two consensus phases as follows:

- **Block Authoring:** It processes the transactions or extrinsic and constructs them into blocks by the set of validators for each discrete time slot. This is usually performed by a chosen validator in a round-robin selection by Authority Round (AuRa) consensus or Blind Assignment for Blockchain Extension (BABE) scheme. These blocks are then subjected to the addition as agreed chain storage where a block header contains a reference to its parent or previous block.
- **Block Finalization:** Blockchain previously authored contains a chain of blocks and can be subjected to forks where two blocks refer to a single parent block. So to resolve the fork or finalize the chain, we need an additional algorithm to select the best chain, which is the longest chain with higher weightage as in Greediest Heaviest Observed Sub Tree based Recursive Ancestor Deriving Prefix Agreement (GRANDPA). This ensures a deterministic finality where a chain can never be compromised, as only block authoring offers a probabilistic finality.

Cross-Consensus Messaging (XCMP) The Substrate framework being inter-operable, has a mechanism for communicating between chains called Cross-Consensus Message Passing (XCMP), where any custom message can be exchanged between the chains or parachains as called in the Substrate terminology. They are classified into *asynchronous*, which is a request-response without any time guarantees, *absolute*, where the message is exchanged in order and efficiently, *asymmetric*, where there is no response back to the sender, and *agnostic* which has no assumption about the message passed. In our architecture, we don't include these messaging systems as we construct a single unified chain focusing on consensus performance with the capability of extending between chains or parachains in the next version.

Authority Round (AuRa) This consensus algorithm [172] has a list of authorities or validators where the block authorship happens at a time slot or step occurring every time

interval. For each step s, a primary node is assigned in a round-robin methodology which proposes a block as highlighted in Figure 3.11. The node selection is based on a modulus operation of S mod N where each step is S and N is the number of nodes. The proposed block is then accepted by the remaining validators, where only a single selected validator can exist per slot. In case of time drift or network synchronicity, the node time slots can overlap, which can cause multiple proposers per unique time slot. It can cause forks in the chain that need to be resolved by an additional finalization mechanism without it offering only probabilistic finality. The communication complexity for this protocol is $O(n^2)$ when considering the block acceptance phase.

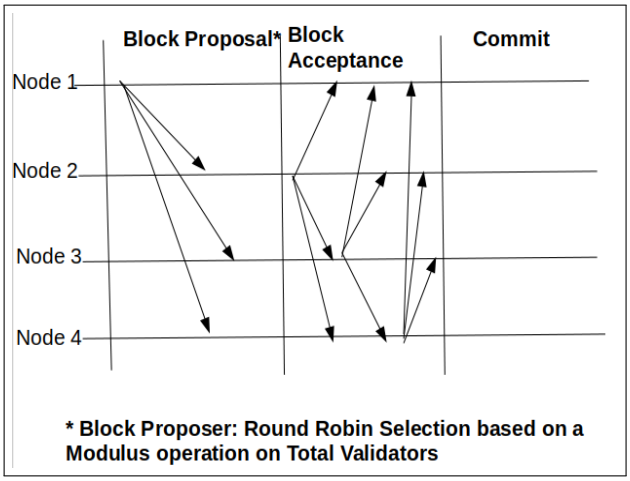

Figure 3.11: Authority Round (AuRa) Consensus

Blind Assignment for Blockchain Extension (BABE) This block production mechanism BABE [173, 174] is inspired by Ouroboros-Praos [111], a proof of stake consensus algorithm. It is also a time slot-based algorithm, as highlighted in Figure 3.12, organized as epochs. Each epoch comprises a set of successive time slots at pre-configured intervals. It overcomes the security threat in AuRa protocol, where a validator at a particular block height can be predicted and can be subjected to attacks as it is based on a modulus operation. The selection of the primary validator is based on a Verifiable Random Function (VRF) with the input of a commonly agreed randomized seed, the current slot number, and the validator's private key. If the VRF output generated by a validator is below a commonly agreed threshold δ, then it is chosen to be the primary validator for the slot. In parallel, a secondary validator is selected as a fallback if the primary doesn't respond for its eligible time slot. The other validator nodes then agree upon this proposed block. But it has only a probabilistic finality and needs to be added with Grandpa finality gadget as the algorithm is termed to transcend as deterministic finality. The communication complexity is O(n) as there is a block proposal phase to broadcast the block.

Greediest Heaviest Observed Sub Tree based Recursive Ancestor Deriving Prefix Agreement (GRANDPA) This GRANDPA finality gadget [175] chain finalization protocol can be coupled with AURA or BABE consensus previously explained for rendering the chain a deterministic finality in cases of forks due to network partition or malicious behavior. As represented in Figure 3.13, it is a protocol where the validator agrees on the chain

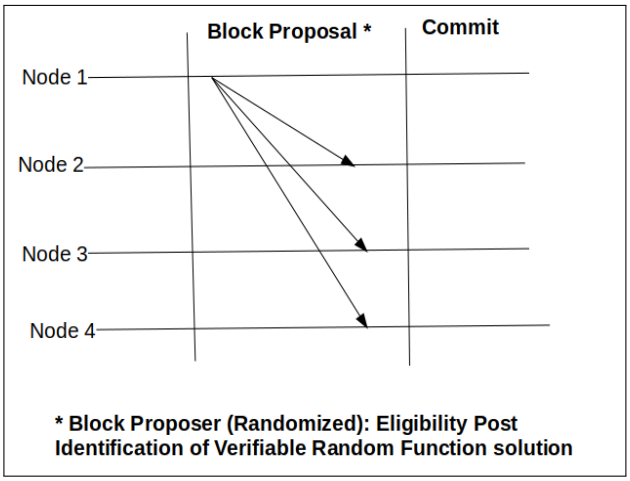

Figure 3.12: Blind Assignment for Blockchain Extension (BABE) Consensus

and not blocks. They apply votes transitively until the block number with the highest votes is chosen to be final. This, in turn, allows the chain of blocks, i.e., a chain, to be finalized at once in a single round. In this protocol, a validator is selected to be the primary broadcast of the highest block. Then during the "pre-vote" phase, each validator endorses a particular block height, confirmed upon a supermajority (2/3) of validators. Then based on the "pre-vote" previous round, each validator casts a "pre-commit" vote upon which they finalize the chain. The communication complexity is $O(n^2)$ based on the messages exchanged in the pre-vote or pre-commit phases.

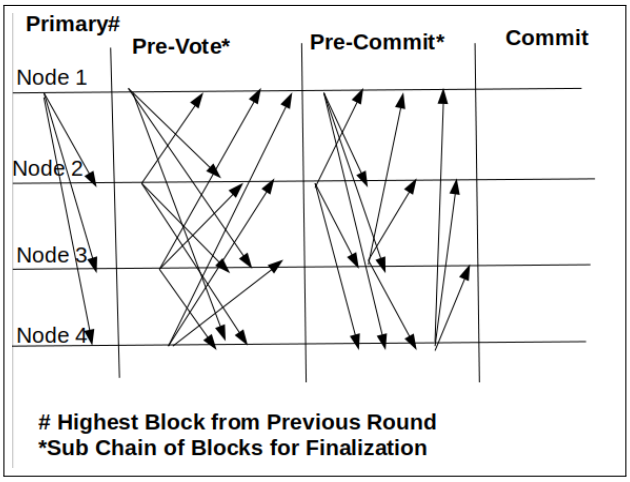

Figure 3.13: Greediest Heaviest Observed Sub Tree based Recursive Ancestor Deriving Prefix Agreement (GRANDPA) Consensus

GRANDPA protocol working is explained in Figure 3.14 where the encircled black block represents the recently finalized chain. While considering the four forks signified by yellow sub-chains, each block has either 2 or 1 as weightage, with fork described as fork 1, fork 2, fork 3, and fork 4. The primary is signified as weightage 1, and the secondary is defined as weightage 2, which indicates either the block proposition by a normal proposer or a fallback proposer. Grandpa algorithm will choose Fork 2 as it has the longest chain with

the highest primaries and is built on the last finalized block. So the factors of chain length, the previous finalized block, and the most primaries will be chosen despite fork 4 having the longest chain with the most primaries but not built on the finalized block.

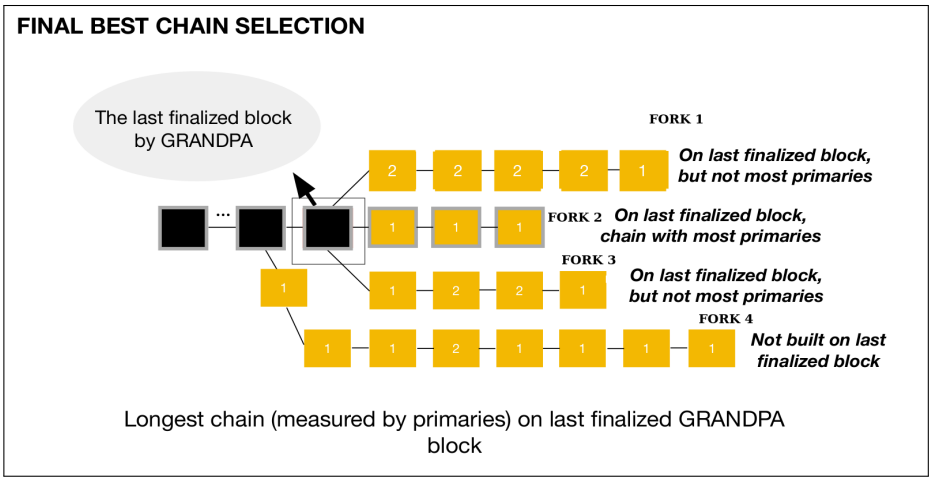

Figure 3.14: GRANDPA Consensus Finalised Chain [175]

3.1.3.5 Use-Case Definition

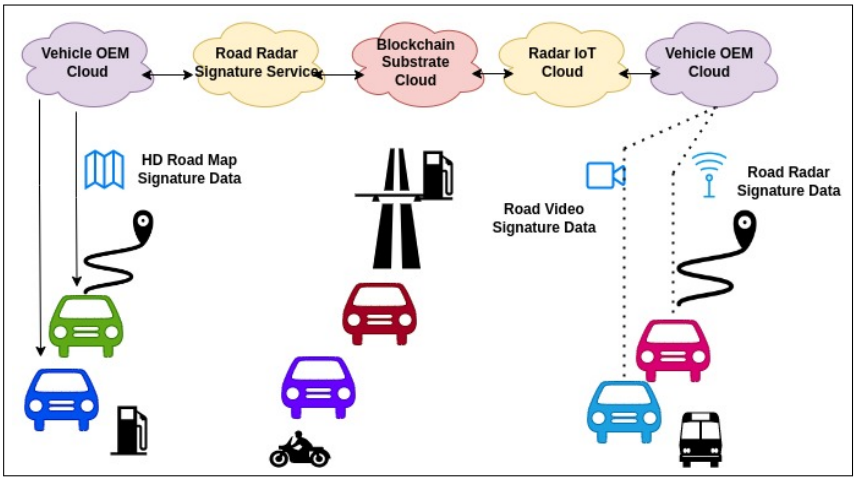

Figure 3.15: Automotive Road RADAR Signature Use Case

In this section, we elucidate the use case of creating an automotive radar data value chain by incentivization and fidelity mechanism. RADAR (Radio Detection and Ranging) data is collected and consumed based on crowdsourcing from numerous vehicles. In the reverse sense, it is offered as a radar data service by enhancing and building map road radar signature data service using it.

3.1.3.5.1 RADAR Significance Automotive RADAR furnishes the essential range and speed data for driver assistance systems, including Long Range RADAR (LRR) adaptive cruise control, automatic emergency braking, cross traffic alert, lane change assist, and

Short Range RADAR (SRR) parking aid, as well as pedestrian detection. The signal processing function involved is Range estimation, Doppler Frequency Estimation, Direction of arrival estimation, and tracking. In normal cases, multiple RADAR channels are required for angular information or to compensate for any information loss. In our case, we adopt a localization-based data collection from multiple vehicles for data fusion and extract any valuable information. The significance of the automotive RADAR market is increasing at 40% Year-on-Year, especially in the premium or mid-segment vehicles. Prominent Automotive RADAR Original Equipment manufacturers include Bosch [176], Continental, NXP Semiconductors, or STMicroelectronics for object detection even at 200 kilometers/hour. Also, for vehicle safety concerns, as we are evolving from L2 autonomous vehicles to fully autonomous L5 vehicles, RADAR is necessary for safety optimization while driving.

3.1.3.5.2 Road RADAR Signature Road signature [176] is a crowd-sourced localization service for autonomous vehicles to detect the relative position of other vehicles and objects in their environment for accurate and reliable localization. As represented in Figure 3.15, the road signature is crowd-sourced or vehicle-sourced in the vehicle surrounding comprising lane markings, gas stations, and guard rails within a decimeter range of the vehicle. The vehicle generates RADAR and video sensor data while in motion, which will be collated at determined intervals via the Telematics Control Unit of the vehicle to the Cloud of RADAR OEM. Radar OEM observes all these received data of RADAR and video to extract the object details and regenerate the environment based on the data. Video and RADAR sensor data complement the rendering of a localization environment. As discussed earlier, the RADAR OEM, e.g., Continental and Bosch, integrate this data with map data to produce an updated and globally consistent map. The generated Road Radar Signature high-definition resolution map, which will be offered as a subscription service back to the vehicle user community, is compromised of three layers:

- **Localization layer:** Vehicles position is determined based on its driving lane and merged with radar road signature, including the video localization map. Also, the object information in the vicinity is compared with the information from the other sensors to get the relative position.
- **Planning Layer:** This layer provides driving planning information comprising the road course, traffic sign, and speed limit, including bends and gradients.
- **Dynamic Layer:** Traffic information, including deadlocks, construction work, maintenance, or parking space availability, is provided.

3.1.3.5.3 Virtuous Cycle of Road RADAR Signature Data As represented in Figure 3.16, the Road Radar signature workflow is facilitated and augmented with acknowledgment, data certification, transparency, privacy, dynamic pricing mechanism accompanied by fidelity and incentivization mechanism. Our data monetization use case will involve the following components realized through the development of pallet smart contracts on the Substrate blockchain framework.

- **Asset Component:** This component creates an asset for vehicle OEM as a Non-Fungible ERC 721 token in the network. RADAR Data of the vehicles for a defined condition of localization, time period, error tolerance, and other miscellaneous meta-

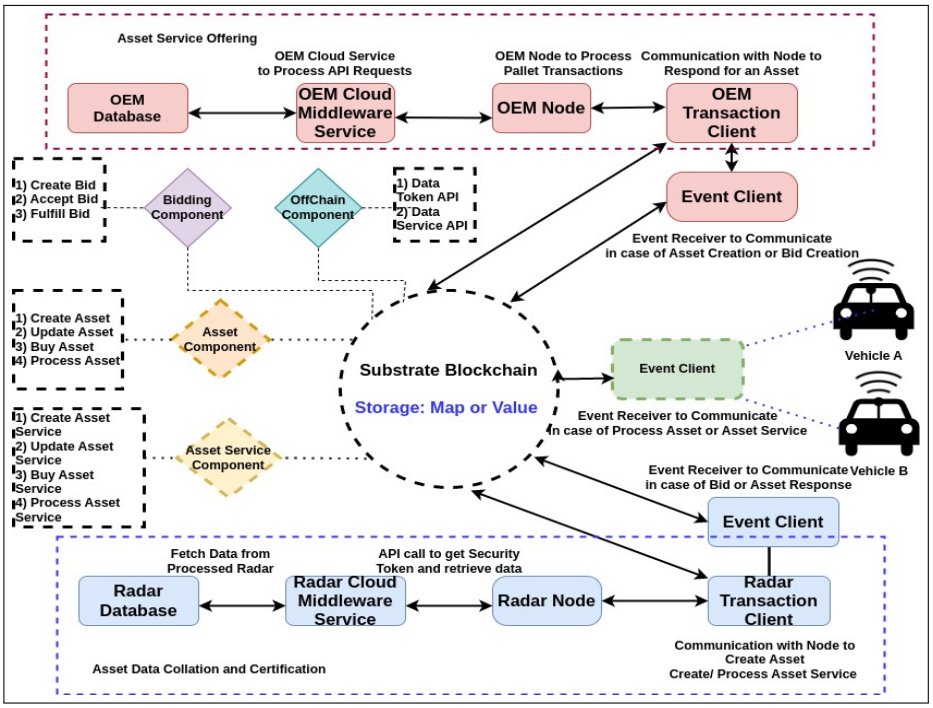

Figure 3.16: Automotive Data Monetisation Architecture

data is represented as an asset along with its floor and ceiling price.

- **Asset Service Component:** The component is similar to the asset component but consecrated towards the Road Radar Signature Map, which is offered by the RADAR OEM to the vehicle users for enhanced safety, autonomicity, and optimized localization information which is accompanied by a ceiling and floor price.
- **Bidding Component:** Each Asset or Asset Service created as an ERC 721 token has its price determined based on a transparent bidding process between the RADAR OEM, Vehicle OEM, and Vehicle Users consented to the transaction.
- **OffChain Component:** As soon as the Asset or Asset Service is sold, data must be offloaded from each vehicle to the RADAR OEM via Vehicle OEM. Vehicle OEM acts as a guarantor or mediator in the blockchain for initial data collation, data cleaning, and obfuscating of any sensitive or private data. These data processing steps are done off-chain in a cloud middleware but closely looped with the blockchain to certify the process and the data.
- **Asset Data Collation and Certification:** Offloaded data to the RADAR OEM is then processed, analyzed, and interpolated with Map data to be offered as a service later completing the asset finalization along with certification along its entire process.
- **Asset Service Offering:** Road RADAR Map signature is offered to the consenting vehicle back in the blockchain ecosystem with an interest of subscriptions collated in the ledger by the Vehicle OEM and transaction completion post the fund transfer.
- **Incentivisation for Everyone:** For each transaction of Asset or Asset Service in-

volving RADAR data monetization and Road Radar Signature Service, each actor in the ecosystem is attributed an incentivization. The asset monetary transaction distributes the commission to the Vehicle OEM, the Vehicle Users, and the Radar OEM. The necessary monetary benefit is given that the data certification, anonymity, and quality are maintained. Moreover, the RADAR data or RADAR signature service represented as tokens ascend in higher price appreciation based on its usage review is submitted to the smart contract.

This completes the explanation of the virtuous data cycle since its inception as an asset from the vehicle to its collation in the RADAR cloud. Completing the closed loop back as an asset service to the vehicle users based on the shared data imbibing fidelity and sustained benefit for everyone in the monetization ecosystem.

3.1.3.6 Architecture Solution

This section looks at the architecture flow in terms of a smart contract transaction sequence diagram for the Non-Fungible Token-based Asset and Asset Service proposition scenarios. The Blockchain ecosystem network comprises nodes representing the Vehicle_OEM, and Radar_OEM as validators with the possibility of extending the membership to other members interested in data. For example, the map provider or the government service as it deals with traffic, road, and toll maintenance. The blockchain network constructed is of a private consortium type where each participant is aware of the public key of the other actors, as each one has a crypto wallet secured by a public cryptography system. In all these explanations, we consider the following principal actors:

- **DataMarket:** Decentralised Smart Contract Pallet realized on Substrate blockchain framework, which orchestrates the entire data monetization transaction subject to hybrid consensus algorithms and its validation.
- **Vehicle_OEM_A|B:** Vehicle Original Equipment Manufacturers like Renault, Stellantis, or Volkswagen, which provide the Offchain cloud infrastructure and participate as a validator in the blockchain consensus. Each participant installs and maintains the blockchain client for data transfer, transaction execution, and liaison of the vehicles.
- **Vehicle_A|B:** Vehicle Users or Autonomous Fleet devices have a wallet-based account in the blockchain process and provide data necessary for monetization and subscribe to the service offered for enhancing driving.
- **Radar_OEM_A|B:** Radar Original Equipment manufacturers like Continental, Bosch, and NXP are interested in the generated RADAR data from the vehicles for enhancing their product offering as well as creating additional business services for customers like Road Radar Signature Map for their end users.

3.1.3.6.1 Tokenized Non-Fungible Asset Data-Set Component The Smart Contract transactions involving the Offchain Sequence for the NFT Asset component involving RADAR data transfer are represented in Figure 3.17 for the initial bidding process, and the asset finalization process is explained as follows:

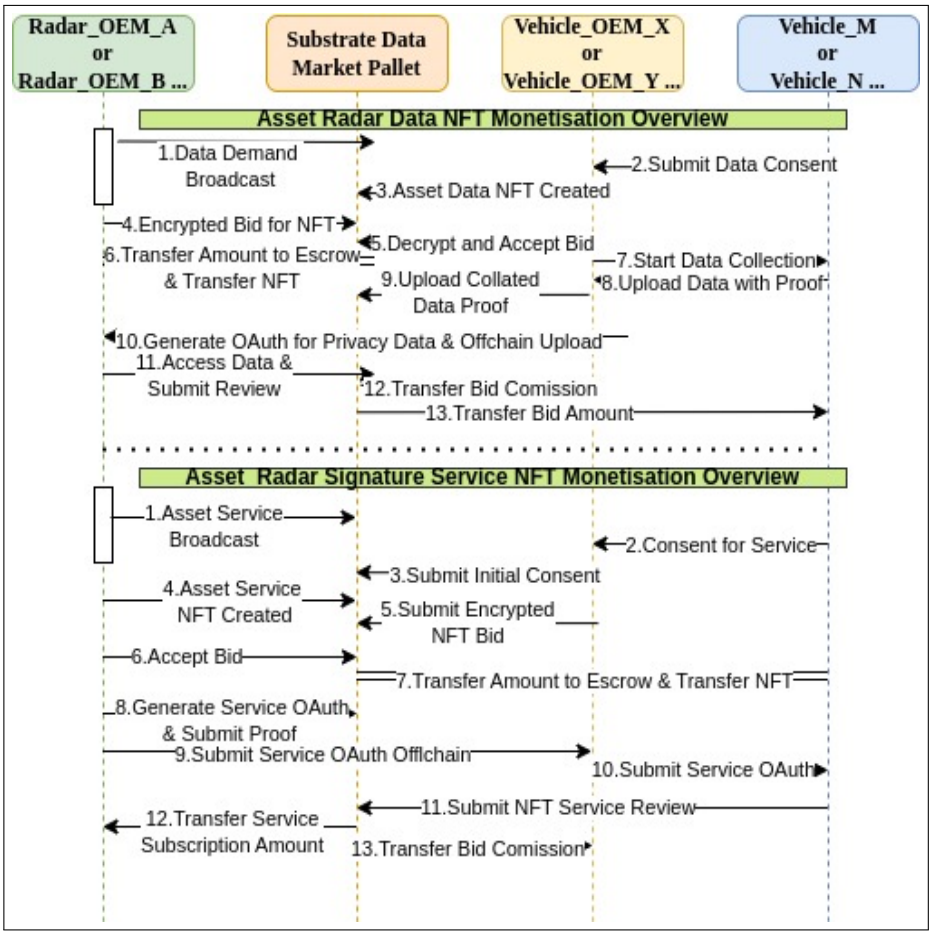

Figure 3.17: Tokenized Asset & Asset Service Monetisation Architecture

- **Data Demand Phase:** The initial phase when the Radar_OEM needs the Radar data, it publishes the demand as an event notification transaction onto the network with the location, vehicle speed, and acceptable price limit requirements. The Vehicle_OEM can then respond to the event by broadcasting the demand to its vehicle clients for its participation contentment.

- **Data Asset Offering and Bid Phase:** Vehicle_OEM_A|B responds individually by creating an ERC 721 token as an Asset with the asset criteria, floor, and roof price. Then to maintain the bid privacy, the Radar_OEM submits an encrypted bid with the public key of Vehicle_OEM_A|B respectively. Then among the bids, one of RADAR_OEM_A is accepted by the Vehicle_OEM_A as the bid of RADAR_OEM_B is not an acceptable price. The fund of RADAR_OEM_A for the bid is transferred to the escrow account as an intermediate transfer which will be transferred to the Vehicle_OEM_A when the bid criteria and the asset are respected along with necessary certification proof submissions.

- **Data Aggregation and Certification:** The Vehicle_OEM_A starts the data collection job from its set of vehicle clients with the necessary criteria. Vehicles start recording Radar data and submitting it with its signature hash as proof to the data pool and smart contract. Then Vehicle_OEM_A processes the data removing the user's sen-

sitive identification data, submits the final hash proof to the ledger, and generates a data access token to a middleware service to be retrieved later.

- **OffChain Data Transfer and OAuth Access:** The blockchain Off-chain worker component then retrieves the OAuth access token from the middleware and submits it as an API POST request to the RADAR_OEM_A if the Vehicle_OEM_A submits all the certification proofs to the blockchain.
- **Fund Transfer including Finalisation of Asset Transfer:** Then the bid amount is transferred by the smart contract from the Escrow to the Vehicle_OEM_A account and Vehicle_Accounts who participated in this monetization process such that the above condition of the access token and proof are respected.
- **Review of transferred Asset:** The RADAR_OEM_A who retrieves the processed data from the data pool utilizing the OAuth access token, reviews the data. The result of the review process is submitted to the blockchain against the Asset ERC721 token issued, which is a fidelity action to augment the price of the data issued by the Vehicle_OEM_A in a later transaction or diminish if the review score is bad.

3.1.3.6.2 Tokenized Non-Fungible Asset Data-Service Component The smart contract transaction sequence for the Asset Service token is represented in Figure 3.17 for the bidding process similar to the asset component. For the sake of clarity, the Asset Service finalization for the Radar Signature Map subscription service is described in Appendix Chapter through Figure 8.9. This is similar to the earlier Asset Data transaction workflow explained but is in the other sense where the vehicles subscribe to the service offered by the Radar_OEM_A|B. It is explained as follows:

- **Road Radar Signature Asset Service Interest Collation:** A particular RADAR_OEM_A who has to build the localization, additional, and planning layer of additional information over the processed data as the Road Radar Signature Map service publishes an event to the blockchain smart contract of its availability. This is then processed by Vehicle_OEM_A|B, where each one gathers the interested vehicle clients for subscribing to this service.
- **Road Radar Signature Asset Service Creation and Bidding:** The RADAR_OEM_A creates an Asset Service NFT ERC 721 Token and the characteristics of the service with the acceptable price range limit.
- **Asset Service Subscription and Proof Generation:** Then each of Vehicle_OEM_A or B expresses the interest in the subscription service on behalf of its clients as an encrypted bid. The Radar_OEM_A accepts the bid of Vehicle_OEM_A along with its vehicles, transferring funds from the vehicles consented to the escrow account.
- **OffChain transfer of Asset Service OAuth API Access Token:** RADAR_OEM_A generates an OAuth access token for its Road Radar Signature Map service along with the hash proof of the service including the raw data location, vehicle speed, and quality. Off-chain Worker retrieves the encrypted OAuth access token from the ledger and submits it to Vehicle_OEM_A.

- **Fund Transfer and Finalisation of Asset Service Transfer:** In parallel, the Off-Chain worker transfers the funds from the escrow for the bid to the Radar_OEM_A. Also, the Asset Service Token ownership is transferred from the Radar_OEM_A to the Vehicle_OEM_A and the interested vehicles.
- **Review of Transferred Asset Service:** The Vehicles or the Vehicle_OEM concerned for the Asset Service token can review the improvement in vehicle driving as a result of Road Radar Signature Map service positively or negatively connoting its price and fidelity accordingly.

3.1.3.7 Implementation

This section discusses our decentralized data monetization solution's development and infrastructure deployment operations.

3.1.3.7.1 Substrate Smart Contract Pallet The Substrate blockchain, as explained earlier in this Section, is an extensible framework allowing the creation of a client node embedded with smart contracts in the form of a pallet module. We designed this smart contract pallet containing ERC 721 token creation logic, updation, ownership transfer, and burn for both Asset Data and Asset Service Road Radar Signature representation. In our case, the Offchain Worker component inside the pallet allows the smart contract to interface with external API, either as Oracles or Token Transfers. Fund transfer operation involving escrow and other actors is also defined with logic for proof and data certification submission handled here. The pallet is included in node runtime for the execution of the data monetization application along with the other default functionalities of transaction validation, consensus process, and state storage. Our code implementation is realized in 4.0.0-dev with node customizations for testing the BABE and AURA consensus algorithm whose code is published publicly.

3.1.3.7.2 Middleware Components The Vehicle_OEM and the Radar_OEM participate inside the blockchain as an actor in the smart contract and node validator. It also participates in Offchain by providing the middleware required for collection, processing, transfer, reception, and OAuth Access token both for raw Radar data and Road Radar Signatures. These business services for Radar_OEM and Vehicle_OEM are implemented in Java JDK (Java Development Kit) 8 language based on Spring Boot Framework. Its architecture is a REST (Representational State Transfer) based Model View Controller middle-ware. It deals with the authentification, authorization, OAuth Access token generation, verification, management, Data Storage, Retrieval, and other API exposition necessary for the blockchain OffChain worker to interact. It also has the API service to encrypt and decrypt privacy bid payload for Asset or Asset Service, which is the Vehicle_OEM or Radar_OEM. The Unified Modelling Language representation (UML) for the middleware is represented in Figure 3.18, which comprises of the following:

- **AuthController:** It is the primary Class interface in the middleware which accepts the Request and redirects to the necessary business service layer for storing assets, asset service data, OAuth token generation, as well as encryption and decryption process for the bidding procedure.

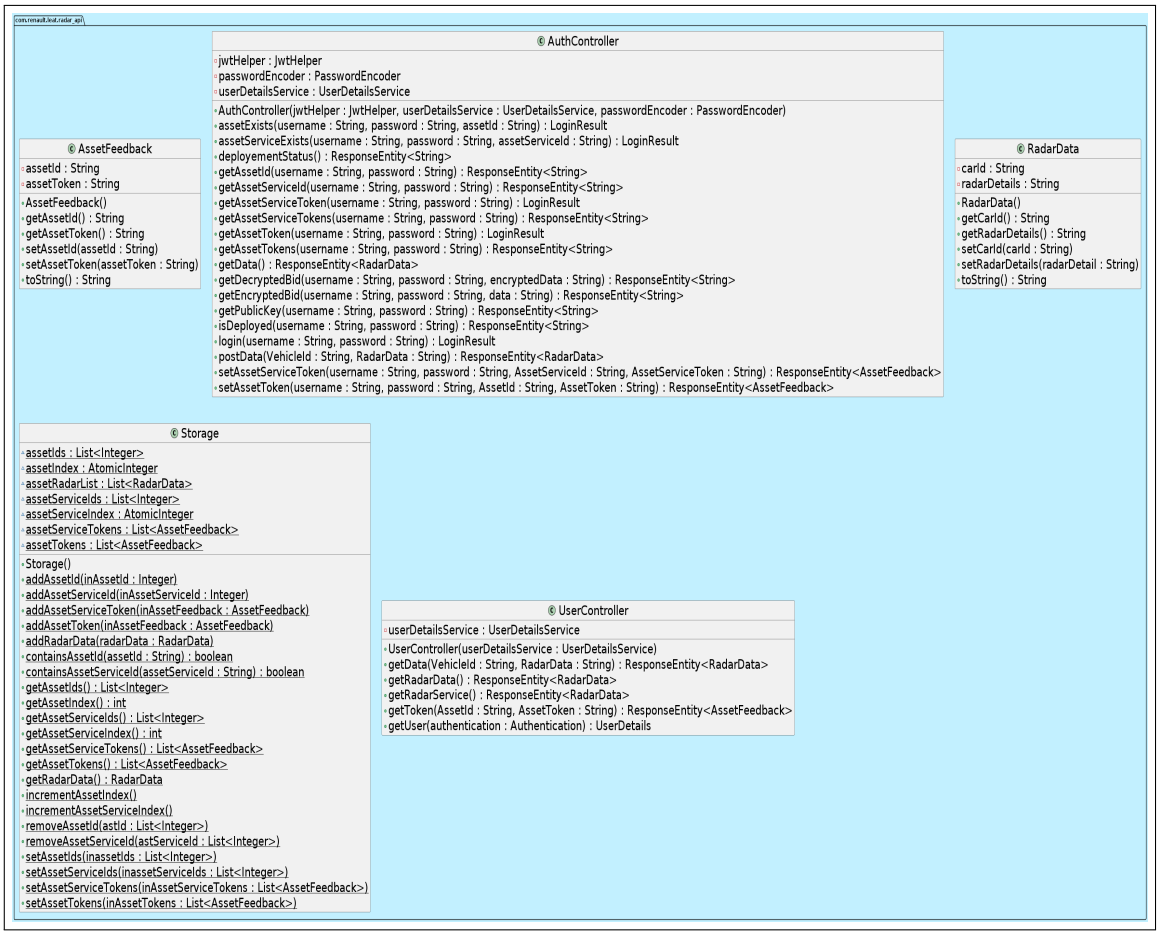

Figure 3.18: OEM Middleware Data Monetisation UML Model

- **UserController:** This Authentication service retrieves Radar and Radar Service subscription data.
- **Storage:** This is the persistence layer for all the details regarding Asset, Asset Service, Authentication, and OAuth Access Tokens.
- **Asset / Asset Service Feedback:** The Asset or Asset Service feedback in the form of review is also stored Off-chain as additional storage.
- **Radar Data:** This contains the data payload representation generated by the vehicle and transacted via the blockchain.

3.1.3.7.3 Infrastructure The decentralized data monetization implementation is deployed in the cloud infrastructure represented in Figure 3.19. As performed earlier for data certification Ethereum-based use-case, we utilize the same TAS cloud infrastructure. The Substrate smart contract pallet containing data monetization logic and the developed middleware are packaged as docker containers. These containers are then deployed on the cloud infrastructure using the Kubernetes orchestration system as pods with necessary extensible data volume, processing, and memory allocations. The client deployed on a separate pod submits a transaction representing Vehicle_OEM or Radar_OEM to a Round

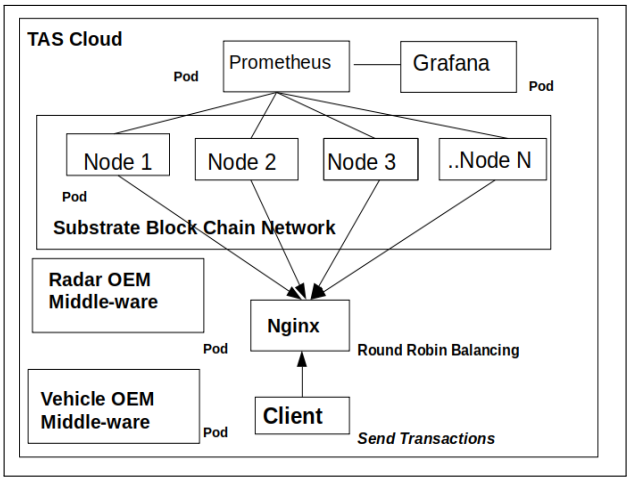

Figure 3.19: Data Monetisation Blockchain Cloud Architecture

Robin-based Nginx load-balancer, which is then redistributed uniformly to the blockchain nodes for validation and execution. The health metrics of the blockchain nodes regarding processor and memory load are stored in the form of the time-series database Prometheus. It is persisted at a pre-determined frequency, then represented graphically in the form of Grafana dashboards deployed in separate pods. The middle-ware containers of Radar_OEM and Vehicle_OEM are deployed running on an Apache Tomcat application server. All the internal networking and forwarding among the pods is managed at the Kubernetes level, along with service discovery and load balancing.

3.1.3.8 Evaluation

In this section, we evaluate the conceptualized data monetization solution from a dual perspective of functional and implementation performance.

3.1.3.8.1 Functional Evaluation Our virtuous cycle of data monetization since the generation from the vehicle traversing Vehicle_OEM to the Radar_OEM and offering a close loop Road Radar Signature service based on the received data improvisation has the functional properties as represented in Figure 3.20. In this discussion, we consider the following representations below for the different types of participants in the ecosystem :

- Set of Vehicle_OEM is represented as Vo = { Vo_1, Vo_2 .... Vo_N } for N participants.
- Set of Radar_OEM is represented as set of Ro = { Ro_1, Ro_2 .... Ro_K } for K participants.
- Set of Vehicles are represented as V= { V_1, V_2 .... V_M } for M participants.
- Processed Radar Data which is monetized by each vehicle belonging to a Vo represented as {D_1, D_2 .... D_L } for L successive data assets which to be exchanged over the data market.
- Processed Road Radar Signature Map (RRS) which a Radar OEM Ro monetizes represented as {Rs_1, Rs_2 .... Rs_P} for P successive RRS asset service which is to be exchanged as a virtuous cycle in the data market.

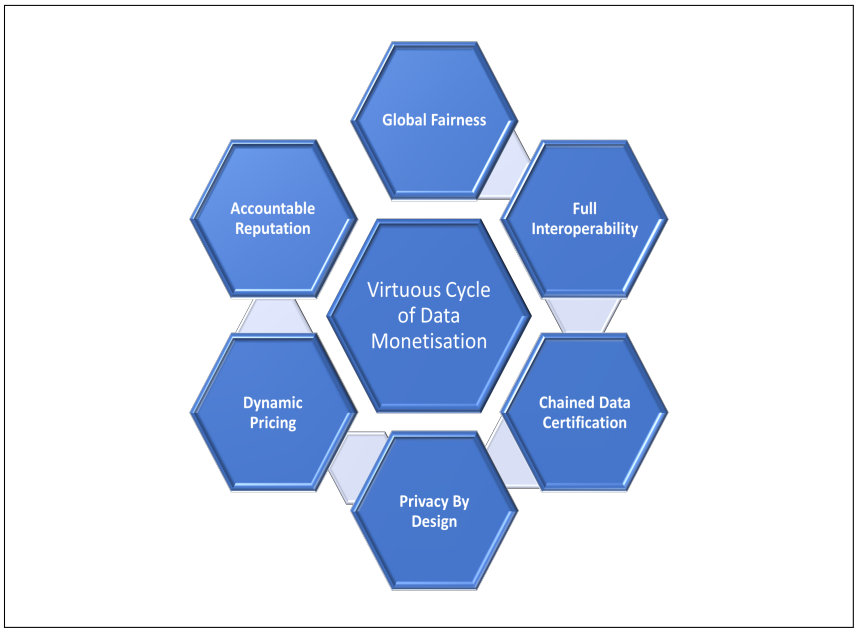

Figure 3.20: Data Monetisation Functional Evaluation

- Bid either for Asset Data or Asset Data Service is represented as $\{B_{x,y,1}, B_{x,y,2} .... B_{x,y,S}\}$ where x signifies the bidder and y signify the asset or asset service tendered.
- Certification for any Asset Data D_L, C_D is defined as the set of hash signatures derived at each stage of processing like individual vehicle data generation, collated raw data in the pool before processing, collated raw data after processing in Vo cloud and then including OAuth Access Token.
- Certification for any Asset Service RSS Rs_P, C_{Rs} is defined as the set of hash signatures derived at each stage of signature formation like Initial Data State, Augmented Data after collation, improvised radar data with signature along with subscription access token.
- Review for any Asset or Asset Service is defined as a score 0...1 where 0 represents the lowest and 1 represents the highest with any median value also possible. This is represented as $R_{i,j}$ where i represents the reviewer and j represents the asset or asset service transferred and reviewed. Based on the review accumulated for Asset Data D_L or Asset Service RSS Rs_P, its future issued Asset or Asset Service can have higher or lesser pricing based on its reputation score mean $RSM_{Radar\_OEM \mid Vehicle\_OEM}$. The reputation score for the asset is proportional to the number of certifications received for the asset or asset service.
- Reputation Score Mean $RSM_{Radar\_OEM \mid Vehicle\_OEM}$ for either Vehicle_OEM or Radar_OEM is the arithmetic mean of historical reviews accumulated. For example Reputation score mean for Radar_OEM or Vehicle_OEM is represented as:

$$\mathbf{RM_j} = mean(R_{i\text{-}2,j}, R_{i\text{-}1,j}, ...R_{i,j})$$

$$\mathbf{RSM_{Radar\_OEM \mid Vehicle\_OEM}} = mean(RM_{j\text{-}2}, RM_{j\text{-}1}, ...RM_j)$$

$0 \leq i \leq N$, i represents the index of the reviewers of either asset or asset service

$0 \leq j \leq N$, j represents the index of asset or asset service exchanged

$\mathbf{RM_j}$, represents the Mean review attained for a particular asset or asset service j

$\mathbf{RSM_{Radar\_OEM\ |\ Vehicle\_OEM}}$, represents the Reputation Score Mean of a Radar or Vehicle OEM.

Its properties are discussed below:

- **Global Fairness:** Fairness in the case of any application is defined by the work of FairSwap [56] "A fair exchange protocol allows a seller S to sell a digital commodity D for a fixed price p to a buyer B. The protocol is secure if B only pays if he receives the correct D." We extend this work in our above design as in [55] for a Global Fairness where every participant in our ecosystem will have the following guarantees:
  - Each Participant is ensured that fund transfer for any D happens only when the certification conditions is satisfied for C_D.
  - Fund transfer is not fulfilled immediately as it is placed in an escrow account ϵ and then, on verification, transferred to the destined participant account.
  - In either case of the virtuous cycle from Radar Data conversion to Road Radar Signature, the facilitators who provision the cloud infrastructure or other value adders can benefit from the commission or fidelity rewards.
  - The Asset or Asset Service Token exchanged results in the influence of a transparent reputation accumulated by either honest or dishonest activity. This is publicly verifiable and can result in the augmenting or decrease of a reputation participant.
  - To ensure fair pricing, the sealed bid mechanism is encrypted with the bid receiver's public key, which results in a competitive remuneration for either the asset or asset service. Also, an asset or asset service issuer's reputation organically influences the probability of data or data signature pricing.
- **Full Interoperability:** Our solution achieves two levels of interoperability which are intra-chain as well as inter-chain, as follows:
  - **Intra-Chain:** The above monetization solution can integrate with external agnostic (centralized or decentralized) systems through OffChain workers for API requests or responses. Also, as we utilize a balanced mix of events, signed and unsigned transactions (extrinsic) to differentiate the priority of message passing between each actor in the system, it avoids unnecessary overhead in the distributed system, especially of consensus.
  - **Inter-Chain:** As it is based on the Substrate framework, this implementation can be extended as a para-chain to integrate with other blockchains or parachains through Polkadot.
- **Chained Data Certification:** As mentioned earlier either for Asset Radar data or Asset Service Road Radar Signature, we ensure the maintenance of the history of

the data provenance along with its hash signature-based certification to avoid any counterfeit and validate the health of the data as well as the service.

- **Privacy By Design:** The solution has granularised the privacy at each level by the following mechanisms:
  - **Account Pseudonymization:** All the accounts are pseudonymized concerning vehicle participants as it is necessary to identify Radar_OEM's and Vehicle_OEM's. A vehicle's original identity is another intermediate identity that can be dissociated in case of necessity. This is to respect the forgetting right of GDPR [23] if the vehicle owner decides to remove his information.
  - **Bid Sealing:** All the bids for either Asset or Asset Service are sealed using encryption, which can be decrypted only by the bid receiver, ensuring competitiveness, transparency, and privacy.
  - **Privacy Data Concealment:** Radar Data from the origin in the vehicle until the data reception by the Radar_OEM is privacy treated with the making of sensitive data, and the hash signature generated is added with controlled noise like location, time, etc. to ensure the authenticity.
- **Dynamic Pricing:** As discussed earlier, pricing is based on a sealed bid oriented proportionally to the participant's reputation and review of the exchanged asset as well as asset service data tokens to avoid any price manipulation. This ensures a reputation-based adjustment of the asset pricing, creating a virtuous cycle in fidelity and incentivization.
- **Accountable Reputation:** Reputation, based on the arithmetic mean of reviewing individual assets transferred as extrinsic transactions diffused through the distributed ledger, is verifiable and transparent for all the ecosystem participants.

3.1.3.8.2 Security In this section, we analyze our solution for some security concerns based on the amalgamated work of OWASP Top Ten (Open Worldwide Application Security Project) security threats for web applications and Blockchain [177] [178]. Also, from the perspective of a smart contract, which, in our case, is an application-oriented blockchain based on a compiled substrate pallet, we analyze its concerns. Their analysis is as follows:

- **Injection:** The blockchain systems can be compromised by using malicious data, which can be in the form of Integer Overflow, Batch Overflow [178]. We ensure adequate checks and balances in the system where external parameterized signed extrinsic are minimized except for encrypted bids.
- **Access Control:** Each participant of Vehicle_OEM, Radar_OEM, Vehicle, and Escrow is based on access control ensured through permission pallets and privileged calls as they are consortium-based, limiting risk exposure.
- **OffChain Manipulation:** Our solution communicates with the OffChain only for retrieval and data job scheduling, not as an oracle for knowledge-based decisions that shield against manipulation attacks.

- **Sensitive Data Exposure:** Data on the blockchain are only related to the original blockchain accounts and their pseudonymized transactions. All the data are managed OffChain, which ensures that sensitive data is hidden and only managed by the blockchain trust mechanism.
- **Smart Contract Security:** In this application, our decision of smart contract deployment is not external and is compiled along with the node and executed along with its runtime. Further, the ERC 721 token specification of Ethereum is adapted for the Substrate FRAME library, and each logic is separated for token creation, OffChain data orchestration, as well as certification validation, which is tested for performance.

The functional evaluation by comparing with earlier mentioned works of Ocean Protocol and Digital Infrastructure for Moving Objects (DIMO) in Section 3.1.3.1. We have a consortium where data is exploited to create a mutual benefit among the ecosystem with a measure of balanced privacy, dynamic pricing, fidelity, transparency, certification, and value addition for the vehicle. It has no external dependency in the form of trusted computing or hardware wallets with integrated OffChain seamless interaction for managing and retrieving from external systems.

3.1.3.8.3 Experimental Evaluation In this section, we experimentally evaluate the monetization architecture implementation as explained earlier in Section 3.1.3.7.2 deployed in the cloud infrastructure. The study is based on the following dimensions:

- Performance Evaluation of the data monetization solution by submitting extrinsic (transactions) and measuring the throughput of finalized and valid transactions.
- Understanding the Hybrid Consensus Algorithm of Aura with GRANDPA or BABE with GRANDPA regarding its parameters and deriving an optimum setting for example: Block Period.
- Analysing the scalability and fork occurrence in the consensus protocol and evaluating its suitability for enterprise mobility solutions.

3.1.3.8.4 Performance Study This section discusses the performance results of the data monetization cloud architecture we have explained. Cloud configuration specification at Kubernetes level is enlisted in Section 4.1.4.3. The Substrate network is tested with a different configuration of 5, 10, 15, 20, and 25 validator nodes that participate in the variations of 1) AuRa and GRANDPA and 2) BABE and GRANDPA. Code Implementation of Substrate pallet along with smart contracts, cloud deployment of Kubernetes, transaction clients, Substrate network configuration files, Radar and Vehicle Middleware Java implementation as well as test results are released publicly in the GitHub repository: https://github.com/scyrilnaves/these-datamonetisation. We organize the test as follows:

- A client that submits the transaction to the load balancer offloads the transaction to a node chosen based on the round-robin algorithm. The client is based on Javascript, which uses the library Polkadot.js for the following purposes:
    1. Establishing WebSocket communication to the Substrate node and retrieving the meta details of the network like pallet information, account details like a nonce,

public key identifier, and other miscellaneous details like chain information, fork information, block details, and transaction details.

2. Transaction (Extrinsic) construction with the signature using the private key of an account, encoding and decoding transaction payload, submission of the transaction, and retrieving its status in the network as finalized or processing.

- For the test, the client submits the transaction of *CreateAsset*, which creates an Asset NFT ERC 721 token in the network. Each transaction has a unique processing complexity based on its purpose logic, affecting its finalization rate or throughput in the network. The test transaction is comprised of the following complexity:

  1. Calculation of Hash for generating a unique Asset Id. Counted as 1 operation.

  2. Storage operation of Asset details in Asset Storage, Storing Asset Count, Asset Index, and ownership details in separate runtime storage structure either as a Storage Map, Storage Double Map, or Storage Value. They are counted as 10 operations.

  3. Read operation from storage to retrieve the existing details before any update. They are counted as 5 operations.

- Transactions (Extrinsic) are submitted by the multithreaded client of 1000 threads signed by 2000 pre-generated accounts.

- Then the transaction is submitted asynchronously at an input rate of 1000 Transactions per second with 50000 total transactions repeated in 3 iterations to avoid any bias in the result. The Block size comprising of extrinsic is maintained at 5 MegaBytes which is optimal as low size can induce message overload and higher can delay by processing bottleneck.

- Signed transactions submitted are checked for finalization, and based on block number, the time difference is calculated for estimating the finalized throughput of the network. Further explanations are provided in Appendix Section 8.4.

In the future section, we will discuss the different test formats across the two consensus algorithm choice variations for the Block proposer: AuRa (Authority Round) or BABE (Blind Assignment for Blockchain Extension) and Block finalization based on chain level agreement: GRANDPA (GHOST based Recursive Ancestor Deriving Prefix Agreement).

3.1.3.8.5 Hybrid Consensus Parameter Analysis: Block Period In this Section, we vary the Block period in the slot-based block proposition consensus algorithm of either AuRa or BABE. The block authoring or proposition happens at fixed customizable intervals termed the Block period. The results of AuRA and BABE block authoring are represented in Figure 3.21 and Figure 3.22, respectively. The inference is based on the following explanations:

- Higher interval of block period directly increases the time associated with block production

- Lower or minimal block period decreases the turn around time for block creation time.

Still, it results in a race condition between transaction verification and consensus operation which overflows the consensus slot time.

AuRA consensus, as represented in the results, is tested across 3, 6, and 12 seconds, exhibiting an ideal throughput of 380 transactions per second for 6 seconds block period time. The input transactions are around 1000 transactions per second against the output of 380 transactions per second as the transaction processing and consensus factor is to be considered. Meanwhile, the throughput ideal for BABE is around 277 transactions per second. BABE has a high block period time of 60, 90, and 120 seconds due to the consensus constraints. Lesser Block time periods of 3,6 and 12 applied for BABE result in a higher amount of forks as a consistency drift is observed in the network. So even though the BABE offers privacy and better security due to its choice of Verifiable Random computation function based selection of proposer, it has a lesser throughput than AuRa due to its computation. Another issue noted here is that there is a GRANDPA level stalling in the network due to the missing pre-commit votes, which occurs in the case of lesser block periods, which signifies the inconsistency of validators assuring the earlier results experienced in works[179, 180].

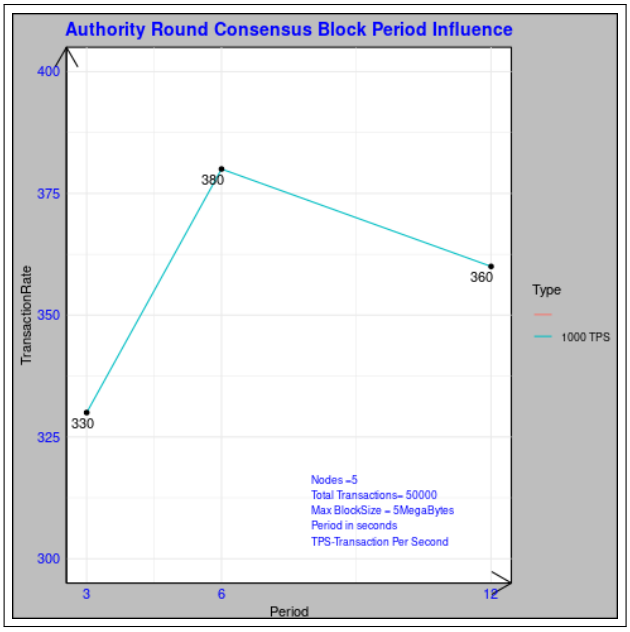

Figure 3.21: AuRa Consensus Block Period Influence

3.1.3.8.6 Hybrid Consensus Scalability Analysis In this section, we analyze the scalability of AuRa and BABE consensus along with the GRANDPA finalization algorithm. The scalability results for AuRA and BABE are represented in Figures 3.23 and 3.25, respectively. The results of AuRa are explained by the $O(n^2)$ message complexity which shows a decreasing throughput in proportion to the number of nodes. In addition, the GRANDPA finalization message complexity bottleneck of $O(n^2)$ further augments the decrease in throughput. Still, the difference in the GRANDPA algorithm is that finalization is performed on the chain of blocks rather than individual blocks of AuRa. Also, another inference based on the result is that higher input Transaction per second (TPS) greater than the optimum 1000 increases the forks in the network as well, which in turn decreases the throughput as represented

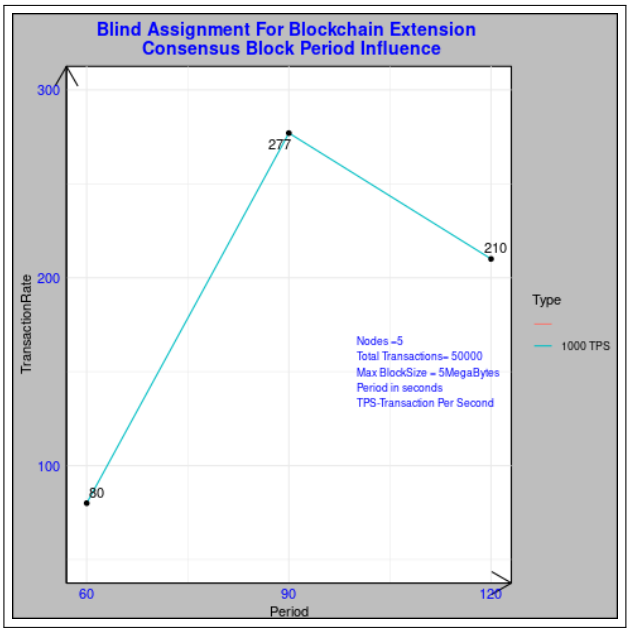

Figure 3.22: BABE Consensus Block Period Influence

in Figure 3.24. This fork is explained by the processing and consensus-induced bottleneck affecting the liveness and consistency of chains but resolved by GRANDPA through additional computation.

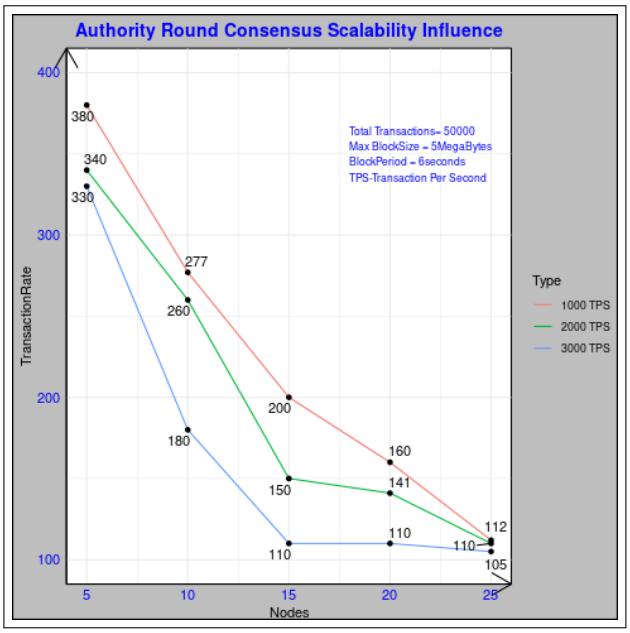

Figure 3.23: AuRa and GRANDPA Consensus Scalability Performance

BABE consensus has an algorithmic complexity of O(n) but suffers a decrease in scalability throughput due to the occurrence of forks in the system, given the susceptibility of the consensus algorithm to elect multiple validators for a single block height. This results in a larger number of secondary blocks rather than primaries which are attributed to the nature

of the algorithm. Accompanied by the computational effort of Verifiable Random Function it affects the throughput of BABE, which is absent in the AuRa algorithm. The fork occurrence increases with higher input transactions per second, as represented in Figure 3.26 augmented by increased transaction processing, message communication overhead, and consistency issues.

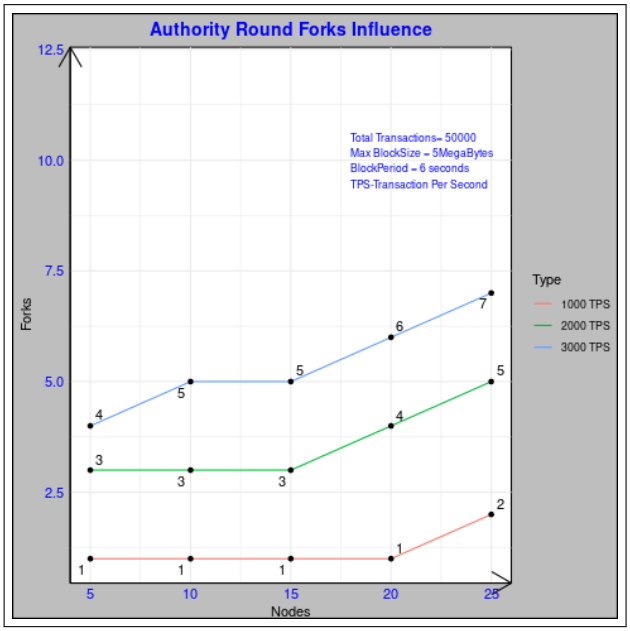

Figure 3.24: AuRa and GRANDPA Consensus Fork Study

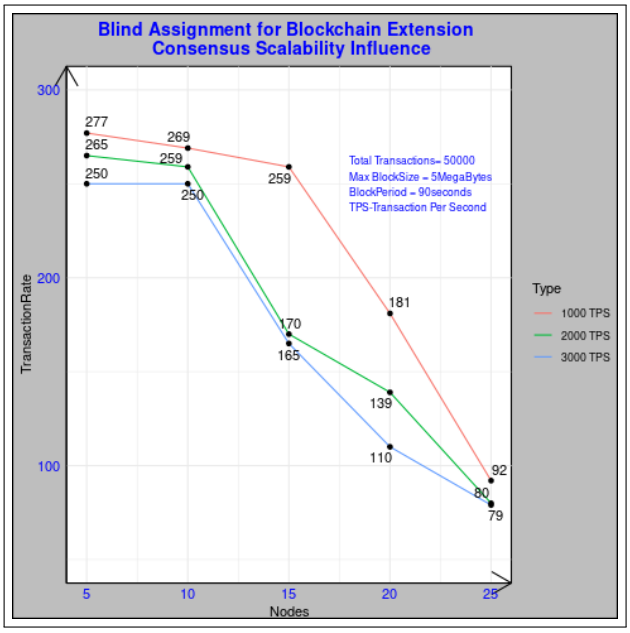

Figure 3.25: BABE and GRANDPA Consensus Scalability Performance

In summary, for the implementation evaluation of Data Monetization architecture, we realize that the BFT algorithms of Aura and BABE have a scalability issue. On a positive

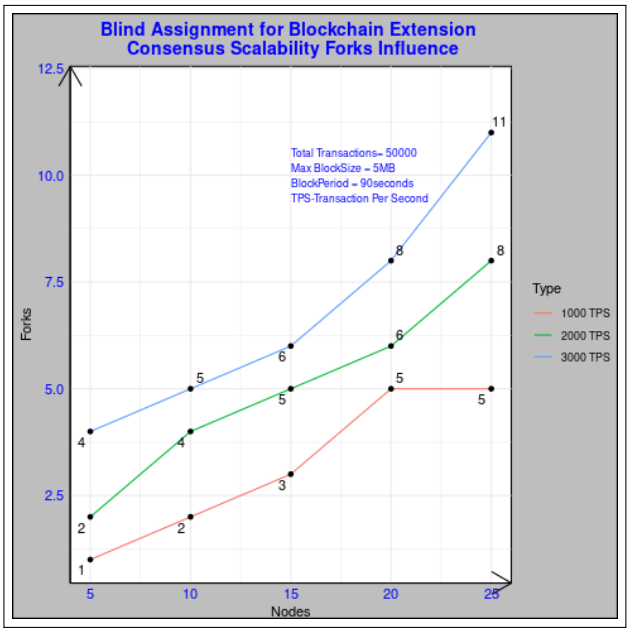

Figure 3.26: BABE and GRANDPA Consensus Fork Study

note due to its hybrid nature with GRANDPA, it offers finalization and better protocol security but at the cost of additional computation. Both protocols are affected by scalability issues where the AuRa consensus is more stable than the BABE protocol. BABE, along with GRANDPA protocol, suffers from consistency issues [179, 180] as explained earlier. It offers a higher degree of security in the form of an impossibility of predicting the next successive block proposer using a verifiable random function. GRANDPA BFT offers the finalization of chains in the presence of forks, a hybrid protocol to work with AuRa and BABE in the substrate framework. It is destined to resolve the forks, but its liveness can be affected if most validators are affected by chain consistency issues. The results can be slightly higher if a more straightforward transaction payload is considered. Still, the throughput of around 300 transactions per second in this data monetization implementation signifies that this architecture is acceptable and can be extended to enterprise mobility use cases.

3.1.3.9 Conclusion

In this work, we analyzed the data monetization use case by creating a virtuous cycle of data flow incentivizing the Vehicle_OEM's, Radar_OEM's, and Vehicles who share the data. We exhibit the extensibility of the architecture to external systems and provide Radar Signature Services offering global fairness, interoperability, privacy by design, asset data reputation, and a secure network. We further evaluate the cloud-based implementation and understand the Substrate Hybrid Consensus from an application-based blockchain with an embedded smart contract, including its BFT family consensus scalability limitations.

3.1.3.9.1 Future work: In this section, we discuss some improvements that can further enhance the data monetization protocol as below:

- **Auction:** Auctions can be made more transparent by a commit reveal scheme where

there are no time-bound constraints and hidden bids are eventually public when all the bids are received and committed.

- **Privacy:** Differential Privacy [181] can be applied for more granular privacy control on the data ensuring more concealment than the masking techniques. Also, account abstraction based on ERC 4337 [182] protocol can be considered for anonymizing network participants as an alternative to pseudonymization in our protocol.
- **Smart Contract:** Smart Contract embedded along with client runtime in our Substrate blockchain has better security than by deployment. But it has not been audited using tools like MythX [177], or legal aspects are offloaded, which can be considered in the future.
- **Interoperability:** Substrate has both Intra-Chain and Inter-Chain communication extensibility. The inter-chain exchange in the form of Parachains has several limitations discussed in [183]. They are due to being monetary-based Proof of Stake Polkadot network for parachain slots, validator election transparency issues and governance problems due to the "prime voter" as well as 13 member council governance affecting decentralization. Also, there are liquidity issues with the Polkadot network due to economic reasons, and many parachains like Lido have ceased their operation.
- **Certification Proof:** Hash-based chained certification can be replaced with Merkle proofs or Zero Knowledge proofs where any user can verify the proof without actual data revealed.
- **Consensus:** Our implementation analyzes the Hybrid Consensus of AuRa, BABE, and GRANDPA available in Substrate necessary for our data monetization use case. Another consensus of Nominated Proof of Stake can be tested, available in Substrate Polkadot based on monetary conditions of currency called DOT'S. It can be applied to our use case but needs monetary-based staking and slashing constraints [183].

3.2 Decentralised Mobility Services Conclusion

In the above two mobility use-case of data certification and monetization, we have discussed the evaluation functionally and through benchmark of cloud implementation. We have analyzed the algorithms of PBFT, Clique, IBFT, and QBFT of Ethereum, and the Hybrid Consensus of AuRa with GRANDPA and BABE with GRANDPA present specific similar challenges. These are the following questions:

- **Scalability:** All the BFT algorithms present a drastic fall in scalability due to communication complexity except Clique. However, we cannot confirm this issue since we notice a bottleneck at the client binary level in the blockchain.
- **Security:** All the testing is performed considering the security of data or assets in the form of tokens which we discussed in the functional evaluation and confirmed. On the contrary, security at the level of blockchain framework in the form of malicious nodes or different network topologies is still not benchmarked, which can impact the consensus messages' finalization throughput and stability.

- **Fault Tolerance:** A pertinent scenario noticed while Ethereum and Substrate is the possibility of the node being stalled in the case of network issues, database corruption, memory issues, as well as virtualized Kubernetes pod issues which can occur and risk the network health. We have not considered these scenarios, as the worst case of recovery would be to restart the nodes affecting the consensus algorithm without any test on the tolerance mechanism.
- **Finalisation or Fork Issues:** All the consensus algorithms exhibit finalization issues with a high presence of forks in the case of Clique. As noted in the above discussions, Aura and BABE are also highly prone to forks resolved by GRANDPA.

The evaluated consensus algorithms of Clique, PBFT, IBFT, and QBFT present an overall scalability issue with the augmentation of the nodes in the blockchain network. Also, we notice issues of chain consistency, deadlock, and liveness problems in the network. Another problem we notice apart from the consensus algorithms mechanism is the binary implementation issues of Geth, Besu, or Substrate, which has limited performance due to technology or database choice.

All these issues are derived from heterogeneous blockchain platforms of Ethereum: Geth, Besu, and Substrate, which present different design considerations as noted by the difference in smart contract construction and processing, database choice, memory management, network management, data structure implementations as well as programming frameworks. So we continue this aspect to further test the BFT consensus algorithms in a homogeneous platform by taking the outcomes of these two use-case implementations as a hypothesis to be confirmed and studied.

So this motivates us in two ways creating a simulator to test the BFT consensus algorithms homogeneously and extending the same simulator to conceive a novel BFT consensus algorithm to solve the earlier uncovered issues.

Chapter 4

Simulated Byzantine Fault Tolerant Consensus Algorithms For Normalised Evaluation

Our brain simulates reality. So, our everyday experiences are a form of dreaming, which is to say, they are mental models, simulations, not the things they appear to be.

– Stephen LaBerge

The previous chapter evaluated the two mobility use case implementations through different blockchain architectures across a family of simple and hybrid BFT family consensus algorithms. We had arrived at an intermediate conclusion which presented challenges or bottlenecks with the performance degradation of the system. They were due to the application-level issues like the complexity of the smart contract or at the blockchain platform level due to the implementation design of the binary. The choice of programming stack or the actual consensus algorithm affected the throughput performance. This problem dissection across multiple layers makes it a bit indistinguishable, further aggravated by different blockchain platforms' design and development by varied open-source communities. So to address this heterogeneity of issues and to concentrate on pure consensus algorithm design issues, we had to conceive a simulator test-bed. It could homogenize the test efforts and help in understanding as well as building a hypothesis around the consensus algorithms of the Byzantine Fault Tolerant family.

4.1 Simulator Formulation

Building a simulator for problem-solving involves the following process:

1. **Problem Formulation:** To define the problem accurately and concisely with all the required assumptions.
2. **Simulation Model Building:** To capture the basic aspects and behaviors of the system where the problem is picturized, which should align with the quality of available data and the degree of validation needed.
3. **Experimental Design and Analysis:** After the model definition and its validation, it can be used to perform experiments that investigate the goals and objectives of the problem.
4. **Evaluate and Iterate:** If the simulation has achieved the objectives, then it should be documented, and recommended solutions should be implemented. Otherwise, it should be iterated further, and other additional data, modeling, or experimentation should be added to solve the desired problem.

We follow the above process in describing our simulation and performance evaluation of BFT consensus algorithms. The idea is to identify and assure ourselves of the real bottleneck at the performance or at the scalability level.

4.1.1 Problem Ideation

The fundamental challenges we overcame in designing the simulator versus the usage of multiple blockchain binaries like Clique of Ethereum, Practical Byzantine Fault Tolerance of Hyperledger Sawtooth, Istanbul Byzantine Fault Tolerance / Quorum Byzantine Fault Tolerance of Hyperledger Besu, Aura, and GRANDPA of Substrate are given by the Table 4.1. Our problem definition in building this simulator would be understanding any consensus algorithm behavior and testing its performance mimicking the real world by abstracting the blockchain platform. It will manage the data creation, validation, verification, and distribution after the agreement among the consenting peers. Since the abstraction is standard across the different algorithms and the evaluation is homogeneous, the simulator framework should enable testing any distributed consensus algorithm by plugging into a module.

4.1.2 Simulator State of Art

In this section, we explore the different works related to the simulation of blockchain platforms. We understand their implementations to analyze if they can be reused to simulate and study the BFT consensus algorithms. We gain inference on the existing works to build motivation and design a new simulator solving our problem objectives.

In the work by A. Albshri and A. Alzubaidi and B. Awaji and E. Solaiman [184], they attempt to capture the available state-of-the-art Blockchain simulators by accounting for their quality factors such as usability, reliability, provided capabilities, and supported features. Figure 4.1 excerpted from their work shows the array of simulators they have analyzed along with their major properties. It represents all the stochastic dynamic simulators that

Property	Blockchain Binary	Homogeneous Simulator
Database	Separated Database Layer using a library like LevelDB for Ethereum-Geth, RocksDB for Substrate	Run Time Memory of Simulation Platform
Transaction Payload	Smart Contract or Crypto Asset Payload	Simplified Message Payload
State Maintenance Overhead	Account or Unspent Transaction Output (UTXO) based maintenance	Simple Account State Maintenance
Encoding / Decoding Payload	Recursive-Length Prefix (RLP) Data Encoding for Ethereum or Simple Concatenated Aggregate Little-Endian Data Encoding (SCALE) and SS58 Public Address Encoding for Substrate	No Special Encoding
Encryption / Decryption	Elliptic Curve Cryptography (ECC) for Ethereum, Elliptic Curve Digital Signature Algorithm (ECDSA), Edwards Curve Digital Signature Algorithm (Ed25519) and Schnorr Signature (SR25519) for Substrate	Simple Encryption like RSA
Hashing Function	KECCAK256 for Ethereum and BLAKE for Substrate Hash function	Simplified SHA256 Hash function
Development Platform	Golang for Ethereum Clique, Java for Ethereum IBFT, QBFT; Python for Hyperledger Sawtooth and Rust for Substrate	Single Platform / Language for normalized testing using Java
Network Layer	Networking Libraries like DevP2P for Ethereum, libP2P for Substrate including varied routing algorithm	Standard Multithreaded Client and Server based on Socket Library tested with Mesh Type routing
Additional Virtual Machine Processing	Ethereum Virtual Machine (EVM) for Ethereum, Web Assembly (WASM) runtime for Substrate or Transaction Processor for Hyperledger Sawtooth	No Additional Virtual layer as all code packaged into a single JAVA archive for execution
Block Transmission	Diffusion of Blocks with all transactions redundantly	Diffusion of blocks with transactions in initial validation and a hash of the same in further steps.

Table 4.1: Differentiation Factors necessitating the Simulation

are discrete event-based. The three simulators of Modeling by Queuing Theory, DAGsim, and Local Bitcoin are virtualization-based. Concerning our core interest in consensus algorithms, the most analyzed algorithm is Proof of Work which is implemented in almost all the simulators leaving Proof of Authority and Proof of Stake to just a single simulator. Bitcoin is the most simulated, with 15 of 21 works based on it. The major metrics being measured by each of the simulators are represented in Figure 4.2 with the consensus layer measuring the amount of generated and mined blocks. Also block verification time, fork resolution, pending transactions, and mining difficulty, are measured in this work. Our focus while analyzing the simulator is to get the metrics of consensus communication overhead, network partition tolerance, scalability, and finalized transaction rate.

Simulator	Year	GitHub code	Prog. lang.	Library	Core	PRF.	Security	Platform	Consensus
Shadow-Bitcoin [29]	2015	✓	Python	N/A	Shadow	✓	✓	Bitcoin	PoW
VIBES [37]	2017	✓	Scala	N/A	N/A	✓	✓	Bitcoin	PoW
Stochastic Blockchain models [38]	2018	N/A	Python	PyCATSHOO	N/A	✓	✓	Generic	PoW /PoS
Behavior and Quality of Blockchain [39]	2018	N/A	Python	SimPy	N/A	✓	✗	Bitcoin	PoW
eVIBES [28]	2018	✓	Scala	N/A	[37]	✓	✗	Ethereum	PoW
Modeling by Queuing Theory [40]	2018	N/A	Java	N/A	[52]	✓	✗	Bitcoin	PoW
BlockSIM [44]	2019	✓	Python	SimPy 3.0	N/A		✗	Ethereum/ Hyperledger	PoW /PoA
DAGsim [50]	2019	N/A	Python	N/A	N/A	✓	✗	IOTA	IOTA
BlockSim [27]	2019	✓	Python	N/A	N/A	✓	✗	Bitcoin/ Ethereum	PoW
FastChain [36]	2019	N/A	N/A	N/A	[29]	✓	✗	Bitcoin	Pow
simBlock [47]	2019	✓	Java	N/A	N/A	✓	✗	Bitcoin	PoW /PoS
Blocksim [43]	2019	✓	Python	SimPy 3.0	N/A	✓	✗	Bitcoin/ Ethereum	PoW
Ext- simblock [49]	2019	N/A	Java	N/A	[47]	✓	✗	Bitcoin	PoW
BlockSim-Net [42]	2020	N/A	Python	N/A	[27]	✓	✗	Bitcoin/ Ethereum	PoW
ChainSim [46]	2020	N/A	Python	N/A	[27]	✓	✗	Bitcoin Ethereum	PoW
Local Bitcoin Network [45]	2020	✓	Python	N/A	N/A	✓	✗	Bitcoin	PoW
SIMBA [30]	2020	✓	Python	SimPy 3.0	[43]	✓	✗	Bitcoin	PoW
Ext 2 - simBlock [48]	2021	N/A	Java	N/A	[47]	✓	✗	Bitcoin	PoW
BlockPerf [31]	2021	✓	Python	N/A	[27]	✓	✗	Bitcoin	PoW
BlockEval [32]	2021	✓	Python	SimPy	N/A	✓	✗	Bitcoin	PoW

Figure 4.1: Summary of Blockchain Simulators along with features [184]

Simulator	Approach used															
	Network layer					Consensus layer							Incentive layer	General		Total
	Block propagation time	Transaction propagation time	Average of block size	Throughput	Network delay	Number of generated block	Number of mined block	Block verification time	The rate of stale (orphan) blocks	Fork resolution	Pending transaction	Mining difficulty	Mining reward	Processing Speed	System stability	
Shadow-Bitcoin [29]	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	1
VIBES [37]	●	○	●	●	○	●	○	○	●	○	○	○	○	●	○	**6**
eVIBES [28]	○	○	○	○	○	●	○	●	○	○	○	○	○	○	○	2
BlockSIM [44]	○	○	○	●	○	○	○	○	○	○	○	○	○	○	●	2
BlockSim [27]	○	○	○	●	○	●	○	○	●	○	○	○	○	○	○	3
SimBlock [47]	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	1
BlockSim [43]	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	2
Local Bitcoin [45]	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○	2
SIMBA [30]	●	○	●	○	○	○	○	●	○	○	○	○	○	○	○	3
BlockPerf [31]	○	○	●	●	○	○	●	○	●	○	○	○	●	○	○	5
BlockEval [32]	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	2
Total	**4**	2	3	**4**	1	3	1	2	3	1	1	1	1	1	1	

Figure 4.2: Metrics of Blockchain Simulators [184]

In work by Faria, Carlos and Correia, Miguel [185], they design a discrete-event simulator called BlockSim to evaluate different blockchain implementations. They used the existing Python framework of SimPy and made their work public. The simulator comprises the node, consensus, ledger, transaction, block, network, and cryptographic layers. The node layer resembles the node initialization in a P2P network, the ledger layer defines the structure of the ledger, the transaction and block layer simulates the transmission and reception of transactions, the network layer defines the communication protocol, and finally, the cryptographic layer manages the encryption functions used. The consensus layer adds a delay in the network instead of block and transaction validation. It also has a block difficulty calculation function where the difficulty in hash calculation increases with the block height, similar to the Ethereum and Bitcoin Proof of Work protocols. The two blockchain networks of Bitcoin and Ethereum are simulated, both Proof of Work protocols. They further evaluate these models with the real-world Ethereum and Bitcoin Networks. They measure the block or transaction propagation times with varied gas limits or block sizes as well as different block encryption and decryption delays. They state in their work that it is an open challenge to adapt BlockSim for Byzantine Fault Tolerant Consensus algorithms.

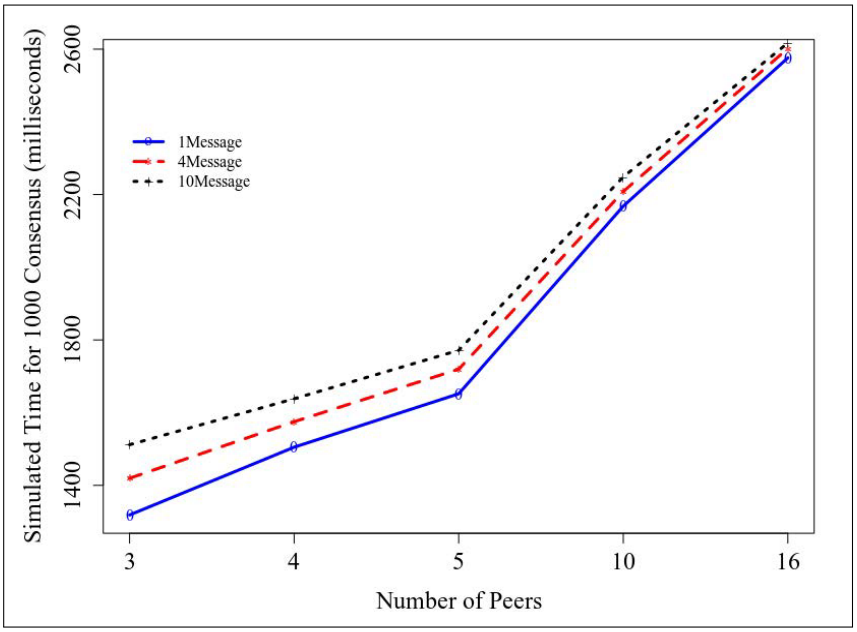

Figure 4.3: Average Confirmation Time for 1000 transactions [186]

The next work is a blockchain simulator for evaluating consensus algorithms by Foytik, Peter and Shetty, Sachin and Gochhayat, Sarada Prasad and Herath, Eranga and Tosh, Deepak and Njilla, Laurent [186] proposing a generalized representation for consensus. They use a discrete event simulation engine to test the scalability and the number of messages exchanged during the consensus mechanism, focusing on the network and consensus layers. The key metrics considered in the simulator are the throughput concerning the number of successful transactions, latency in transactions, network fault tolerance, and checking its behavior with heterogeneous devices and network conditions. They utilize the Network Simulator-3 (NS3) to simulate this blockchain system. The consensus mechanism implemented is Raft [187], which is a simple protocol to understand. They test it with different numbers of messages like 1, 4, and 10 with an increasing number of peers starting from 3

until 16 as represented in Figure 4.3. The results shown there depict the increasing time to process the message with more peers, but at the same time, the gap between the different sizes of messages decreases with increasing peers. This reveals that the number of peers affects the scalability more than the number of messages transmitted.

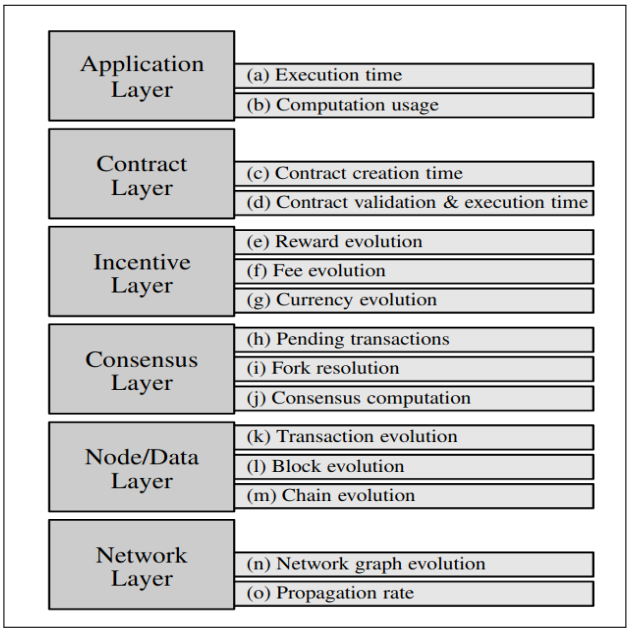

Figure 4.4: Blockchain abstraction layer model and associated metrics in BlockPerf [188]

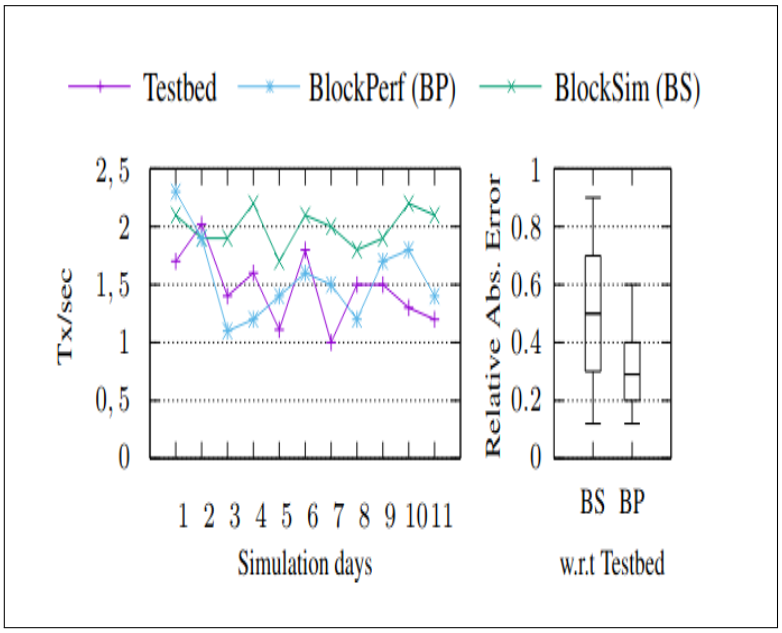

Figure 4.5: BlockPerf Transaction Throughput [188]

The next work to be discussed is an improvement over the previously discussed work of BlockSim [185] called BlockPerf [188] by Polge, Julien and Ghatpande, Sankalp and Kubler, Sylvain and Robert, Jérémy and Le Traon, Yves wherein a real network infrastructure layer

is used albeit a simulated layer. They have modeled the blockchain as a six-layer model represented in Figure 4.4, which shows the metrics that can be derived from each blockchain layer. It emulates the network layer while simulating the node, consensus, incentive, and application layer based on statistical data modeling, which is different from BlockSim [185] approach. The consensus mechanism tested is Proof of Work, similar to the BlockSim [185]. Still, the implementation differs wherein instead of simulating the behavior of the consensus algorithm, each of the mining nodes selects a random number, the pending transactions in the queue, and the previous block hash to calculate the next block hash resembling the Proof of Work algorithm. The improvised BlockPerf[188] is tested over the BlockSim and the real-world Bitcoin test bed implementation bitcoind. The evaluation of BlockPerf's throughput compared to BlockSim shows that it is better in simulating the real-world testbed of blockchain setup with less relative absolute error, as shown in Figure 4.5.

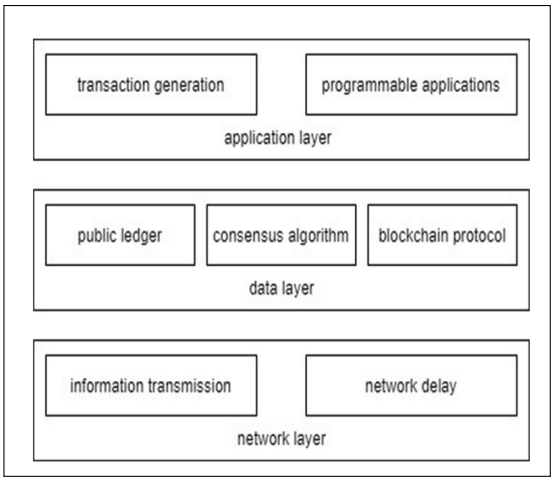

Figure 4.6: ChainSim [189]

Following is a simpler blockchain simulator framework called ChainSim [189] proposed by Wang, Bozhi and Chen, Shiping and Yao, Lina and Wang, Qin, which can simulate multiple nodes in a single computer using lightweight threads. The module structure of the simulator is shown in Figure 4.6 with the application layer consisting of transaction generation and programmable applications, the data layer consisting of a consensus protocol, and the network layer consisting of different information transmission methods such as unicast, broadcast, multicast or gossip. The three different protocols implemented are Bitcoin, Ethereum, and IOTA. The consensus algorithm simulated is Proof of Work, an older version of the algorithm not used by the actual community of Ethereum and IOTA anymore. The simulation results are compared with the actual public network data of the latter, which denotes that the error is within the acceptable range. Still, the actual implementation details are missing, which poses some challenges regarding the ability to reuse this simulator.

In the next work by Wang, Ping-Lun and Chao, Tzu-Wei and Wu, Chia-Chien and Hsiao, Hsu-Chun, we look at a proposition of a simulator for plain BFT consensus Protocols [190] without the structure of a blockchain. The simulator's design is implemented in JavaScript as represented in Figure 4.7, where a controller launches a test specified in a configuration file in the form of events. These events are then sent to a consensus module where the node participation as either honest or malicious can be modified to test the protocol's

resilience. Then the consensus module forwards the message via the network module to the attacker Module and, finally, the controller. The simulation can deploy network models of Synchronous, partially synchronous, and Asynchronous networks accompanied by attacks of network delay, dropping, and inserting new messages. They simulate the protocols of ADD+ BA [191], Algorand [192], Async BA [193], PBFT [73], HotStuff [83] and LibraBFT [84]. They analyze that these BFT protocols suffer from view synchronization and network stability issues when the latency is high. The above-discussed selected state

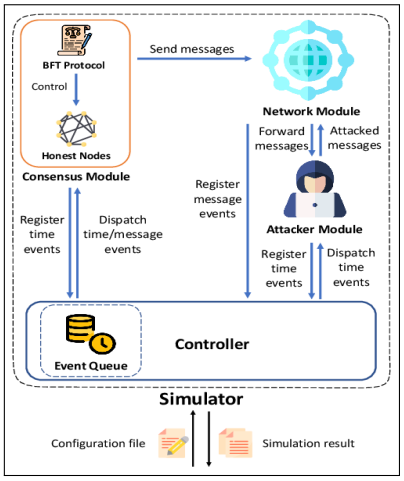

Figure 4.7: BFT Simulator Tool Infrastructure [190]

of works presents the following general observations from the perspective of reusability to test consensus algorithms:

1. Simulations are coupled to a framework like SimPy or NS3 with artificial estimated delays, which can present a bias that varies drastically compared to a blockchain binary.
2. Works discussed above lack modularity, open-source availability, and documentation necessary to test new consensus algorithms.
3. Network, cryptographic, and node layers are abstracted to perform predefined functions with no implementation to test their performance with real message payload.
4. The control of network conditions in the simulator for capturing asynchronous, partially synchronous, and synchronous networks with varying latency is not available.
5. Ability to simulate different types of adversaries in a BFT setting is not followed in the blockchain-based simulator, which might help us validate the resilience of any algorithm.

These challenges inspire us to develop our simulator to test the consensus algorithm with a controlled, homogeneous implementation similar to a real-time binary, avoiding the above-mentioned issues.

4.1.3 Simulator Methodology

Simulation of a blockchain should start from the communication layer enabling the actors or nodes to pass and receive messages between them in a peer-to-peer network. The shared

messages must be cryptographically modified, ensuring security and privacy through encryption and hashing techniques. The messages herein, called transactions, are then stocked in a data structure resembling a queue, a temporary buffer. The transactions are then packed into a block structure, including the cryptographic digest, which is then proposed and distributed by a node or participant to other peers. The next step is the consensus process on the block message, which can be of different types like Practical Byzantine Fault Tolerance, Clique, Istanbul Byzantine Fault Tolerance, or Quorum Byzantine Fault Tolerance as explained in earlier Chapter 2. Finally, after the consensus process, the finalized block is stored in a chain of blocks structure where the finalized block is linked with the hash of the previous finalized block.

4.1.3.1 Design Goals

In line with the above skeletal explanation, we outline certain design goals to be followed in a distributed system simulator similar to SimGrid Design Goals [194] like *reproducibility* to re-execute the simulation, ensuring the same outcome, *speed* ensuring a fast simulation, *versatility* to simulate any distributed systems and *scalability* to deal with large simulations. We extend these design goals in our blockchain simulator construction by setting the following properties to be imbibed in our proposal.

1. **Modularity:** Simulator should delineate the different components, each having a separate logic like networking, cryptographic, transaction production, consensus decision layer, blockchain ledger storage, and node account maintenance. Our organization of the simulator into separate packages based on logic differentiation will achieve this property.
2. **Don't Repeat Yourself (DRY):** This software design principle will be kept in mind to avoid any repetition of configuration or code and make abstractions as much as possible to get a generic functional code with minimal or no duplication.
3. **Separation of Concern:** As the Turing Award computer scientist Edsger Dijkstra quotes *"It is what I sometimes have called "the separation of concerns," which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of"*.
   This software development principle is a logical successor of the previous DRY principle, which will be used for our simulator. Here the division of the software is into separate logic modules and further into Classes, Interface, and Abstract Classes to enable polymorphism. It avoids duplicity and is essential for maintaining the coherence and reusability of the deliverable code.
4. **Extensibility:** Architecture should allow other modules to be added to the existing ones to add more logic. Each package or module we build will be loosely coupled with abstractions avoiding concrete implementations, allowing future extension of classes either by inheritance.
5. **Replicability:** Simulation should be platform agnostic and easily archived to run as an executable or application server, generating results that should be the same at each run for a given configuration of infrastructure and simulation. The final simulation code base will be containerized to be run on any cloud infrastructure with a predefined set of configurations allowing the easy re-run of the simulation test.

6. **Elasticity:** Simulation should allow for an extensible workload regarding node participation, network topology variance, and message payload size for varied performance testing, including scalability. The container running on an application server will be cloned in as many pods or deployments to scale the test. It is ensured as we focus on testing the algorithm performance with a variable number of node participation, message payload, or network topology change.

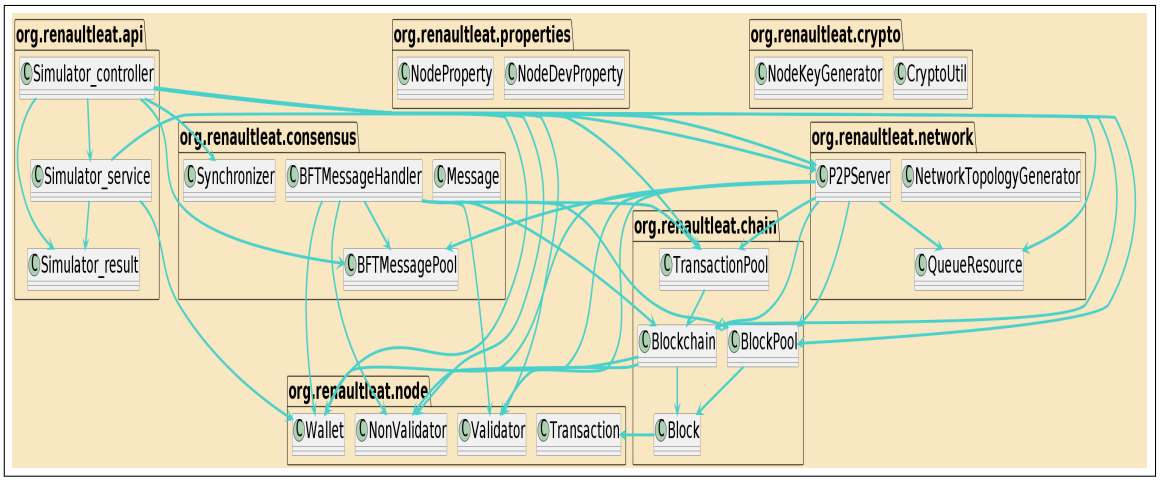

Figure 4.8: High-Level Architecture of Simulator

4.1.3.2 Design Principles

Following the above properties, we conceptualize the simulator architecture as represented in Figure 4.8. It is organized as packages with the JAVA naming convention of org.renaultleat.* considering this thesis project is a partnership between the laboratory LEAT and Renault. The overall structure of the architecture is as follows:

1. **org.renaultleat.api**: This Application Programming Interface (API) package encapsulates all the interactions with the simulated blockchain node enabling it to set or retrieve its configuration, start the simulation, or retrieve the results.
2. **org.renaultleat.node**: It is used to maintain the account of the participating node, including its cryptographic keys, sign the data, and create the payload of a signed transaction.
3. **org.renaultleat.chain**: The chain package is more like a storage data-structure part organized into a blockchain data structure, transaction queuing, block queuing before finalization, and block payload creation.
4. **org.renaultleat.crypto**: A cryptographic package is more like a utility that deals with cryptographic operations of encryption, decryption, hashing, signature creation, and verification. It includes the node authorization checks to participate as a consensus validator or not.
5. **org.renaultleat.network**: This package handles the transmission and reception of any message payload between the node participants. It is established by peer-to-peer networking with the flexible topology change of either fully connected mesh, lattice, or Watts-Strogatz.

6. **org.renaultleat.consensus**: The consensus package layer is split across storage for synchronization in a multi-threaded environment, temporary message storage, message payload protocol, and the consensus algorithm logic handler.
7. **org.renaultleat.properties**: Configuration of the nodes, network, simulation, and performance workload are defined through this package.

The idea behind this concept is the clear modular division of the simulator with separate blockchain working parts to work in harmony. It enables us to plug and test the core of the simulation, which is the consensus layer ensuring a homogeneous testing condition for all the algorithms. In the following explanation, we will divulge more about each component in each package and its organization.

4.1.3.2.1 Application Programming Interface Package: This package component is illustrated in Appendix Chapter 8 through Figure 8.10 and Figure 8.11, which highlights the pivot from where we can derive information related to the blockchain simulation. The total validator's participation, node behavior type as malicious or benign, transactions processed, blockchain storage, and rate of transaction confirmed or processed are specified here. The consensus properties like round change timeout, sending of messages to Peers, and the switch to start and stop the simulation are managed in this package.

4.1.3.2.2 Model View Controller (MVC) Pattern: This package follows the MVC pattern comprising of Simulator_controller, Simulator_service and Simulator_result classes. This is an architectural pattern where the control for deriving information through standard Hypertext Transfer Protocol (HTTP) GET, POST, and PUT methods is delegated from the Simulator_controller to Simulator_service. This service, in turn, derives or stores the information with the HashMap data structure storage of Simulator_result class and from other storage structures of packages such as chain and node. The delegation is not limited to the configuration of simulation data but other functionalities such as sending messages through the network package, starting the simulation, or initializing cryptographic keys.

4.1.3.2.3 Chain Package: This package is more of a storage entity comprising the four classes represented in Appendix Chapter 8 through Figure 8.12 and Figure 8.13 whose explanation is as follows:

1. **Transaction Pool:** This contains the ArrayList data structure for storing the verified transactions after the node participants receive the transaction from other peers before converting them to blocks. This class handles transaction validation, maintaining the block size, evading duplicate transactions, and round progression or block-height increment.
2. **Block:** The structure of a fundamental block data is defined here with the essential details of block number, timestamp, consensus messages relating to the block, the signature, transaction details, and validation rules.
3. **Block Pool:** Similar to the transaction pool, this stores the proposed block in a queue before being finalized by consensus protocol. It is more like an intermediate class before confirming the blocks in the blockchain principal storage. The logic for avoiding

duplicates, invalid blocks, and periodic cleaning of the finalized blocks for memory management is inserted here.

4. **Blockchain:** This is the principal data structure that maintains the 'chain of blocks' in a CopyOnWriteArraylist. It is a thread-safe synchronized ArrayList used in our consensus layer to be multi-threaded for optimized transaction and block processing. The operations of adding a new finalized block, block validation, and storage for identity management of the validator or non-validator participants are defined here.

4.1.3.2.4 Consensus Package: This core package comprises classes as illustrated in Appendix Chapter 8 through Figure 8.14 and Figure 8.15 that define the consensus message protocol, message pool construction, and consensus message handler whose explanation is as follows:

1. **Message:** This defines the consensus message payload structure with consensus round properties of the actual block height, block hash, message sender's public key, message sender's signature, and message type. For example, PRE-PREPARE, PREPARE, and COMMIT messages for PBFT of the current round are subject to consensus. Round here means the latest height or position of the blockchain for which a block needs consensus to be enchained on the existing blockchain structure.

2. **BFTMessagePool:** Being similar to the classes of TransactionPool and BlockPool, it acts as the storage with a structure of HashMap embedded with block hash as key and CopyOnWriteArraylist as value to store the different consensus messages. This representation can be extended to any consensus message storage and manipulated. The operations built into this class are adding, removing, getting the size of messages accumulated, and clearing messages for memory management.

3. **BFTMessageHandler:** This is more like a sentry with the principal method or function **handleMessage()**, which looks at the consensus message being received. Depending on the algorithm, it ensures that the messages are sorted, received, and proceeding to the next round if consensus is completed. The check of the message threshold for each consensus phase is applied, and relaying the unique consensus messages to the peers. Since this is the core class of the consensus algorithm, we followed a more abstract or generic design to accommodate any BFT family algorithm by following the Strategy Design Pattern, which will be discussed next.

4.1.3.2.5 Strategy Design Pattern: This behavioral design pattern is oriented towards designing a family of algorithms and plugging one instead of the other making it interchangeable. This is represented in Figure 4.9, where the Simulator Controller Class is the deciding place to replace one algorithm with another. All the BFT family algorithms we have implemented are concretized versions of the principal interface BFTMessageHandler. The concretized versions of this interface implemented as PBFTMessageHandler for PBFT Algorithm [73], CliqueMessageHandler for Clique Algorithm, IBFTMessageHandler for IBFT Algorithm, and QBFTMessageHandler for QBFT Algorithm [121]. Each implementation overrides or implements the method signature **HandleMessage()** where the logic for each of them is defined.

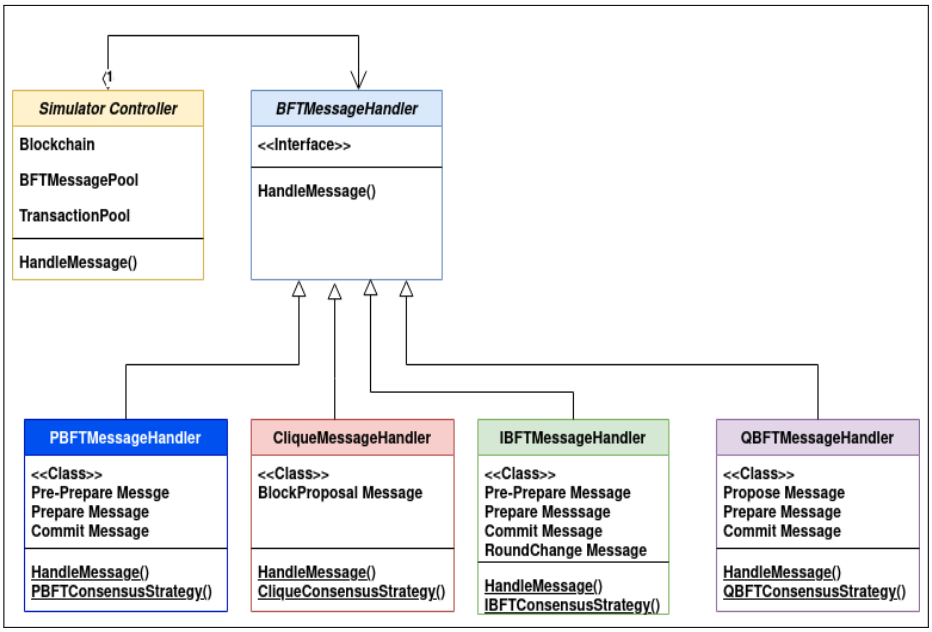

Figure 4.9: Simulator Consensus Module Strategy Design Pattern

4. **Synchronizer:** Since BFTMessageHandler is multi-threaded, synchronized data structures are utilized to maintain the thread safety and concurrent operation of the consensus phases, which is ensured by a boolean for completion.

4.1.3.2.6 Cryptography Package: This utility package is organized into NodeKeyGenerator and CryptoUtil classes. It comprises cryptographic functions that generate Rivest-Shamir-Adleman (RSA) keys, Secure Hashing Algorithm (SHA-256) bits hashing, encryption, decryption, hash verification, and generation of signatures. This package is utilized while sending a transaction, block, or consensus operation during the message's generation, transmission, validation, and reception.

4.1.3.2.7 Network Package: This module establishes all the network functions that bind the entire Peer-to-Peer network. All the nodes are connected by the socket communication-based multi-threaded channel to pass and receive messages simultaneously. The module is split into three components:

1. **NetworkTopologyGenerator:** This class generates the participating nodes' network topology or routing details. The topology generated is a triplet <0,1, true>, which signifies that node 0 will be connected to node 1 depending on the next boolean, connected if true or ignored. The following part will explain the different topologies we have limited to our simulation.

   4.1.3.2.8 Dynamic Network Topology: Different Dynamic Network Topologies need to be tested in any simulation ranging from an optimistic to a realistic scenario. The node topologies we generate for the test are:

   (a) **Fully Connected Mesh Topology:** This is an ideal network but expensive in

terms of communication as represented in Appendix Chapter 8 through Figure 8.18. Each node is inter-connected, directly opening a socket communication, one for transmission and the other for reception. This ideal scenario helps us test a network's upper bound or max performance and the consensus algorithm. We focus on consortium networks with limited participation, so this topology choice will be close to our envisaged production network.

(b) **Ring Lattice Topology:** This is a limited connection model as represented in Appendix Chapter 8 through Figure 8.19. It has a high clustering coefficient and path length if we consider a big network but less than a fully connected mesh network. Clustering means that nodes with more connections will be more feasible to connect further while other nodes are more likely to be deserted. The path length is defined as the average distance between any two nodes in the network.

(c) **Watts Strogatz Topology:** This topology can be termed a model with high clustering and short average path length. As explained earlier, it is built on top of a lattice network where the node connection is rewired randomly with a probability. In our representation as in Appendix Chapter 8 through Figure: 8.20, we have a degree of 4 with a rewiring probability of 0.05, ranging from 0 signifying regular to 1 for complete randomness. This topology is not a realistic model but a generative model of a social network where nodes are connected randomly with distant nodes, making it a small world network reducing the path length.

2. **P2PServer:** (P2P: Peer-to-Peer) This class is a networking class with logic embedded in NodeCommunicator and NodeCommunicatorThread classes to open socket communication, sends messages via the opened socket, and finally listens to the incoming socket communication messages. This class then passes the information to other logical instances of Blockchain storage, Transaction Pool, Block Pool, BFTMessage Pool, and QueueResource, which will be discussed next. Multiple Channels of NodeCommunicator and NodeCommunicatorThread are opened parallel to increase the bandwidth of message communication transmitted among the peers.

3. **QueueResource:** As our simulator is multithreaded, multiple threads will likely read the same message in parallel if it is broadcasted directly and received by another listening thread. To overcome this problem, this resource class stocks all the messages from the P2P server class socket connection in a Blocking Queue data structure, ensuring thread safety operation. All operations are atomic and utilize an internal lock.

   All the other logic threads of consensus, transaction pooling, and block pooling look at this queue for new messages. It will be allowed to pick a unique message despite multiple threads reading on the same queue. The queue resources created are separate for each concern, like one for general messages, one for consensus, and the other for transaction queues. Like NodeCommunicator and NodeCommunicatorThread having multiple channels, we maintain the same resource queues in this module.

4.1.3.2.9 Node Package: This package is for Node Identification and participation composed into four classes as represented in Appendix Chapter 8 through Figure:8.21. The explanation of each is as follows:

1. **Wallet:** As signified by its nomenclature, it holds a node's cryptographic keys for signing a message, a unique node identifier, an instance of node property, and the function of creating a signed transaction for publishing in the network.
2. **Validator:** This class holds the verification logic if a message from a validator or block proposer is eligible to participate in consensus.
3. **NonValidator:** This class holds the verification logic if a message from a validator or block proposer is ineligible to participate in consensus but participates as a transaction issuer in the network.
4. **Transaction:** This is more like a factory class for transactions where the transaction payload is prepared by a node starting with a transaction identifier, message hash, timestamp, message signature, and node identifier.

4.1.3.2.10 Property Package: This module is used to set the behavior of a Node and the Blockchain network in general as illustrated in Appendix Chapter 8 through Figure 8.22. The node property of being a validator or not, socket communication port, the block size in terms of the number of transactions, network type, consensus message majority threshold, and peer information are all stored in this static class for a single one-time property definition here.

4.1.4 Technology Decisions

In the earlier section, we explained the different classes' architectural decisions, module design, and organization, giving us a comprehensive overview of the simulator. In this section, we present the technology stack utilized in our simulator, the user interface of our simulator, the infrastructure we exploit for our test bed, and the deployment process.

4.1.4.1 Simulator

The technologies used in the simulator are represented in Table 4.2, where an object-oriented language, JAVA, is chosen to suit our abstraction, polymorphism, and strategy design pattern. To build the REST API layer, we build the controller using the Jersey JAX-RS library. We utilize the default available library from the JAVA security package for security aspects and the Google Guava library for hashing functions. The entire source code is built using the Maven dependency management tool, generating a Java Archive. The Archive can be run on any Java Runtime Engine (JRE)-based environment. During the simulator development, the IDE used was visual studio and Git for the source code repository management.

Module	Technology/Platform/Library	Version
Core Development	Oracle Java Development Kit	8
Application Programming Interface	Representational State Transfer (REST) Jersey JAX-RS	2.19
Encryption/Decryption	Java Security	8
Hashing	Google Guava	20.0
Executable	Java Archive on Apache Tomcat	8
Build Tool	Maven	3.5
Integrated Development Environment	Visual Studio Code	1.74
Source Code Management	BitBucket / Git	2.3

Table 4.2: Simulator Technology Stack

4.1.4.2 Simulator-User-Interface

The simulation is rendered with a modest UI developed to visualize the test's key vitals at first sight. Its snapshot for 5 nodes cloud deployed configuration is represented in Figure 4.10 and Figure 4.11. It is built using Javascript libraries which interact with the API layer to get the desired properties or simulation results as a Javascript Object Notation (JSON) data and render it on the User Interface (UI). The technology used in the Simulator UI is represented in Table 4.3.

Module	Technology/Library	Version
Core Page	Javascript	ES2022
Visualization	Chart.js	2.19
Cascading Style Sheets	Bootstrap	3.3
RunTime Environment	Node.js	16
Integrated Development Environment	Visual Studio Code	1.74

Table 4.3: Simulator UI Technology Stack

4.1.4.3 Infrastructure

All our simulator tests were performed on the TAS Cloud, whose data centers are in Sophia Antipolis, France. As a partner of the Smart IoT for Mobility (SIM) project, where this thesis is supported, TAS offers Infrastructure as a Service (IaaS). The resource available for the simulator test is represented in the following Table 4.4. The dashboard of the Rancher management tool is illustrated in Figure 4.12.

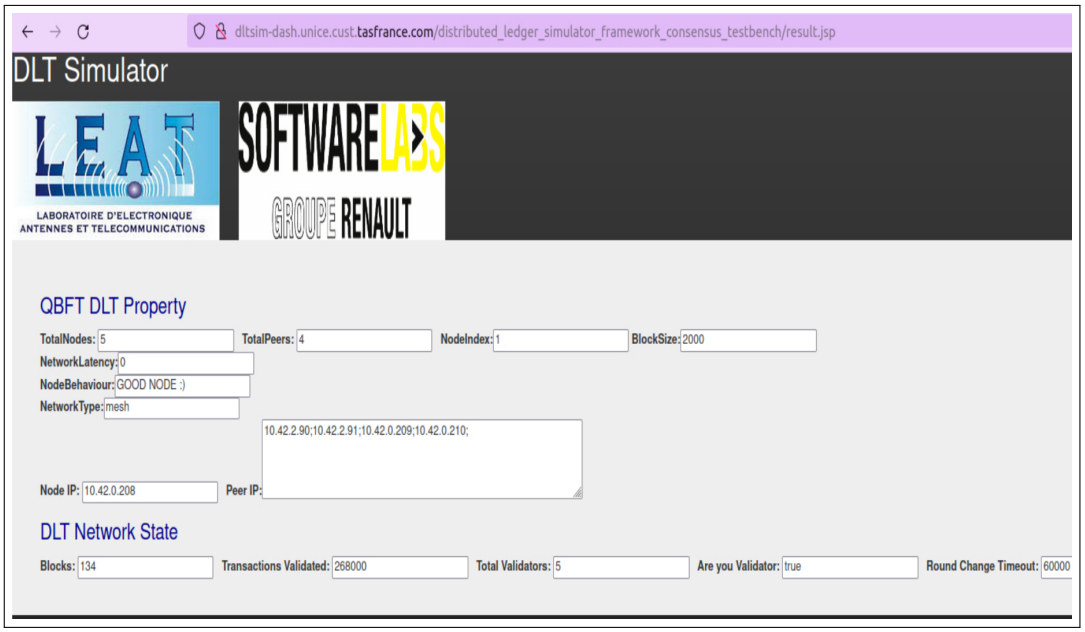

Figure 4.10: Simulator Explorer Dashboard User Interface Snapshot 1

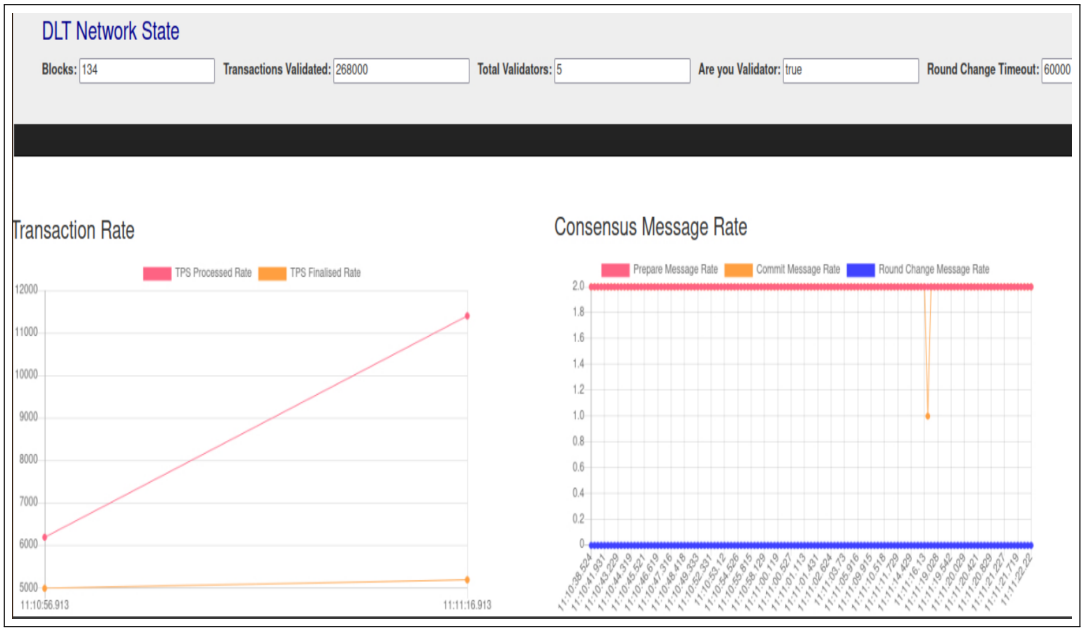

Figure 4.11: Simulator Explorer Dashboard User Interface Snapshot 2

4.1.4.4 Simulator Deployment

The test deployment workflow of the simulator on the infrastructure is represented in Figure 4.13. The technology used for deployment is listed in Table 4.5. All this workflow is automated by shell scripts, starting with the building of a simulator into a JAVA archive. The archive and dependencies are ported to an Apache Tomcat Docker Container Image, which is pushed to DockerHub. Then the Kubernetes YAML script is generated to deploy pods containing the earlier pushed docker containers. Then the launch script for the test starting the transmission of the transactions is launched from the test orchestrator machine.

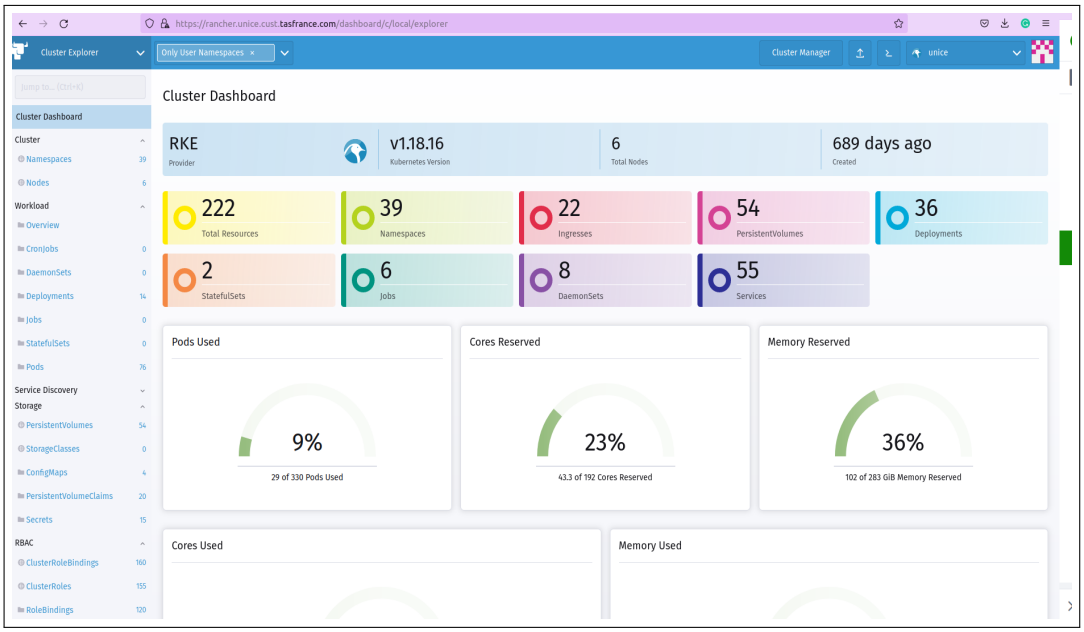

Figure 4.12: TAS Cloud Cluster Dashboard

Configuration	**Value**
Total Worker Nodes	3
Worker Node OS	Linux Ubuntu 20.4
Memory per Worker Node	94 GB
CPU per Worker Node	64 Cores
Pods per Worker Node	110
Container Orchestration Layer	Kubernetes 1.24
Container Orchestration Management Layer	Rancher 2.5.6

Table 4.4: TAS Cloud IaaS Configuration

Deploy Scripts	**Technology/Platform**
Binary	Java Archive
Containerization	Docker
Container Repository	Docker Hub
Deployment Scripts	Kubernetes YAML

Table 4.5: Simulator Deployment Technology Stack

The simulator user interface, as well as the Rancher dashboard, is verified to assure us of the health and statistics of the network, including infrastructure usage at the time of the test.

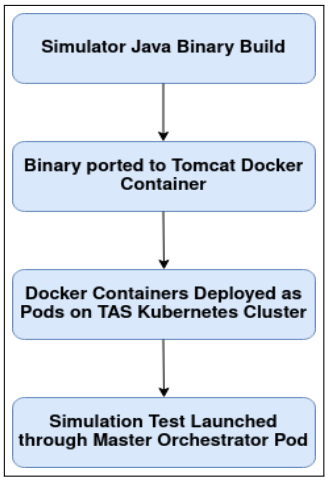

Figure 4.13: Simulator Cloud Test Deployment Workflow

4.1.5 Test Bed Architecture

The Cloud architecture of the simulator test is illustrated in Figure 4.14, which consists of a sample configuration for 5 simulator nodes. The master orchestrator node represented here signifies the Linux machine inside the cloud network, which can be used to configure the other different nodes along with their properties and network topology. It can be used to launch the test for transaction transmission and retrieve the results as Javascript Object Notation (JSON) data for analysis later using Rscripts.

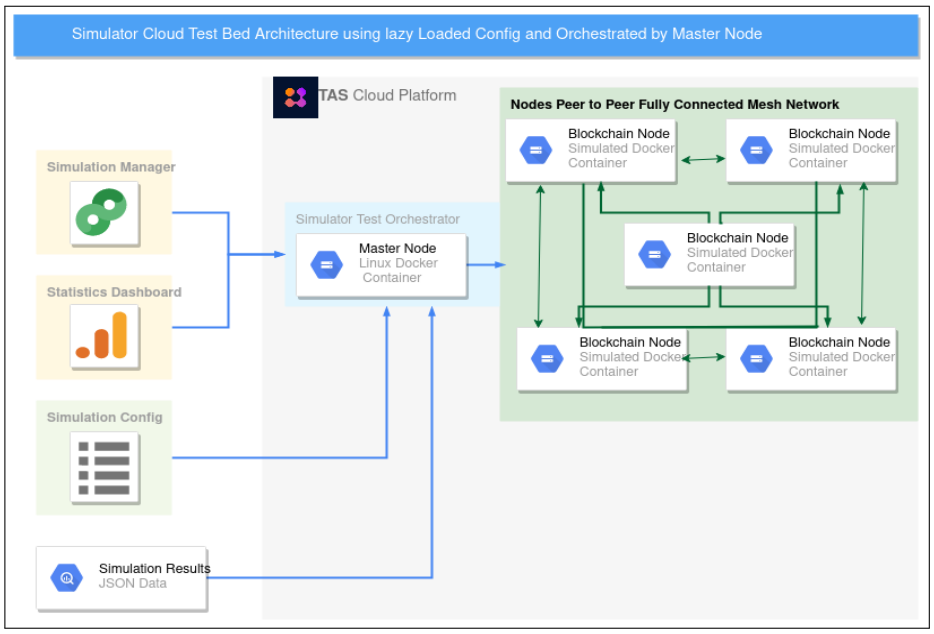

Figure 4.14: Simulator Cloud Test-Bed Architecture

4.2 Simulator Validation

Having discussed the simulator exigence, design principle, architecture, technology stack, and cloud testbed infrastructure, we proceed to validate the same by simulating the BFT consensus algorithms of Clique, PBFT, IBFT, and QBFT. We then validate these results against the real blockchain implementations of data certification and monetization results.

4.2.1 Choice of BFT Algorithm Study

We select certain algorithms to validate if our simulator behaves similarly or at least close to the actual binary implementation, enabling us to study the BFT algorithms profoundly. The following are the justifications for selecting Clique, IBFT, PBFT, and QBFT consensus algorithms for our simulation validation.

1. All these algorithms belong to the BFT consensus family, which is well implemented and maintained by prominent blockchain frameworks of Ethereum Geth, Hyperledger Sawtooth [75], and Ethereum Besu.
2. All these sets of algorithms belong to the variant of Proof of Authority wherein a set of well-identified limited participants is chosen to participate in the consensus and create a private consortium network that interests us.
3. We don't consider the AURA algorithm for our simulation, which is implemented as a standalone consensus algorithm by Ethereum Parity or as a Hybrid format along with GRANDPA [195] or BABE [196] finalization algorithm by Substrate Blockchain Framework. The reason is that AURA consensus by itself has two pitfalls in either standalone or hybrid as follows:

   (a) In the work by Shi, Elaine [197], they analyze the algorithm mathematically where it cannot tolerate up to 1/3 resilience parameters and suffers from consistency issues in the case of forks. They point out that 1/3 adversary resilience is tight when resolving forks of the same length, leading to consistency issues instead of recommending a tolerance level between 1/6 and 3/8. They have disclosed this recommendation to the Parity Team as well.

   (b) This work by De Angelis, Stefano and Aniello, Leonardo and Lombardi, Federico and Margheri, Andrea and Sassone, V. [117] analyses the AURA consensus based on the Consistency, Availability, and Partition (CAP) Tolerance Theorem. Any distributed system satisfies at most two CAP properties, CA, CP, or AP. Since AURA is based on epoch time for round progression and clock skews can affect the node synchronization, it can lead to inconsistency. In the case of inconsistency for blockchain leads to unresolved forks, and the transaction or blocks are bound to be unfinalized at any time.

4.2.2 Simulation Test Methodology

As explained in the earlier section 4.1.4.4, we vary our network configuration concerning the number of nodes for each consensus algorithm to test its performance and scalability. We maintain an optimistic, Fully Connected Mesh Network Topology scenario with

a block size containing 2000 transactions across all the tests. The simulation starts with firing the transactions from each multi-thread node and then forwarding them to the consensus layer for finalization into blocks constructed on the blockchain. We measure the performance in terms of transaction per second (TPS), where we calculate the time lapse between the launch of the transaction and its finalization onto a blockchain divided by the total finalized transactions. In the next section, we discuss the simulation results for each algorithm. Code Implementation of the java blockchain simulator for Clique, IBFT, PBFT, and QBFT, along with Kubernetes cloud deployment files, transaction clients, docker configuration files, simulation result visualizer, and test results are released publicly in the GitHub repository: https://github.com/scyrilnaves/these-blockchainconsensussimulator

4.2.3 Discussion on Results

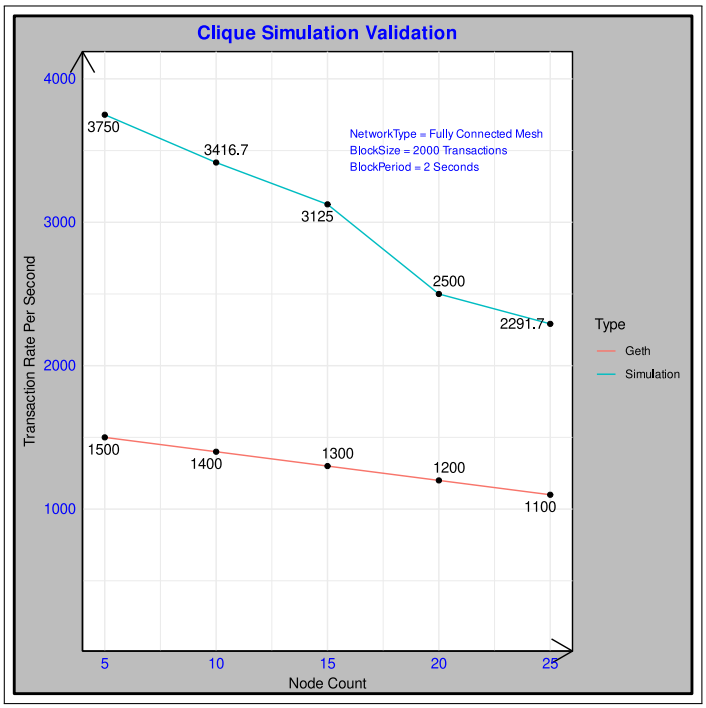

Figure 4.15: Clique Consensus Algorithm Simulation Validation

The simulation results of the consensus algorithms Clique, PBFT, IBFT, and QBFT are represented in Figures 4.15, 4.16, 4.19 and 4.20. The results of implementing different consensus algorithms are re-visualized and compared as discussed in Chapter 3.

In the case of the Clique Algorithm, the simulation results and the actual implementation results from Geth Binary are represented in Figure 4.15. It shows a decreasing trend in transaction throughput as the number of node participants increases. The scalability problem is attributed to communication cost but in the case of Clique, it is less compared to other consensus algorithms discussed here. Instead in this consensus protocol, it is just the propagation of the proposed block to the other node participants. The performance of the actual Geth implementation is less drastic than the simulation's gradual drop because there is an additional bottleneck problem at the level of Ethereum Virtual Machine, which

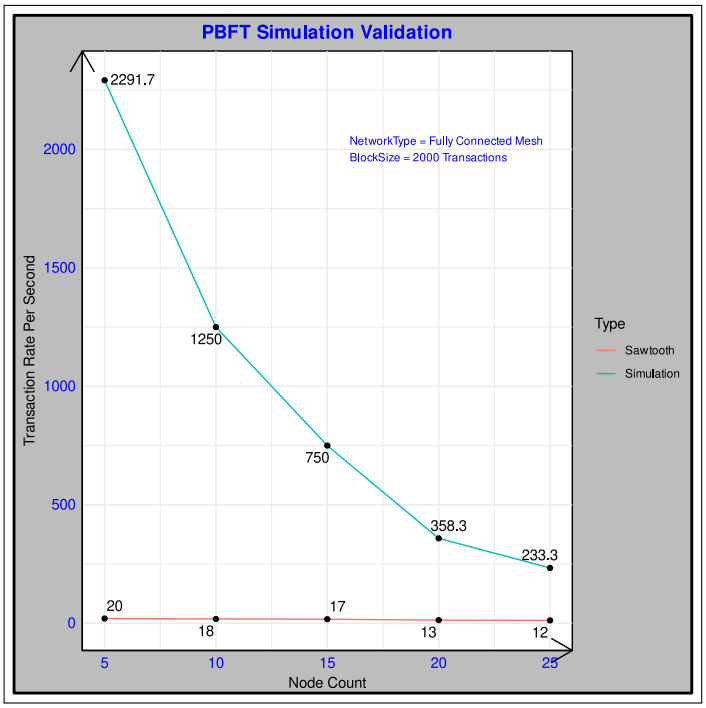

Figure 4.16: PBFT Consensus Algorithm Simulation Validation

is already discussed in Chapter 3. This limitation prevents the actual Geth implementation from having a higher transaction rate and shows minimal change with the scalability factor. In terms of the real transaction rate, the difference between the simulation and the actual is due to transaction type being EVM smart contracts, Go implementation, encoding methodology, data type, encryption technique, and other factors discussed earlier in Table 4.1.

Next to look closer is the PBFT algorithm results as in Figure 4.16, where there is a huge difference in the actual Hyperledger Sawtooth versus simulation transaction rate and a drop in scalability. The vast difference in transaction performance is due to the Hyperledger Sawtooth implementation having a collective drawback of choosing Python, which is an interpreter language, docker containerization of the binary, and intra-communication bottleneck between the REST API module, consensus engine, and transaction processing. The algorithm, in general, has a steep drop attributed to its costly communication complexity of $O(N^2)$ and $O(N^4)$ in normal and view change cases. To cross-verify our simulation result due to the high range of performance difference, we study an external work on PBFT consensus by Tang, Song and Wang, Zhiqiang and Jiang, Jian and Ge, Suli and Tan, GaiFang and Wang, Yong and Zhong, Meiling and Cheng, Tong [198, 199].

In this work, [198], the authors improvise the version of the classical PBFT [73] algorithm by introducing a trust equity scoring mechanism between the nodes to increase the performance safely and efficiently. Based on this score, they adjust the list of consensus nodes that can or cannot participate in block finalization. By increasing the number of nodes, they compare the performance and scalability between the existing and improvised as illustrated in Figure 4.17. These results corroborate with our simulation results as represented in Figure 4.16 but differentiate directly from the actual Hyperledger Sawtooth re-

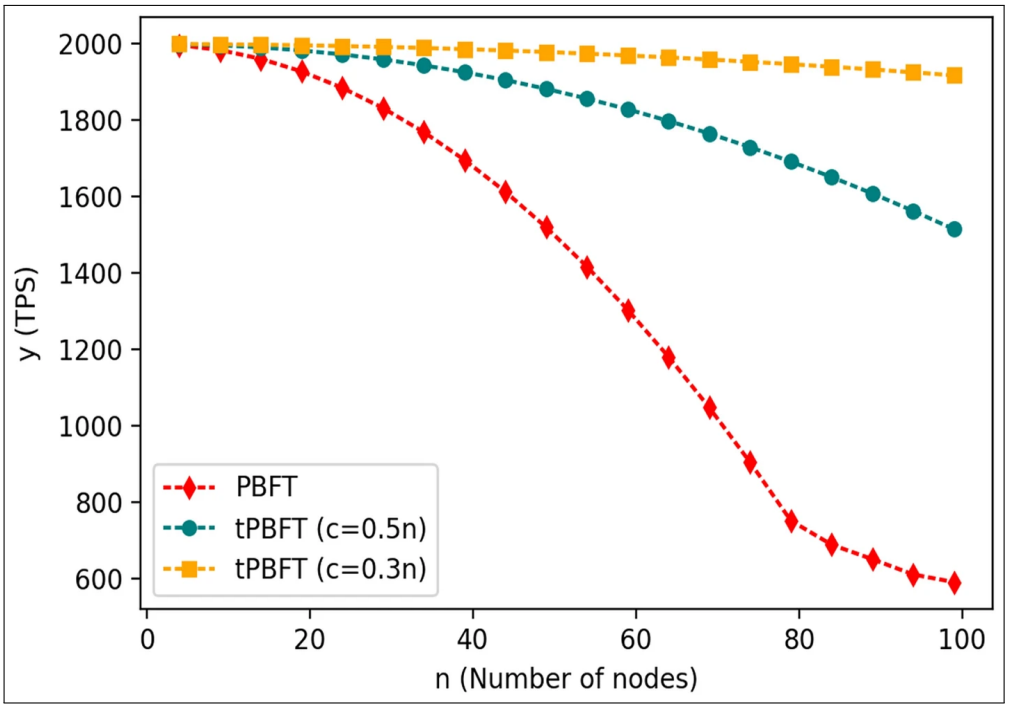

Figure 4.17: Improvised PBFT & Classical PBFT Algorithm Simulation Performance [198]

sults, which validate our argument of programming language choice, binary development, and containerization problem of Sawtooth.

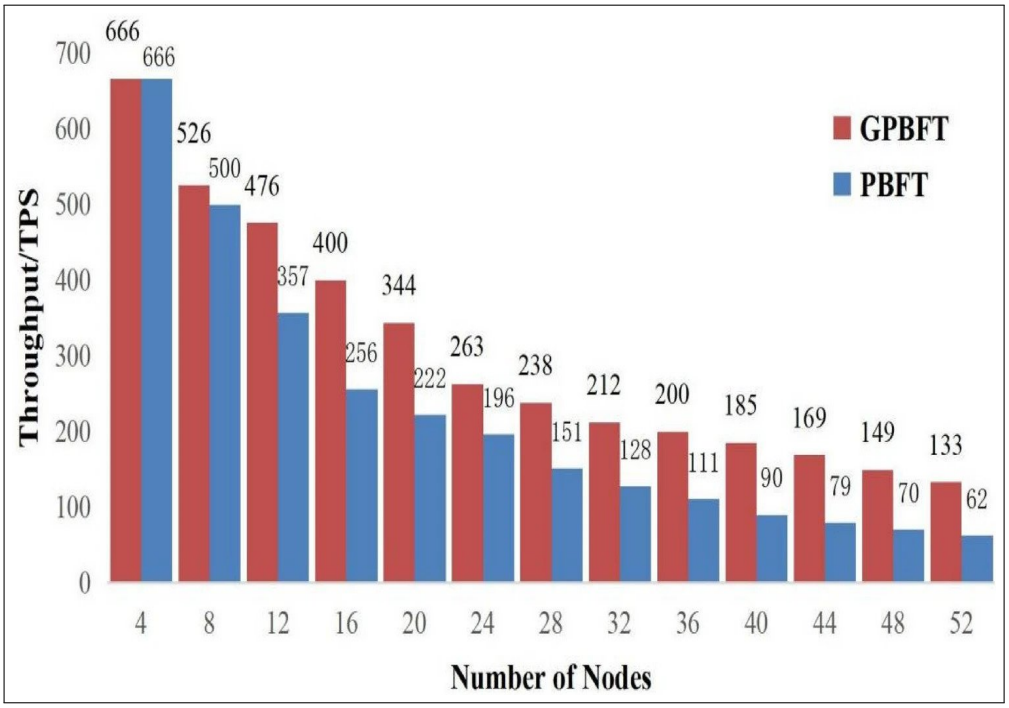

Figure 4.18: Grouped PBFT & Classical PBFT Algorithm Simulation Performance of Wang, Yong and Zhong, Meiling and Cheng, Tong in [199]

In their article [199], Wang, Yong and Zhong, Meiling and Cheng, Tong have proposed a

Grouped PBFT algorithm (GPBFT) in which the node's trust degree is evaluated based on an Eigen Trust Model. The trust degree of nodes is used as the basis for the election of nodes divided into several groups to form a consensus opinion. The group formation reduces the communication overhead, leading to better throughput as represented in Figure 4.18. On the other hand, the results presented for the classical PBFT algorithm represent a scalability drop similar to our obtained simulation results in Figure 4.16. These two comparisons of our simulation with the works of Tang, Song and Wang, Zhiqiang and Jiang, Jian and Ge, Suli and Tan, GaiFang, Wang, Yong and Zhong, Meiling and Cheng, Tong [198, 199] validate our simulation procedure behavior.

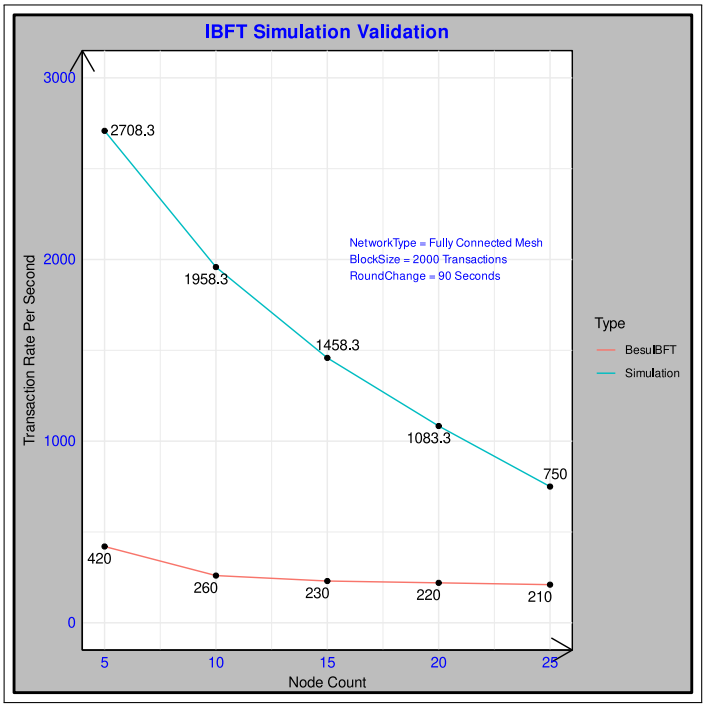

Figure 4.19: IBFT Consensus Algorithm Simulation Validation

The next validation is the Istanbul Byzantine Fault Tolerance Algorithm as represented in Figure 4.19. The Hyperledger Besu Binary, which implements IBFT consensus, has a hashing bottleneck, already discussed extensively in Chapter 3. Due to this identified issue, the Besu implementation performance gradually drops between 5 and 10 nodes, aggravating further with more nodes and then settling with a horizontal plot. The simulation has a drop in performance with increasing nodes which is explained by the increased consensus phases of Prepare, Commit, and Round Change. As the phases are reduced compared to PBFT, it introduces the problem of forks where multiple blocks can be proposed at the same block height, which is solved by its block-locking mechanism. These factors slow the performance, and the augmenting communication complexity with increased peers explains this observed behaviour.

Similar to IBFT, the following analysis on QBFT [121] is built on the same Hyperledger Besu code base and inherits the same hashing problem discussed earlier. The performance results of the simulation and actual as represented in Figure 4.20 show a better performance algorithmic-wise as it is an improvement over IBFT. It has a lesser communication

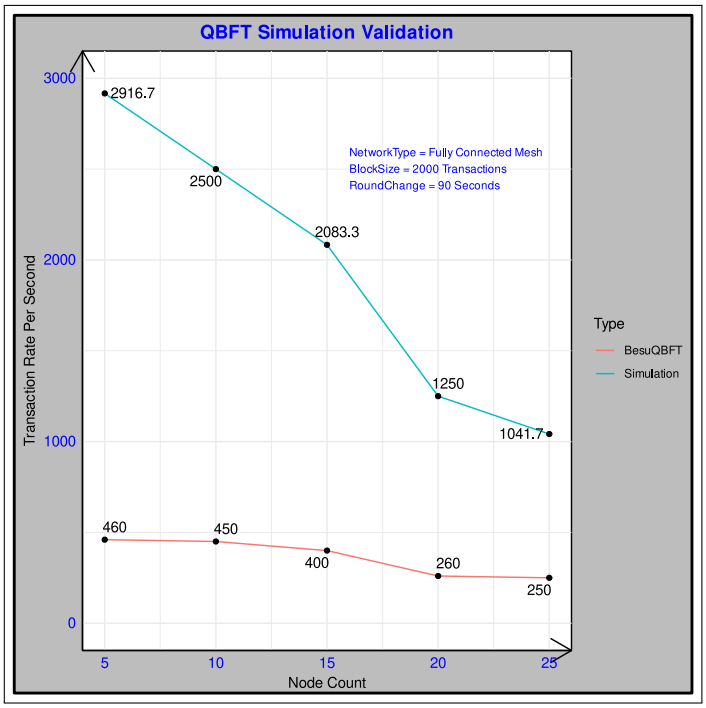

Figure 4.20: QBFT Consensus Algorithm Simulation Validation

phase with Prepare, Commit, and a scheduled timer-based round change. The implementation performs better than IBFT and has a lower scalability drop but suffers from the same plateaued throughput effect discussed earlier. The simulation also shows a decline with increasing nodes, but QBFT simulation has a higher performance and scalability tolerance than IBFT, similar to the actual. The higher throughout observed compared to Besu is due to the homogeneous streamlining of simulator decisions avoiding bottlenecks of binary implementation or hashing functions.

All the BFT algorithms discussed earlier are simulated and verified with the actual implementation. It shows a similar and valid performance drop in scalability and consistency due to the following factors by utilizing a homogeneous test bed:

1. **Communication Complexity:** BFT algorithms of PBFT, IBFT, QBFT, except for Clique, always operate in phases (like Pre-Prepare, Prepare, Commit, and Round Change) and suffer from communication problems, augmented with increasing nodes. However, even proposed consensus improvisations like reducing the phases can lead to consistency and liveness issues.
2. **Fork Issues:** On the other hand, the Clique algorithm has a single phase of block proposal that alleviates the communication problem but introduces fork consistency issues, leading to deadlock or liveness problems in the network.
3. **View Change Complexity:** All the algorithms except Clique have nodes that progress through views or state transitions. However due to asynchronicity or failure of the network, each node or set of nodes can have a different view or state of the network. If there is no synchronicity in terms of view, then it leads to varying perceptions of the network or blockchain, leading to forks and entailing deadlock.

4. **Strong Ideal Case Assumption:** Degraded performance, even in case of benign nodes or perfect synchronous network assumption, can further deteriorate the consensus algorithm in case of malicious nodes or network latency as well as failures which needs to be pre-emptively resilient.

4.3 Conclusion

In this chapter, we have introduced the necessity of building a simulator that arises due to homogenous blockchain binaries available in the community. We then discuss the related state of the art on blockchain simulators and their shortcomings in reusing or adapting them for our needs. Following this, we ideate our simulator design principles, architecture, and design pattern for conceiving our DLT simulator to deploy on any cloud-agnostic platform. We then test the simulator against the real blockchain binaries of Hyperledger Sawtooth, Ethereum Geth, and Besu to validate our simulator. This verification enables us to embark on the next step of developing our own consensus. The result discussions on the simulator shed light on the actual core problem we encounter in this family of BFT algorithms with the suitability for consortium networks.

The analysis motivate us to design our consensus algorithm to address the issues discussed and be resilient to attacks or benign failures. The various literature around the scope of improvement in BFT consensus algorithms deliberate the ideas of reducing communication bottleneck, pipelining across nodes [85] or better view synchronization [190]. A node scoring mechanism [198], Hardware improvements of utilizing Trusted Execution Technology [200], or even adopting SmartNICs (Network Interface Controller) and deploying the protocol on FPGA (Field-Programmable Gate Array) [201] inspires us to approach these shortcomings. In the next chapter, we solve the BFT consensus problems from a consortium blockchain perspective and try to improve the algorithm's performance, scalability, reliability, and resilience.

Chapter 5

CUBA: An Evolutionary Consortium Distributed Ledger Byzantine Consensus Algorithm

The said truth is that it is the greatest happiness of the greatest number that is the measure of right and wrong

– Jeremy Bentham

In the previous Chapter 4, we identify and list the issues in the set of BFT algorithms tested. Now we would like to discuss our problem statement and the various approaches we can follow to solve the problems of scalability, performance, failure, and security resistance in the BFT protocols. Then we look profoundly into these approaches and identify that there are still some challenges to be answered comprehensively. These factors inspire us to conceive our novel consensus algorithm CUBA. CUBA expands to Contesting Utilitarian Byzantine Agreement which evaluates and valorizes the actions as metrics of the gamified participants in the network. The obtained metrics are used as feedback to reorganize the network either for faster performance of the network consensus or to be resilient to the malicious activity noticed. This consensus protocol is designed to both sustain or increase the Utilitarian happiness in a Byzantine environment of identified participants for the "Liveness, Safety, Security, Performance and Scalability" of the network.

5.1 Problem Statement

Here, we define our problem statement concerning our Byzantine Fault Tolerant Consensus notion. We set the conditions and interests under which the network participants would behave. In addition to the Byzantine Agreement among the peers, which is the expected convergence in the network for data or transaction finalization, we expect certain other challenges to be resolved as part of it. We define each of them in the following subsections.

5.1.1 Network Definition

The distributed or Peer-to-Peer network which we assume is a consortium blockchain network. This type of network is chosen because private enterprises like Renault, Mercedes-Benz, Porsche, IBM, or other Original Equipment Manufacturers prefer consortium networks to balance transparency and privacy while creating inter-organizational synergies. This, in turn, makes the network to be formed of a limited count and well-identified participants with an established reputation.

We consider K-designated Node participants in the consortium network identified by N_0 .... N_{K-1}. The participants send and receive transactions that need a common agreement on the transaction order, validity, and immutability. The transactions agreed upon are, in turn, to be stored as a block structure. We consider a Transaction represented by T and a Block B of size M containing a set of transactions $\{T_0 \ldots T_{M-1}\}$. The Blocks are enchained as a blockchain structure of height S represented as $\{ B_0, B_1\{H(B_0)\} \ldots B_S\{H(B_{S-1})\} \}$. H is the hash function digest of the block data, which is usually the previous block's hash to be included in the current block.

Next is the network assumption concerning synchrony. We consider the model by Dwork, Cynthia and Lynch, Nancy and Stockmeyer, Larry [72], the eventual synchrony model. It is an asynchronous network where the messages may be delayed but eventually turn synchronous, delivering them in a Global Stabilisation Time δ (GST). GST is an unknown but finite time interval bound. This model respects the safety and liveness properties as the network can eventually stabilize within a limited number of byzantine nodes. The downside is that during a sustained asynchronous period, the protocol may default the liveness property known as FLP Impossibility by Fischer, Michael J. and Lynch, Nancy A. and Paterson, Michael S. [26] in case of a node failure. But we also welcome other extraneous components, like fixed communication timeout for failure detection or randomized election, to overcome this impossibility.

5.1.2 Information Broadcast

The nodes broadcast the transaction, including signature and other application data, through a reliable channel that tries to rebroadcast the transactions unless delivered successfully. The nodes need to follow the Atomic broadcast protocol, which satisfies the property of Atomicity, Total Order, and Termination [202], which was defined for a physical time U. We extend this protocol to the consensus mechanism by the following properties [203, 204, 205] of all eventually delivering within a GST δ :

1. **Validity:** If an honest node broadcasts a Transaction T_i then it is eventually delivered.
2. **Agreement:** If an honest node broadcasts a Transaction T_i then all the nodes deliver the transaction.
3. **Total Order:** If two honest nodes N_i and N_j deliver Transactions T_a and T_b, then N_i delivers the transaction T_a before T_b if and only if N_j delivers the transaction T_a before T_b.
4. **Integrity:** A honest node N_i delivers the transaction T_a at most once if it was previously broadcast by another Node N_j.
5. **Resilience:** If a transaction T_a is broadcasted to honest nodes, it is eventually rebroadcasted by all other honest nodes except the malicious or faulty nodes. Also, It implies that an honest node or network of honest participants should be able to identify the malicious or faulty nodes and either suspend their action or remove them from the network based on consensus. This retrospective adaptation or reorganization of the network is based on benign or non-benign failures to render the network consensus performance, consistency, partition tolerance, and liveness as usual.

5.1.3 Participant Behaviour:

Even though the consortium blockchain network is an ideal case of benign reputed participants with no malicious intent, but its quite a strong assumption. As noted by Ariely, Dan and Davis, Michael, Buterin, Vitalik [206, 207] who are behavioral economists stating that even within well-reputed organizations, dishonesty is common. We can notice the examples of Volkswagen, Meta, and Vodoo, which attracted several million dollars in sanctions due to human error. They argue that human psychology justifies cheating actions if its minute and not in an aggrandized form. This attitude is more of a starting point for the bigger ones to arrive, ransacking the whole organization, which might be true in our case despite reputed consortium members. They state that "everyone cheats a little from time to time. But most major betrayals within organizations – from accounting fraud to doping in sports – start with a first step that crosses the line" [207].

So our consortium network for a byzantine agreement needs proper reinforcement through economic or other means to handle this adverse behavior. On the other hand, the obvious benign network failures and delays are considered in our system or problem statement definition. If there are f dishonest or faulty nodes in a network of N nodes, the consensus protocol must be tolerant for the condition f$\leq (N-1)/3$.

5.1.4 Consensus Finalisation

If an honest node or participant N_A adds a block B_m containing a set of valid transactions $\{T_1 \ldots T_K\}$ after approval from other participants. The block is placed at a particular index M in the blockchain then no other node N_B can add another block B_n to the same index at any point of time [208] even in the presence of forks subject to eventual resolution.

5.1.5 Quality of Consensus Protocol:

In addition to the basic requirement of Consistency, Availability, and Partition Tolerance in any distributed consensus protocol, we add other desirable properties similar to quality of service in physical networks. These are from the blockchain perspective [209] like transaction finalization time, scalability, consensus decentralization measure, and resilience capacity of the protocol against impending malicious attacks.

Scalability in the blockchain is of two types: Vertical scaling and Horizontal scaling [210]. Vertical scaling involves augmenting the processing power of the overall chain by increasing the computation power of individual nodes or participants. On the other hand, Horizontal scalability magnifies the blockchain throughput by accommodating more nodes in the network at the expense of communication and consensus bottleneck. In our evaluation, we would measure the two aspects to improve individual node participation and, at the same time, consider the openness to add more nodes in the private consortium network for democracy and resilience.

And the last but not the least goal is the understandability of the protocol in similar terms to RAFT Protocol [38], where we try to measure the algorithm's simplicity in solving a complex problem. The idea should be clear and compartmentalized for easier understanding and verification of the protocol opening further intuitions.

5.2 Cross-Section of Byzantine Fault Tolerance Consensus Problems and Approaches

This section discusses the methodologies or propositions to improve the Byzantine Fault Tolerant consensus protocols' scalability, performance, and resilience in line with the Consistency, Availability, and Partition Tolerance Theorem. We further evaluate them to ascertain whether they can be a probable solution to our problem statement discussed earlier (in Section 5.1) or inspire us to continue. The probable solutions are posed as a suitability question to understand and evaluate their relevance as below.

1. **Can we reduce communication complexity?**
   This methodology aims to reduce the communication bottleneck through various methodologies but introduces additional problems. Single Phase Protocol and Linear Communication protocols like Zyzzyva [76] have a single phase of PRE-PREPARE message from a primary. Then it expects the replicas to acknowledge the message. It counts the protocol to work based on speculation that the replicas have no failures or communication loss. But this assumption is strong and has liveness and consistency issues as the network is always unpredictable. The work by [211] feature a BFT protocol with linear communication in the normal case and change to $O(n^2)$ [208] in the case of network problems.

   Similar work to Zyzzyva is SBFT [212], which follows a linear communication cost in the normal mode. However, in the case of faults, it adopts a linear PBFT utilizing threshold signatures, achieving linear cost. However, these protocols are more optimistic as normal case operations are ideal to consider and can suffer communi-

cation overhead in fallback cases of view change necessitating leadership selection. Also, the protocol of Clique and Aura [213] follow the same pattern of a single chosen leader. It assumes the replicas or followers function without any issue but in reality, is prone to chain fork problems or network deadlock if two leaders propose a block at the same height.

2. **Can we parallelize transaction processing through multiple chains or sharding?**
   To achieve better scalability and throughput, the single blockchain is split into multiple shards or chains [214]. Then in each shard, the node participants are split through voting participation. BFT protocol is applied for unique or non-conflicting transactions distributed in either shard through a four-phase protocol like PBFT. Then a 2-phase protocol is performed to merge the data across the shards. This approach is not only limited to blockchains, but parallelized state replication execution [208, 215, 216] has already been attempted, which might be a leeway for us to take inspiration from. It has better throughput, but the time to synchronize between the shards is not instantaneous, and the finalization time is comparatively higher. Also, network failure or delays can affect the consistency and availability of the protocol.

   The assignment of small sets of nodes to a particular shard can affect the resiliency as launching an attack in a small network occupied with a shard is easier. In the case of Elastico [217, 218], the sharding technology is used to parallelize the consensus network into smaller committees, each processing a disjoint set of transactions. It performs an intra-committee consensus to agree on a single set of transactions. It is a probabilistic consensus where each participant solves the Proof of Work hash based on epoch randomness. A relay is established to synchronize the transactions among the shards, which is reconfigured every epoch. However it suffers from the small committee problem due to multiple shards as it can reduce fault tolerance to 1/4 participants rendering risk for liveness property.

3. **Can we reduce the effective participants from the global set of participants limiting the consensus communication?**
   In this methodology, as proposed in Algorand Blockchain [219] from a universal set of validators in the network, a block proposer and an associated committee is chosen at random. It is chosen using a verifiable random function based on the individual economic stake as input. A similar approach is followed in Proteus Protocol [220] by Jalalzai, Mohammad M. and Busch, Costas and Richard, Golden G., where a set of committees is chosen randomly from the total validator set and is allowed to participate in the consensus. This minimizes communication to a randomized set of participants, ensuring higher throughput. But the protocol is based on the stake and incentive mechanism, without which it can lead to centralization and liveness issues [221, 222]. This economy-based protocol to select the proposer or committee will be irrelevant in the case of non-stake networks.

4. **Can we adopt a randomized BFT approach?**
   Protocols based on a randomized approach can be either through a randomized selection of transactions as in HoneyBadger BFT protocol [220] or randomized selection of validator committee from a total set of validators [220] in the case of Proteus BFT. These randomization protocols offer better performance and scalability for 100-200

nodes, but simultaneously, they question the efficiency and decentralization of the randomization process.

A random generation system should respect fault tolerance, unbiasedness, reliability, verifiability, unpredictability, and decentralization properties. A public centralized random generation system called Randomness Beacon by NIST (National Institute of Standards and Technology) exists. However, a decentralized system can be compromised if it relies on a centralized actor. Also, the factor of randomness in the sense of BFT Protocols cannot guarantee deterministic termination. It can only achieve a termination with a high probability, like in the case of Ben-Or's protocol; it achieves consensus only with high probability termination in a crash failure model [217, 223].

5. **Can we choose to run our consensus protocol on specialized hardware for efficiency?**
   In the case of István, Zsolt and Sidler, David and Alonso, Gustavo and Vukolic, Marko [224], they propose a consensus mechanism for maintaining a consistent and efficient data center. They move the Zookeeper consensus at the network level to run on a Field Programmable Gate Array (FPGA). They demonstrate the practical case of a hardware consensus and a main-memory key-value data store on FPGA microservers to perform better.

   In the next by Poke, Marius and Hoefler, Torsten [225], we explore the idea of improving traditional Replicated State Machine (RSM) protocols using Remote Direct Memory Access (RDMA) Primitives. They evaluate this concept on a strongly consistent key-value store and obtain a performance improvement along with log access management. We maintain the cloud-based infrastructure resources we had previously simulated and experimented with in Chapter 4. We want a simplified architecture and a symmetry between cost and efficiency, which might be disrupted as FPGA or dedicated hardware can be quite monetarily demanding with scaling participants.
6. **Can we choose a rotating leadership to improve faulty leader issues?**
   As discussed earlier in the case of PBFT type protocols [73, 76], the performance already degrades due to communication complexity. Further, it deteriorates if a fault is detected due to the leader's failure. To resolve reliance on a designated primary or leader and reacting a posteriori, the protocol by Veronese, Giuliana Santos and Correia, Miguel and Bessani, Alysson Neves and Lung, Lau Cheuk [226] rotates or spins the leader for each round. Faulty leaders are recorded and blacklisted to avoid future obvious failures. In our analysis, the protocol is robust against attacks on the leader or avoiding monopolistic tendencies of the single primary. However, choosing a leader per round is quite heavy on message-passing, even in benign cases, and impacts performance.
7. **Can we inspire ourselves to follow leaderless BFT protocols?**
   Leader-based algorithms create a single point of failure, leading to invoking the view change case, as we noticed in PBFT Leader-based Type algorithms. Also, the leader acts as a co-ordinator communicating to all the nodes back and forth for an agreement which is quite consuming in terms of communication and computation. In the work of ezBFT [227] Arun, Balaji and Peluso, Sebastiano and Ravindran, Binoy, they propose a leaderless protocol to minimize the client-side latency in WAN networks.

There is no primary replica that orders the requests, and instead, all the participating replicas can equally order any incoming message. Like PBFT-type protocols, they propose two variants called fast path and slow path for optimistic case and failure scenarios, respectively. This protocol forwards the request from a client to the closest primary by latency which then broadcasts to its peers. Each primary performs the individual ordering and then sends the results to the client. Upon attaining the threshold of messages, the client replies to the replicas with a commit message. This protocol is 40% latency efficient compared to PBFT [73], or Zyzzyva [76].

In the next seminal leaderless cryptocurrency protocol, Avalanche by Rocket, Team and Yin, Maofan and Sekniqi, Kevin and van Renesse, Robbert and Sirer, Emin Gün [228], they propose Snow protocol. It achieves high throughput by a sub-sampling approach where a node interrogates only a random subset of nodes for the opinion of a message lightening the communication. The other key factors are metastability which tilts the blockchain towards an agreement without any leader. It works by a node performing repeated sub-sampling to influence its decision about data, either augmenting or decreasing the data score. If there is a satisfactory confident score about the data, then it is finalized. The same behavior is propagated as an epidemic to the entirety of the network, thereby tilting the network towards the data being accepted or not.

These protocols are efficient as they are both leaderless and phaseless, but the uncertainty of the opinion in case of malicious nodes can lead to failure. As these protocols in production are added with the staking protocol to enable slashing of the funds in case of dishonesty, it presents a protocol based on economic concerns. Another observation is that although the communication is phaseless, repeated subsampling for each data among the peers as it scales can dampen and affect the network's performance.

8. **Can we improve the communication topology to diffuse the message and improve throughput?**
   As we had noticed in PBFT [73], the scalability is an issue due to a single leader being occupied with the direction of the consensus [229]. Although certain algorithms improve by leader collecting the message and later broadcasting it to others, reducing the communication complexity to linear O(n) [208, 230].

   The next innovation is in the arrangement of nodes in a hierarchical manner such as a Tree structure [229, 230, 230]. In this, multiple signatures are merged into a single collective signature and disseminated top-down from the leader node in the root level to the children, the intermediate node, and finally, the leaf nodes. The response from the leaf is then sent back in the bottom-up approach from the leaf node to the apex root node, thereby economizing on the message traversal cost. Here the complexity is O(1) as only the collective signature is transferred and received.

   Next, we understand the more natural approach of Gossip by the protocols of Algorand [192] and Gosig [231] where a node passes the message to N other random nodes. This is done in epidemic-type dissemination, where the virus carrier or message propagates the disease. In the case of topologies discussed, the rigidity in the topology can affect liveness, at least in the case of a tree. Consider a malicious actor

in the leaf or root that can affect the entire structure, which needs additional redundancy as fallbacks. In the case of Gossip, the same malicious factor is considered, and also, the message transmission is to random neighbors, which makes the protocol probabilistic lacking finality.

9. **Can we use cryptographic primitives to improve the signature or cryptographic techniques?**
   The most prominent improvisations [229] in the BFT protocol can be listed as follows:

   (a) **Threshold Signature:** This technique merges several message signatures into a single entity where the communication is drastically reduced [83, 212]. It holds the condition that the threshold signature (t,n) [229] has at least t participants of total n which produce the signature.

   (b) **Collective Signatures:** In this protocol [232] the scalability is achieved by constructing Schnorr multi-signature [233]. The signature uses a tree topology from the top-down and bottom-up approaches. The improvement is the ability to combine the signatures reducing payload and communication. Also, signature verification is easier as a single combined signature is enough to be verified rather than n different signature.

   (c) **Threshold Encryption:** In this encryption scheme, similar to the threshold signature (t,n), at least t participants must decrypt a message from a total of n participants. This ingenuity is implemented in HoneyBadger Protocol [99] to improve the Asynchronous Common Subset (ACS) primitive as the Honeybadger transmits a unique or disjoint set of random transactions between the peers. This set of transactions is encrypted using a threshold scheme, and then an agreement is made on the cipher without revealing the transactions to avoid censorship. So the communication is optimized with less payload and also linear transmission.

   (d) **Verifiable Random Function (VRF):** This function chooses a subset of nodes without any interaction between the nodes. This protocol is used for privacy protection through blind auctions or in blockchain for cryptographic sortition to choose actors secretly and Domain Name System Security Extensions. Protocols like Algorand [192] use this function to check a node's eligibility to participate in the consensus committee by including one's private key and a common seed shared with all participants. After the eligibility is checked, the proof is generated and shared for verification without sharing the private key or performing any additional interaction.

   The cryptographic protocols discussed offer lighter or linear message communication, decreased payload, and even privacy at minimal interaction in the case of VRF. But the idea to minimize the message is justified, but the cost of higher complex cryptographic calculation needs to be considered. The VRF offers privacy which can solve many security issues as well. Still, the lack of standardization in the protocol leads to key disclosure, falsified proofs, digest collisions, or unacceptable randomness.

10. **Can we adopt technical improvements in BFT protocols for scalability and performance?**
    Along with the intrinsic factors in the protocol, as discussed in the above points, we

now explore the idea of parameterized factors in any blockchain. Some of the factors are as follows:

(a) **Block Size:** A BFT consensus's performance is directly proportional to the message payload involved. In the case of blockchain protocols, each block containing transactions plays a critical role in the message size. With the block size increase accommodating more transactions, there is a performance drop. It is due to higher network communication and increased signature or transaction verification, affecting per-unit block finalization through consensus. It would be ideal to parameterize an optimal block size by testing the protocol implementation.

(b) **Block Frequency:** Certain BFT protocols perform consensus at pre-defined intervals for a harmonious output of blocks like Clique, Aura [213], IBFT [119] and QBFT [121]. These fixed time intervals for each block round can be either elevated or lowered depending on the implementation, as we also need to consider transaction processing and verification. A shorter frequency will be inadequate to perform the initial transaction processing, construction of blocks, and then performing the consensus. A balanced time interval frequency should be determined, avoiding the race condition between consensus and block creation.

(c) **Transaction Processing:** The transactions submitted to the blockchain via a client's Application Programming Interface (API) are normally through Remote Procedure calls. To rationalize these calls, a client can submit transactions in a batch format, optimizing the requests. In another pattern, the client can be multithreaded, allowing multiple sockets or web service connections to the blockchain node, enabling faster transaction submission for better throughput.

(d) **Number of Validators:** This is an obvious parameter as the number of participants increases in the BFT consensus participation, the message communication, signature verification, and consensus participant threshold also increases. So an ideal level of participation is to be chosen considering the democratic nature of the network and also the resilience of the protocol against attacks. Although a less number of participants can offer high performance, the idea of a distributed protocol ensuring transparency, validation, security, and audit by all the participants is the goal of the distributed system.

(e) **Layer 2 Solutions:** These are secondary solutions built on top of primary blockchain networks to offload the computation from the main chain, ensuring faster finalization of the transactions and lesser computational resource consumption. The following are the solutions created for established cryptocurrency blockchain networks such as Ethereum and Bitcoin. The former uses the Proof of Stake, and the latter uses the Proof Work BFT protocols.

    i. **Off-chain:** In this technique, the transactions are completely obscured from the main blockchain. It can be the transfer of private keys to a designated wallet instead of the actual cryptocurrency transfer, thereby avoiding the cost of transfer and transaction finalization time. These are implemented for Bitcoin-based blockchains involving crypto-money transfers. Another protocol variant involves transmitting information through push

notifications between multiple chains for cross-chain messaging between Ethereum, Polygon, and BNB Chain. For example, Lightning Network and Liquid Network.

ii. **Side Chains:** These chains seamlessly integrate with another chain type with different consensus mechanisms and account management. These are interplayed to maintain transaction efficiency on one chain and then transfer the final outcome transaction to the main chain. Examples of these side chain frameworks are Plasma and Polygon.

iii. **State Channels:** These are similar to off-chain, but in this case, a state channel is created for a particular state of the blockchain network and agreed upon using a smart contract on the main chain. Then a set of transactions are performed in the state channel, which is faster as there is no consensus, and then upon finalization, the state is updated on the main chain smart contract. Examples of these state channels are Trinity Network and Raiden Network.

The above set of techniques or parameterization discussed, impacts the Byzantine Consensus Protocols' performance, computation resource, and scalability [208, 234]. The parameter improvisation, like block size and frequency, are minor value additions to the protocol improvement. However, a more comprehensive study is needed on the heart of the consensus protocol to improvise and adapt to our problem statement. Likewise, in the case of Layer 2 scaling solutions, it is more oriented towards masking the transaction from the main blockchain risking the transparency, consensus, validation, and democratic nature of the BFT protocols, which cannot be suitable to our perspective of the problem.

5.3 Contesting Utilitarian Byzantine Agreement (CUBA)

In the previous chapters on the State of Art (Section 2.2), Use Case Oriented BFT Evaluation (Chapter 3), Simulator Applied BFT Evaluations (Chapter 4) and in the previous BFT cross section (Section 5.2) we have been able to understand the various consensus protocol innovations, successes, and drawbacks. This has enabled us to sieve our focus towards the core of our previously explained problem statement. We are motivated to propose our own consensus algorithm philosophy, which can hypothetically answer our challenges. This philosophy will be further concretized into algorithmic operation and evaluated if it can respond to our problem statement.

5.3.1 Philosophy

In this section, we construct the metaphorical aspect of our consensus by imbibing various well-established philosophies. This construct will act as the guiding beacon while we design our consensus protocol to solve the problem stated earlier. We consider the earlier definition of the network in subsection 5.1.1.

It assumes a K designated Node participants in the consortium network identified by N_0 .... N_K. The participants send and receive transactions that need a common agreement on the transaction order, validity, and immutability. The transactions agreed upon are, in turn, to

be stored as a block structure. The blocks are enchained in a blockchain structure according to a well-agreed order of precedence.

5.3.1.1 Democracy Conundrum

During 507 BC in Ancient Greece, Cleisthenes introduced the concept of democracy in the aftermath of *End of Thirty Tyrants*, who were an oligarchy terrorizing their citizens. Although he belonged to an aristocratic family, he despised the idea of his heritage. He wanted to absolve the power of the aristocrats and hand them over to the common people or *Demos*, which we enjoy now. But the other side of the coin has to be analyzed as well.

A great Greek philosopher, Socrates, was skeptical about democracy. He attributes his decision to the process of election, which without an understanding, can wreck anarchy in the state. An election process must be intellectually driven rather than an adult franchise. Although he doesn't undermine the idea of having a democratic electorate, the wisdom of the people is still doubtful. A tragic incident in his life is illustrated in Figure 5.1 where Socrates is condemned to death in 399 BC by handing over a cup of poison hemlock. He was tried for corrupting the youth, and the jury comprised 500 Athenians. After the voting process on his trial by the jury, the verdict was directed against him as he lost by a narrow margin of votes. Athenian concept of democracy by itself was short-lived and insidious. With a selected few of 30000 male citizens to make decisions on political issues, the rule was very oppressive. This enunciates that democracy needs other friends in its circle, such as education, intellect, and self-regulation, to prevent itself from falling back to anarchy or autocracy.

Figure 5.1: Death of Socrates by Jacques Louis David [235]

Consensus Intuition

Similarly, our consensus protocol formed by the participants N_0, N_1..... N_{K-1} in consortium C needs a mechanism for informed decisions based on intellect. Intellect needs to be translated from an abstract point of view to computing space which will be a node's knowledge about other nodes in its universal set C. Knowledge will contain attributes about the behavior of the nodes concerning their transaction signature, validation, block proposal, block

acceptance, and maintaining honesty in the network. Knowledge of a Node N_K can be represented as π_K, which will be a set of individual behavioral perceptions on other nodes in C represented as $\{B_0, B_1.....B_{K-1}\}$

Similarly, each node will know about other nodes based on their past actions or behaviors in the network. This helps each node to make an informed consensus and modify the participant list if necessary. Using an informed decision, the democratic equilibrium is sustained and ensures the network is always secure, valid, and robust against incoherent behaviors.

5.3.1.2 Need for Utilitarianism

Utilitarianism was popularized by Philosophers Jeremy Bentham and John Stuart Mill in the 19th century, although its roots go back to ancient Greece like Aristippus and Epicurus [236]. The modern interpretation of it states its fundamental axiom as "the greatest amount of good for the greatest number." It is a moral theory that aims to maximize the happiness of society at all costs. Its limitations sometimes conflict with ethics, as happiness for most of society cannot be achieved by exploiting a poor few. But its proposition still holds in determining the right and wrong decisions for the ulterior motive of "happiness." As illustrated in Figure 5.2, the Good Samaritan is a Biblical parable of a man who cares for a wounded and robbed stranger despite his own mission in the journey as other passersby ignore the suffering person. He takes the wounded person to a shelter and pays for his expenses, which detracts from his own activity. This can be viewed as a utilitarian act that brings out a sense of pleasure for all at his own expense. We try to imbibe this same allegory in our consensus objective. It will be in coordination with a competition mechanism that enforces a sense of discipline and motivation between people in maintaining utilitarianism in the network.

Figure 5.2: The Good Samaritan by Eugène Delacroix [237]

Consensus Intuition

In the consortium network C, each Node N_K has an intellectual knowledge π_K about its peer nodes in the form of an effective utilitarian score. This score represents the cumulative effort of each node, along with others, in maintaining the utilitarianism of the network. Util-

itarianism is referred to metaphorically as honesty in the network to ensure the network's liveness, availability, partial tolerance, and resilience. This can be ensured by attributing an effective score to each node based on his knowledge and actions of the node in the past.

Consider a Block B_i at index i which contains a set of N transactions $\{T_0, T_1....T_{N-1}\}$ proposed by a Node N_J where J is an individual node identity. The Block contains a set of transaction emitters Te who have signed and issued it represented as set $\{Te_0, Te_1... Te_{n-1}\}$. The block has to be validated for consensus, taking into account the following example:

- If there is only a unique block at the destined index i.
- If the block is proposed by a valid node and provided a signed hash digest.
- If the block upon consensus attracts the necessary votes or approvals from other benign nodes.
- If the block is propagated and added to the index by all the nodes without any conflict or forks, ensuring a single persistent blockchain.
- If all the above actions are performed, i.e., communication or response in an expected time interval δ individually to ensure the stability and liveness of the network.

All these individual actions can be attributed to a utilitarian score U_a, U_b, U_c where a, b, c are considered actions, for example, enumerated above. These actions are then cumulatively managed in the network for each node as it performs according to the enumeration. Then, we measure each node's individual utilitarian UT_i score. This score is also maintained in the negative sense as a measure of the above-expected actions that are not performed or ignored by an expected node. It is represented as M_a, M_b, M_c where M represents misbehavior and a, b, c reprises the former notation significance. The negative scoring mechanism ensures a sense of discipline and competition as it can reduce the score of any supposedly utilitarian node. The Effective Utilitarian Score EU_n, n indicates the block height or index level in the chain measured as

$$\sum_{n=0}^{k} EU_{\mathrm{n}} = U_{\mathrm{a}}^{\mathrm{n}} + U_{\mathrm{b}}^{\mathrm{n}} + U_{\mathrm{c}}^{\mathrm{n}} - M_{\mathrm{a}}^{\mathrm{n}} - M_{\mathrm{b}}^{\mathrm{n}} - M_{\mathrm{c}}^{\mathrm{n}}$$

Here the effective utilitarian score EU_n consolidates the positive actions score subtracted from the negative actions score. An effective utilitarian score signifies the utilitarian health in the network, a higher one signifies positive utilitarianism and a lower score signifies the presence of malicious participants hampering the network. In addition, the competition through disincentivization of benign and malicious bad actions motivates each node for proper behavior.

5.3.1.3 Panopticon Complement

Panopticon was introduced by French Philosopher Michel Foucault in his book "Discipline and Punish: The Birth of the Prison" where he introduced the concept of surveillance. He states that "Visibility is a trap," which underlines the concept of invisible surveillance. It means that a single security guard can surveil prisoners around him where he is invisible to others, but others are visible among themselves. It is more of a conscious visibility of the subjects to be monitored without knowing who, where, and how they are being monitored.

This was further widened by the work of Jeremy Bentham, who architected a prison as represented in Figure 5.3 based on the former's work. It consists of a central tower that is dark and invisible, where the security guard is positioned, but others cannot predict his sight view. On the other hand, surrounding are the prisoner cellars who cannot discuss among themselves and are not aware of anything happening inside the central tower. This organically develops a conscious behavior of the prisoner to behave properly as he can at all times be watched by the invisible guard. This is a self-induced correction mechanism for the fear of being constantly watched or surveilled. A practical example of this surveillance is the Sovereign States monitoring their citizens. This is the case of the National Security Agency of the United States Government surveilling its own citizens reported in 2013.

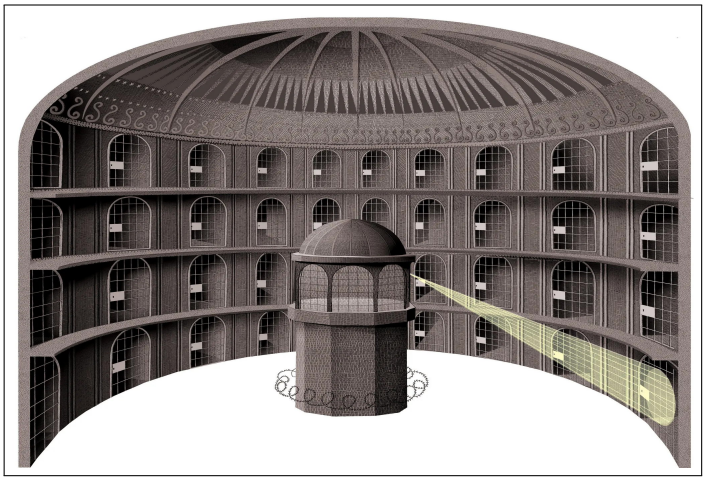

Figure 5.3: Panopticon Representation by Adam Simpson[238]

Consensus Intuition

We consider the previous intuition of Effective Utilitarian score EU_n of each node being calculated at each block height level consensus. The score is accumulated at the end for measuring each one's contribution to network-level utilitarianism or 'proper finalization of blocks.' This score calculation is decentralized as and when upon block propagation, each node can recalculate the utilitarian score attributed to each node due to a particular block creation. So each node maintains knowledge of other nodes in terms of the Utilitarian Score and can surveil each other, which induces the necessity of conscious self-correction among nodes if they behave maliciously.

Let us assume that the Total or Net Effective Utilitarian Score NEU_i of a Node N_i at the latest block height H is given as

$$NEU_{\mathrm{i}} = \sum_{b=0}^{H} EU_{\mathrm{b}}$$

where b signifies the block height starting from genesis at 0 to the latest height H and EU represent the Effective Utilitarian Score measured for a Node at each block height b. This score about all the node participants is maintained by each node once when it receives a valid or invalid block. Let us consider a Node N_i maintains a set of Net Effective Utilitarian Score about all the K nodes in the network as $\{NEU_0, NEU_1, NEU_2 \ldots\ldots NEU_{K-1}\}$. All

nodes maintain this set of scores, and when they need to reorganize their network, they can unanimously consider their individual score. This reorganization or self-correction is the invisible guard, and the NEU_n Effective Utilitarian Score of the Nodes are the visible ones that need to be maintained positively for most cases to render the blockchain consensus protocol correct.

5.3.1.4 Sisyphus Quotient

French Philosopher Albert Camus, in his book "The Myth of Sisyphus," quotes "A man wants to earn money to be happy, and his whole effort and the best of a life are devoted to the earning of that money. Happiness is forgotten; the means are taken for the end." This explains the absurdity of human life, where the action is often repeated, resulting in the relentless pursuit of material or immaterial happiness. This action is absurd as it has no meaning, as we are all subjected to the same kind of monotony.

In Greek Mythology, Sisyphus was a king and tyrant subject to a rigorous punishment by the God Zeus. The punishment represented in Figure 5.4 is for him to push the boulder up a mountain. But the gods decide to punish him severely by making the boulder roll back to its original position when Sisyphus has reached the summit. This makes him repeat the task to complete it, falling into a forever loop without realizing it is absurd. This can be applied to human life, where we try to achieve something greater in our lives even after repeated attempts to ignore or learn from the previous failures or achievements of the past. This allegorical understanding will be applied to our consensus protocol for it to consider the more recent actions, learning from the recent past and ignoring or forgetting gradually our very old actions. It also avoids the trap of affluence influencing democracy and skewing the governance in their favor [239] as noted in public policy democracy research. In our consensus approach, we entertain this Sisyphus quotient to limit the influence of the historically honest nodes with higher utilitarian scores in the recent block consensus approach and bring in a level playing field with lesser score utilitarians.

Figure 5.4: Sisyphus Myth illustrated by Tiziano Vecellio [240]

Consensus Intuition
We consider for the blockchain network of consortium C where each node has the same set

of Net Effective Utilitarian Scores of the K nodes at the latest block height H represented as {NEU_0, NEU_1, NEU_2.....NEU_{K-1}}. Here we would like to apply the concept of Sisyphus to our Net Effective Utilitarian Score, where we apply the principle of weightage for each Effective Score EU for a Node N_i attributed to a certain block height H. This is to give more weightage to the recent block height EU scores, lesser weight to less recent block height EU scores, and least weightage to the relatively ancient block height EU scores. This is in the sense of introducing absurdity to past actions and making them relevant to our current position. This makes the consensus protocol consider a Node N_i with more of its recent activity in the network rather than the history. This is considered as there is a fair chance of a node turning from benign to malicious and vice versa at any point in time.

We consider the total block interval from Genesis or the first block at height 0 to the latest block height H. We divide this interval into interleaving block heights to apply the weightage principle. Each interval of block heights is to be I_K where $K \geq 0$, and with an increase of K value, it gains more relevant weightage. It is to follow the Sisyphus principle of assigning more weightage to the recent blocks. Each Block B is to be at a certain height J (B_J) up to the latest block height H (B_H) for $J \geq 0$.

$$I_0 = \text{Consider Block } B_J \text{ if } J \leq H/3$$

$$I_{K-1} = \text{Consider Block } B_J \text{ if } H/3 \leq J \leq 2H/3$$

$$I_K = \text{Consider Block } B_J \text{ if } 2H/3 \leq J \leq H$$

In this case, for each of the above intervals, we apply the degree of weightage principle as follows with the base multiplication factor called Sisyphus Forgetting Quotient S where $S \geq 0$. S increases gradually for more recent block Effective Utilitarian calculations making the old blocks less relevant. Here EU_b is the Effective Utilitarian of block B, whose height starts from genesis at height 0 to latest height H, * is the multiplication operator.

$$\text{If Block } B_J \text{ in } I_0 \implies EU_J * S/3$$

$$\text{If Block } B_J \text{ in } I_{K-1} \implies EU_J * S/2$$

$$\text{If Block } B_J \text{ in } I_K \implies EU_J * S$$

This necessitates reformulating the equation of Net Effective Utilitarian Score NEU_i including the Sisyphus Forgetting Coefficient. For a Node N_i at the latest block height H, this equation becomes:

$$NEU_i = \sum_{b=0}^{H} EU_b * (S * b/H))$$

This multiplicative factor Sisyphus Quotient of S * b/H will place more relevance on the recent blocks and force the older blocks or actions to have less relevance. At the same time, we harvest the knowledge of each node in the consortium network.

5.3.1.5 Swarm Instinct

Starlings are medium-sized birds appreciated by the science fraternity for their amazing flocking skills. In Figure 5.5, we see a thousand starlings undergoing Murmuration, a "scale-free correlation" or, in other words, self-organizing themselves [241]. This has been applied in the research of hidden climate patterns, awarded the Nobel Prize for Physics in 2021. Each bird interacts with a chosen set of its bird peers to adapt its own perception for perfect consensus among the whole flock [242]. These biologically inspired behaviors have been studied and applied to various computing problems of Artificial Intelligence or Swarm Robotics. This is not only limited to birds but can be observed in the case of fishes, ants, or even spiders, where they adapt in the form of closed-loop feedback for faster and more efficient synchronization of groups. Even viruses communicate and cooperate for the utilitarian or altruistic benefit of infecting or influencing the host better, which was reported in the Nature journal [243]. This is more efficient than voting based approach in humans as it is a static approach on a given set of options [244, 245]. This form of rapid feedback dynamic mechanism observed in this starling is better and applied in many problems [246, 247]. We take this behavior of convergence by learning from the peers in our consensus algorithm protocol to evolve our blockchain network. In a sense, this evolution maintains robustness against any faults or attacks and ensures the increase of Utilitarianism or higher finalization throughput of blocks.

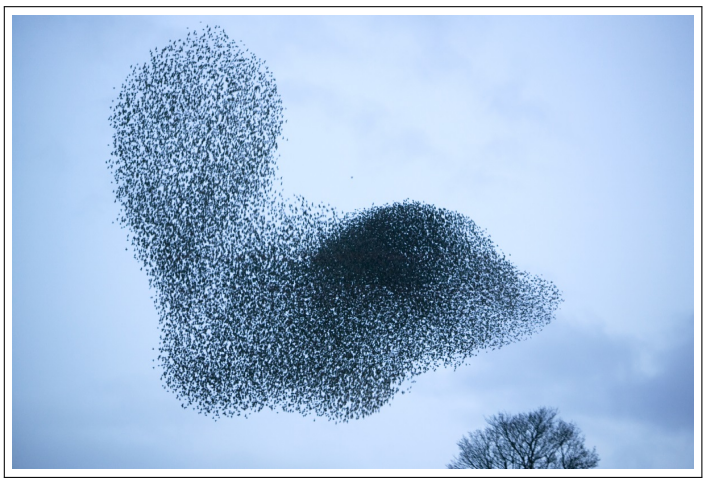

Figure 5.5: Starlings Swarm Behaviour photographed by Ashley Cooper [248]

Consensus Intuition

Our consensus protocol applies the Swarm principle to our consortium network C composed of K nodes or participants. N_i signifies the individual identity of the node in the consortium network where $0 < i \leq K$. At the start of the network or genesis phase, we organize these individual participants into committees or Quorums. Assuming M Quorums are in the network, each is represented as Q_J where $J \leq M$. So it implies the organization of the entire consortium ledger network would be a set of Quorums, each containing a set of chosen nodes as represented in the following equations.

$$C=\{Q_{M\text{-}n}.....Q_{M\text{-}2}, Q_{M\text{-}1}, Q_M | \text{ n is any positive value}\}$$

The selection of the nodes for any Quorum during the initial genesis phase would be based on a randomized selection as we don't have any initial knowledge of the node's behavior as represented below Q_J.

$$Q_J=\{N_{i-3}, N_{i-2}, N_{i-1}, N_i\}$$

where i is selected based on a controlled randomness algorithm, J is any quorum identifier.

Each Quorum is occupied with the block proposal through intra-quorum communication followed by an inter-quorum consensus mechanism to finalize a block. In future block rounds, as the nodes' knowledge is obtained, successive organization of the node membership to a quorum will be based on its Net Effective Utilitarian Score.

.

First, this involves the classification of the nodes into a set of different classes represented by { L_1, L_2.... } where each class represents a quality of the Node's Utilitarianism achieved in the network. For a concrete and simplified adoption, we assume the following classes: Ideal Utilitarian class, Utilitarian class, Fair Utilitarian class, and Weak Utilitarian class. This is a necessary classification performed at a given frequency of block height or epochs for understanding and organizing the network for efficient finalization. Based on the Net Effective Utilitarian Score NEU_i, where i $\leq$K, K Nodes in the consortium network can be sorted in descending order. As we need 4 different classes of Ideal Utilitarian class, Utilitarian class, Fair Utilitarian class, and Weak Utilitarian class to be formed, we assign each sorted node from the highest to the lowest based on the sorted rank. If we assume the node ranks from highest to lowest, represented as

$$C=\{R_{0,i}.....R_{K-2,i}, R_{K-1,i}, R_{K,i}\}$$

where R is the rank of any node (rank, node identifier) identified by node i preceded by its rank in descending order of their Net Effective Utilitarian Score.

$$\text{Rank\_Split\_Index (RSI)} = K/5$$

$$\text{Ideal Utilitarian Class (IU)} = \text{Consider Node } N_i \text{ if } 0 \leq \textbf{Rank} \leq RSI$$

$$\text{Utilitarian Class(U)} = \text{Consider Node } N_i \text{ if } RSI \leq \textbf{Rank} \leq RSI * 2$$

$$\text{Fair Utilitarian Class (FU)} = \text{Consider Node } N_i \text{ if } RSI * 2 \leq \textbf{Rank} \leq RSI * 3$$

$$\text{Weak Utilitarian Class (WU)} = \text{Consider Node } N_i \text{ if } RSI * 3 \leq \textbf{Rank} \leq RSI * 4$$

$$\text{Very Weak Utilitarian Nodes* (VWU)} = \text{Consider Node } N_i \text{ if } RSI * 4 \leq \textbf{Rank} \leq RSI * 5$$

After this classification into different utilitarian quality classes of Nodes, they are used to form the M Quorums. Each Quorum Q_M for any positive value M has to be filled with Nodes belonging to the same class. This ensures that a major quality of IU, U, FU, or WU represents each Quorum. This is basically to judge the perception of the node based on their classification. Then nodes depending on their honesty and active participation in the network, can move from a weak utilitarian to an ideal or any other class. If a node is in

the WU class for successive epochs, it will be suspended from the consortium network. This reorganizes the network based on healthy participants for maximum finalization and robustness.

The Very Weak Utilitarian nodes classification identifies faulty nodes (benign or malicious) without improving utilitarian scores for each successive epoch. These very weak utilitarian nodes are intermittently suspended from the network for certain epochs as they degrade the network. If these very weak utilitarian nodes improve their behavior, they are added to the weak utilitarian class. So the organization of the network also evolves dynamically based on the node's behavior calculated for improving or sustaining the Utilitarianism in the network.

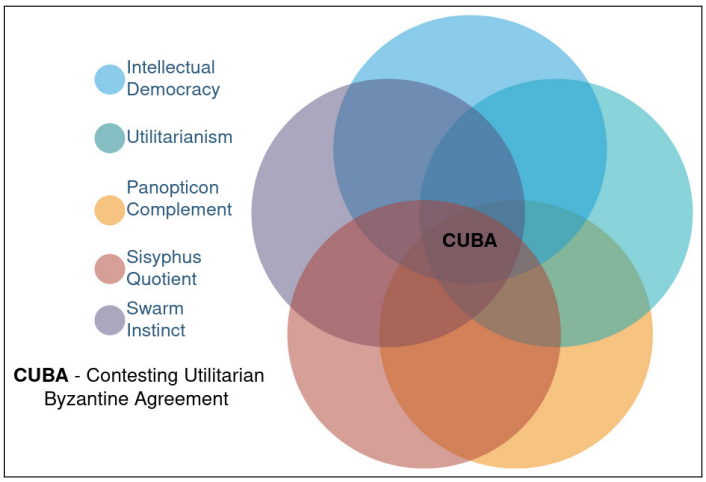

Figure 5.6: CUBA Philosophical Skeleton

As represented in the Venn Diagram 5.6, our CUBA consensus protocol lies at the intersection of Intellectual Democracy, Utilitarianism, Panopticon Complement, Sisyphus Quotient, and Swarm Instinct values. These philosophical attributes act as guidelines in the protocol for ensuring a better, scalable, and fault-tolerant Byzantine agreement between the consortium nodes.

5.3.2 CUBA Consensus Algorithm

In this section, we describe our concrete implementation of the Contesting Utilitarian Byzantine Agreement Protocol by inheriting the earlier philosophical considerations we have explained. First, we explain the system model we assume for our protocol concerning the network and nature of participants. Then we explain the protocol's overall working, abstracting the finer details for a basic understanding. Finally, we explain the detailed fabrication of the protocol to solve the problem from our perspective.

5.3.2.1 System Model

We reprise the System Model definition from the section on Problem Statement (Section 5.1) explained before. It defines the Network Construction, Information or Message Broadcast, Participant Behaviour, the Notion of consensus Finalisation, and the expected quality

metric in the protocol. In addition, we consider that each node N_i where i $\leq$K for K participants in the consortium network has a private and public key. The system has a public key infrastructure, and we consider that no adversary can exploit the classic cryptographic methods of hashing, encryption, and signatures to one's advantage. We define certain basic terms or principles that would be frequently utilized in this section as follows:

1. **Node:** A participant or Node N_i in the consortium network can be identified as one with its own proper identification, which cannot be duplicated or cloned.
2. **Block Height:** A Block Height or number n is the index at which the block B_n is finalized or to be finalized where n is any index in the blockchain
3. **Round:** A Round is the process of a consensus for a certain upcoming block height position in which all the nodes undergo agreement if they are synchronized or within a timeout δ.
4. **Epoch:** An epoch in the blockchain refers to a period or frequency of blocks, during which a fresh set of quorums is constructed and rearranged. This process involves the selection of a controlled, randomly chosen group of participants based on their utilitarian score.
5. **Network Organisation:** Consortium Network C of K participants is organized as a set of R Quorums $\{Q_1, Q_2, ...Q_R,\}$.
6. **Quorum:** A Quorum is a set of participants who belong to a closed unit of participants chosen based on their utilitarian classification and score. Each Quorum Q_p where p $\leq$R consists of at most T Nodes.
7. **Block** : A Block B_n in the blockchain at any index n comprises several partial block units corresponding to the number of Quorums in the network. So it is identified as $B_n=\{PB_{n,1},PB_{n,2}....PB_{n,R}\}$ for R Quorums.
8. **Partial Block:** A partial block is identified as $PB_{n,o}$, the smallest unit for storing unique transactions. Here n is the major Block Number, and o is the minor Partial Block Number or index. Each Quorum Q_R proposes a particular partial block at the major Block Number and minor Block Number. A partial block is composed of a unique set of transactions as $PB_{n,o} = T_1, T_2.....T_S$ where T is the transaction and S is the maximum partial block size. So a Blockchain A is given by A $\circ$ {B...} where each Block B $\circ$ {PB...}, ... represents a set in any given order by agreement and $\circ$ represents *the comprises of* relation.
9. **Ephemeral Chain:** A blockchain that contains the set of blocks, which in turn contains a set of partial blocks that is agreed only at the intra-quorum level but needs to be further finalized at the inter-quorum level is termed Ephemeral.
10. **Finalized Chain:** A blockchain that contains the set of blocks, which in turn contains the set of partial blocks with a byzantine agreement both at the Intra-Quorum and the Inter-Quorum level, is termed as finalized.
11. **Heart Beat:** Each Node communicates at a given frequency T its availability message to all other nodes in the network. This is ensured to distinguish a node between

benign network failures and malicious behavior, which is counted during network reorganization. Although we consider an eventual synchrony model [72] within δ Global Stabilisation Time, we consider the extreme factors of high periods of asynchronicity. So, we place this mechanism to solve the impossibility problem in reaching the consensus in this case as in [26].

12. **Vote:** This is a message protocol to participate in the consensus mechanism, which a node attests for acceptance by a valid message signature. The required threshold of vote messages is necessary for a consensus on a particular block to be considered finished.

5.3.2.2 Consensus Overview

Notation	Description
ζ	Consortium Blockchain Network of K Nodes
C_g	Client who submit transactions where g $\in \mathbb{N}$
N_i	Node i identification where i $\in$ {1...K} for K Nodes
Q_r	Quorum r identification where r $\in$ {1...S} for S Quorum
ρ	Total number of Quorums in the Consortium Network ζ
σ	Size of each Quorum up to which it can accommodate Node or participant.
K	Total Number of Members in the Network.
T_χ	Unique Transaction issued by a client C_g where $\chi \in \mathbb{N}$
π_r	Transaction Pool to stock a relevant transaction by a Quorum Participant in Q_r
$P_{k,l}$	Partial Block unit where k is the Block Height, and l is the index position within a block or Quorum Index in the Blockchain (Ephemeral or Finalised)
B_k	Block unit where k is the Block Height in the Blockchain (Ephemeral or Finalised)
κ	Partial Block size upto which transaction can be packaged
α_ψ	Ephemeral Blockchain having a size or latest block height of ψ
β_ω	Finalised Blockchain having a size or latest block height of ω
ϵ	Epoch Identification
ϵ_O	Genesis Epoch Identification or Initial Epoch
Υ_I	Ideal Utilitarian
Υ	Utilitarian
Υ_F	Fair Utilitarian
Υ_W	Weak Utilitarian
T	Heart Beat Message Time Frequency
M_ι	Intra-Quorum Message
M_I	Inter-Quorum Message

Table 5.1: CUBA Protocol Notation Description

CUBA protocol overall working is represented in Figures 5.7 and 5.8. This is more a bird's view of its working with certain details to be explained in later sections. Its working is

represented as the sequence of steps as follows:

1. The protocol starts with the emission of a set of transactions by a client, which is validated and then distributed uniformly to quorums based on the modulo operation of the transaction hash Unicode.
2. For a new block round, each quorum receives a unique set of transactions distributed from the previous step in its queue and then forms a partial block based on partial block size. This is then subjected to a consensus agreement within its quorum members. The network will pipeline the partial blocks and blocks, including the ephemeral and the final chains, for faster processing.
3. The partial block confirmed from each quorum is placed in an Ephemeral Blockchain at the index of block number round and its quorum index. Its index is a tuple <Block Number, Quorum Index>. This block is placed in the Ephemeral chain and transmitted between all the participants.
4. The Ephemeral Block is then subjected to the intra-quorum consensus proposed by a Block proposer among the set of participants and transmitted.
5. The transmitted block containing all the partial blocks is verified, and a utilitarian score is calculated based on the signature and vote accumulated in the blocks. If a new epoch is reached at the end of the current round, the nodes are sorted in descending order based on their utilitarian score.
6. The sorted nodes are then classified based on their score. The first range of nodes will be accommodated in Ideal Utilitarian and the rest of the ranges in Utilitarian, Fair Utilitarian, and Weak Utilitarian in descending order.
7. Then, a new set of quorums is proposed based on the classification for the new epoch, and the nodes with the very lowest scores are temporarily suspended from the network. This is to optimize the network with the most utilitarian participants for a faster and more efficient blockchain network.

5.3.2.3 Detailed Protocol

In this section, we dive deeper across the breadth of the CUBA consensus protocol, starting from the transaction processing until the network evolution for self-optimization based on the utilitarian behavior of each node in the network. The working of the protocol should be read with Table 5.1 for notational descriptions.

5.3.2.3.1 Transaction Processing In this section, we look at the starting point of the protocol, which is transaction processing by the blockchain network. As represented in Figure 5.9, each of the client C_g where $g \in \mathbb{N}$ submits a transaction T_χ where $\chi \in \mathbb{N}$ to the network which is forwarded to the nodes. Each node rebroadcasts the transaction to the whole network in the desired network topology. Network Topology can be Fully Connected, Ring Lattice, or Watts Strogatz, as explained in Simulation Chapter 4. To understand how the transaction is further handled, we need to understand the organization in the protocol.

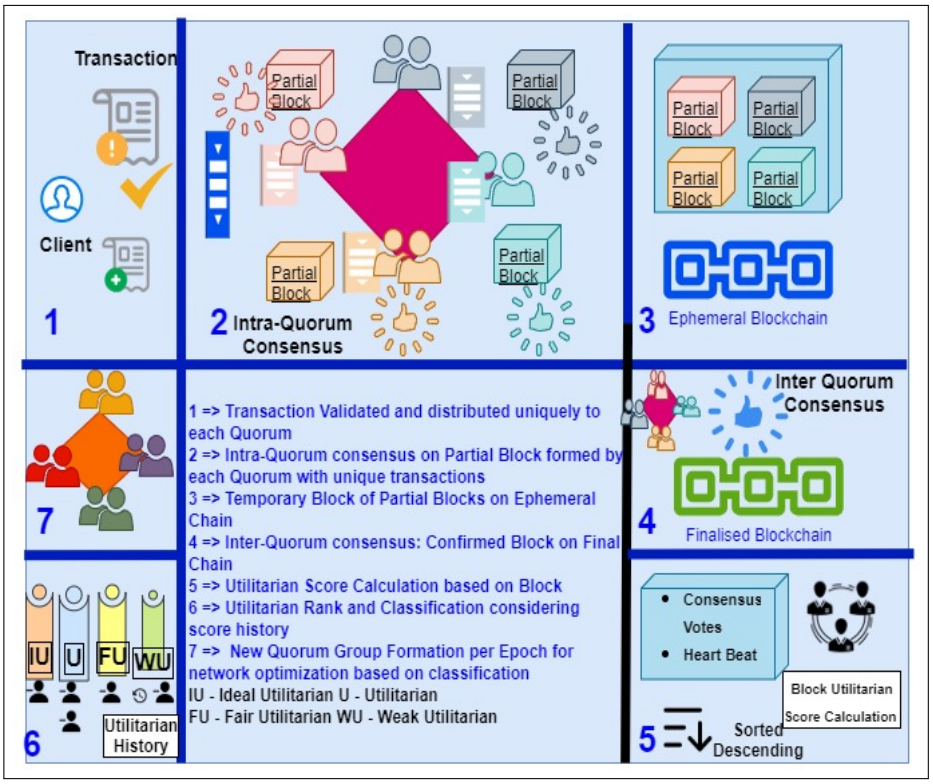

Figure 5.7: CUBA Consensus overview explains the lifecycle of the consensus process from transaction issue, Intra-Quorum, Inter-Quorum Phase, calculating each node's utilitarian score based on previously finalized blocks proceeding with quorum reorganization.

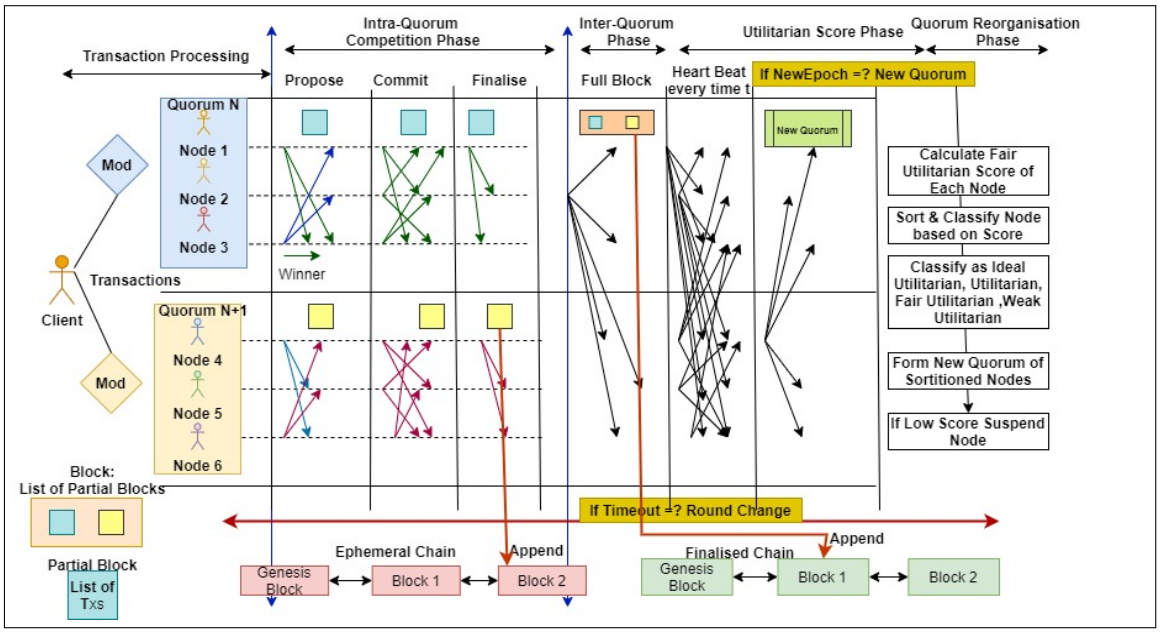

Figure 5.8: CUBA Consensus Message Exchange Overview

Protocol Network Organisation In the Consortium Network C of K participants, who are organized as a set of S Quorums {Q_1, Q_2,....Q_s,}. Each Quorum comprises a set of partic-

ipants who belong to a closed unit of nodes chosen based on their utilitarian classification and score. Each Quorum Q_r where r $\leq$s consists of at most σ Nodes. The Quorums can be explained by three characteristics in the CUBA protocol as follows:

1. **Quorum Membership:** Each quorum can accommodate members N_i belonging to more or less similar Net Effective Utilitarian Score at the end of each epoch. Quorum Membership changes for every epoch to optimize the network in removing malicious or faulty nodes. Each quorum can accommodate until a quorum Size σ up to the total number of quorums in the network ρ.

   In order to award a node N_i where i$\leq$K, membership for a new Epoch ϵ, then it needs to be classified into Ideal Utilitarian Υ_I, Utilitarian Υ, Fair Utilitarian Υ_F and Weak Utilitarian Υ_W. For the Genesis Epoch ϵ_O, the classification would be a randomized sortition of nodes N_i as we don't have any prior evaluation or utilitarian state to consider. Subsequent epochs ϵ would be evaluated based on the Net Effective Utilitarian score for the preceding epoch, which will be discussed in detail in the next section.
2. **Quorum Communication:** Quorum can communicate or pass messages in two ways through Inter-Quorum Message M_I and Intra-Quorum Message M_ι. This section will explain its detailed phases for Intra-Quorum and Inter-Quorum, respectively. Its overall function is as follows:

   (a) **Intra-Quorum Message M_ι**: This is the preliminary level of consensus message passing where the primary data structure unit is Partial Block $P_{k,l}$ k being the Block Height, and l is the index position of a block or quorum as agreed upon by the quorum members of l. $P_{k,l}$ comprises verified transactions belonging to quorum l for the round or block height k.

   (b) **Inter-Quorum Message M_I**: This is the agreement message on the whole block B_k which is composed of several finalised partial blocks. The participants across the quorum attest to this message, and they accept the block if it comprises all the partial blocks from all the quorums and is finalized by a valid proposer.

Idea of having an organization of Nodes N_i under each quorum and the significance of the network organization in the protocol is based on the following reasons:

1. **Reduced Communication Complexity**: In this protocol, the consensus is reduced in two phases where Intra-Quorum has a message complexity of O(σ^2), and Inter-Quorum has a complexity of O(1). Here σ is the quorum Size of participants it can accommodate, significantly less than the total K participants.

2. **Parallelization and Pipelining:** Each quorum verifies its set of transactions and creates a particular partial block $P_{k,l}$ for its index without waiting for the other quorums. This makes the processing faster as two chains of ephemeral and finalized ensures the pipelining of multiple blocks easier without any bottleneck. It removes the issue of relative dependence on the previous blocks for the consensus operation on the current block round, which will be discussed in detail.

3. **Utilitarian Classification and Optimisation**: Quorum enables clustering of clas-

sified participants as Υ_I, Υ, Υ_F and Υ_W. This clustering enables us to identify the participant's Utilitarian behavior and witness their evolution in the network. It also gives the scope for fairness in motivating N_i to move from a lower to a higher classification by attributing extra improvement quotients in the utilitarian score. Also, on the contrary, it enables the suspension of nodes who always are in the lower classification successively with no improvement affecting the network optimization.

Having understood the network organization as represented in Figure 5.9, we then continue with the transaction processing from the client's C_g origin to each of quorum Q_r in ζ. A node that forms part of a particular Q_r can gossip or broadcast any transaction T_χ. But for Q_r to consider T_χ as part of $P_{k,r}$ for round k and index r, it has to follow the two conditions as follows:

1. Each T_χ must be unique and original with H_{T_χ} = Hash(T_χ) $\notin \beta_\omega$,α_ψ.
2. For considering in its partial block $P_{k,r}$ its quorum index should match with the value on modulus operation based on the codepoint of H_{T_χ} over the total quorum members.

The algorithm pseudocode for transaction processing is detailed in Appendix chapter 8, Section Algorithm3 for brevity.

Transaction Flooding Attack This section highlights the limitation or scope of our protocol's transaction processing, as explained earlier. The most simplistic attack in any distributed or centralized system is Distributed Denial of Service [249]. Each transaction T_χ is checked for authenticity and duplicity before being rebroadcasted to other nodes in the consortium network ζ. In addition, a more straightforward mechanism would be to limit the transactions received through the Remote Procedure Call Application Programming Interface per time period. This limits the possibility of the transaction rate emitted, avoiding the DDoS attack vector as a whole. This section is more on the implementation aspect, and we keep it aside in our protocol explanation to focus more on consensus protocol.

Data Structure Definitions This section examines the various data structure definitions necessary to understand our protocol.

The fundamental unit in our protocol is Partial Block $P_{k,l}$ where k is the Block height and l is the index position within a block or quorum index. The data structure of $P_{k,l}$ is represented in Figure 5.10. Each $P_{k,l}$ is proposed by a quorum l after the intra-quorum consensus for a given round k. It consists of a set of transactions self-assigned as part of its quorum membership. It also has the other identifiers of timestamp, previous block state hash, current partial block hash, and proposer's signature. After the Intra-Quorum consensus, it will be placed inside a block container B_k of Ephemeral Blockchain α_ψ for the k,l index and broadcasted to the consortium network. B_k is represented as in Figure 5.11 where each quorum needs the partial blocks to be proposed. As the partial blocks are received for the block structure, then the Temporal Hashing is updated incrementally. This will be explained in detail in the upcoming section. It is more to ascertain the state of an unfinalized container block containing a set of partial blocks before being updated with another partial block. This is done to maintain a history of state progression, as we always need non-repudiation in the network. After all the partial blocks are received for a Block k, it undergoes the

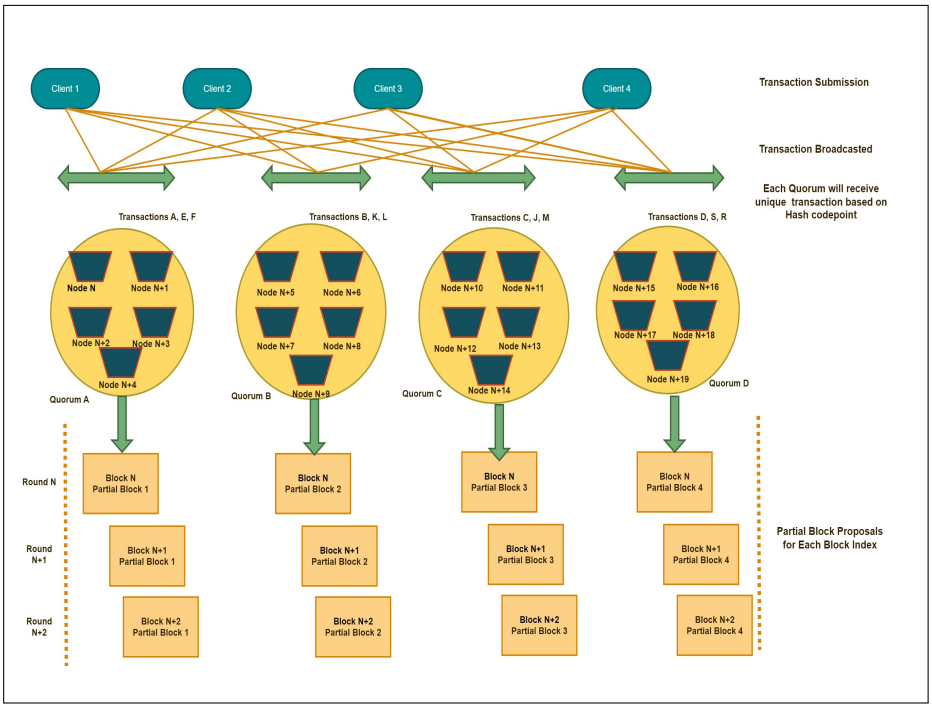

Figure 5.9: Transaction Processing

secondary consensus of Intra-Quorum finalizing the block on the finalized blockchain β_ω.

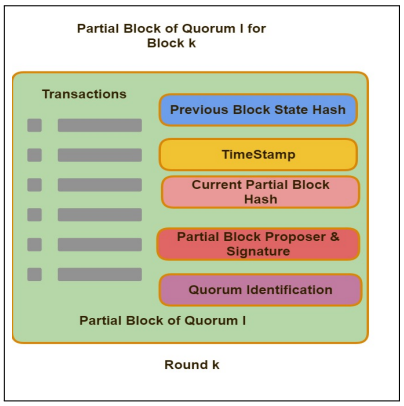

Figure 5.10: Partial Block Data Structure

CUBA Data Structure Significance The idea to have two units of structure as Partial Block and Block is given by the following reasons:

1. **Faster Processing**: By splitting the transaction processing into the units of partial blocks $P_{k,l}$ which are coalesced to form a block B_k among multiple quorums fastens the process. The effort to finalize a block by this modification has a better approach in two ways:

   (a) Reducing the task through ρ lighter partial block units instead of a single block B_k.

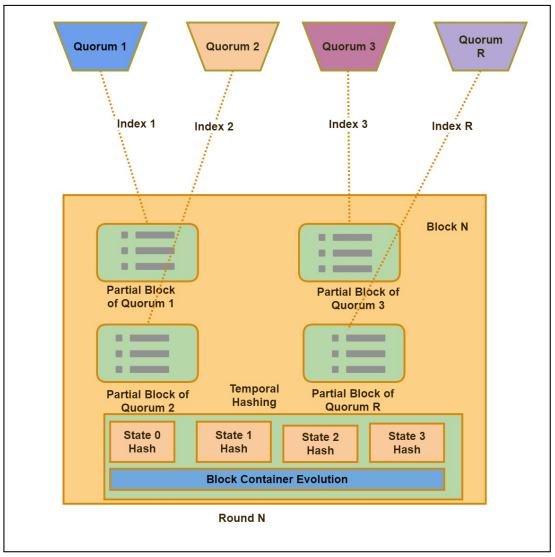

Figure 5.11: Block Data Structure

(b) Distributing the task among ρ Quorums with each Quorum having σ members.

$$\textbf{Effort}_{\textbf{Normal}}=|B| \div K$$

$$\textbf{Effort}_{\textbf{CUBA}}=((|P| \times \rho) \div (\sigma \times \rho))$$

$$\textbf{Effort\_Simplified}_{\textbf{CUBA}}=((|P|) \div (\sigma))$$

$$\textbf{Gain}_{\textbf{CUBA}}=((|B| \times \sigma) \div (K \times \rho))$$

|B|=> Size of data structure unit of Block

|P|=> Size of data structure unit of Partial Block

The $\text{Gain}_{\text{CUBA}}$ is significant as the partial block size is less than a block data structure.

2. **Partitioned Failure Risks**: Since we have multiple ρ quorums in the protocol, the risk for liveness in terms of failure tolerance and network failures gets redistributed among them. The redistribution is due to the sub-network of quorums with a reduced majority threshold compared to the large K network members.

Pipelining In this section, we explain the pipelining aspects of our protocol with the modified data structure presented earlier. The pipeline decomposes a repeated sequential process into subprocesses, each of which can be executed efficiently on a special dedicated autonomous module that operates concurrently with the others. In the domain of distributed or blockchain systems, the block and partial block can be considered analogous to the process and subprocess as in the definition before.

As represented in Figure 5.12, the pipelining in CUBA protocol can be applied in two phases as follows:

1. **Intra-Quorum Phase:** This operates on finalizing a partial block structure within the quorum members. As in Figure 5.12, each quorum creates the partial blocks as and when proposed. For example, each partial block at multiple Rounds of 4 and 5

is being processed. In addition, each is processed by different quorums; in this case, we consider 3 Quorums to be parallelized. The partial block finalized can be in any temporal order, and storage in block structure does not guarantee temporal order to match the index order. Figure 5.13 represents an ephemeral or intermediary chain. This represents an intermediate chain structure for finalizing the partial block before being finalized in the finalized blockchain structure as in Figure 5.14. To guarantee non-repudiation by hash, an integral property of any blockchain protocol, we maintain the *temporal hashing data* in each block. As represented in the following temporal hashing for Round R, we store it as a partial block index I mapping structure against the hashing state. The hashing state is given for a time T, block container hash at T (BC), node identity who updates the state (N), and list of partial blocks added (L).

$$\textbf{Temporal Hashing}_{R} = \textbf{Mapping<Partial Block Index I, Hashing State}_{R,I}\textbf{>}$$

$$\textbf{Hashing State}_{R,I} = \textbf{[T, BC, N, L]}$$

For each Round R, an intermediate ephemeral block has indexes to be filled by partial blocks after their individual intra-quorum consensus. As we note in Figure **??** for round 1, all the partial blocks are filled, and the state 0,1,2,3 hash is updated accordingly as and when it is appended to the block structure. Similarly, for Round 2, we have the state 0,1 hash filled representing the completion of inter-quorum consensus for Quorum 1 and 3, not necessarily in order awaiting the Quorum 2 and 4 partial blocks. So the inter-dependence between the partial blocks across rounds and quorum indexes is relaxed, enabling individual quorums to process in parallel.

2. **Inter-Quorum Phase:** Following completing all the quorum indexes in any block, we undergo a consensus protocol across quorum members to finalize a total block. Here the finalized block is proposed, signed by a proposer, and transmitted to all the participants. This phase is partly parallelized as the block container formation is pipelined with no relative dependence. But in the case of a finalized block for a finalized chain, we have a strong chained previous index to the current index hashing as exhibited in any standard blockchain structure. However, since it is a lighter message exchange of O(1), the relative dependence bottleneck on the blocks is not penalizing.

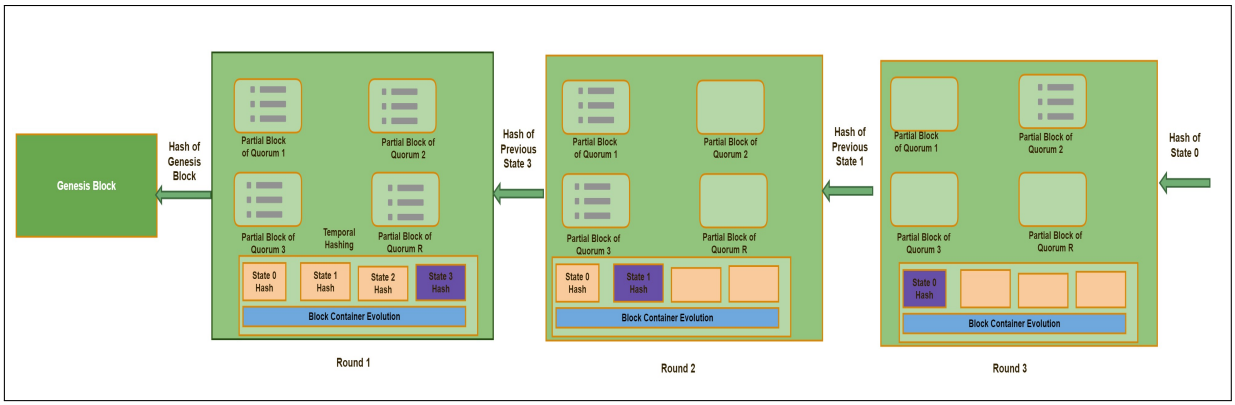

Figure 5.13: Ephemeral Blockchain Structure

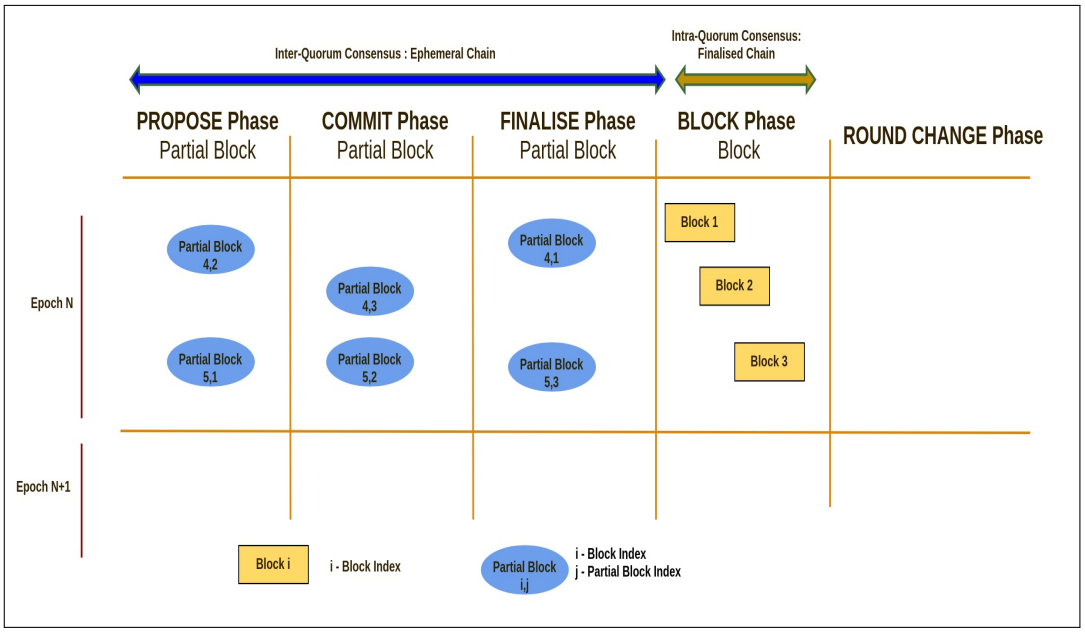

Figure 5.12: Pipelined Consensus Protocol

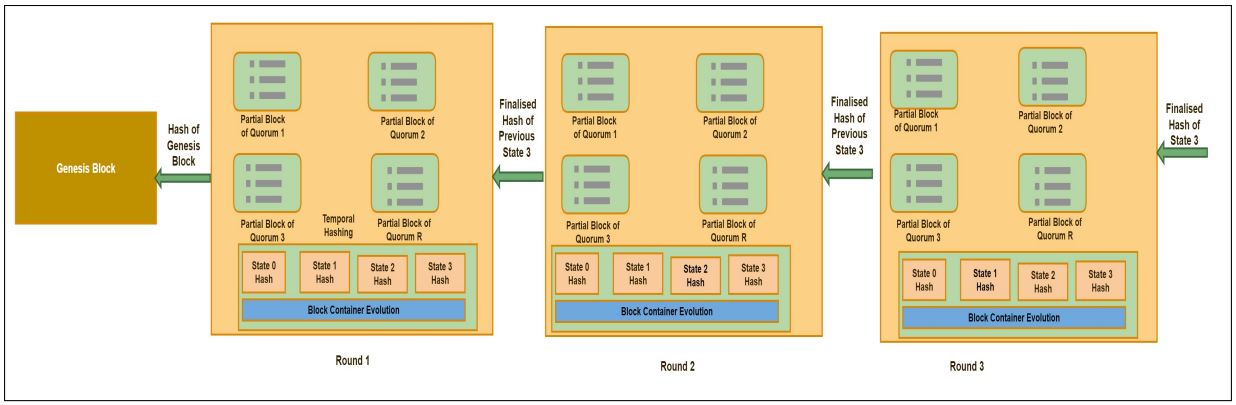

Figure 5.14: Finalised Blockchain Structure

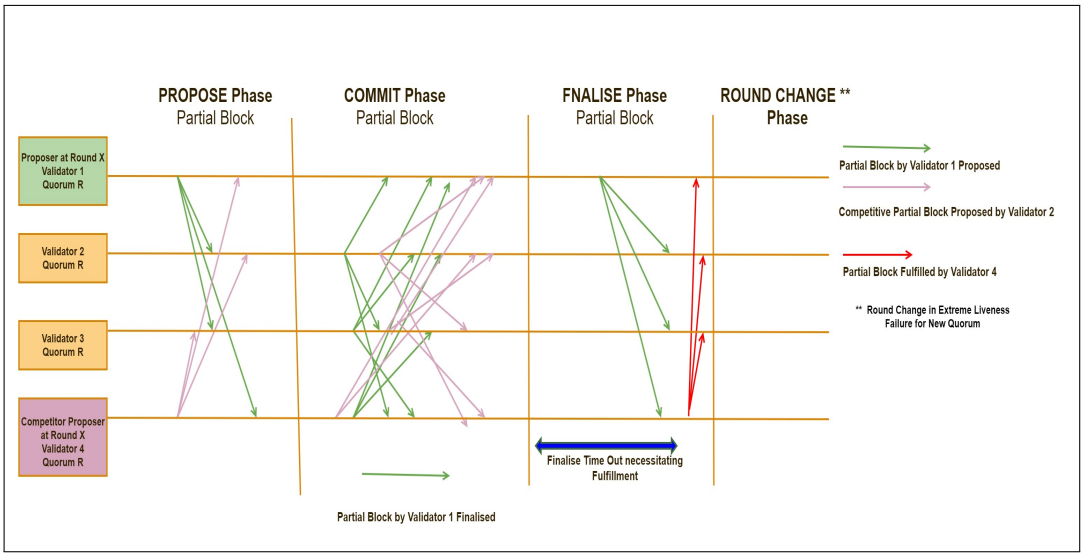

Figure 5.15: Intra-Quorum Consensus Protocol

Intra-Quorum Consensus After the initial discussion around transaction processing, network organization, and pipelining aspects, we explain the next important section of our consensus in detail for finalizing a partial block. The Intra-Quorum consensus operates in sub-phases of PREPARE, COMMIT, and FINALISE as illustrated in Figure 5.15. The algorithm for Intra-Quorum protocol is listed in Appendix chapter 8, the section of Algorithm 5, 6 and 7. Its working is as follows:

1. Each Quorum Q_r upon receiving the transactions in its pool up to a partial block size κ proposes a partial block $P_{k,l}$. Suppose the node is a partial block proposer based on the codepoint of block height K's hash. To create a gamified protocol, a competitor partial block proposer is also chosen among the quorum members to propose a similar partial block $P'_{k,l}$.
2. During the PROPOSE sub-phase, the block proposed by the proposer or competitor is propagated in the quorum network. $P_{k,l}$, which achieves the threshold of 2/3 votes in the COMMIT phase, is confirmed.
3. During the FINALISE sub-phase, the finalized proposer is chosen to add the partial block to a block container and then broadcast it to the quorum network. If the finalized block is not achieved within a timeout Δ, then a Fulfiller is selected to finalize the block.
4. In all the cases of Intra-Quorum and Inter-Quorum phases, when there is a blockchain state deadlock either in the ephemeral or finalized chain, a ROUNDCHANGE phase is initiated. This will be explained in the later section of Round Change 5.3.2.3.1, at which point of the consensus, a Quorum Reorganisation takes place, and the nodes that have failed or are malicious are removed for optimization in the subsequent round.

Intra-Quorum Competition Significance

1. The idea of having two proposers: partial block proposer and competitor, during the

PROPOSE and COMMIT phase, is to increase the liveness of the network. Although the blocks $P_{k,l}$ and $P'_{k,l}$ are conflicting introducing forks, they are limited to just two phases of COMMIT and PROPOSE.

2. Even when there is a failure due to network tolerance, the partial block that propagates faster in the network and achieves the threshold votes among the quorum members is considered to be finalized. The partial block proposer or competitor which wins the votes has an increase in utilitarian score and a decrease for the loser as explained in Appendix chapter 8, section of Inter-Quorum Algorithms 8 and 9.

3. The idea of having to PREPARE, COMMIT, and followed by FINALISE is based on the concept introduced by PBFT [155, 250]. The PREPARE phase ensures the ordering of the transactions for a single view, but the COMMIT phase provides a unique Partial Block $P_{k,l}$ is selected at the index k,l. This is necessary when there is a view or round change detected, and the reorganization of the quorum is undertaken. The COMMIT phase maintains the same state when there is a round change.

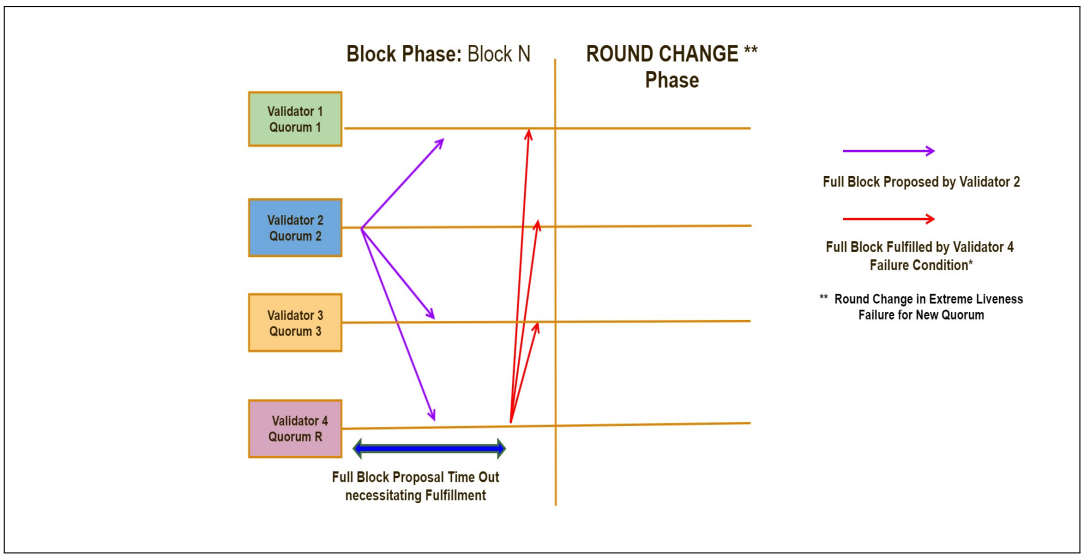

Figure 5.16: Inter-Quorum Consensus Protocol

Inter-Quorum Consensus This section focuses on the following important phases across the different quorum members. A block B_K is finalized among the effective K members across the quorum. The protocols work as follows:

1. In the previous phase of Intra-Quorum consensus in section **??**, we understood the partial block proposal between the participants of each quorum. Each partial block is filled for a whole block where individual partial blocks are placed in the block container $B_{c,k}$. Then the temporal hash is updated in the container as and when all the indexes of a partial block in the block B_K are fulfilled after respective intra-quorum consensus. When a total number of ρ partial blocks equivalent to the number of quorums in the network are received in the ephemeral chain α_ψ k, then a full block proposer is chosen to propose the finalized block B_K.

2. Block B_K is formed considering the last temporal hashing state as indicated by the block proposer. In case the proposer fails to complete the block, then after a timeout of η a fulfiller is chosen. Fulfiller proposes the block B'_K, which is updated. In case of conflict between the blocks B_K and B'_K, then the time is compared to resolve the conflict. The block confirmed is stored in the finalized blockchain β_ω.

3. In case of non-fulfillment of blocks, a RoundChange phase is initiated as a default view change for Quorum Reorganisation, which will be explained a little later.

The algorithms pseudocode for Inter-Quorum consensus are detailed in the Appendix chapter 8, Section of Algorithms 8 and 9.

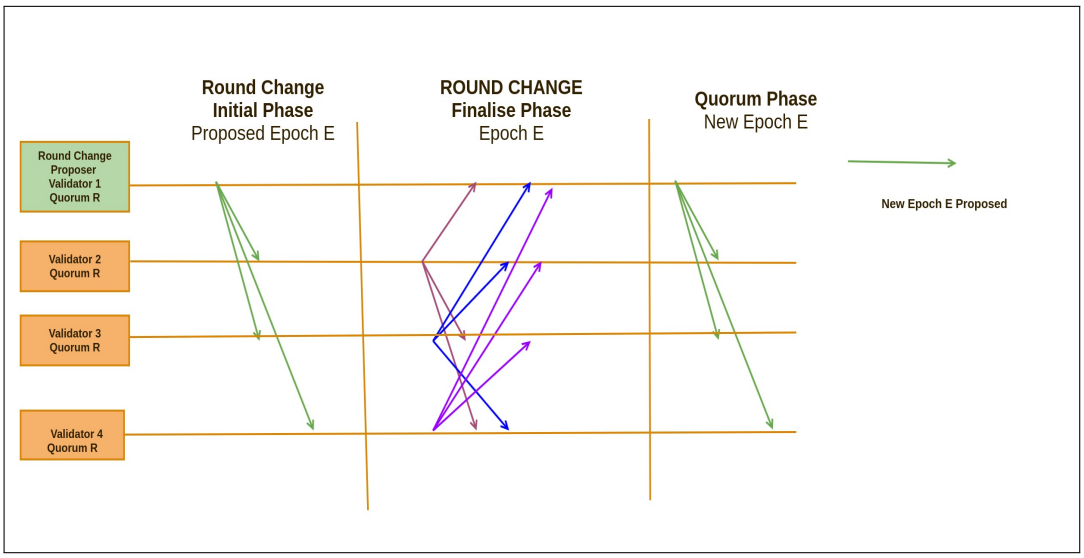

Figure 5.17: Round Change Protocol

Round Change Protocol The Round Change protocol is applicable during the entire transaction lifecycle, either in the inter-quorum or intra-quorum when the blockchain state is stalled or unprogressive. The state considered here is of the Ephemeral chain α_ψ or the finalized chain β_ω which is checked by the function *IsRoundChangeNeeded()* in Algorithm 10 detailed in Appendix chapter 8, section 8.5. The timeout ξ is tolerated until the round-change function is invoked and the consensus phase starts.

1. A round change proposer is chosen if the condition is satisfied when there is no state progression and initiates the round change message as illustrated in Figure 5.17.

2. The rest of the validators can consent to the round change if everyone agrees to progress for a new epoch E. The round change is finalized if a threshold greater than $1/2(K)$+1 is reached.

3. Once the new epoch is finalized, a new quorum is formed considering the latest heartbeat received and the past utilitarian score of the nodes. The nodes are then reorganized to form new quorums for epoch E, which will be explained later in the section 5.3.2.3.2 on Network Evolution.

Heart Beat Exchange Protocol An integral part of the protocol is this heartbeat exchange which can maintain the liveness of the blockchain consortium network. As illustrated in Figure 5.18 with a Node N_i at a defined interval of frequency t. As explained in Appendix chapter 8, section 8.5 in the Algorithm 11, the nodes exchange HEART_BEAT message every t interval, propagating throughout the network. This message is used as a metric during quorum reorganization for node classification and formation. In case of a failure of a node, then it can be detected easily as it can fail to exchange the heartbeat and be utilized as a deciding factor to prove its adverse condition. Eliminating the failure or misbehaving nodes by this mechanism can lead to the optimization necessary in the network.

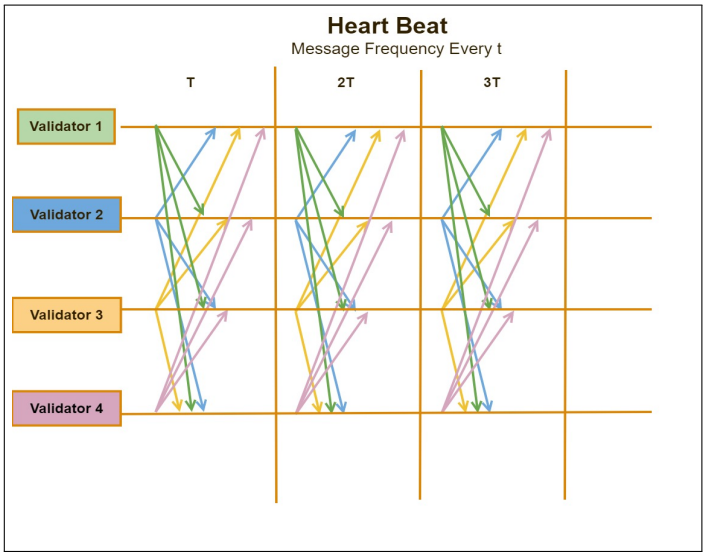

Figure 5.18: Heart Beat Message Protocol

5.3.2.3.2 Network Evolution This section explains the reorganization necessary in the network at the end of each epoch E to form the nodes in each quorum for better optimization. This can be at the end of each epoch for a gradual increase in block height or due to round change, which can invite an extraordinary shift in node participation. This is explained through different components whose functioning will be described as it aims to reach an optimized blockchain network for better throughput, scalability, and resiliency.

Utilitarian Inference Continuing the section from Inter-Quorum where a Block B_K is either formed by the block proposer or fulfilled and propagated throughout the network. Each Node N_i maintains an internal register of the utilitarian score derived for each Block. The post-mortem on the Block can be used to attribute the utilitarian score as explained in Appendix chapter 8, section 8.5 through the algorithms 12 and 13. Its overall working is as follows:

1. Each block is composed of partial blocks wherein each partial block has several attributes for differentiating its utilitarian score among all nodes. First, the Inter-Block time coefficient is calculated concerning the previous block as it measures the real effort maintained in the network. Then this coefficient is utilized as a multiplication factor for each positive or negative score.

2. The positive score in block B_K can be attributed to an action if it improves the utilitarian score as the following:
   (a) Partial block proposal winner if a proposer succeeds in the PROPOSE and COMMIT sub-phase for the concerned quorum.
   (b) Commit win for those who participated by votes in reaching the desired threshold faster than the competitor partial block.
   (c) Heart beat score for those who have sent their ping message to all the nodes to establish their liveness metric.
3. The negative score for B_K can be derived as the actions in the opposite sense as listed below:
   (a) Partial Block Proposal Loser if a proposer loses in the competition race against a peer inside its quorum as a penalization score.
   (b) Commit Loser for those expected to endorse a winning partial block but failed, as noted by their vote signatures.
   (c) Heart Beat Missed score for those who failed benignly or malicious behavior to cascade their liveness message for fixed frequency interval.
   (d) Malicious score if the block or partial block propagated is invalid with forged signatures, hash, or votes.
4. Each score is consolidated at the block level multiplied by the inter-block time coefficient. Each score is based on the node's previous disposition as either Ideal Utilitarian, Utilitarian, Fair Utilitarian, or Weak Utilitarian and a fairness score is added and explained in the next section. Then the net score is calculated as a difference between the positive and negative scores.

Fairness Component This component is explained by the algorithm 13 in the function *getFairnessScore()* listed in Appendix chapter 8, section 8.5 for detailed working. In this component, a variable attribute is added when a node is attributed a utilitarian score in the positive connotation for partial block proposal, heartbeat win, or commit win. This component attributes a quotient in inverse to the node's utilitarian classification in the previous epoch. In the case of the Ideal Utilitarian, a minimum score is added, and Utilitarians are awarded a little higher for a fair and weak utilitarian, respectively. This is based on the philosophy of avoiding the scenario similar to the Pareto 80-20 rule, where we want weak utilitarians to move up the ladder if they exhibit positive behavior and do not remain forever weak. The final scores are then stored for each epoch concerning each node in the EffectiveScore storage.

Swarm Optimisation In this section, we explain how we utilize each node's previously accumulated utilitarian score knowledge for each subsequent epoch in evolving our network. The evolution targets identifying the node's behavioral change or more of the consensus actions it has performed in the past and in recent times. Weak or malicious nodes are suspended temporarily as a penalization and included after the interim suspension pe-

riod. This helps optimize the network performance due to any failures or attacks. Attacks like Distributed Denial of Service where a network as a whole or a particular node can be targeted to compromise the network can be identified based on individual utilitarian scores. This limits either in suspension of malicious or affected nodes that can join in later epochs on restoration.

Sisyphus Quotient As explained earlier in this section, our earlier achievements are absurd in life, and we strive for the future, as noticed in the life of Sisyphus. So we imbibe the same notion in the Utilitarian Score detailed through the Appendix chapter 8, the section in Algorithms 14, 15 and 16. In particular, the two functions of *FormForgettingCoefficient()* and *UpdateForgettingCoefficient()* hold this logic. It works as follows:

1. For a current epoch E, before forming the next epoch E+1, we need an effective understanding of the Utilitarian Score across nodes with varying epochs.
2. The essence of this component is to give more weightage to recent epochs and lesser weightage to past epochs. This will gradually descend if we move from the recent epoch to the genesis epoch.
3. In the function *FormForgettingCoefficient()*, the values of the forgetting coefficient are calculated concerning the current epoch. This calculated coefficient is stored in the storage Forgetting_Coefficient. It is calculated as follows by iterating for each previous epoch since the current epoch:

$$\text{Forgetting\_Coefficient}_{\text{PreviousEpoch}}$$

$$= \text{1-((CurrentEpoch-PreviousEpoch) / CurrentEpoch)}$$

4. Next, in the function *UpdateForgettingCoefficient()*, the calculated forgetting coefficient for each previous epoch is applied by multiplying each with EffectiveScore obtained in the previous section. This reduces or augments the relevance based on the score obtained in earlier or recent epochs.

Panopticon Complement and Classification This section works on the inter-block time coefficient plus fairness normalized score applied with forgetting quotient to classify the nodes. The Appendix chapter 8, section Algorithm 16 has the function *ClassifyandReorganiseQuorum()*, which performs this logic and is detailed over there. It works as follows:

1. The Nodes are first sorted in descending order based on the Utilitarian Score, which is then split into 4 equal intervals of nodes based on their rank.
2. The First interval of nodes is marked and placed into the Ideal Utilitarian List, the Second Interval into Fair Utilitarian, the Third into Utilitarian, and the fourth into Weak Utilitarian. The lowest scorers are marked as Very Weak Utilitarians but not classified. As obvious by their nomenclature, the Weak Utilitarians, especially Very Weak Utilitarians, are the nodes susceptible to failures and have continuously degraded the network's performance.

3. In case a node has been in the Very Weak Utilitarian list for Z previous epochs, then it is suspended for an epoch as a kind of penalization and remediation at the same time.

4. Heart_Message is expected of a node every θ frequency for inclusion in the classification and the proposition of the new quorum. The node is suspended for the epoch in progress if a heartbeat message is not received.

5. The suspension factor, which completes the panopticon complement, works on the transparent Utilitarian Score and Heart Beat Message to enforce the vitality of the blockchain network.

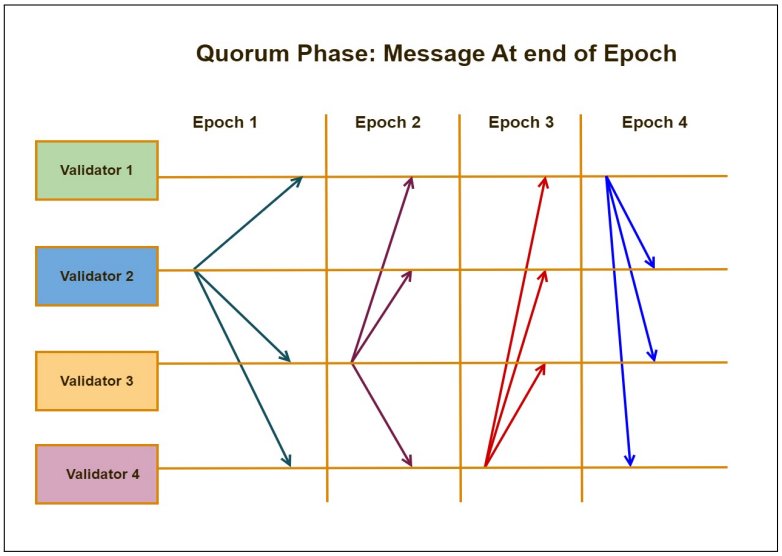

Figure 5.19: Quorum Message Phase Protocol

Quorum Proposition This is the final action to be performed as part of every epoch's quorum reorganization. Its detailed working is listed in Appendix chapter 8, section 8.5 of Algorithm 16 for the last part of the function *ClassifyandReorganiseQuorum()*. This works as follows:

1. For each new epoch E based on a selected quorum proposer on past epochs of performing Intra-Quorum, Inter-Quorum consensus, Utilitarian Calculation, Fairness and Inter-Block Coefficient Application, Sisyphus Quotient, and finally Panopticon Complement, we sort each classified node into its own quorum.

2. Blockchain network requires ρ quorums wherein each quorum needs to be filled with nodes.

3. The Quorums are filled with each up to σ number of nodes in their order of ascending ranks. This will roughly organize the Ideal Utilitarian, Utilitarian, Fair Utilitarian, and Weak Utilitarian in homogenous groups of each quorum.

4. The very weak utilitarians are marked in the network for later suspension if they show sustained failures. A minute heterogeneity can be observed since there cannot

be a perfect sortition of quorums of nodes of the same utilitarian classification due to suspension or failure of nodes.

5. The newly formed QUORUM_MESSAGE<E_{ϵ}> with quorum reorganizations for the new Epoch is proposed by the quorum proposer as in Figure 5.19 to the other peers in the network for evolution improving optimization and resilience.

CUBA Protocol Variants After discussing the CUBA protocol with its standard definitions and their evolution in the network, we notice that most message exchanges are PROPOSE, COMMIT, and FINALISE within the Intra-Quorum level. In this section, we propose an alternative design of the CUBA protocol for faster processing of transactions and blocks by having certain strong assumptions.

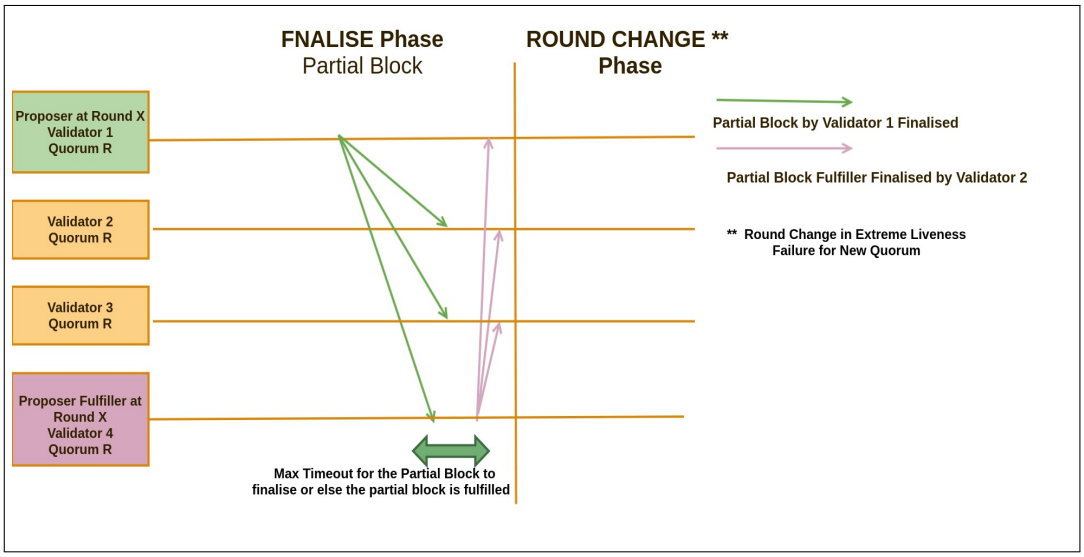

Figure 5.20: Intra-Quorum Implicit Consensus Protocol

Implicit Agreement In this, the vanilla CUBA protocol component at the Intra-Quorum level for finalizing a partial block is lightened as in Figure 5.20 with one sub-phase of FINALISE instead of the PREPARE, COMMIT, and FINALISE sub-phases. Here we assume implicitly that a node being honest in a consortium setting will propose a valid partial block avoiding the extra agreement sub-phases. It is detailed in Appendix chapter 8, section 8.5 of Algorithm 18, which works as follows:

1. Each Quorum Q_r, upon receiving the transactions in its pool up to a partial block size κ finalizes a partial block $P_{k,l}$.

2. A quorum member is chosen to be a partial block proposer finalizer. It proposes the partial block $P_{k,l}$, which is then propagated to the network.

3. In case the proposer finalizer fails to respond within a timeout, the block fulfiller is chosen to finalize the partial block at the same index.

Implicit Agreement Limitation CUBA - Implicit has a message complexity of O(n) for n nodes inside a quorum compared to $O(n^2)$ of the vanilla variant, but it has certain

limitations as follows:

1. The choice of a single proposer finalizer at a round can affect liveness as it is a single point of failure. Although there is a partial block fulfiller, the eventual recovery of the original proposer at a given time t can introduce two conflicting blocks for the same index. This can lead to frequent forks as common in single proposer algorithms like Clique [118]

2. The reduction from a 3 sub-phase to a single phase can affect the consistency of the protocol, with each node having different views. For the same reason of state consistency, PBFT had to include COMMIT phase [155] for a uniform state. We can proceed with this consensus agreement if a protocol considers availability and partial tolerance over consistency.

5.3.2.3.3 Implementation In the previous sections, we discussed the CUBA Vanilla protocol and its variant with a detailed algorithm and description. We implement these consensus protocol designs into our simulator discussed in the previous Chapter 4. We re-utilize the existing simulator architecture (in Section 4.1.3) for plugging our CUBA protocol and its variant. In the following subsections, we explain the modified packages in the simulator to incorporate the CUBA protocol and its variants.

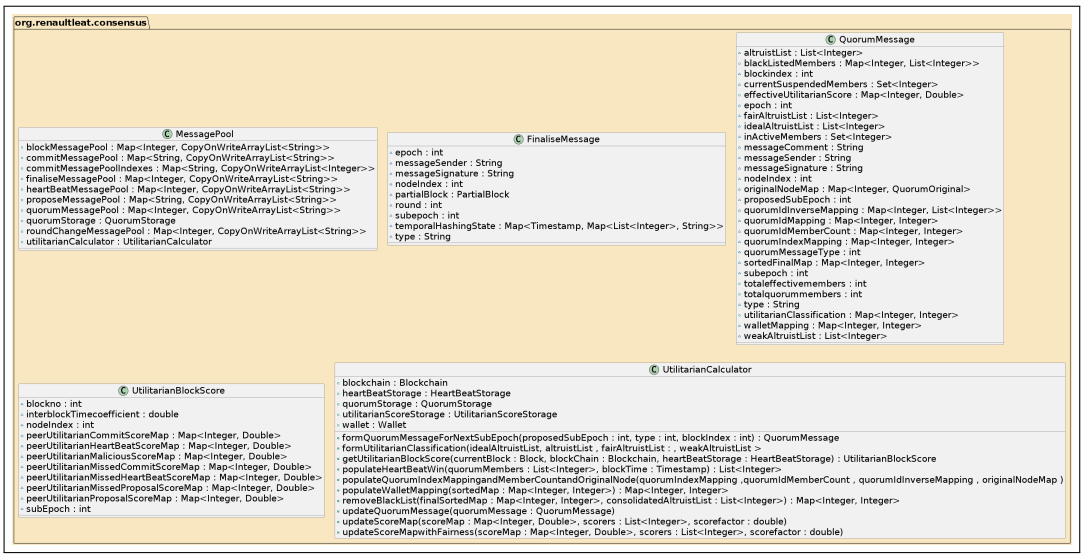

Figure 5.21: Simulator CUBA Consensus Package

Simulator CUBA Consensus Package In this consensus package, we have the following modifications or additions concerning the CUBA protocol classes as follows:

1. **UtilitarianBlockScore:** It is the data structure that holds the utilitarian score of the block according to its PROPOSAL, COMMIT, and FINALIZE votes, either gained or lost, including the malicious score.

2. **UtilitarianCalculator:** This class consolidates the score for each node for every epoch and then applies the normalization in items of inter-block time, fairness, Sisyphus coefficient, fairness, sorting, and classification of nodes into individual quorums.

3. **QuorumMessage:** This is the integral data structure necessary for performing the reorganization at the start of each epoch which has the classification, mapping of node index to its quorum indexes, and quorum metadata.

4. **FinaliseMessage:** This message differs from the conventional message of only votes and signatures like PREPARE and COMMIT as it holds the block container, hashing state, partial block, and votes.

5. **MessagePool:** This is the storage for all the message pools exchanged during the consensus agreement for PROPOSE, COMMIT, FINALISE, BLOCK, QUORUM, HEARTBEAT, and ROUND CHANGE.

6. **ConsensusMessageHandler:** It handles the different consensus messages for intra-quorum and inter-quorum phases, maintaining its threshold votes, consensus conditions, and validation check.

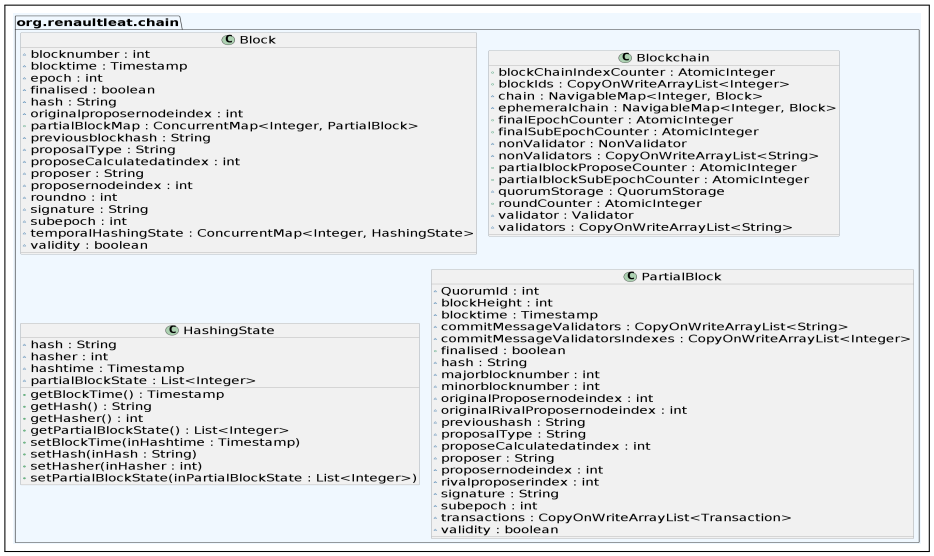

Figure 5.22: Simulator CUBA Chain Package

Simulator CUBA Chain Package: Similar to the consensus package, we explain the class additions for the chain package due to CUBA on our simulator as follows:

1. **PartialBlock:** This is the fundamental data unit in our blockchain protocol which agglomerates to form a block. It has the transaction list, proposer's data, and signature details, including the competitor proposer's data.

2. **Block:** The data structure change due to CUBA is implemented here as a Map of Partial Blocks is maintained with their index positions. Also, the hashing state instance is updated for the block when a partial block is appended.

3. **HashingState:** This structure intervenes with the block structure to maintain the temporal hashing during the ephemeral chain state. Also, the partial block state is recorded and updated simultaneously to have proof of addition.

4. **Blockchain:** It has the Ephemeral and Finalised blockchain state maintained by an

individual node. The epoch increments, quorum updates, round change necessity checks, and block round increments are performed here and synchronized between different processes accessing it.

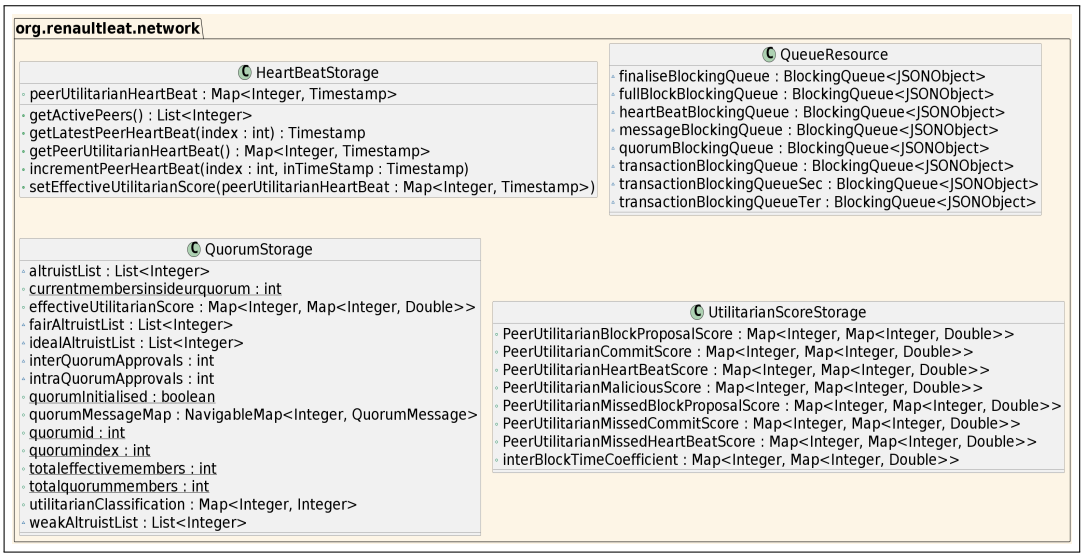

Figure 5.23: Simulator CUBA Network Package

Simulator CUBA Network Package: The network package for CUBA has certain updates concerning the storage as follows:

1. **HeartBeatStorage:** Heartbeat messages exchanged across the different nodes are stored here by each node queried during the quorum formation.
2. **QueueResource:** The messages exchanged across intra-quorum, inter-quorum, heartbeat, and quorum messages are received into a blocking queue for maintaining synchronicity and thread safety as our simulator is multi-threaded.
3. **UtilitarianScoreStorage:** All the Utilitarian scores, both normalized and raw, are stored and mapped to each node. It is stored at the end of each epoch based on the node's participant behavior in the blockchain consensus.
4. **QuorumStorage:** Similar to Utilitarian storage across Epochs, each epoch's quorum formation is stored here and is a historical reference for each node's quorum membership.

5.4 Conclusion

This chapter introduces the overall working of the Competing Utilitarian Byzantine Agreement protocol by explaining its problem statement and detailing its different phases. The protocol is conceptualized by considering the various alternatives in the literature, which is gradually forged philosophically. Democracy, Utilitarianism, Panopticon, Fairness, Sisyphus, and Swarm are applied to our problem statement and its justifications. Then the protocol is explained algorithmically through its intra-quorum consensus, inter-quorum consensus, and utilitarian score calculation, followed by the quorum formation. Also, the

variant of CUBA implicit is discussed with its limitations, and then finally, the implementation on the simulator is detailed. In the next section, we evaluate our CUBA across the spectrum theoretically and experimentally to understand its robustness and alignment with network throughput, scalability, security, and resiliency parameters.

Chapter 6

CUBA Evaluation

Evaluation is creation: hear it, you creators! Evaluating is itself the most valuable treasure of all that we value. It is only through evaluation that value exists: and without evaluation, the nut of existence would be hollow. Hear it, you creators!

– Friedrich Nietzsche

In this chapter, we look at our designed protocol CUBA and understand its complexity from the perspective of theoretical assessment covering the aspects of its message complexity and CAP theorem-based analysis. Also, we evaluate its real implementation on our simulator discussed previously across different scenarios to understand its performance, security, resilience, and adaptability metric using terms of utilitarian score in response to threats.

6.1 Theoretical Evaluation

In this section, we assess the Intra-Quorum, Inter-Quorum, Utilitarian Evolution protocol phases, and the adverse scenarios inviting round change. This is extended to the quorum reorganization for each epoch and heartbeat message exchange. It is further analyzed through the lens of CAP theorem and Blockchain Trilemma to discover its limitations, especially in an eventual synchrony model.

6.1.1 Algorithm Complexity

This section considers the CUBA protocol's critical path, and then we dissect each phase by looking at its algorithmic design and complexity. We analyze the complexity from two perspectives in our blockchain, aka "Distributed systems," as follows [251, 252]:

1. **Message Passing or Communication Complexity:** The message complexity of an algorithm for either a synchronous or an asynchronous message-passing system is the maximum overall execution of the total number of messages sent.
2. **Time Complexity:** The time complexity of an algorithm for a synchronous message-passing system is the maximum number of rounds in any execution until it has terminated.

In this analysis, we consider the following notation reprisals from the previous chapter:

1. K - Total Members in the network ζ.
2. ρ - Total number of Quorums in the network
3. σ - Total allowed member participation in each Quorum
4. $P_{k,l}$ - Partial Block unit where k is the Block Height, and l is the index
5. B_k - Block unit where k is the Block Height in the Blockchain (Ephemeral or Finalised)
6. ξ - Round Change Timeout
7. Γ - Block Height for progression to new Epoch
8. ε - Latest Epoch Height for which we consider since the Genesis Epoch

6.1.1.1 Intra-Quorum Message Exchange

This first phase of CUBA protocol has message sub-phases of PREPARE, COMMIT, and FINALISE. When we consider this protocol asymptotically for message communication, we observe that the COMMIT protocol involves quadratic complexity $O(\sigma^2)$ while the PREPARE and FINALISE are in the order of $O(\sigma)$. We consider the dominant overhead of the COMMIT sub-phase rather than the PREPARE or FINALISE.

In COMMIT, a partial block proposer expects a commit message for the proposed partial block from other quorum members to finalize the partial block $P_{k,l}$. Another point to consider is the competitive partial block proposed, which produces $O(\sigma^2)$ in addition. Summing

up we obtain O($2\sigma^2$) which we normalise to O(σ^2).

Next is the time complexity for the algorithm. It is oriented around message passing and validation until the majority threshold is reached. Considering the validation effort on each partial block received for the ephemeral chain, we can represent it as O(ρ). We exclude round change from intra-quorum and inter-quorum analysis as it needs to be dealt with separately in the worst-case scenario of the liveness problem in the network.

6.1.1.2 Inter-Quorum Message Exchange

Next is the secondary phase of the protocol, where the full block B_k is finalized by the block proposer and broadcasted to the entire network at scale. So the asymptotic message complexity is O(K). The time complexity is O(ρ) as it is more of a validation effort to be performed by the nodes on each partial block component of the block which is computationally costly.

6.1.1.3 Round Change

The Round Change message exchange happens in the case of blockchain ephemeral or finalized remaining unprogressive until ξ timeout. In the round change finalize sub-phase, there is a one-to-one communication between all the network members K for revising the quorum organization. So the complexity is quadratic and quite expensive with O(K^2). Time complexity is more of O(1) since there is no heavy computation except for message accumulation until the threshold is reached. A point to note is that the round change entails a utilitarian message exchange which normally happens at the start of a new epoch.

6.1.1.4 Utilitarian Message Exchange

In contrast to earlier phases or sub-phases, the quorum message exchange or proposition phase happening at the start or end of each epoch has a time complexity higher than the message complexity. The heartbeat message exchange has an O(1) time complexity as it is a simple message payload and O(K) communication complexity for delivering the message to the consortium network. Message complexity for quorum proposition is O(K). To understand the time complexity of the quorum proposition phase, we organize it as below code steps, each with its time complexity as follows:

1. Calculation of Effective Utilitarian Score for recent epoch iterating Γ blocks where each has ρ partial blocks in turn - O($\Gamma\rho$)
2. Calculation of Active Peers calculated by interrogating the heart-beat storage for each node - O(K)
3. Calculate the forgetting coefficient for each previous epoch concerning the current epoch and apply it to the effective score for each node - O(ε)
4. Apply the fairness coefficient to each of the nodes based on its historical classification - O(K)
5. Sorting of nodes based on Effective Score in descending order by iterating each node.

This is based on the adaptive merge sort algorithm available in the collections library of Java - O($K \log K$)

6. Classification of nodes based on Utilitarian score and rank - O(K)
7. Form Blacklist and suspend the malicious node if it is in very weak utilitarian list for Z previous consecutive epochs - O(Z)
8. Form the ρ quorums with the classified nodes - O(K)

In the above list of steps, the dominant time complexity element we consider asymptotically is O($K \log K$).

6.1.1.5 Pipelining Effect on Protocol

In this section, we analyze the pipelining by changing our data structure of a Block B_k as a set of $\{P_{k,1}...P_{k,\rho}\}$ partial blocks. Also, having a differentiated ephemeral chain from the finalized chain, we can progress to a round i+1 even if round i, i-1... is still being processed. In avoiding the sequential execution of block processing, we can achieve an amortized set of blocks in the queue to be eventually finalized without having a waiting time. But at the end of an epoch, the progression of the round can only be possible if the last epoch is filled up to the latest finalized block height. This is necessary as a quorum reorganization is multidimensional based on the recent epoch, heartbeat, and a gradual view of historical epochs. A finalized consistent state is necessary for quorum reorganization, but this is a parameterized factor of deciding the frequency of epoch, which will be studied in detail during experimental evaluation.

6.1.2 Consistency, Availability, and Partition Tolerance Analysis

This section applies the CUBA analysis through the Consistency, Availability, and Partition Tolerance Theorem (CAP Theorem). The three fundamental properties to be respected by any distributed system are discussed as follows:

- **Network Partition Tolerance:** It is a communication problem caused by message delay or loss within a distributed system, and the system should continue to be functional despite the issue.
- **Consistency:** All nodes in a distributed system should be synchronized, and data should be replicated uniformly where a client should access the data from any node as if it were a centralized system. In the case of blockchain, it should be resilient to forks achieving total order and agreement.
- **Availability:** This property should ensure that if several nodes are faulty or down, the client can retrieve the data without any issue, as the other nodes should act as backup.

CAP Theorem states that the distributed system can achieve only two properties out of these three. This can be explained by:

- **No Availability (CP):** In this case, the system favors consistency and partition tolerance over availability. When a partition occurs, the inconsistent node will be shut

down, and the system will be unavailable.

- **No Consistency (AP):** Availability and Partition Tolerance are prioritized over consistency here. When a partition occurs, the system is available, but the unsynchronized slice of the system will reply with a stale version of the data.
- **No Partition Tolerance (CA):** As the previous two here, partition tolerance is compromised as the system favors consistency and availability, which cannot be achieved in a partitioned network and cannot tolerate faults.

6.1.2.1 Misconception and CAP Revisited

There is a misconception with the CAP theorem as stated by Brewer, Eric [253] that the system choice of CP, AP, and CA is not a binary choice of 2 out of 3. Still, it only limits certain aspects. It instead prohibits only "perfect availability" and consistency in the presence of partitions. This implies the following:

- AP relax consistency but are not completely inconsistent.
- CP sacrifice has a tolerable availability and is not completely unavailable.
- So, it reveals that AP and CP systems can offer an acceptable level of consistency and availability in the presence of partition tolerance.

An AP system can achieve best-effort consistency by adopting web caching or DNS to have an intermediate consistency offering. Or, in the case of the CP system, it can be a best-effort availability like Google Chubby service. So organizations like Facebook and DropBox adopt eventual consistency instead of the strong consistency model to have a tradeoff between Consistency, Availability, and Partition Tolerance. Strong Consistency ensures that the same updated value is returned across the system after a state update. Eventual Consistency guarantees that if no new updates are made to the state, then eventually, after a time δ we can get the updated value.

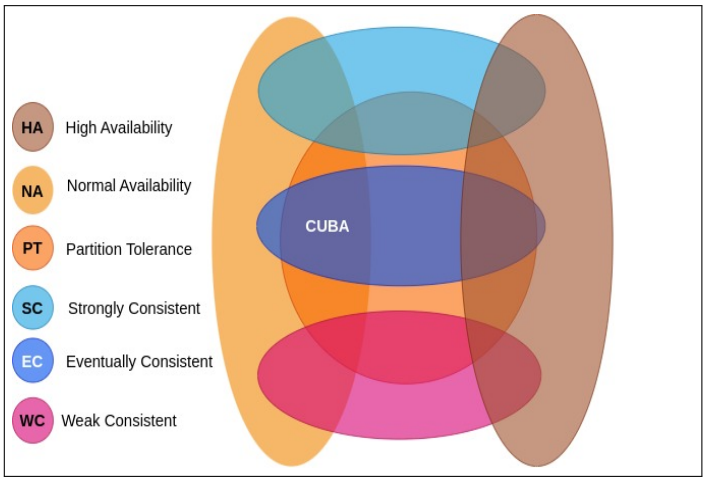

Figure 6.1: CUBA CAP Analysis

We analyze the CUBA protocol to understand its limitations in line with the above perspective. CUBA works in an eventual synchrony model by progressing the epochs where

a new set of ρ quorums, each having σ members inside a quorum. As the progression of the finalized chain is based on the block height and epoch, it is resistant to clock skewness to an extent. But in case of unexpected anomalies of partial block fulfillment timeout, full block fulfillment timeout, and round change timeout, the protocol depends on clock synchronizations. This can lead to an intermittent liveness issue until the node recovers within a global timeout δ. So essentially, our protocol cannot offer strong consistency but an eventual consistency model. Also, in case of an extended liveness issue, a round change is invoked where the quorums are reorganized, and failure nodes can be suspended for certain epochs rendering the network resilient. It is evident by the heart-beat message and the inter-block time coefficient the utilitarian score can diminish the overall score of a failure node. As Figure 6.1 shows, our protocol offers normal availability, eventual consistency, and partition tolerance. Let us understand the protocol by the following questions:

- **Why not high availability?**
  In the protocol, we can sustain until $2f + 1$ node for f failure nodes in the overall network as we need benign nodes to progress, or we would repeatedly fall to the round-change phase. This makes the system reliant on the minimum $2f + 1$ nodes for safety and liveness to render the blockchain available normally. A highly available system would be immune against any threshold of bad actors, which CUBA cannot offer in our byzantine setting.

- **How does CUBA trade between partition tolerance and eventual consistency?**
  A blockchain protocol should mandatorily be partition tolerant either a priori like PBFT [73] protocol where there is no possibility of forks. Or be tolerant aposteriori like Clique, Aura protocol [213] which uses a fork resolution algorithm like GHOST in addition to resolving the blockchain partitions. In CUBA, we avoid the presence of forks as there is a checkpoint mechanism of quorum reorganization every epoch interval, which detects the drift and auto-corrects to avoid the adverse scenario. Also, in terms of multiple quorums, each limited to a particular index of the partial block, the organization cannot create a long chain to produce a fork without consensus from the benign participants. In addition, The intra-quorum phase has three sub-phases, making it eventually consistent with an acceptable timeout. The corner case of more than $2f + 1$ malicious nodes can create a partition that renders the system prone to byzantine failure, an acceptable limitation within the threshold.

6.1.3 Blockchain Scalability Trilemma Analysis

This principle states that any blockchain system can have at most two properties of the following:

- **Decentralization:** All the nodes participating in the network will have the same resource possession or authority level, such as validator.

- **Scalability:** It refers to the maximum participant in the network who are involved in the consensus of transactions and their ability to scale in transaction processing.

- **Security:** It refers to the resistance of the node against attacks which depends on the consensus protocol and majority threshold to control the network.

To cite an example of Proof of Work, blockchains like Bitcoin or Ethereum [254] have scalability and security properties but lack decentralization as nodes hold block mining with high computational power. But layer 2 or sharding solutions can overcome this trilemma

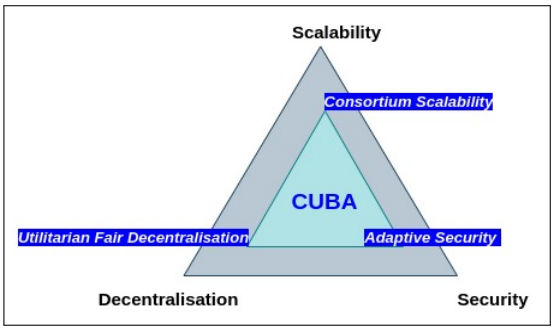

Figure 6.2: CUBA Scalability Trilemma Analysis

limitation, which has all three properties. In our case for CUBA protocol, as represented in Figure 6.2, we can arrive at a median satisfying the three concerns as follows:

- **Scalability**: CUBA protocol tackles scalability by the trilogy of pipelining, of the blockchain data-structure change, and reduced communication complexity through the quorum mechanism.
- **Decentralisation**: All the nodes can participate if it respects the utilitarian notion through intra-quorum and inter-quorum consensus. Also, fairness to improve the utilitarian score of a node is embedded in the protocol design and ensures the consortium network of egalitarian participation.
- **Security**: As the quorum organization is revised at an interval of each epoch based on a node's contribution to the previous epochs, the mechanism of heart-beat, forgetting coefficient, and fairness help in identifying the failure or malicious node in the network. This is similar to a feedback loop system where the current or past state determines future actions.

In summary, our CUBA consensus achieves a middle ground among the three vital metrics of a blockchain protocol. This theoretical understanding will be further verified in the future section during the experimental section discussion.

6.1.4 Utilitarian Fairness Evaluation

Fairness from the point of computer science was first studied by Francez, Nissim in the context of concurrent or parallel systems, which can be extended to our blockchain protocol similarly. CUBA protocol ensures fairness in its approach by the following mechanism:

- **Fairness of Participation**: Any node N_i where $i \in \{1...K\}$ belonging to consortium ζ can join the network and is not limited to any set of actions like the proposal of the partial block, full block, quorum or validation. But it has diverse roles based on their utilitarian score and probability of being selected based on the codepoint mechanism. This ensures a fair decentralization with a uniform probability of occupying varied roles and utilitarian levels at any block height of an epoch.

- **Fairness of Resource**: Any node has to keep a check on its virtual resource of utilitarianism rather than any external asset-based resource like in Proof of Stake or hardware dependence in case of Proof of Work. This directly influences the optimization of the network, and no node is limited to giving its honest participation in the network without any resource or asset-based biases.
- **Fairness of Utilitarian Mechanism**: All the nodes can achieve an egalitarian, utilitarian score if it respects the voting conditions, signature, and heart-beat. Any node cannot selfishly accumulate a utilitarian score and improve its position. Further, the calculation of this score is decentralized and transparent, and each participant can verify with the historical finalized blocks. Also, the Sisyphus forgetting coefficient makes the system converge more towards recent actions with less influence on the earlier disposition making it completely evolutionary and fair.
- **Fairness of Classification and Recovery**: Utilitarian classification mechanism as Ideal Utilitarian, Utilitarian, Fair Utilitarian, and Weak Utilitarian is to enhance the score of participants. Those with a relatively lower score are attributed a fairness coefficient inversely to their rank and classification for each epoch. It helps the participants in the lower range of Fair and Weak Utilitarians to gradually reach the level of Ideal Utilitarianism if and only if they align with the optimization of the network.

6.1.5 Adverse Scenario Evaluation

In this section, we evaluate CUBA across broad contours of the system's resilience against probable adverse mechanisms. Our considerations for this section are:

1) We constrain ourselves to attacks intrinsic to blockchain and assume attacks at the network level, infrastructure like DDoS, phishing, security key theft, and key cloning to be solved by other standard mechanisms available.
2) We focus on the working of the consensus protocol abstracting the choice of signatures, smart contracts, wallets, and other extraneous scenarios.
3) The threshold of honest nodes we assume is $2f + 1$ for f dishonest nodes.

It is analyzed by considering the following cases:

- **Case 1: Ineffective Participation** In this attack, a certain set of nodes can abstain from sending or passing messages or voting in intra-quorum or inter-quorum phases. This can be detected based on the score calculation post the block finalization and can lead to penalization of commit, heart-beat, and proposal, reducing the node's utilitarian rank. This leads to further node suspension, making the network resilient at the start of the next epoch.
- **Case 2: Berserk Nodes** These nodes send different messages to different nodes, which will be conflicting. This equivocation can be detected as we have a closed set of quorum participants with a verification mechanism of the message received from other nodes. Also, our protocol is not based on gossip or opinions about information, but it's a binary vote of acceptance or refusal for a particular block, making it resilient. Reiterating the mechanism of the utilitarianism score augments the protocol to evade

this case.

- **Case 3: Partition Attack** If we assume an intermittent partition less than the global stabilization time of δ for an eventually synchronous network, then liveness can be affected during that period. It leads to the exceptional scenario of round change if the round timeout is exhausted and leads to a protocol operating within a set of live-connected participants.
- **Case 4: Threshold Breach** In the case of more than 1/3 and less than 2/3 adverse nodes, round change will be initiated to reorganize the network. In the worst scenario of more than 2/3 nodes, consensus will be impacted. The obtained quorum organization will be ineffective and make the round change recursive, or the malicious blocks will be finalized in the network.
- **Case 5: Fork Handling** The presence of forks can be handled with a partition of fewer than 1/3 participants. It is the maximum threshold where a new Round Change can proceed to reorganize the quorum or the protocol is stalled until the partition is reinstated. In the case of an acceptable fork within a range of 1/3 participants, then we assume two partitions created as P_1 and P_2. P_1 partition is assumed to contain 2/3 members, and P_2 will have the remaining participants. P_1 will proceed with a round change and new quorum evolution. P_2 will not be able to proceed as it lacks the minimum threshold of participants to proceed and will join the network in the future epoch if the partition is resolved by that time.

We extend the study on the security of CUBA protocol by implementing them in the simulator. We perform the empirical analysis to ascertain the immunity level of the protocol and test its optimization through network evolution.

6.2 Experimental Evaluation

In this section, we investigate the CUBA protocol across a broad spectrum of scenarios which we discuss utilizing simulation results. We analyze the protocol in each situation and derive its key metrics regarding performance, quorum organization, node evolution, and utilitarian score impact.

6.2.1 CUBA Protocol Parameterisation

This section involves the list of CUBA protocol parameters that we set by default in our simulation analysis as listed in Table:6.1.

6.2.2 Methodology

We inherit the same methodology to test our simulator concerning classical algorithms of Clique, PBFT, IBFT, and QBFT as explained earlier in Chapter 4. We test the performance of the CUBA protocol by scaling the participating node count in a pattern of 5, 10, 15, 20, and 25 by measuring its throughput in transactions per second. We also observe the other metrics of the utilitarian score during the test to gauge the evolutionary aspects of the consensus. Code Implementation of the blockchain simulator for CUBA and CUBA Implicit,

Notation	Parameter	Value
PB_Proposal_win	PartialBlock_Proposal_Win_ScoreUnit	0.5
PB_Commit_win	PartialBlock_Commit_Win_ScoreUnit	0.3
HB_Score_win	HeartBeat_Win_ScoreUnit	0.1
PB_Proposal_loss	PartialBlock_Proposal_Loss_ScoreUnit	0.3
PB_Commit_loss	PartialBlock_Commit_Loss_ScoreUnit	0.3
M_Score	Malicious_ScoreUnit	0.5
HB_Score_loss	HeartBeat_Loss_ScoreUnit	0.1
IU_{sc}	Ideal_Utilitarians_ScoreFairnessCoefficient	0
U_{sc}	Utilitarians_ScoreFairnessCoefficient	0.05
FU_{sc}	Fair_Utilitarians_ScoreFairnessCoefficient	0.1
WU_{sc}	Weak_Utilitarians_ScoreFairnessCoefficient	0.15
Full_κ	Full_Block_Size	2000
ξ	Round_Change_TimeOut	60 seconds
Δ	Ephemeral_State_TimeOut	5 seconds
η	Finalised_State_TimeOut	5 seconds
θ	HeartBeat_Frequency	5 seconds
Z	SuspensionEpoch_Depth	5
c	Assign Transaction to a Quorum Codepoint position	5
c_1	Partial Block Proposer Codepoint position	2
c_2	Partial Block Competitor Codepoint position	6
c_3	Partial Block Finaliser Codepoint position	3
c_4	Partial Block Fulfiller Codepoint position	8
c_5	Block Proposer Codepoint position	6
c_6	Block Fulfiller Codepoint position	5
c_7	Quorum Proposer Codepoint position	4
c_8	Round Change Proposer Codepoint position	7

Table 6.1: CUBA Protocol Parameters

along with cloud deployment of Kubernetes, transaction clients, docker configuration files, simulation result visualizer, and test results, are released publicly in the GitHub repository: https://github.com/scyrilnaves/these-cuba

6.2.3 Infrastructure

The CUBA simulation test infrastructure is identical to the previous simulation test of classical consensus protocols, as explained in Chapter 4. We use the CUBA docker image of the implementation discussed in the previous chapter 5 for our container.

6.2.4 Result Discussions

In this section, we focus on the experimental evaluation through varied scenarios. These scenarios are chosen to interrogate the CUBA system and understand its improvements and limitations.

6.2.4.1 What can be the optimum epoch limit for network self-optimization?

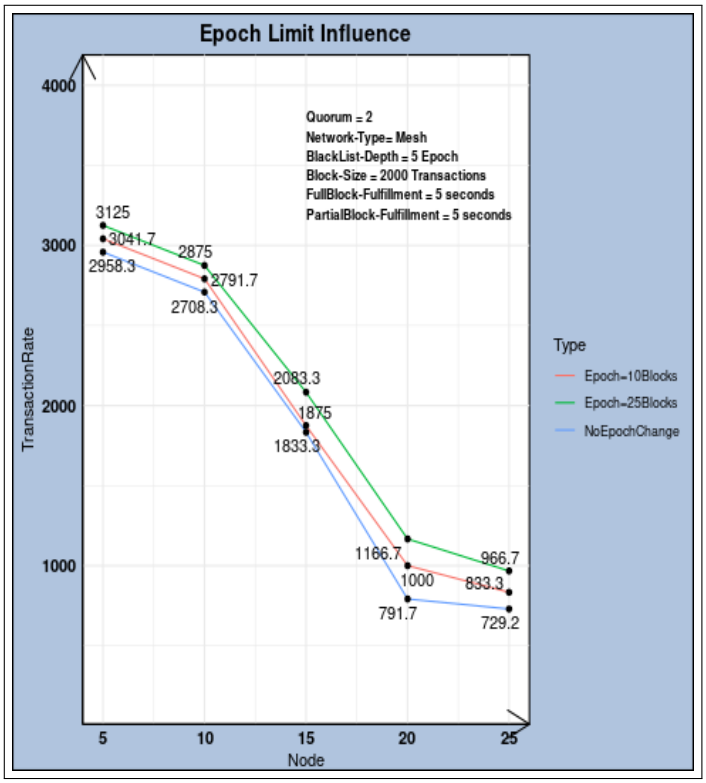

Figure 6.3: CUBA Epoch Limit Analysis

In this scenario, we vary the epoch duration of the CUBA protocol, which in turn affects the quorum reorganization frequency, a quite heavy operation in terms of time complexity for a chosen node to be the quorum proposer. Meanwhile, node scalability is increased gradually from 5 nodes to 25 nodes across the epoch change, and throughput is measured. The number of quorums is maintained at a minimum of 2 uniformly throughout the test. This is chosen because it is the best communication configuration when considering inter-quorum consensus messages. The remaining CUBA network configurations are marked as annotations in Figure 6.3 representing the results obtained during this test scenario. The epoch variations performance are analyzed as follows:

- **No Epoch Change:** This has the least performance as the quorum optimization does not take place, and if the node fails, it can lead to an unstable network without scope for the evolution of the network for good.
- **10 Block Epoch Change:** This test has an epoch change every 10 blocks, which is computationally expensive as discussed in 6.1.1.4. So the gain due to optimization is

neutralized by the equally high computation cost, which is not a wise choice. Moreover, the nodes are fairly honest and don't have a short frequency of epoch evolution.

- **25 Block Epoch Change:** In this analysis, a 25-block epoch change is more optimal than the former.

All nodes experience a drop due to increasing nodes which is more drastic after 15 nodes due to increasing message complexity. But this scalability drop will be further optimized as we are in the exploratory phase to choose the right configuration.

6.2.4.2 What is the heuristic for choosing the number of quorums?

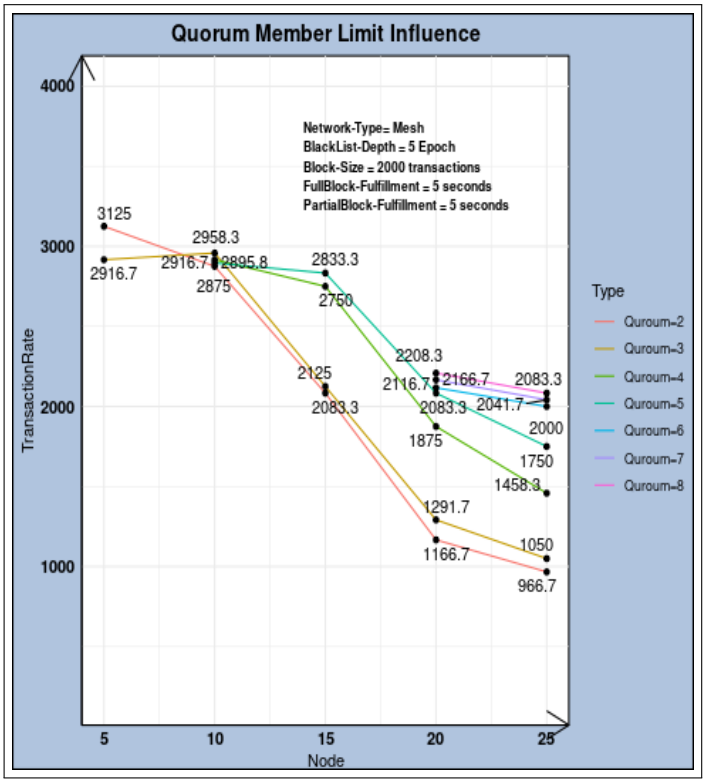

Figure 6.4: CUBA Quorum Member Limit Analysis

The next important configuration choice we need to understand is the heuristic of the number of quorums ρ, which is inversely proportional to each quorum size σ. We maintain the configuration as represented in Figure 6.4. The Epoch limit Γ is set to 25 blocks, as we have noticed it is an optimal configuration from the previous results. Quorum choice is based on two factors as below:

- Similar to the consensus protocol related to quorums [95, 256] when the number of quorum members increases, it increases the failure tolerance or liveness.
- In our CUBA protocol, the intra-quorum phase has a relatively higher communication complexity sub-phases than the inter-quorum. Increasing the quorum size, on the one hand, can increase the resilience of the protocol but can degrade the much-needed lighter communication.

As represented in Figure 6.4 we notice that we test the different number of quorum patterns across increasing nodes to evaluate their performance. In the case of fewer nodes, the number of quorums being 2 & 3 has higher performance as it is a small network such that the intra-quorum phase has less impact. But in case of a higher number of nodes starting from 15 nodes, the Number of Quorums = 4 & 5 has higher performance than others. For large participation configurations like 20 & 25 nodes, the scalability is optimized by choosing a relatively equivalent value of Quorum = 7 & 8 configurations. So based on these results, we infer the following:

$$\text{Number of Quorums } 1/\propto \text{Quorum Size}$$

$$\text{Quorum Size} \propto \text{Liveness or Fault Tolerance in Network}$$

$$\text{Quorum Size} \propto \text{Size of Network}$$

$$\text{Quorum Size } 1/\propto \text{Communication Complexity}$$

6.2.4.3 Can the network topology have an effect on the CUBA protocol?

As explained earlier in Chapter 4, we have built our simulator to construct different topologies of peer-to-peer networks like Mesh, Lattice, or Watts-Strogatz model. The communication influence for these different topologies and their configuration parameters are presented in Figure 6.5. The epoch change is maintained at 25 blocks, and the number of Quorums is chosen to be 5, an acceptable figure between liveness and communication complexity. Its results are discussed as follows:

- Mesh Topology is relatively better than lattice in the case of 10 and 15 nodes and scales down from 20 nodes. The initial gain is due to direct connection, but when the network size increases, the gain is minimized due to socket connection overload for receiving and sending messages.
- Between Lattice and Watts-Strogatz, the latter performs better due to the shorter average distance between nodes.
- For comparing a large network size homogeneously, we notice that Watts-Strogatz is better than mesh or lattice as the distance is shortened due to the lesser load on socket communication.

Although Watts-Strogatz represents an ideal picturization of any blockchain network, we will maintain ourselves to mesh topology for further evaluation. Our protocol is designed for a consortium network with more or less optimized network communication mesh ensuring that the nodes are closely connected. An additional reason for mesh is the introduction of fault nodes advertently during our evaluation which might hamper the communication and bring in a cascading effect of latency to other nodes.

6.2.4.4 How effective is the protocol resistant to Node failures?

In the previous section, we were able to decide the fundamental factors in the CUBA protocol of Epoch Limit, Number of Quorums, and the Network Topology. In this section, we perform the empirical analysis more profoundly on how our network is resilient to attack scenarios, tolerant levels, utilitarian score evolution, and how it performs the much-needed network evolution for optimization.

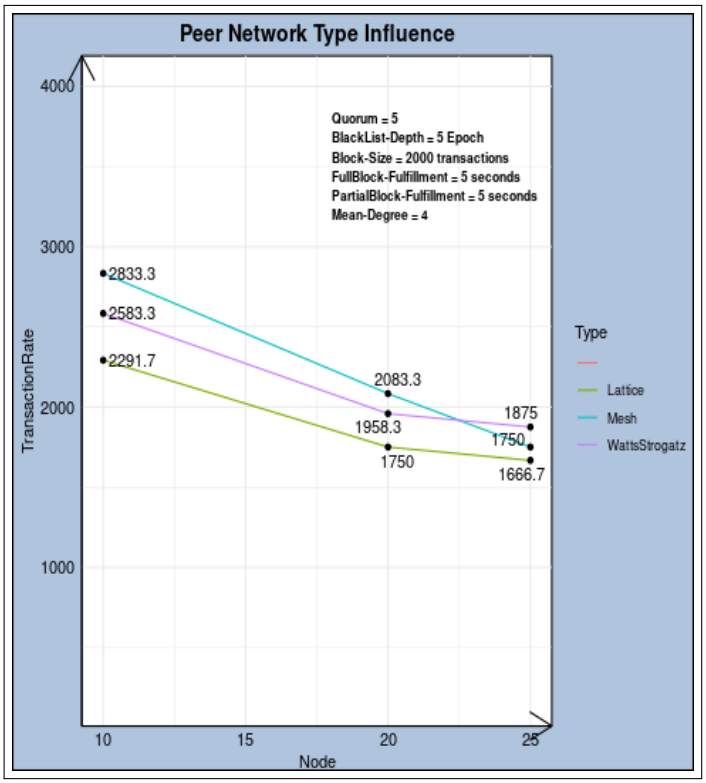

Figure 6.5: CUBA Network Type Analysis

6.2.4.4.1 Case1: Benign Normal Scenario In this scenario, we have a network that is comprised of only honest nodes, but they can be prone to benign or unconscious failure. As represented in Figure 6.6 we test a network of 10 nodes with 2 Quorums. There is an observable ascent in terms of throughput until 60 seconds, and then we notice a drop and stabilization. The reason for the intermittent drop can be explained with Figure 6.7, which denotes the actual effective utilitarian score calculated by the CUBA protocol during the test run. We notice that the least performing nodes are node 3 and node 8. This can be further drilled down in Figures 6.8 and 6.9, which present the partial block score win and loss utilitarian score, respectively. As noticed, their scores have deteriorated as they have missed proposing partial blocks in several consensus rounds. This is similar to the case of missed commit votes for the partial blocks relatively higher than others as in Figure 6.13. While the heart-beat score representation in Figure 6.12 for all the nodes is more or less equivalent with no major incident. As represented in the utilitarian classification of Figure 6.13, we notice the genuine failure of nodes 3 and 8 due to their partial block competition loss.

In the utilitarian classification of Figure 6.13, we analyze that node 0 is in the initial Weak Utilitarian phase. Still, due to sustained recovery, it can achieve utilitarian classification at the end of Epoch 10. On the contrary, node 3 descends from utilitarian to weak utilitarian, which is explained by the reason for the missed partial block and commit score. As observed in Figures 6.10 and 6.11 for commit utilitarian and missed commit utilitarian, node 3 has a low score in the positive sense and high in the negative sense which explains its faulty nature. Nodes 4, 7, and 8 have relatively balanced progress between Ideal Utilitarian,

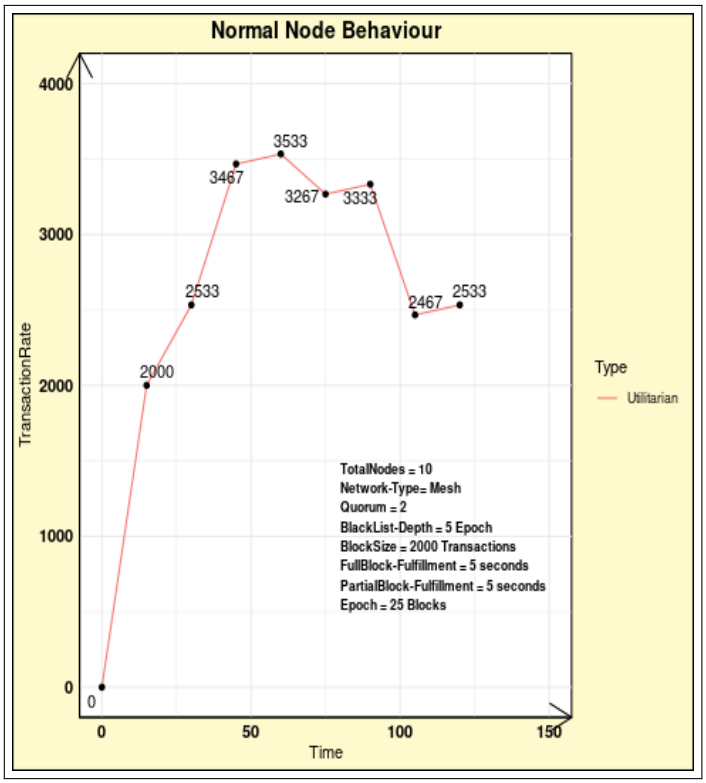

Figure 6.6: Case1: CUBA Normal Benign Node Performance

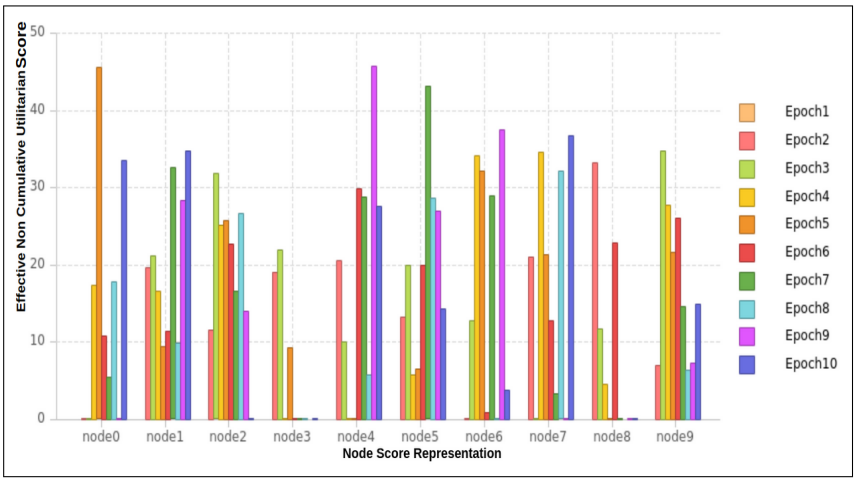

Figure 6.7: Case1: CUBA Benign Effective Utilitarian

Fair Utilitarian, and Weak Utilitarian levels, representing a constant evolution in proportion to their positive or negative actions. In conclusion, we can verify during the test that every utilitarian action missed or performed has a direct reflection on the network and its individual score.

6.2.4.4.2 Case2: Benign Latency Scenario In this scenario, we introduce the honest node network, but we introduce communication latency of 100 milliseconds for 3 Nodes of Node 7, 8, and 9, both in their send and receive functions. This delays their consensus

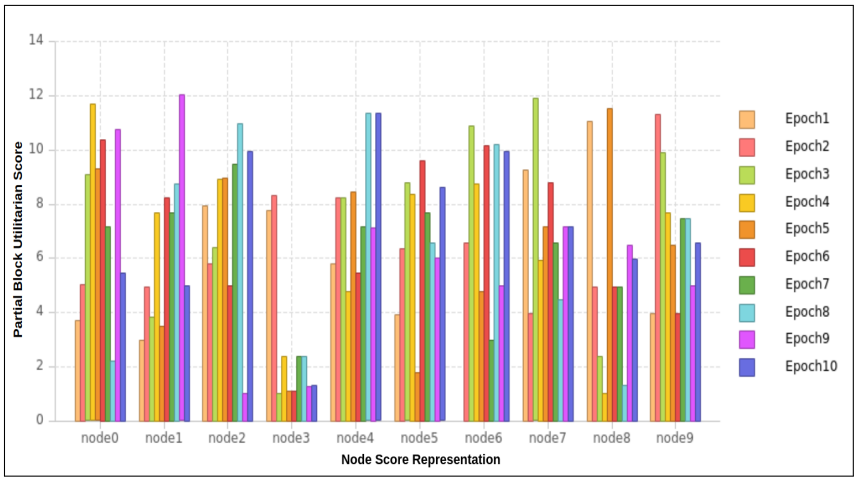

Figure 6.8: Case1: CUBA Benign Partial Block Utilitarian

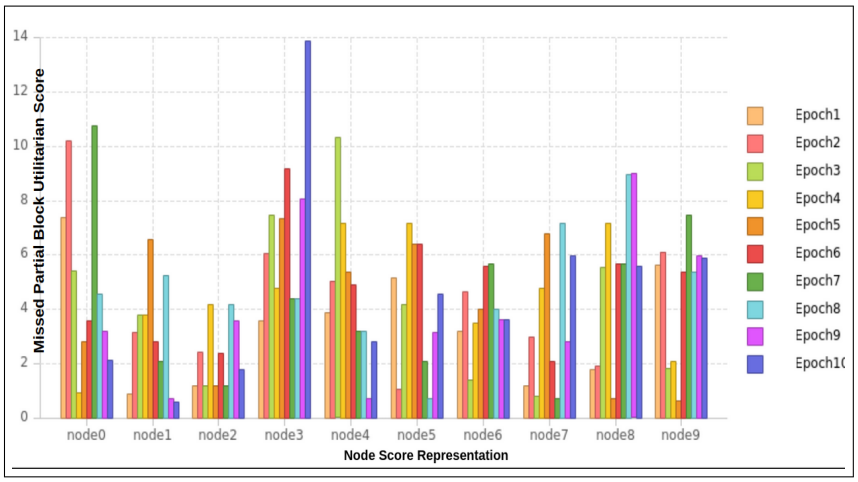

Figure 6.9: Case1: CUBA Benign Missed Partial Block Utilitarian

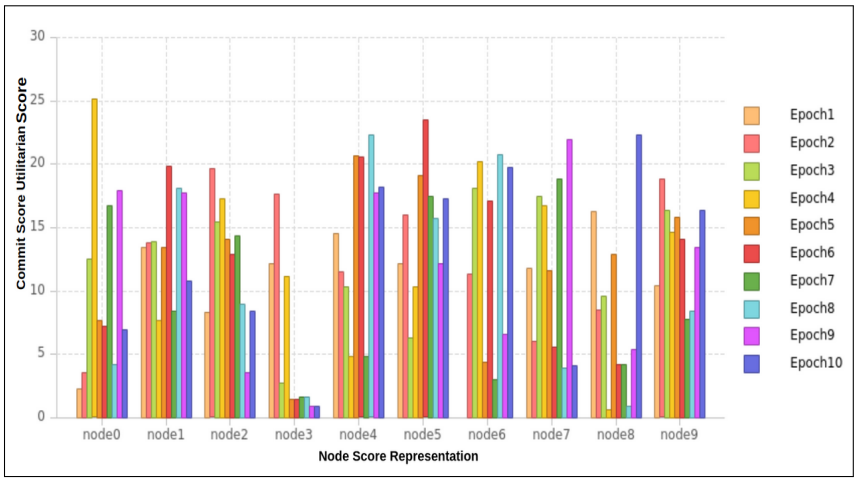

Figure 6.10: Case1:CUBA Benign Commit Utilitarian

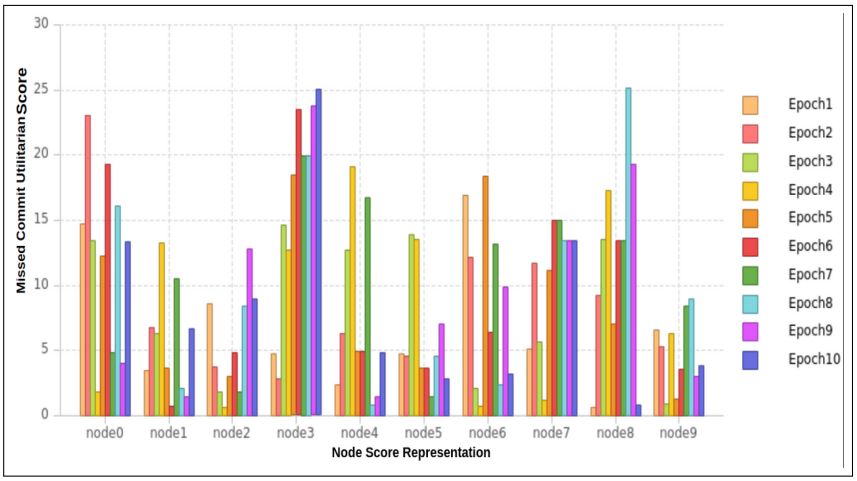

Figure 6.11: Case1: CUBA Benign Missed Commit Utilitarian

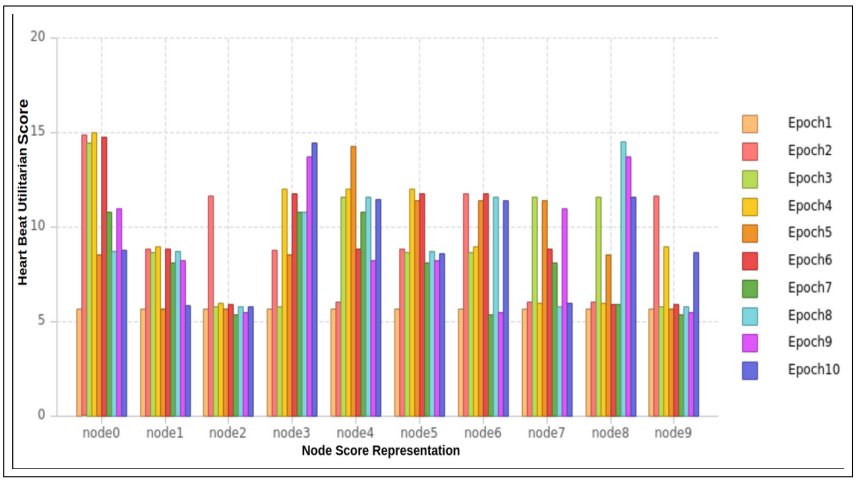

Figure 6.12: Case1: CUBA Benign Heart Beat Utilitarian

messages and the message it relays to other nodes in the peer-to-peer network of 10 nodes. We do not picture this as an adverse scenario. Still, the CUBA protocol will identify and classify the nodes inflicting this delay as in their performance represented in Figure 6.14, which degrades the throughput more than the former test result on total benign nodes. The plot shows a relatively slower ascent until 2900 transactions per second and stabilizes around 2200.

This network performance can be explained by the Figures 6.15 on Effective Utilitarian, which demarcates the Node 7,8,9 due to their latency introduction in the network. Complementary evidence is produced by Figure 6.16 on missed partial block and by Figure 6.17 on missed commit utilitarian score respectively. Failure to respond to the message in time, delays the consecutive actions for the nodes, in turn delaying the consensus. The Utilitarian classification as in Figure 6.18 marks these nodes between the Weak and Fair Utilitarian levels prone to network suspension. In an ideal case, we keep the epoch suspension period to 1 epoch as we assume it is a friendly consortium network. Still, we can increase it to higher periods of suspension if needed, as it is parameterizable.

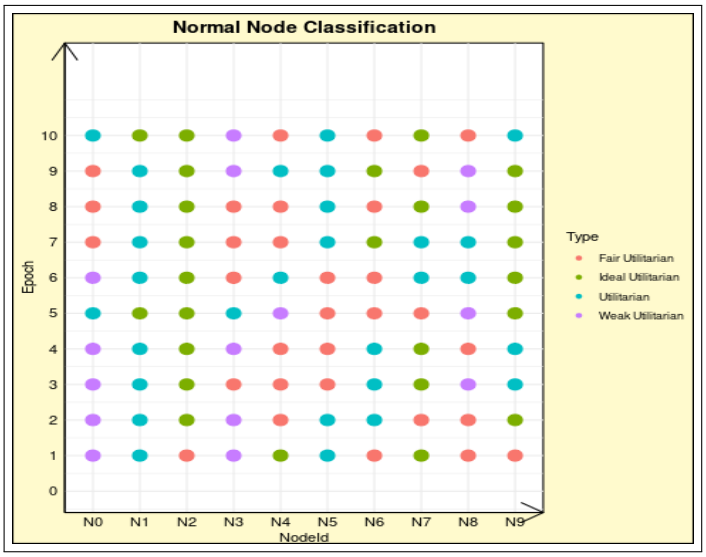

Figure 6.13: Case1: CUBA Normal Benign Node Utilitarian Classification

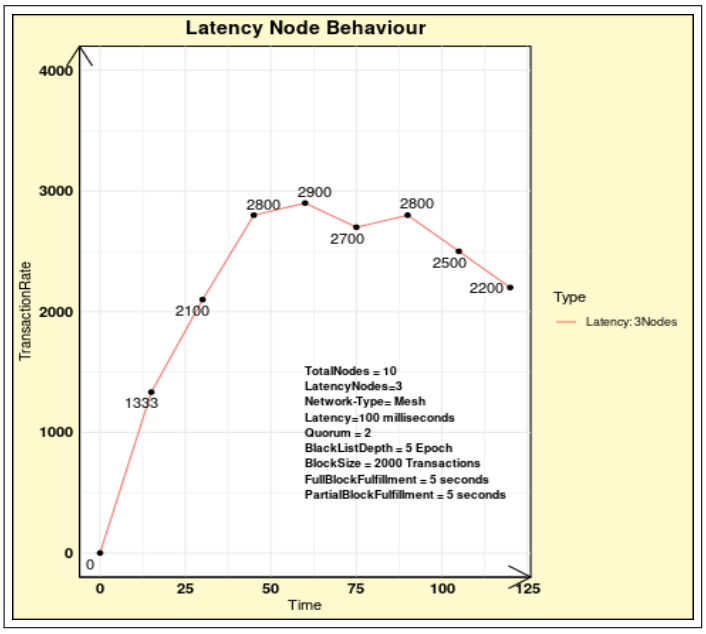

Figure 6.14: Case2: CUBA Normal Benign Latency Node Performance

6.2.4.4.3 Case3: Malicious Partial Block Latency Scenario In this scenario, we analyzed the CUBA protocol for malicious partial block proposal during the finalized sub-phase of intra-quorum consensus formed out of compromised signatures by Nodes 7, 8, and 9 and an additional latency of 200 milliseconds individually. Its performance is represented in Figure 6.19 where there is a slower rise to the plateau, around 1500 - 1600 transactions per second. This is a more perilous behavior as it severely degrades the system due to the unfilled ephemeral chain impacting the finalized chain. As this is a voluntary sustained malicious action, it is delineated clearly in Figure 6.21 on Effective Utilitarian and Figure 6.23 on Missed Partial Block. The Malicious Utilitarian score marks the action of Nodes 7,8, and 9 in Figure 6.22. Only Nodes 7, 8, and 9 have their malicious score filled due to their

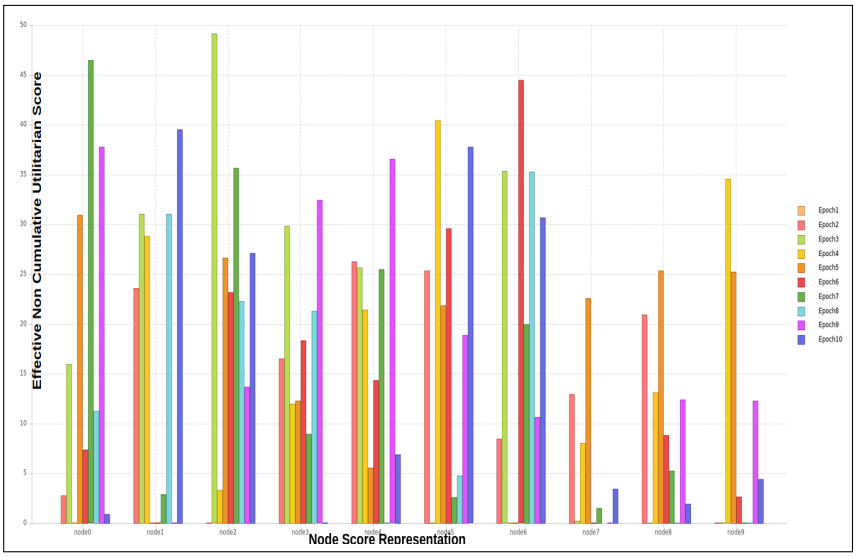

Figure 6.15: Case2: CUBA Latency Effective Utilitarian

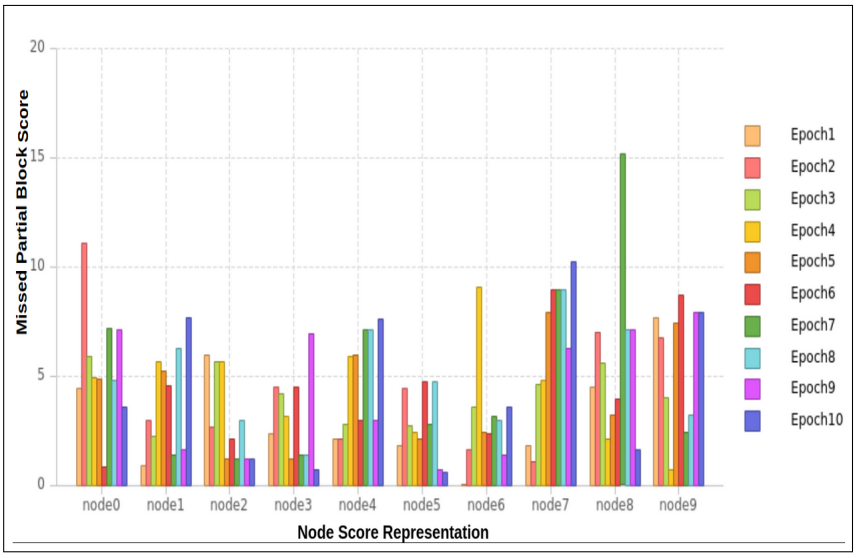

Figure 6.16: Case2: CUBA Latency Missed Partial Block

solo action of producing malicious partial blocks, while other nodes are honest. This is also further evidenced in the case of Figure 6.24 where Nodes 7, 8, and 9 have a high negative score of missed commit utilitarian by their conscious malicious behavior. This purported behavior affects their classification as they are suspended for 1 epoch and continuously hover between weak and fair utilitarian. Apart from these nodes, Node 4 progresses from a Weak Utilitarian to an Ideal Utilitarian and Node 0 from a Fair to an Ideal Utilitarian due to the fairness in the CUBA protocol.

6.2.4.4.4 Case4: Malicious Full Block Latency Scenario Next is a lighter but relatively impacting adverse action by the proposal of malicious Full Block during the inter-quorum phase by Node 7, 8, and 9 and a communication latency of 200 milliseconds. Its performance is represented in Figure 6.25 where a V-shaped recovery of around 2000 trans-

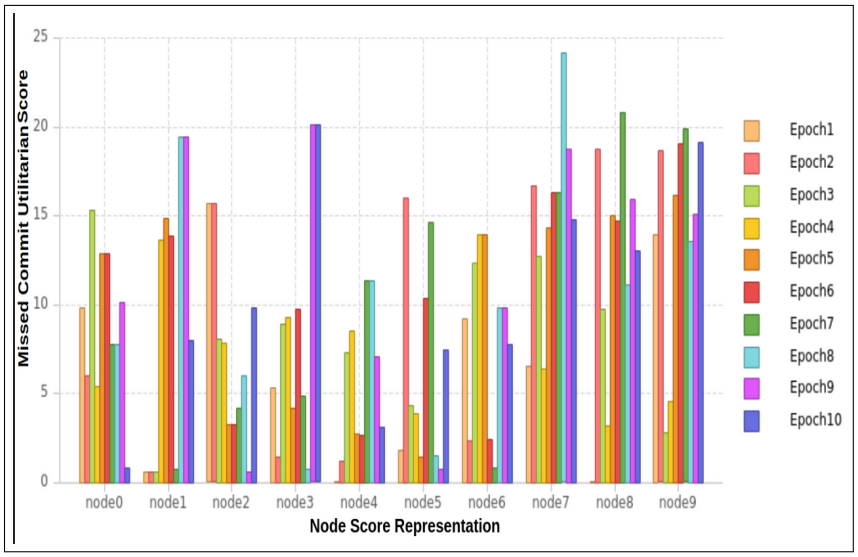

Figure 6.17: Case2: CUBA Latency Missed Commit Utilitarian

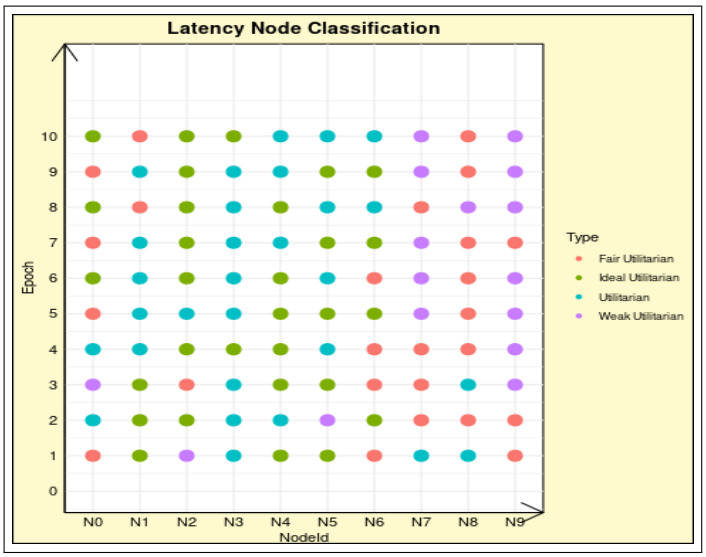

Figure 6.18: Case3: CUBA Normal Benign Latency Node Utilitarian Classification

actions per second is noticed due to the counterbalancing action by the Block fulfilled, which is chosen to replace a full block index at fixed intervals. Similar to the previous adverse scenarios, the Nodes 7, 8, and 9 are affected in their Figure 6.26 on Effective Utilitarian, Figures 6.27 and 6.28 by due to their malicious as well as latency behavior. Malicious action is represented in Figure 6.29 for Nodes 7, 8, and 9 by their dual negative actions of malicious full block and latency.

Regarding Utilitarian classification represented in Figure 6.30, the nodes 7, 8, and 9 lie in their least range of Weak Utilitarian and get suspended for 1 Epoch. In contrast, Nodes 0 and 4 can progress from Fair Utilitarian to Utilitarian level, which is a positive factor. In the result comparison of both the malicious scenarios of partial block during Intra-Quorum consensus in Figure 6.14 versus Full block during Inter-Quorum phase in Figure 6.19, we

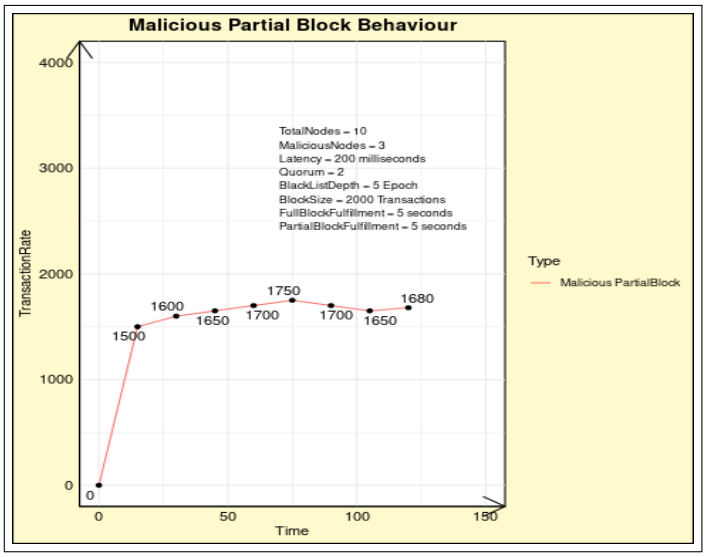

Figure 6.19: Case3: CUBA Malicious Partial Block Latency Node Performance

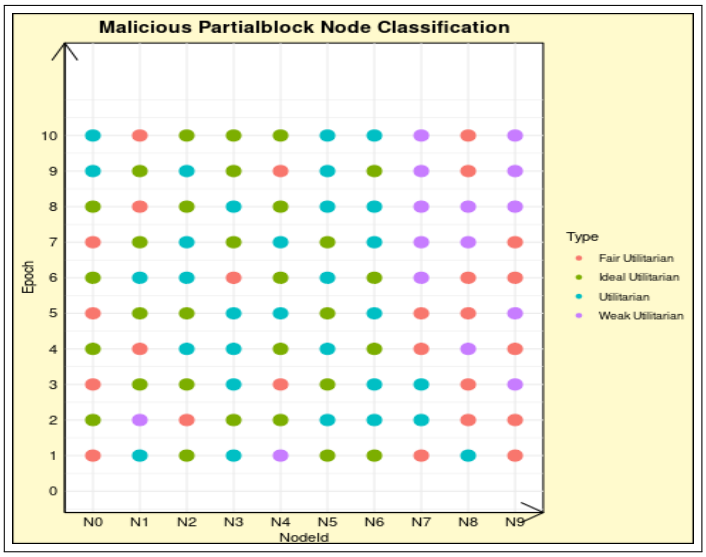

Figure 6.20: Case3: CUBA Malicious Partial Block Latency Node Utilitarian Classification

observe the influence of failure of partial blocks degrading the performance to be more. The higher penalization of the consensus throughput is more in the former as it is a vital unit in the blockchain that can break the ephemeral chain. The proposal of a finalized malicious partial block at the end of 3 sub-phases makes the previous 2 sub-phases of Propose and Commit with 2/3rd of quorum votes received for the partial block round consensus futile. But in the case of a full block, it is a single sub-phase of the proposition that can be compensated discreetly by a block fulfiller.

6.2.4.4.5 Case5: Berserk Malicious Full Block Latency Scenario Next, we measure the resilience of CUBA for the Berserk node behavior of Nodes 7, 8, and 9. We design the berserk malicious behavior in the form of sinusoidal frequency where they propose a malicious full block and add latency of 200 milliseconds in the range of 50-100 and 150-200

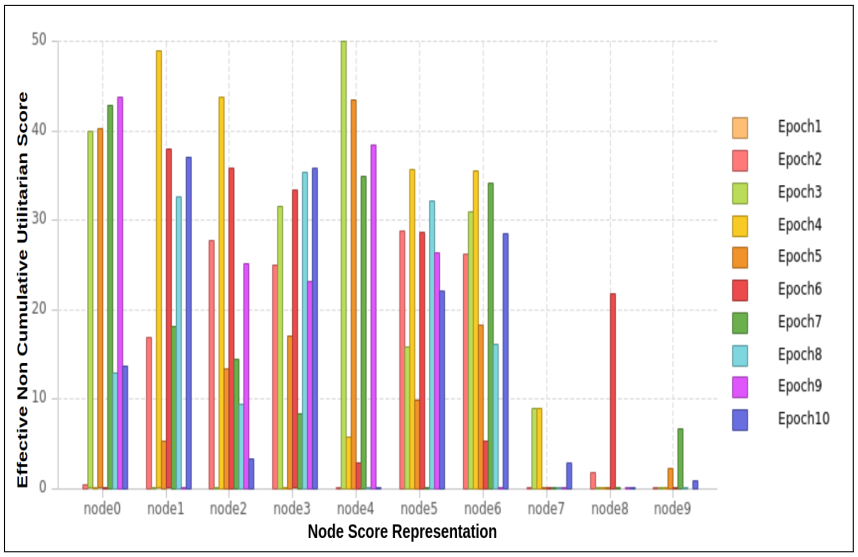

Figure 6.21: Case3: CUBA Malicious Partial Block Effective Utilitarian

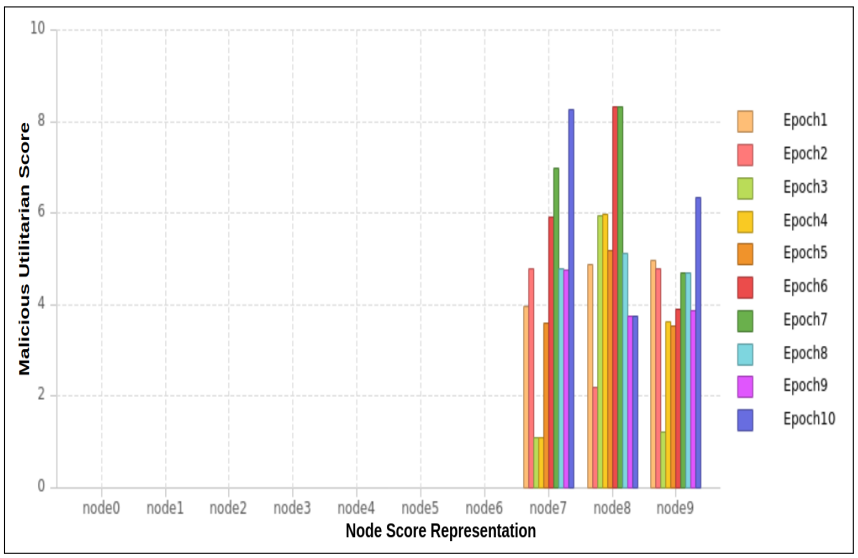

Figure 6.22: Case3: CUBA Malicious Partial Block Malicious Utilitarian

block heights, and for the remaining intervals of 0-50, 100-150, 200-250 they remain honest as represented in Figure 6.31.

As noted in their throughput performance in Figure 6.31, we have a period of crests and troughs indicated by the period of honesty and dishonesty. As expected, we have a sinusoidal effect of Nodes 7, 8 and 9 on their utilitarian score as represented in Figure 6.33 for Effective Utilitarian, Figure 6.34 for missed partial block, Figure 6.35 for missed commit and Figure 6.36 for a malicious full block.

In the representation of Utilitarian classification as in Figure 6.37, we see that despite other scores having a sine wave effect, the Nodes 7, 8, and 9 are always in the lower Weak or Fair Utilitarian and are unable to conceal their activity easily despite their intermittent malicious action. This points out the resilience of the protocol and the capability of it to identify

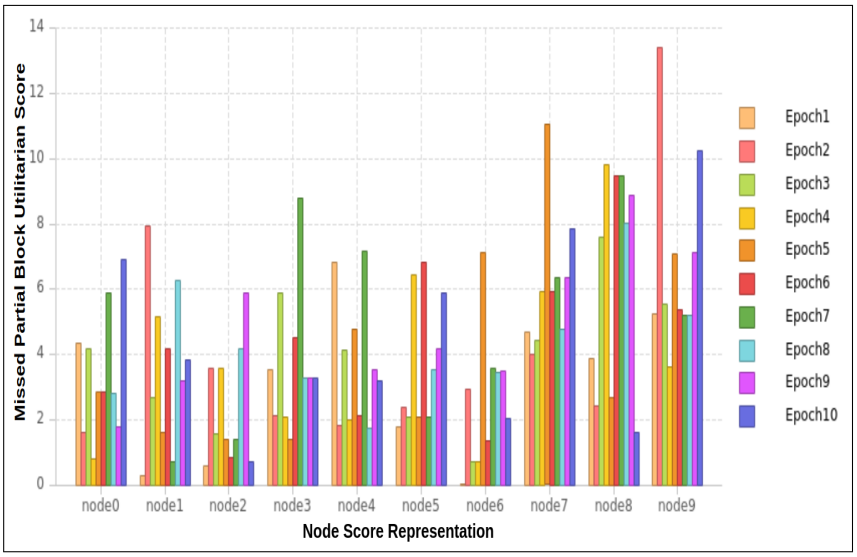

Figure 6.23: Case3: CUBA Malicious Missed Partial Block Utilitarian

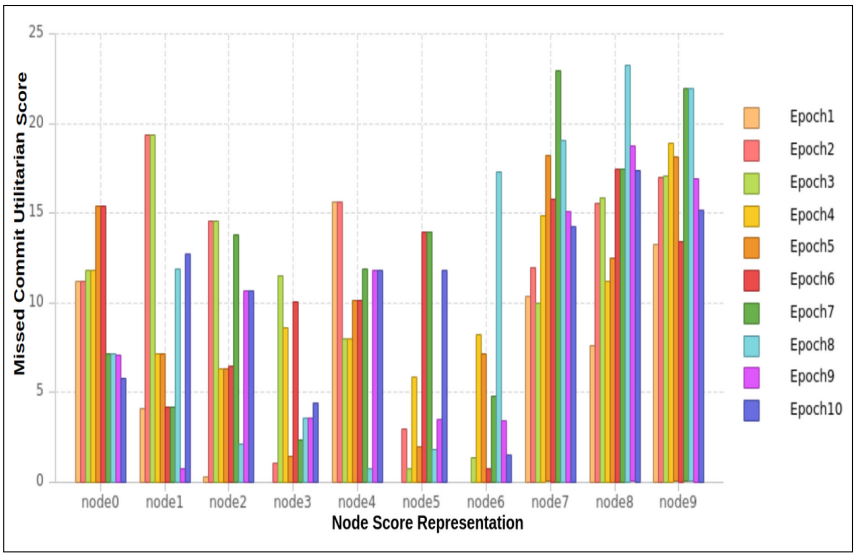

Figure 6.24: Case3: CUBA Malicious Partial Block Missed Commit Utilitarian

and suspend a bad actor. In comparison to the continuous malicious behavior throughput performance as in Figure 6.25 versus the berserk or sine behavior performance in Figure 6.32, the latter has a slightly higher performance as dishonesty is intermittent. Still, the CUBA identifies the weak utilitarians to optimize the network.

6.2.4.5 How is the performance of CUBA Implicit Variant?

This section tests the alternative variant we proposed in the previous chapter as part of our protocol definition of CUBA Implicit in Section 5.3.2.3.2. Its performance is identified in Figure 6.38, where we compare the vanilla CUBA protocol against the implicit variant of fully honest nodes.

As implied by the throughput performance analysis in Figure 6.38 we notice a slight per-

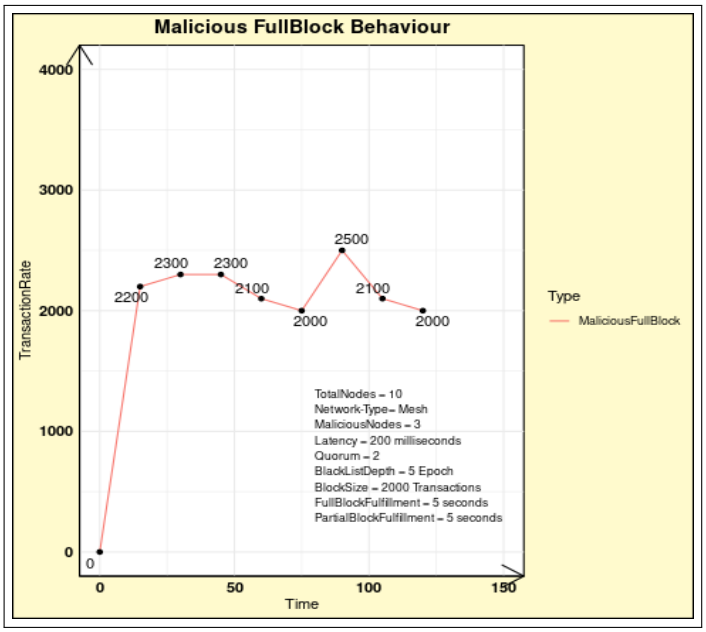

Figure 6.25: Case4: CUBA Malicious Full Block Latency Node Performance

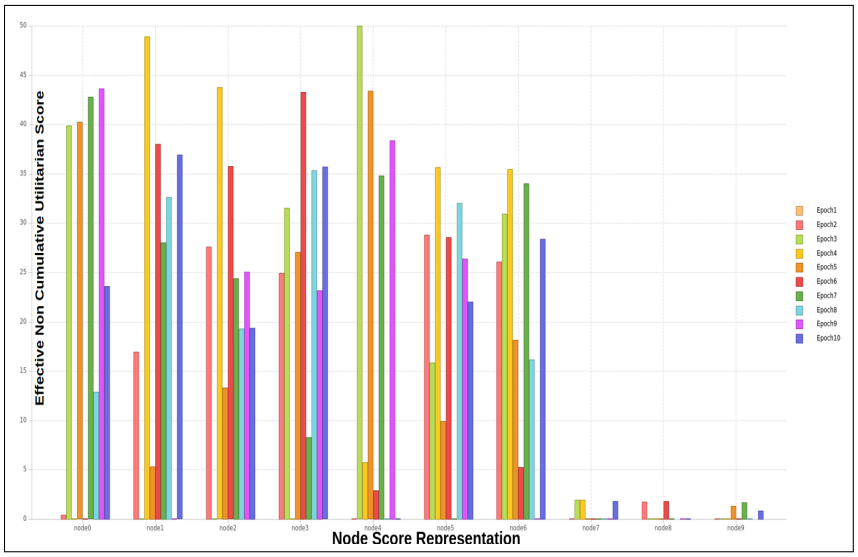

Figure 6.26: Case4: CUBA Malicious Full Block Latency Effective Utilitarian

formance improvement in the variant. The following factors can explain this:

- Implicit Variant reduces the Intra-Quorum communication sub-phases from PROPOSE, COMMIT, and FINALISE to a single FINALISE sub-phase broadcasted to the Intra-Quorum members. This implicit enveloping of the three sub-phases by a single sub-phase reduces communication complexity to O(K).
- Advantage of lesser communication bottleneck affects the synchronization and liveness as discussed earlier in Section 5.3.2.3.2, which is a tradeoff factor.
- As noticed in Figure 6.38, the difference in performance gain widens as we measure it

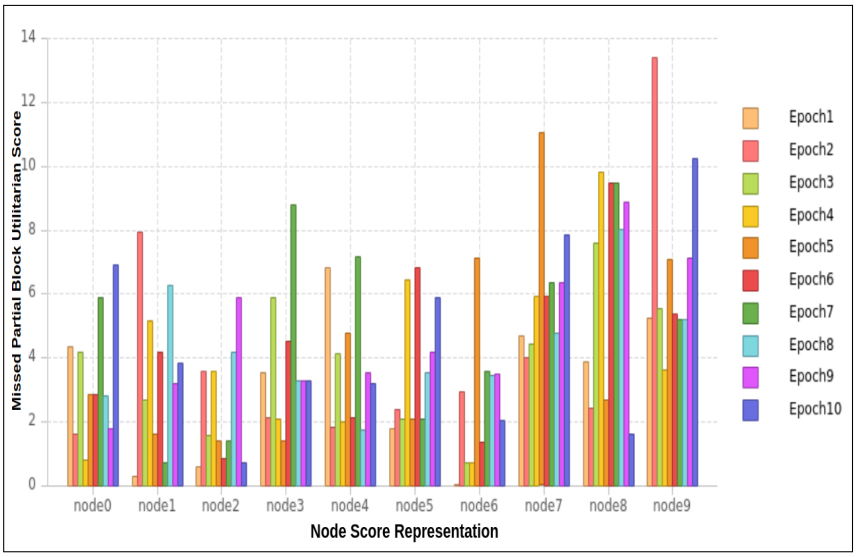

Figure 6.27: Case4: CUBA Malicious Full Block Latency Missed Partial Block Utilitarian

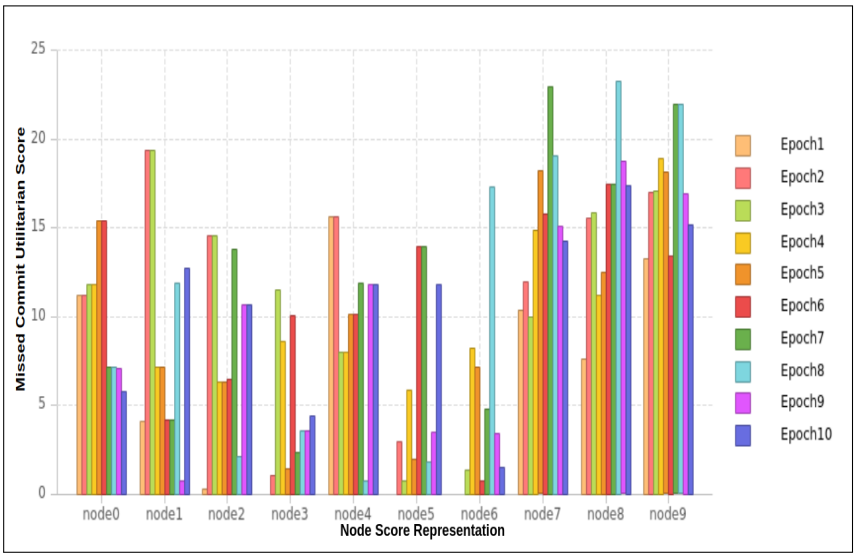

Figure 6.28: Case4: CUBA Malicious Full Block Latency Missed Commit Utilitarian

relative to the scalability of nodes which increases the magnitude of communication bottleneck higher for CUBA vanilla compared to the Implicit variant.

6.2.4.6 Distributed Denial of Service

Distributed Denial of Service (DDOS) Attacks are considered in this work at all levels from the network level implementation to the CUBA consensus level. CUBA can be resistant to this attack by adopting an intrinsic solution based on the Utilitarian score to identify the compromised node participant. Also since our transaction processing involves codepoint operation to assign transactions to a particular Quorum it acts as a limiting factor to the incoming transaction messages. Further, our consortium network of verified participants acts as an authentication mechanism to avoid any anonymous unverified transactions. A

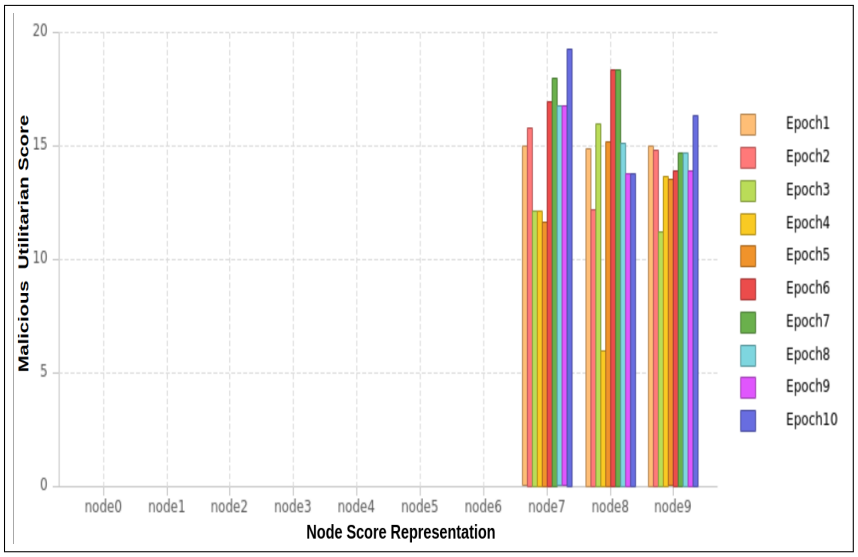

Figure 6.29: Case4: CUBA Malicious Full Block Latency Malicious Utilitarian

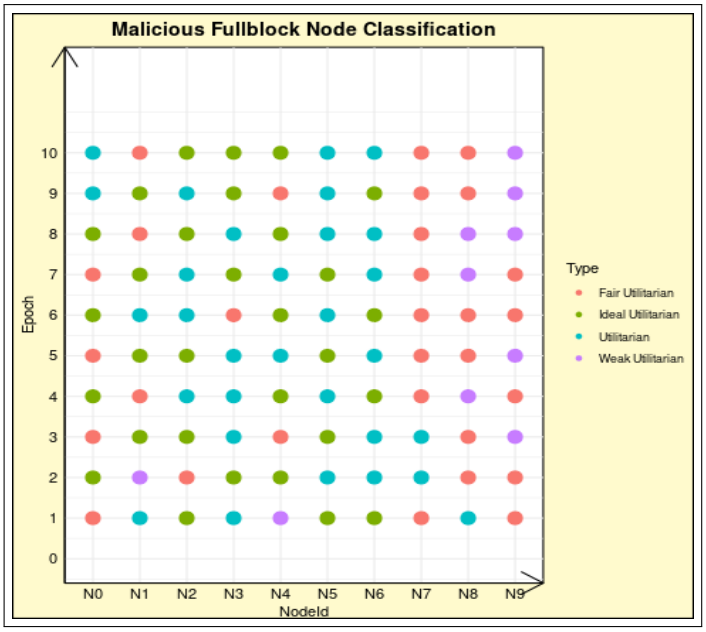

Figure 6.30: Case4: CUBA Malicious Full Block Latency Node Utilitarian Classification

more extrinsic solution can be extended at the network level [257] like traffic filtering, rate limiting, or Anycast technologies. The following techniques can be adapted to be DDOS resilient:

- Rate Limiting: to restrict the number of requests a user or IP address can request.
- Traffic Filtering: Firewall, Intrusion prevention system.
- Loadbalancing / Anycast DNS: Distribute incoming traffic across multiple servers or nodes
- Utilitarian tracking: Based on individual identification can be disincentivized from

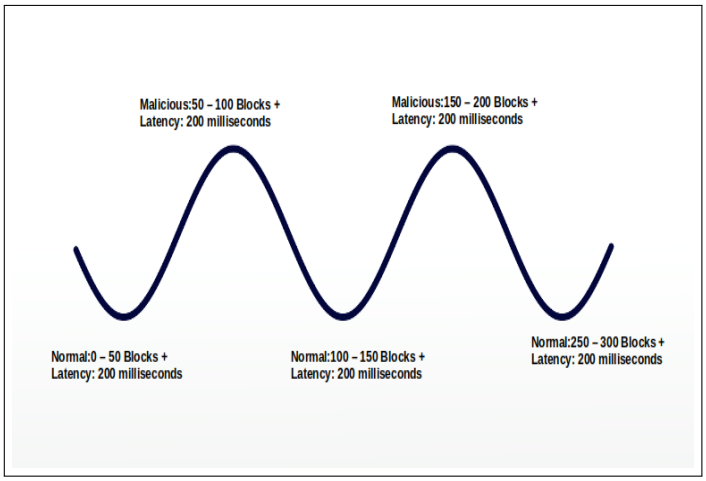

Figure 6.31: Case5: CUBA Malicious Full Block Berserk Behaviour

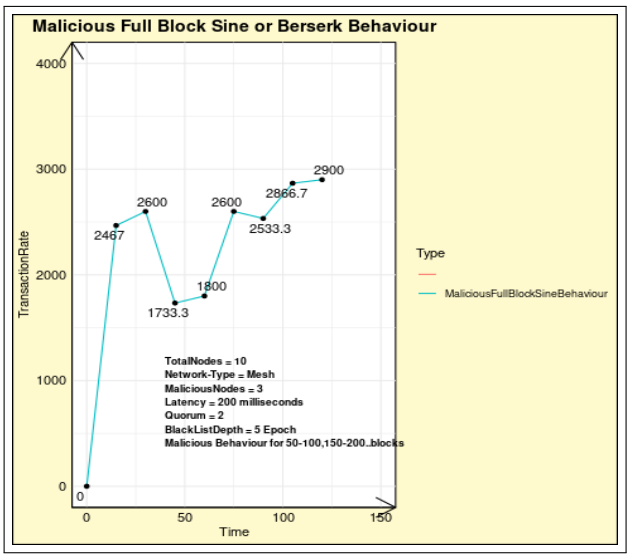

Figure 6.32: Case5: CUBA Malicious Full Block Berserk Behaviour Performance

issuing transactions

- Smart Contract Based systems: To blacklist IPs which aim to sabotage the network by emission of transactions

The processing of transactions is based on Mempool as represented in Figure: 6.39 where multiple threads in parallel are created for Transaction Message Pool, Commit, Propose, Finalise, and Full Block Message Pool. It aims to utilize the memory pool for faster and independent processing of different consensus message transactions. Also, the check for duplicate messages when processing the transaction in the data structure volatile Map<String, CopyOnWriteArrayList<String» proposeMessagePool corresponding to the block or message hash avoids the creation of transaction bottleneck and transaction queue overflow.

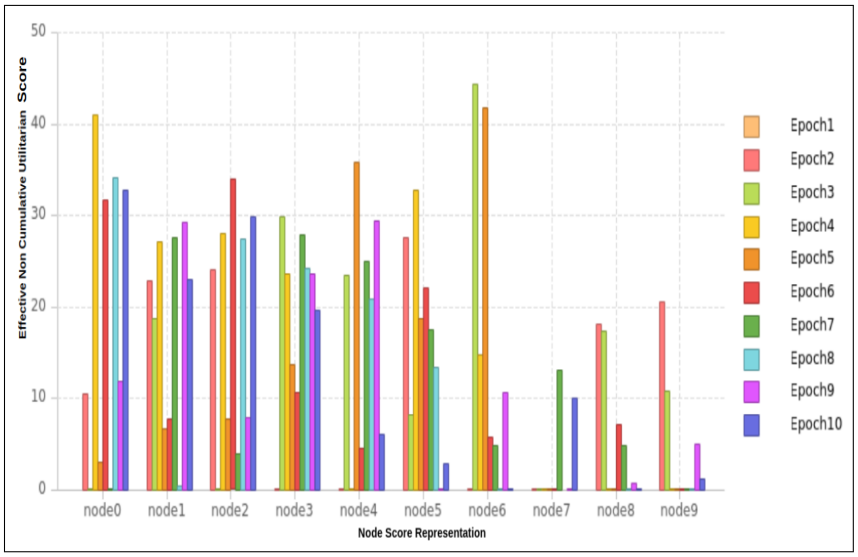

Figure 6.33: Case5: CUBA Malicious Full Block Berserk Behaviour Effective Utilitarian

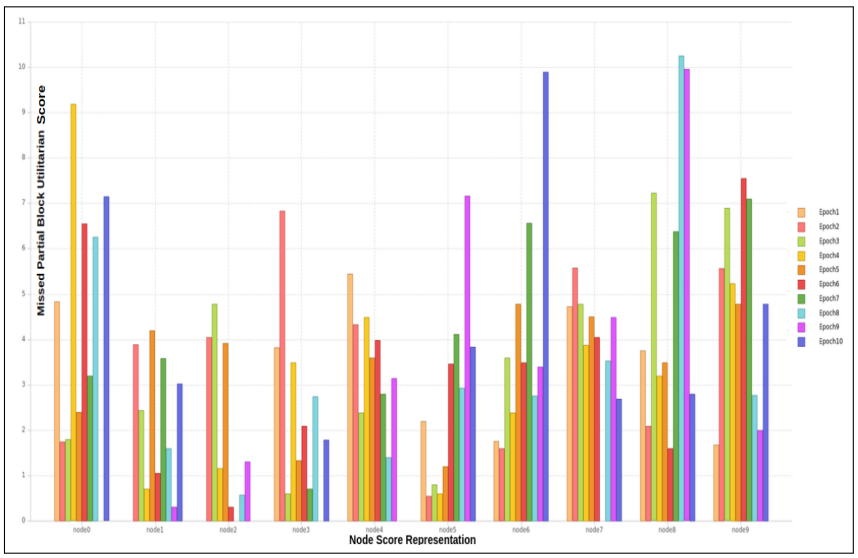

Figure 6.34: Case5: CUBA Malicious Full Block Berserk Behaviour Missed Partial Block Utilitarian

6.2.5 Overall Classical BFT Comparison

We compare the CUBA protocol's throughput performance, utilitarian evolution, and resilience of the protocol through our previous experimental evaluations against the classical consensus simulation performed in Chapter 4 and Section 4.2.3. We represent the results in a consolidated Figure 6.40, which compares CUBA along with its implicit variant and other classical BFT algorithms simulated of Clique, IBFT, PBFT, and QBFT. The analysis can be discussed as follows:

- Clique shows a relatively smooth drop in scalability but not a steep drop, as it's more of a proposed block propagated to the entire network. But this single phase can introduce the presence of forks if there is a noticeable latency in the network [258,

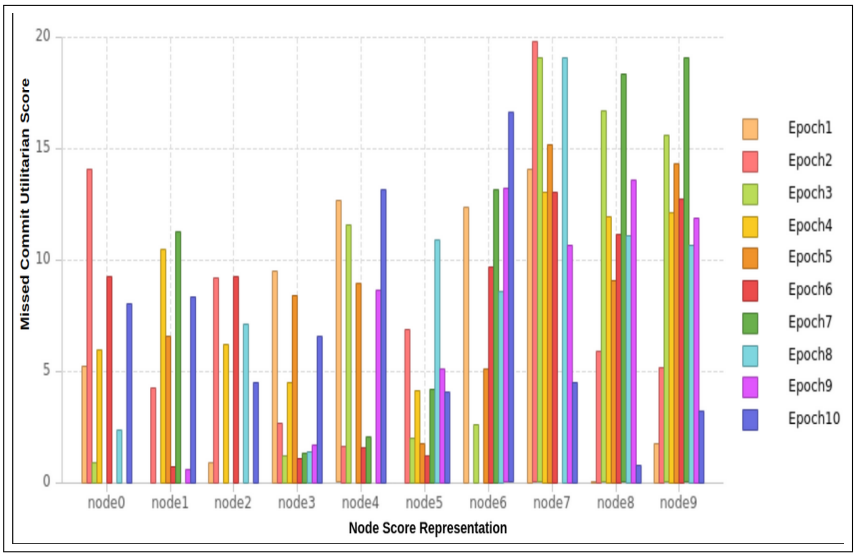

Figure 6.35: Case5: CUBA Malicious Full Block Berserk Behaviour Missed Commit Utilitarian

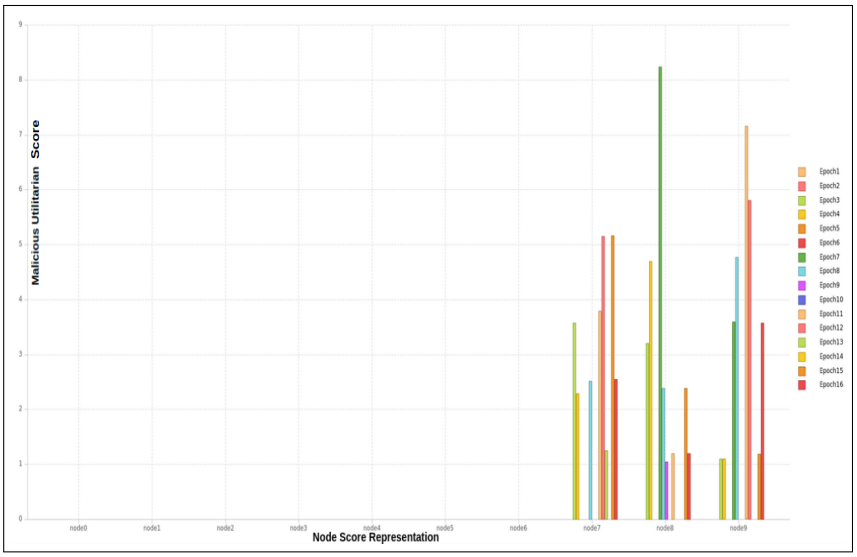

Figure 6.36: Case5: CUBA Malicious Full Block Berserk Behaviour Malicious Utilitarian

158].

- PBFT with 4 phases of PRE-PREPARE, PREPARE, COMMIT, and ROUND CHANGE augments the message complexity proportional to the number of nodes.
- IBFT is a phase-reduced version of PBFT with PREPARE, COMMIT, and ROUND CHANGE. Round change occurs only in case of a liveness issue, which is better than PBFT but can be prone to duplicate block propositions at the same height. This issue is already acknowledged [259] and needs to be solved by block locking.
- QBFT is similar to IBFT but scales better as the block-locking mechanism is removed and replaced by selective round change.

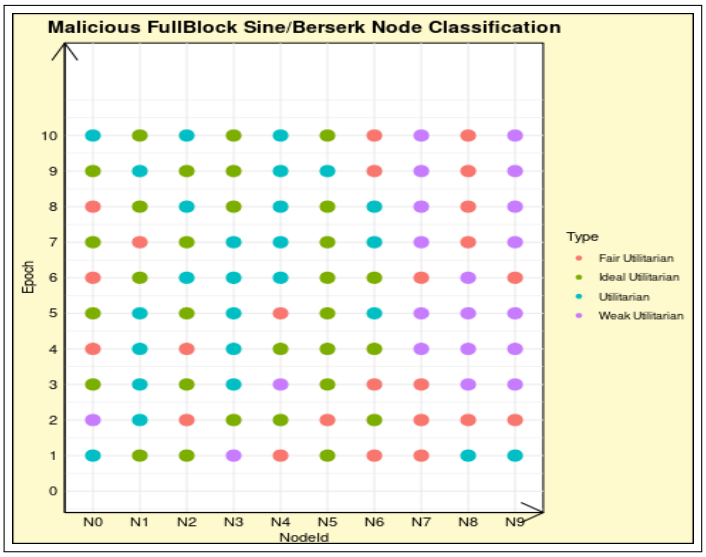

Figure 6.37: Case5: CUBA Malicious Full Block Berserk Behaviour Node Utilitarian Classification

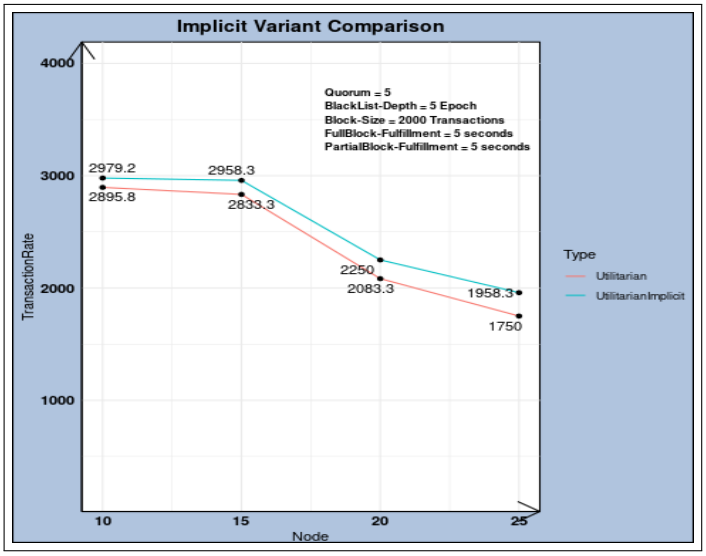

Figure 6.38: CUBA Implicit Variant Analysis

- CUBA Vanilla performs more than its BFT protocol peers except for Clique. Clique is a single-phase consensus algorithm that has the problems of Forks, Chain Inconsistency, lacks chain finality, and is prone to network deadlock, as we noticed in our previous tests of the Data certification use-case as well as through simulation.

  Moreover, the idea of designing Clique as specified by the Geth Ethereum Team is towards Kovan or Rinkeby test-net for faster block processing and testing the smart contracts faster before deploying in main-net Ethereum networks. It doesn't assure the same properties of a classical BFT algorithm of Consistency, Availability, or Partition Tolerance.

  CUBA protocol offers security and resilience against malicious actors by evolving

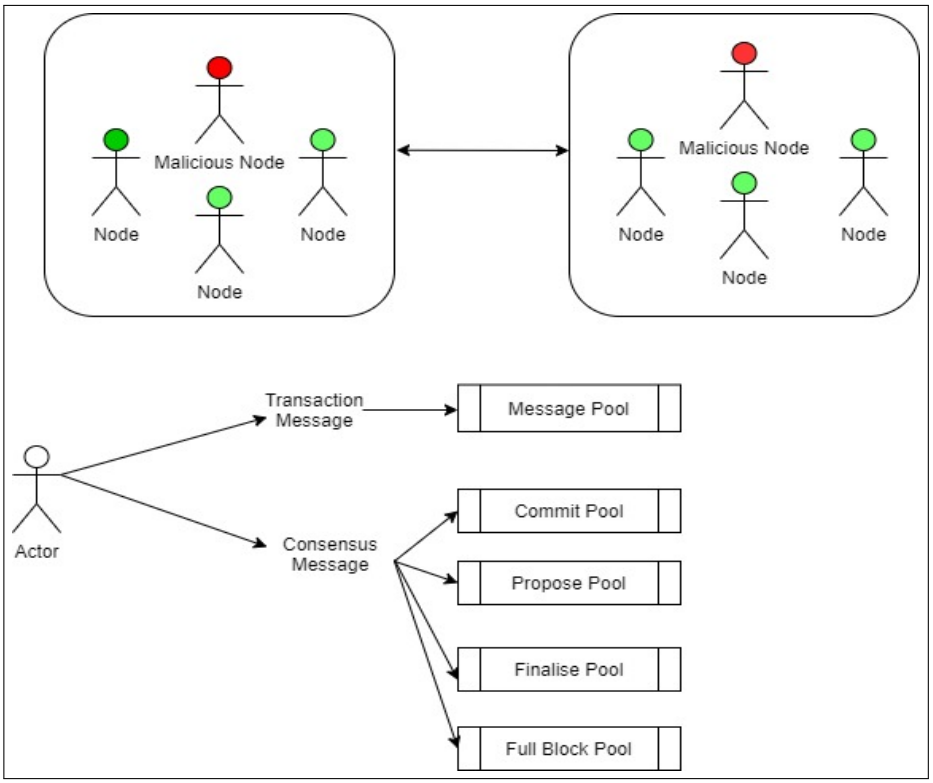

Figure 6.39: CUBA Distributed Denial of Service

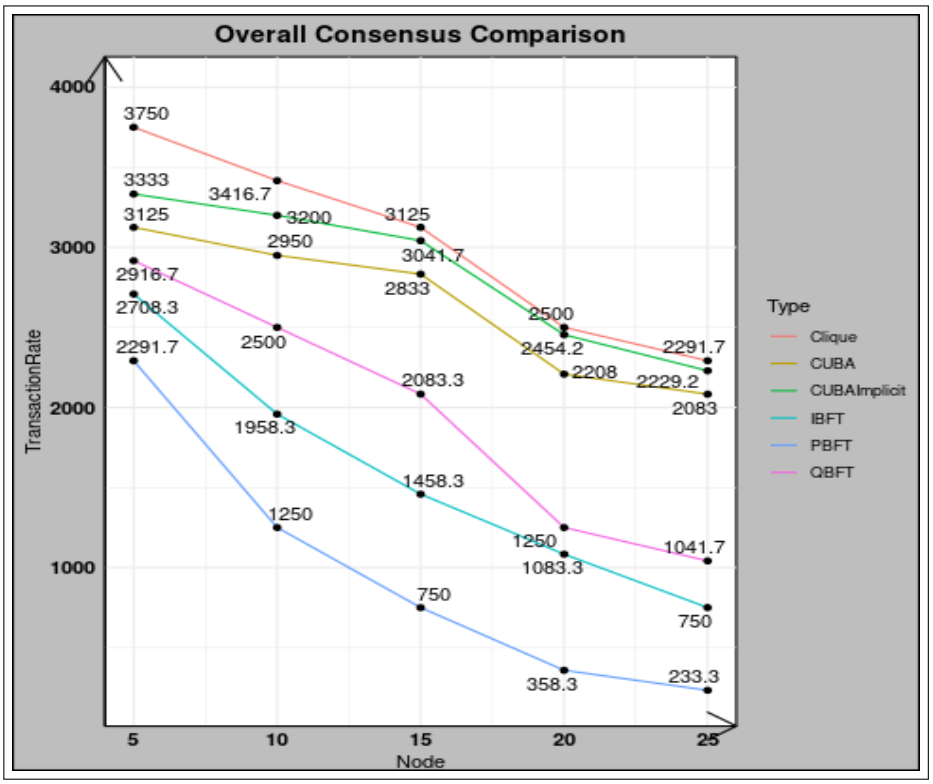

Figure 6.40: CUBA Overall Comparison Analysis

its network, which is not the same case for Clique as it cannot handle attacks. So a direct comparison between CUBA, CUBA Implicit versus Clique cannot be done only in terms of performance throughput. As noticed, CUBA for 15 Nodes, sustains the throughput performance for 6 Quorums but slips around 2000 at the same setting. This is relatively better than PBFT, IBFT, and QBFT.

- CUBA implicit variant performs close to Clique and can offer finality but can suffer synchronization issues which we discussed earlier, but still considerable if we can guarantee network performance, especially for fiable consortium networks, as this is more of an intrinsic requirement and can be obtained rather than protocols dependent on extrinsic Token Assets or Hardware requirements.

6.3 CUBA amongst recent BFT consensus protocols

In this section, we compare how CUBA fares in the universe of BFT consensus protocols where there is no shortage of contributions by the computing sorority or fraternity. In the earlier section on Overall comparison 6.2, we considered the classical BFT protocols of PBFT, IBFT, QBFT, and Clique. We conclude by comparing the recent works in the field of BFT consensus protocol innovation to understand the position of CUBA in the landscape. We analyze and compare our work's results against the results of recent works. It may not be an ideal comparison, but we would like to acknowledge the recent innovations to improve our observations. It is discussed as follows:

- **HotStuff** [83] is well known for its linear message communication using a threshold signature scheme and solves the hidden lock problem. Hidden lock is a liveness problem where a leader doesn't wait for all honest nodes' responses to get the proper highest lock. It is solved by adding a precursor round to the view change. It operates in phases of Prepare, Prepare-Commit, Commit, and Decide with a view change. Its view change is linear, but in the case of adverse conditions, it needs extra rounds to finalize the correct value, which makes it quadratic.

  In comparison to CUBA, HotStuff has a higher throughput and scalability performance. Still, the test results seem to be Kops /sec, translating to operations per second as per our assumption. This differentiates how we measure our performance as transactions per second finalized by block construction. The idea of linear communication through threshold signature is also a step we have placed in our future works. However adopting threshold signatures can add a bottleneck, as decoding the signature message for each individual node is expensive. This protocol doesn't address the resilience against malicious attacks as experimented in our CUBA consensus.

- **MirBFT** [94] is another innovation in the case of BFT protocols which allows parallel leaders simultaneously and avoids duplicate transactions through hash space division buckets. It gives an intuition for parallelizing the block creation, which is otherwise sequential with monolithic processing. CUBA handles the acceleration of the transaction processing differently: pipelining, hash codepoint sortition of a transaction, and multiple quorums. It has a throughput of 60000+ req/secs, similar to Bitcoin-like transactions, which is quite appreciable.

But when we reflected on CUBA for change in the data structure as partial blocks agglomerated to blocks divided into an ephemeral and finalized chain, we stumbled the chain creation as the primary bottleneck in parallelization. As the chain of the previous block is necessary to form the interconnection for future blocks, it cannot be finalized until its precedent is finalized. We are not sure if the implementation and evaluation of this protocol were considered during their design with the inclusion of adversarial resilience. It is always the case for Chained-BFT protocols like HotStuff or Streamlet, which is somewhat relaxed in the case of Directed Acyclic Graph (DAG) based Distributed Ledgers [260] to be discussed in the next section.

- **GOSIG** [231] is a scalable BFT protocol that capitalizes on a gossiping network to broadcast the block in each round. Multiple blocks are passed by gossiping onto the network for a round, and the block accumulating the required signature is deemed finalized. This is an alternative way to lighten the communication in BFT protocol which depends on a single leader to collate the votes and perform the consensus orchestration, causing a bottleneck. They use the Verifiable Random function to choose the block proposer, making it unpredictable for an attacker to target a node. They rely on signature aggregation for vote collection during consensus like earlier protocols which can be a cryptographic overhead we need to evaluate before considering for CUBA. They apply a version of transmission pipelining to gather signatures or votes on the block headers before even receiving the actual blocks. They have an impressive throughput of 3000 transactions per second for 5000 nodes.

  It relies on gossip, which we can consider for CUBA but can lead to two pitfalls as conflicting blocks can be finalized for a single round, especially in the case of their pipelining feature. Also, gossiping can lead to redundancy of message communication which might impact the throughput, which we try to solve by displacing leader-based protocols. Another observation in the case of a network partition is what would be the fallback of gossip protocols for eclipse attacks when attackers mask a node's view for selfish benefits.

- **Round-based DAG BFT Protocols**: Directed Acyclic Graph-based consensus has been chosen to replace the fundamental blockchain-based consensus for faster throughput and finality of transactions. We consider several protocols closely related to each other as they all operate in round-based DAG. These are Aleph [261], DAG-Rider, Narwhal, and Tusk [262], Bullshark [263].

  In the case of Narwhal and Tusk [262], a mempool structure seeks to parallelize the transaction processing. Each node has a set of workers and primary. Each worker has a unique transaction to process and communicates with a similar worker from other nodes for consensus. If there are n workers in a single node, it communicates with n workers for different nodes to perform a parallelized consensus. Each worker, in parallel, transmits its data digest to the primary, which interacts with the primary of other nodes. The primary broadcasts the final digest and collects its certificate from other primaries, leading to construction in the form of DAG. Then a leader is elected based on a random coin for a round and interprets the causal order locally. A block is considered finalized if it has $2f + 1$ certificates and all the other associated blocks are also completed. All this is done locally for zero message communication which is

quite efficient. Aleph and Bullshark perform this local interpretation as well.

Narhwal achieves a throughput of 130000 transactions per second, Bullshark around 125000 transactions per second, which is relatively high but has some drawbacks, as acknowledged by the protocol developers. It cannot distinguish between honest and dishonest who delay the system by adding latency inadvertently, which is a pitfall. Also, a client needs to resubmit the transaction as in DAG there is a high risk of losing if the transaction processed does not fall in a data unit that has high certificate attestations. Narhwal relies on inter-process communication between worker threads, which makes retrieving transaction data in case they are lost cumbersome and expensive. Lastly, these protocols require a transaction engine as efficient as the protocol to be developed in the future. DAG system also has a problem with account maintenance of nodes, and conflicting transactions are difficult to counter and costly as we need to traverse the entire graph again.

- **CAROUSEL** [264] is a BFT protocol with about 70000 transactions per second, improving the leader selection process during the consensus. It tends to avoid crashed leaders in a crash-only execution system. Assuming the protocol operates in round $r-1, r$, and so forth, selecting a leader for round r should be based on the condition they should have endorsed in the preceding round $r-1$. But among this selection, they remove f latest leaders of the committed blocks to avoid the byzantine actors. This achieves leader utilization, which limits fault leaders being chosen in the network and improves the chain quality, which minimizes the byzantine blocks on the chain.

  CUBA has a different methodology to reach the goal of leader utilization and chain quality by measuring the utilitarian score from all distinct actions in the previous round and throughout the chain's history with a Sisyphus forgetting and fairness coefficient.

6.4 Future Work

We propose in this section certain skeletal mutations for CUBA in the future, which we have considered for now. They are listed as follows:

- **Dynamic Quorum Size:** BFT protocol literature community constantly desires to improve the overhead of message communication [265]. In line with this, CUBA is no exception which can be achieved by having active quorum participation. Assuming that our network has ρ quorums in the network at any given Epoch E, then at any time during the epoch, we can allot ρ/A where $A \in \mathbb{N}$ which is a set of natural numbers. This allows a sub-sample quorum out of ρ quorum members to participate for certain block intervals and then choose a new set of active quorums. This is quite intuitive in lessening message communication as it can randomize selection while having the inherent BFT properties of CUBA.

- **Utilitarian Weightage Assignation:** Each finalized block carries an effective utilitarian score attributed to the participant after the finalization. This effective score can be accumulated to calculate the total effective utilitarian of the chain. It can help in

resolving forks similar to GHOST [89] protocol where arbitrary weights are applied. Also, the utilitarian score can add weight or credibility to a consensus message for a lighter verification system, bypassing the mandatory checks that are repetitive for faster processing.

- **Optimised Selection of Proposers:** Since, in CUBA, each participant has a utilitarian score which one gains during the consensus process and the choice of partial block proposer, competitive partial block proposer, quorum proposer, fulfiller is independent of one's score. It is based on the codepoint of hash in the current proposition, which can be combined with the utilitarian score to choose from certain whitelisted members with a higher network score. It is to be at the cost of fairness but can be an optimization factor.

- **Linear Authenticator Complexity:** In a work by Zhang, Yupu and Dragga, Chris and Arpaci-Dusseau, Andrea and Arpaci-Dusseau, Remzi [267] they propose the notion of no commit proofs. This improves the Authenticator Complexity, which calculates the number of signatures or votes needed for single-phase finalization. They apply this to the view change phase of their consensus protocol Wendy and on HotStuff, making it linear. In the normal case of HotStuff [83], the view change is linear, assuming non-byzantine actors who do not consider false positive Quorum Certificates (QC) and choose the highest value arbitrarily. But this is an ideal case, and it falls back to quadratic complexity, which the authors improve by encoding the QC as a single data structure that needs a linear authenticator for a phase. It is done by performing a BLS Multi-Signature [268] on the difference of QC. Since in CUBA, we perform parallel finalization of multiple blocks, and we can combine different messages across different block heights from a single validator to lighten the message payload in the network. Any validator can then decode this to retrieve the unit of messages to apply each in their particular block height.

- **Fairness of Ordering:** In BFT protocols like PBFT or CUBA, we have a receive order fairness which does not apply any specific ordering protocol on the transactions. This usually depends on the leader who receives the transaction, and the network assumes it to be the order. In the work of Kelkar et al. [269], the authors propose a partial synchronous leader-based BFT protocol that performs order agreement on the transactions. It consists of the Gossip, Agreement, and Finalisation stage in ordering the transaction. It is done by each node broadcasting a partial graph of its own causal view on transaction order to a leader. The leader then receives the partial graph from a unified transaction order graph. In the case of cyclicity or Condorcet paradox [270] in the graph, the strongly connected components are separated, and it is resolved. We can apply the same construct to the CUBA protocol to order the transaction in the partial block among the quorum members by the partial block proposer or finalizer. This can bring a canonical order to the transaction, rendering it more robust for financial applications.

6.5 Conclusion

This chapter evaluated the Contesting Utilitarian Byzantine Agreement (CUBA) protocol from a multi-dimensional theoretical and experimental metrics perspective. We analyze each phase of the algorithm based on the CAP theorem, Scalability Trilemma, Fairness, and adverse scenarios. We further investigated the experimental implementation in a public cloud environment and measured its scalability and resilience with simulated adverse conditions. We compared the results bench-marked against classical protocols and the recent state-of-the-art BFT protocols with promising results and improvisations. The CUBA protocol can be concluded that in an ideal quorum-to-member ratio, it shows a more acceptable scalability condition with the recovery due to quorum optimization at the end of each epoch. In the case of advertent behavior, as discussed earlier, the nodes can ensure resilience by ensuing suspension, reorganizing the network, and ensuring the liveness as well as the performance of the network.

Chapter 7

Conclusion and Perspectives

The beginning of wisdom is found in doubting; by doubting, we come to the question, and by seeking, we may come upon the truth.

– Pierre Abélard

In this chapter, we analyze our thesis work from the perspective of strengths and opportunities to conclude our work and suggest future directions that might enrich the work further.

7.1 Conclusion

In our thesis work, we traversed the following iterative pathway where a question from the genesis of the work leads until the modest conclusion of our work with space for improvements as follows:

How do we develop a scalable and optimized Mobility Service Implementation using DLT? Is the choice of the BFT Consensus Algorithm from a consortium perspective suitable for our solution?
This has been the principal question of the thesis. It led us to formulate our initial hypothesis by studying the various techniques and architecture available in the literature as a first step to analyzing its positives and negatives. We further designed a mobility solution based on data certification and monetization. They were individually implemented and evaluated using the Ethereum and Substrate platforms. The BFT consensus algorithms of Clique, PBFT, IBFT, and QBFT were studied in the process. In order to evaluate them uniformly, a blockchain simulator was developed, and the algorithms were studied on a homogeneous platform. They exhibit scalability issues due to the inherent nature of message communication bottleneck in BFT protocols and consistency issues. This led us to develop the CUBA consensus protocol which solves the issue by pipelining and reducing message communication through intra-quorum and inter-quorum phases. The node participant's behavior in CUBA is evaluated using a utilitarian mechanism and adapted based on its positive or negative behavior in the previous epochs.

What is the current landscape of DLT-enabled Mobility Services and the Distributed Consensus State of Art?
This chapter on the State of Art 2 explored the architectures of Mobility as a Service (MaaS) complemented by DLT technology, use-case on vehicle sharing, data-market strategies, and tokenization-enabled mobility economy. Challenges to these solutions in security, privacy, fairness, encryption schemes, and incentivization mechanisms were also studied. In a deeper dive into DLT technologies, we backtracked the consensus algorithms since the age of distributed systems. From Byzantine General's Problem and Practical Byzantine Fault tolerance until more recent works dedicated to blockchain protocols like Hotstuff, and Tendermint were explored in this study. We analyze the problems solved by each of the protocols as well as the bottlenecks and network suitability of each of these protocols to get a fundamental overview before we start the construction of our mobility solution.

How do we build a Mobility Data Certification Service in a consortium network?
In this work on Chapter 3, Section 3.1.2, we built a solution to certify the data by proposing two architectures of Data-Registry and a Non-Fungible Token approach. We implement a cloud-based solidity smart contract solution in the Ethereum technology from a consortium perspective. Evaluation from the functional and performance was carried out where we demarcated the problems concerning client binary of Geth, Besu, OpenEthereum, and consensus issues. The consensus algorithms studied are BFT Proof of Authority Algorithms of Clique, Istanbul Byzantine Fault Tolerance (IBFT), and Quorum Byzantine Fault Tolerance (QBFT). We responsibly report the binary client problems to the open-source community for further improvements. We extend the work to accidentology use-case on Hyperledger Sawtooth based solution for analyzing Practical Byzantine Fault Tolerance (PBFT) consen-

sus and study it in detail. We ascertain that the algorithms degrade in their scalability performance, fork issues, and message complexity, leading to finding an alternative solution.

Is there any alternative to classical BFT consensus algorithms-based DLT platforms?
In this part of Chapter 3, Section 3.1.3, we confront the problem further by solving a virtuous data monetization problem based on Substrate technology. Substrate enables us to implement the smart contract directly in the client binary without needing deployment post the network launch like in Ethereum base networks. Here the solutions revolve around the Hybrid consensus split into Block authoring and Block finalization. We test the Non-Fungible Tokenised economy smart contract pallet cloud solution across the pair of consensus algorithms: 1) Authority Round (AuRa) and GRANDPA (Greediest Heaviest Observed Sub Tree based Recursive Ancestor Deriving Prefix Agreement) 2) Blind Assignment for Blockchain Extension (BaBe) and GRANDPA. We notice that BABE solves the security and privacy problems by masking the block proposer beforehand, but along with AuRa, it descends gracefully in scalability performance. Although the chain forks emanating from AuRa and BABE are successively resolved by GRANDPA, its throughput scales down, further motivating us to scale up our thesis to a detailed homogenous study of BFT algorithms.

How do we homogenize our study on BFT algorithms by being blockchain platform agnostic?
In Chapter 4, To answer the above question, we developed a blockchain simulator in Java which is constructed generically, allowing plug and play of any consensus algorithm by inheriting a common interface. The motivation is to test all the BFT algorithms in a cloud-based infrastructure and dissect their work and bottlenecks. We design and construct our simulator to test Clique, IBFT, QBFT, and PBFT. It independently derives its statistics and pinpoints its multifaceted problems to communication complexity, fork issues, view change complexity, and resilience to malicious and benign failures.

How do we solve the classical BFT consensus problems uncovered?
To reply to this challenge, in Chapter 5, we propose our consensus algorithm Competing Utilitarian Byzantine Agreement (CUBA) and its implicit variant. CUBA expands to Contesting Utilitarian Byzantine Agreement which evaluates and valorizes each consensus action as a Utilitarian metric of the gamified participants in the network. The obtained utilitarian metrics are used as feedback to reorganize the network for faster performance of the network consensus or for being resilient to the malicious activity noticed. This consensus protocol is designed to sustain or increase the Utilitarian happiness in a Byzantine environment of identified participants for the network's liveness, safety, performance, and scalability. We evaluate its algorithmic complexity comprising message complexity across the three phases in intra-quorum, inter-quorum, and Network Evolution. We identify that the algorithm has an expected availability, eventual consistency, and partition tolerance from the CAP theorem perspective. We study the protocol's fairness, blockchain trilemma, and failure resilience from a theoretical perspective.

How does CUBA protocol fare in cloud implementation benchmarks, honest or malicious failures, and against its peer BFT protocols?
We test the implementation of CUBA in Chapter 6 through our simulator, erstwhile devel-

oped and deployed to the cloud infrastructure for different network participation configurations. We test the benign cases, malicious scenarios comprising a partial or full block, and berserk adversarial scenarios. We track the utilitarian score evolution and throughput performance. Evaluation results show an improved throughput, scalability, and malicious resilience compared to Proof of Authority protocols like PBFT, IBFT, and QBFT, as well as comparable to Clique for consortium Distributed Ledger networks. The implicit variant is closer to clique performance but has certain assumptions of strong network performance and is less prone to failures. In our work, we discuss the comparison of CUBA to recent prominent blockchain protocols of HotStuff, MirBFT, and Round-Based DAG BFT protocols, which present a slight upper hand for the latter.

7.2 Perspectives

In this section, we discuss the future direction which help us in evolving this thesis for further enrichment and contribution to the community. These perspectives are to be read in conjunction with the sectional future works proposed at the end of each major work of Data Certification 3.1.2.6, Data Monetisation 3.1.3.9.1, and CUBA Consensus 6.4. These are highlighted with a certain level of modesty and yearning as we are bound by the time constraints of the thesis as follows:

- **Privacy:** To further increase the level of privacy, multiple solutions like ZkSnarks, Homomorphic Encryption, as well as Multiparty Computation can be embedded in our mobility solutions to achieve different means of verification and proving of data. While we have discussed extensively Non Fungible Tokens in our Data Monetisation and Certification solutions we need to further explore them along with selective data sharing and pseudonymity concerns [271]. Accounts in the blockchain can be abstracted as proposed in ERC 4337, and alternatives like Biometrics can be considered for account authentication in the blockchain network [272].
- **Layer Solutions:** CUBA protocol aims to accelerate and increase the resilience of the native blockchain network. But specific network layers over the blockchain can be conceptualized like Offchain State Channels and Layer 2, which can perform transactions faster conducted offline, and the final proof can be recorded in state channel [273].
- **Sharding Scaling:** Sharding is the recently adopted technique of creating a shared state of the primary network called shards. Consensus is performed in parallel at each shard level and then accumulated at the whole network level, which enables faster processing of the transactions in the network. CUBA can be further extended at the level of individual shards consensus and at the global level for combining the results.
- **Open Source Adoption:** CUBA consensus is evaluated through the blockchain simulator, which can be proposed to the open-source community for further developments and implementation in a consortium or enterprise blockchain platform.

7.3 Overall Analysis

Strengths	**Opportunities**
Tokenized and Privacy By Design Mobility Data Certification **Ethereum** study.	Development of the mobility solution to a **production enterprise architecture**.
Hybrid Consensus: **Grandpa, AuRa and BABE**, along with run-time Smart Contract for Virtuous Data Monetisation Substrate study.	Benchmarking of Mobility Solution for **wider network participants** and vehicle fleet.
Study of BFT Algorithms and its characteristics: **PBFT, IBFT, QBFT, and Clique.**	Data space complexity of the **CUBA** consensus algorithm to be understood.
Conception of **Blockchain Consensus Simulator** platform for homogenized cloud testing.	Collaboration with Open Source Community for **enhancement and adoption** of CUBA protocol.
Proposition and benchmarking of **Competing Utilitarian Byzantine Agreement (CUBA)** consensus for scalability and resiliency.	**Enhance privacy** in the network at the level of transactions as well as consensus messages.

Table 7.1: Thesis Analysis

In this section, we identify the strengths and opportunities of our work as represented in Table: 7.1. We discuss them as follows:

- **Strengths:**
  - Thesis extensively discusses state-of-the-art. It proposes new mobility business models around tokenized data certification and monetization, which can be extrapolated to similar use cases.
  - Consensus of Ethereum Proof of Authority basket and Hyperledger Sawtooth are studied. The hybrid consensus of Substrate and embedded smart contract pallets are analyzed extensively, dealing separately with the block proposal and finalization.
  - Blockchain simulator platform for testing and analyzing from multiple scenarios of consensus protocols is designed and implemented.
  - CUBA consensus protocol aiming to solve the inherent problems is proposed and tested from multiple security and network configuration perspectives, contributing to a novel BFT consortium consensus algorithm. It is adequately analyzed from the theoretical viewpoint as well.
- **Opportunities:**

- Our mobility solutions and CUBA consensus proposals can be tested in enterprise-grade adaptation to demarcate infrastructure, energy consumption, network load, data storage, and legal liabilities risks.
- We have benchmarked our solutions from a consortium perspective which presents a limited scope but our security and privacy propositions are directed towards wide public network exercise.
- We can propose these solutions to the open-source community for greater participation and brainstorming to modulate for a community-grade software release.
- Since our work is scoped to Group Renault and its partners of Universite Cote D'Azur, LEAT Laboratory it will be of increasing interest to pitch this solution to the internal product team for consideration in their timeline to onboard this as a real-world solution.
- Blockchain has been transforming through parachains, cosmos hubs, or shards which might slightly contrast our solutions. Nevertheless, all our solutions can be very well integrated into the future course as these are classical contributions either of the Ethereum smart contract, Substrate pallets, or CUBA distributed ledger consensus.
- One thought that we anticipate as a nice-to-have feature would be to mutate the CUBA protocol from a normally distributed consensus to a privacy consensus protocol. It will be to perform agreement on the distributed ledger, only sharing proof and verifications instead of conventional consensus messages. This might be a distant aspiration but a research axe on it would be interesting.

Chapter 8

Appendix

The more I learn, the more I realize how much I don't know

– Albert Einstein

This chapter contains additional explanation and essential algorithms, which is necessary to read in line with the main chapters of the thesis.

8.1 Data Certification Ethereum Discussion

8.1.1 Analysis of Existing Testing Tools

This section discusses the existing well-known frameworks or tools for testing the Ethereum network from a performance perspective.

8.1.1.1 ChainHammer

It is a test suite developed in Python for creating a blockchain network, deploying smart contracts, and testing the transaction load with varied types of synchronous and asynchronous methods. It is primarily developed for PoA network clients like Geth, Parity, and Quorum.

8.1.1.2 Calliper

It is a tool developed by the Hyperledger Linux foundation with varied workloads and adapters for blockchain networks such as Geth, Besu, and Fabric. It is developed in Node.js with rate controllers for sending the transactions to the network and then observing the throughput of the blockchain network.

8.1.1.3 BlockBench

In this work [274], the authors have contributed to a framework for analyzing private blockchain via APIs against varied workloads for Geth, Parity, and Fabric. It is developed in C++ and Node.js with macro and micro smart contract benchmark mechanisms.

8.1.2 Proposed Testing Tool Architecture

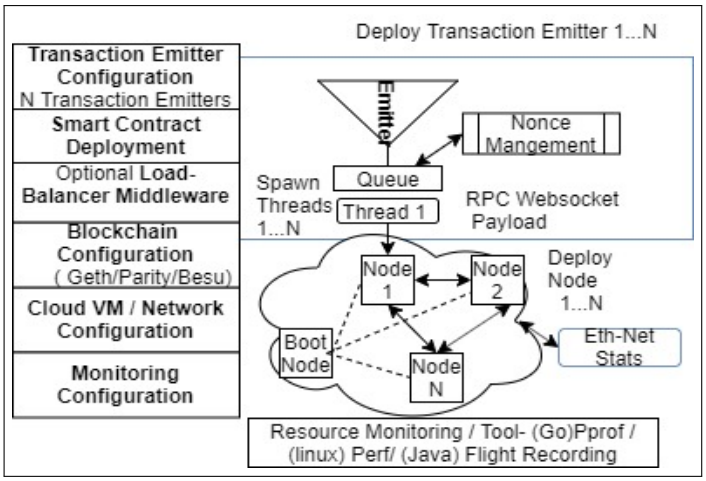

Figure 8.1: Proposed Testing Tool Architecture

Having studied the above various works, we wanted to remove the drawbacks and perform a comprehensive and uniform performance test. The main issues noticed with the earlier

tools are solved in our proposed tool as in Figure 8.1. Features of our tool are:

1. **Transaction Emitter Pattern:** The Asynchronous Multithreaded Producer-Consumer pattern of sending transactions is better than other tools as it effectively controls the client load. Each client has an RPC API interface for sending transactions. To test the load limit of each client binary, it was necessary to be fired as fast as possible in the multithreaded version. To control the rate of transaction firing and reliability, we placed the request in the queue to be worked upon by processes in a producer-consumer model.

2. **Nonce Management:** As per Ethereum Improvement Proposal 155, each transaction must have incremental nonce to avoid a replay attack. The nonce for each transaction is managed by the tool in incremental order, which is not the case for other tools or works. Proactively sending in the transaction payload speeds up the transaction processing and avoids duplicate nonce transactions or unordered or gapped invalid transactions. The tool effectively calculates the valid nonce by mutually excluding the nonce increment process, even when multiple transactions send instances attached to a single blockchain node to avoid duplication.

3. **Choice of Communication Protocol:** The third most important factor to consider was the protocol choice for RPC API, where raw web socket performs much better than raw HTTP or commonly used web3 libraries. Each HTTP request opens a new connection, whereas, in a web socket, the connection is maintained until explicitly closed with only a handshake at the beginning to open a new connection. Also, the raw web-socket connection used in our tool is more efficient than the web3 library wrapper used by other means to avoid bottlenecks.

The proposed tool consists of a monitoring layer for our testbed comprising of pprof for Go (Geth), Perf for Linux (OpenEthereum), and Flight Recorder for Java (Besu). Above this layer, we have the cloud network configuration layer developed using Terraform to deploy Azure VMs and connect them through an overlay network. The next layer is the blockchain part using shell script and Python to configure the network and genesis block to deploy the required nodes in the network. In the next layer, we developed a load-balancer middleware in the tool for uniform and reliable transaction distribution, which will be explained in section 8.1.10. We have a smart contract deployment layer depending on the load needed and the client layer for sending transactions.

8.1.3 Evaluation of Existing Testing Tools against Proposed Testing Tool

We evaluate the existing testing tools against our proposed tool to understand their performance on an Ethereum network with 3 Nodes. The blockchain network is set up on the Microsoft Azure cloud platform with 3 Nodes of B2ms Ubuntu-18 virtual machines, each having 2 VCPUs and 8GB RAM. The reason for choosing a cloud network compared to a local one aligns with production scenarios, where most deployments occur on the standard cloud. Although the RTT of a cloud network is 5 times more than the local network noticed in our earlier work, we choose it to mimic a real enterprise scenario. We test each of the tools by deploying a smart contract function of simple addition like the below:

```
function addTest() {
-> Three variable increment = 38149 Gas
maintxsetcounter = maintxsetcounter+1;
secondtxsetcounter = secondtxsetcounter+1;
thirdtxsetcounter = thirdtxsetcounter+1;
}
```

Even though the function chosen is simple and superficial, it can be intuitively extrapolated to other functions for tokens like balance increment or automotive use cases like car odometer data storage. The above function incurs 38149 gas at the EVM level. At the consensus level, we configure 2 seconds as the Block period, Block Gas Limit to 228894000, to get a block size of 6000 transactions and a transaction pool queue size of 10000000. The versions of Linux 64-bit blockchain clients tested are: i) Geth - 1.9.18 ii) OpenEthereum - V3.0.1 ii) Besu 1.5.0. The performance of existing tools evaluated are i) ChainHammer-V59 ii) Calliper 0.4.0, iii) Blockbench (Source code at July 2020)

We launch the testing tools in an asynchronous multithreading mode with 100 threads from another Ubuntu machine in a similar configuration as blockchain nodes. The reason for 100 threads is due to RPC API limitation, as we faced issues in the existing tools either by nonce or HTTP timeout errors. None of the current testing tools manages the nonce a priori by sending it to the transaction's payload. Instead, they leave it to the client binary to calculate, which slows the transaction processing. We perform the test of each tool for 600000 transactions in 3 iterations. The Blockbench tool could not be tested as the tool suite ran into web3 library errors. Calliper could not be used to test Parity as it did not have an adapter implemented. Also, the ChainHammer tool is not developed for Besu clients, so the results do not include them.

8.1.4 Stress Test with proposed tool

In this section, we vary the thread count incrementally in the proposed tool from 10 up to 10000, with the rest of the configuration being the same to stress test further. We see an incremental improvement in performance with the number of threads for Geth and comparably for other clients in Figure 8.2. This is because the multithreading in the tool fills the nodes with more transactions to process. This shows a linear growth in the performance of Geth and Parity, with the former being steeper due to its better design implementation as well as the consensus efficiency of the Clique. In Clique or IBFT 2.0 implementation of Besu, there is no visible difference with thread variation as there is a bottleneck in the implementation of the client itself and not at the consensus level, which is discussed in section 3.1.2.5. In the case of 10000 threads, Geth and Parity seem to be stable, but Besu fails due to design and implementation limitations. Geth performs well compared to others due to better cache and EVM implementation. Also, the other reason for better performance with Geth is the Clique consensus which has better message communication complexity and immediate sealing compared to Aura or IBFT 2.0.

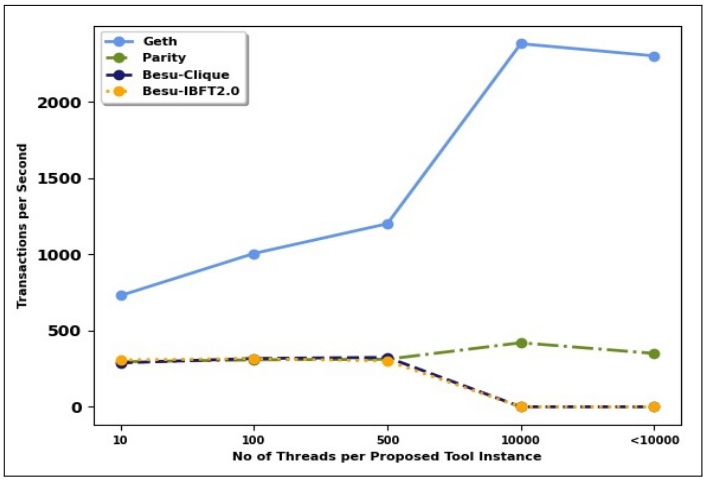

Figure 8.2: Custom Client Test Tool Thread Variation Performance

We infer **100** threads as the best configuration for all clients to test as we notice that Besu runs into stability issues for higher threads like 10000. We could not test beyond 10000 threads which is considerably large for all clients as the Ulimit of the Linux system limits us. With more than 10000 threads as a trial, we noticed an upper bound performance of Geth at 2300 transactions per second(tps) by profiling using pprof. We noticed it inflated the transaction pool and consumed time on the EIP155 signer at the functional level, which is the root cause of several other issues. We discuss the bottleneck issues and root causes we noticed between the clients in the following section 3.1.2.5, which we have reported responsibly to the community as well [159] [160] [161].

8.1.5 Evaluation of Ethereum specific Performance and Behaviour Factors

In this section, we vary the different Ethereum-specific performance parameters of the client binaries and study their behavior. We test the same Add smart contract transaction and retain blockchain network configuration as in the previous section 8.1.3.

8.1.6 Block Gas Limit

Block Gas Limit is a critical performance factor in analyzing by varying its ratio based on the transaction's gas. We set the block period as 2 seconds and tool threads as 100 for now as we focus on the block gas limit's impact on the network. The test was done on different ratios of block gas limit from 1000 up to 15000 incrementally. Consider an Add transaction for 3000 ratio, then the block gas limit is 38149 * 3000 = 114447000, enforcing a block size of 3000 transactions. A higher block gas limit accommodates more transactions to be included in a block but at the same time increases the transaction execution time on the EVM as well. In the results as in Figure 8.3, we notice that at block gas ratio of **6000**, Geth achieves the best performance with 1196 transactions per second. Parity and Besu's performance under the same block gas limit is less than Geth's since the implementation of their EVM and transaction pool are less efficient than the former.

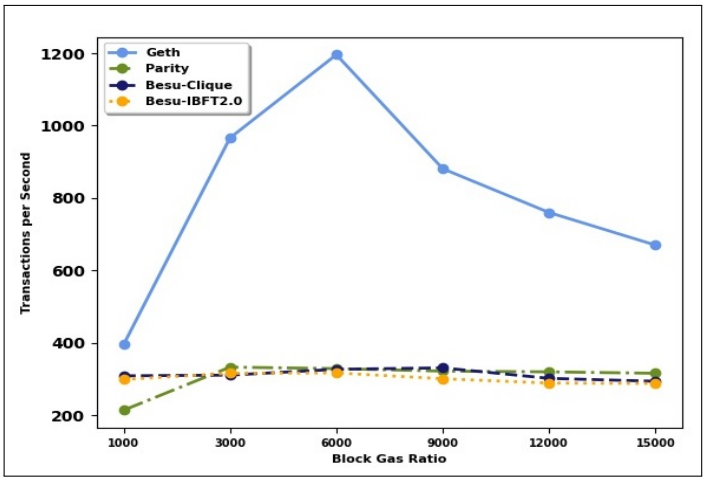

Figure 8.3: Block Gas Limit Variation Testing

In the case of higher block gas limits like 9000 and above, all the client binaries drop performance due to more transactions being executed concurrently. The bottleneck is noticed at the client's EVM level and the transaction pool, which impacts block processing. In the earlier version of Geth V1.9.12, for block gas ratio greater than 10000, we found stalling of transaction execution at the EVM layer, and we reported this as an issue to the Go Ethereum team. The team subsequently solved it, which we noticed while testing the new version mentioned in this article. So, we identify a block gas ratio 6000 as the best parameter for testing. We discuss the behavior issue of dropped throughput due to higher gas limit in section 3.1.2.5.

8.1.7 Block Period

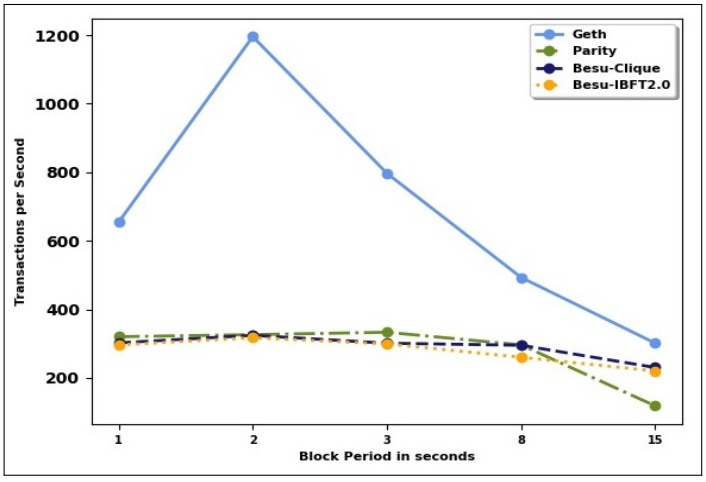

Figure 8.4: Block Period Variation Testing

In this test, we evaluate varied block periods from 1 to 15 seconds for the block-sealing consensus mechanism of the clients. We retain the same configuration and Add transactions as the previous tests, and the block gas ratio is fixed at 6000. We notice from our

results in Figure 8.4 that at 1 second block period, the time window is too short, and the (Geth) Clique consensus faces stability issues. We reported this stability issue at extremely lower block periods [160] to the Go Ethereum team, which is discussed in section 3.1.2.5. Instability means many forks and reorganizations were noticed in the chain as nodes could not synchronize and process consensus in this short period. But at block period 2 seconds, it stabilizes for all the different client binaries at the consensus level. So the block period of **2 seconds** is the fastest and most stable, as higher block periods can delay block sealing.

8.1.8 Transaction Type

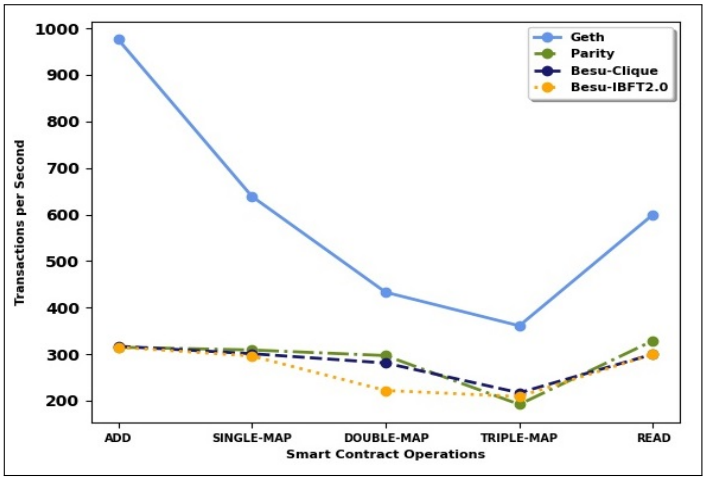

Figure 8.5: Transaction Type Variation Testing

In this test, we understand the impact of varied transaction types based on complexity or gas cost. The significance of smart contract transactions chosen is a simple Map operation that performs key, value-based functions. Still, the reason we include in our test reflects that these operations are the most invoked in smart contracts by enterprises as a common data storage structure. Another transaction is the Add operation which holds significance as explained in the earlier section 8.1.3. Types of transactions are: 1) Add Test as in the previous section. 2) Map test, which functions like a hash map operation to store <key, value> pairs in the smart contract with increasing complexity sub-tests like a) single, b) double, and c) triple map operations like in the below snippet. 3) Read operation, which does not cost gas or complexity. For this test, we retain the best parameters of block period 2 seconds and block gas limit ratio 6000.

```
function MapTest() {
-> Single Map Operation = 47239 Gas
firstmap[firstcounter] = first_value;
firstcounter = firstcounter+1;

-> Double Map Operation = 72880 Gas
secondmap[secondcounter] = second_value;
secondcounter = secondcounter+1;

-> Triple Map Operation = 99916 Gas
```

```
thirdmap[thirdcounter] = third_value;
thirdcounter = thirdcounter+1;
}
```

Inference from the test results in Figure 8.5 shows that the performance decreases proportionately in all the clients with increasing levels of complexity. Geth has a visible impact on transaction variation. Still, it is more gradual for other clients as they are already limited at the implementation level, with a lesser throttle on transactions processed. Read operation, which has nothing to do with transaction processing and no gas cost, has quite the same performance across clients, with Geth performing better than others. As a result of this test, we have finalized the best performance parameters for the block period at 2 seconds, block gas limit ratio at 6000, and behavior with transaction types. We then proceed with these parameters to test the scalability of the network with different clients. We discuss further the behavior in section 3.1.2.5

8.1.9 Scalability

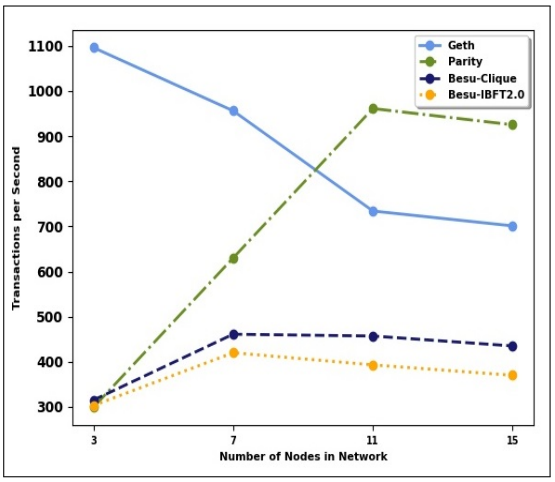

Figure 8.6: Scalability Behavior Testing

In this section, we retain the Add operation, the block period of 2 seconds, and the block gas ratio of 6000 and understand performance concerning the scalability of nodes from 3 up to 15. From results in Figure 8.6, Geth's Clique performance decreases with increasing nodes as more validators per block are eligible to validate, increasing the probability of forks. Except for Geth, whose throughput decreases with more than 3 nodes, Parity and Besu have an improvement in scalability initially. In Parity, up to 11 nodes, and Besu, up to 7 nodes, we notice an increase and then a drop. This is attributed to the increase of nodes, which increases the reception of more transactions as it avoids the bottleneck in the API layer noticed in previous tests. But even though this is solved, it has an internal processing bottleneck, and more nodes augment the communication complexity at the consensus level, which results in a downfall after an increase.

Clique has a single message round compared to two for Aura, making it lighter. But in Clique, more validators per round can create forks in case of out-of-order sealing. In Par-

ity's Aura, just one elected validator per block makes it less susceptible to forks but also scales down. For Besu, Clique's performance is better than IBFT 2.0 due to the efficiency of the algorithm, but still, Besu overall faces a bottleneck at the implementation level as in previous sections, which gets further aggravated for scalability test, making it lesser than Geth and Parity, which will be discussed in section 3.1.2.5

8.1.10 Loadbalancer Middleware Integration with proposed tool

In Ethereum, there is a particularity in load-balancing of transactions which is different from the classical ones as certain design decisions in the clients limit them. The requirement of nonce increment, effective load calculation by monitoring the transaction queue, and also effectively handling a node failure were the ones that led to developing our novel middleware and integration with our testing tool explained earlier in section 8.1.2. In this section, we describe the load-balanced distribution of transaction firing among the nodes in the blockchain network. This architecture will interest enterprises more as the transaction firing would be more reliable and evenly distributed. Even though the nodes might fail or crash, the queue stores the incoming requests and can be processed on eventual recovery.

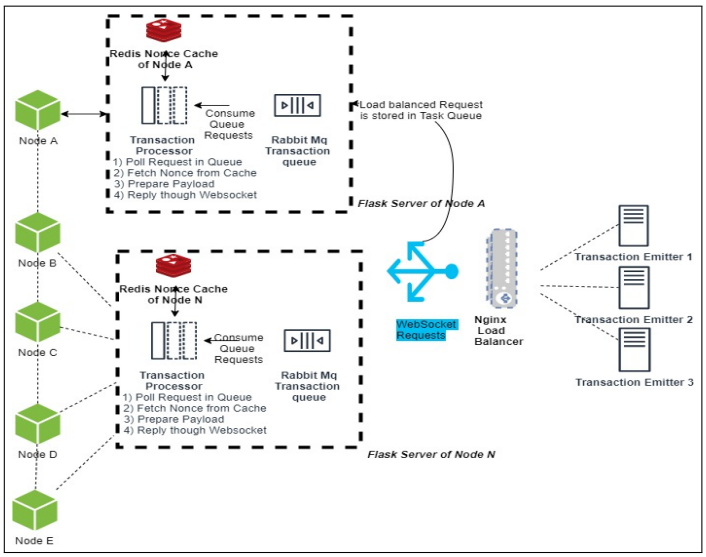

Figure 8.7: Ethereum Client Loadbalancer Middleware Architecture

As in figure 8.7, we create a middleware layer for each node with a web service endpoint. This layer will comprise workers acting on a FIFO queue of message requests forwarded by a round-robin-based load balancer. All workers manage nonce by incrementing in a mutually exclusive cache between them to avoid duplication errors. We test for Add transaction and best parameters of block period 2 seconds, block gas limit ratio 6000, and by a varying number of balanced nodes from 3 up to 15. Figure 8.8 highlights test results for this architecture.

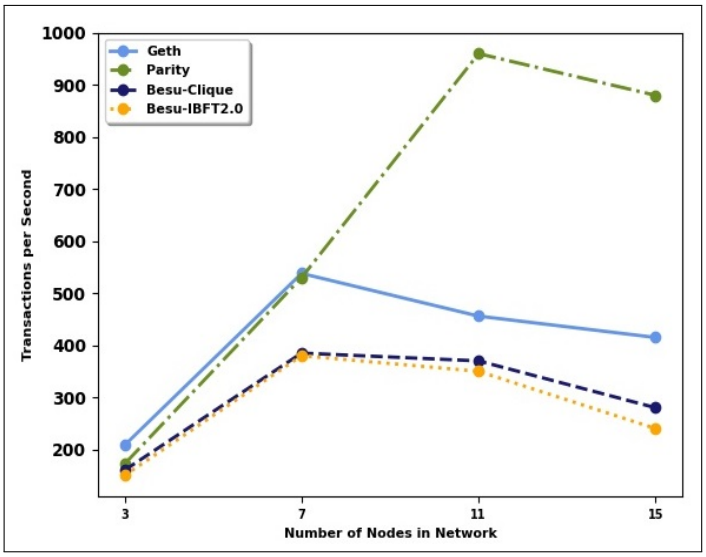

Figure 8.8: Load Balanced Behavior Testing

We notice a lesser throughput in general than previous non-balanced tests as we have limitations in this architecture to work upon in the future. Its queue model and nonce management design limit the transactions reaching the network. So, in general, the nodes receive a lesser rate than earlier ones which suits better for Parity and Besu. It matches their API level bottleneck in the negative sense. Geth has maximum performance on 7 nodes while Parity's performance increases with more nodes up to 11, a similar behavior explained in earlier test on 8.1.9.

8.2 Data Monetisation Discussion

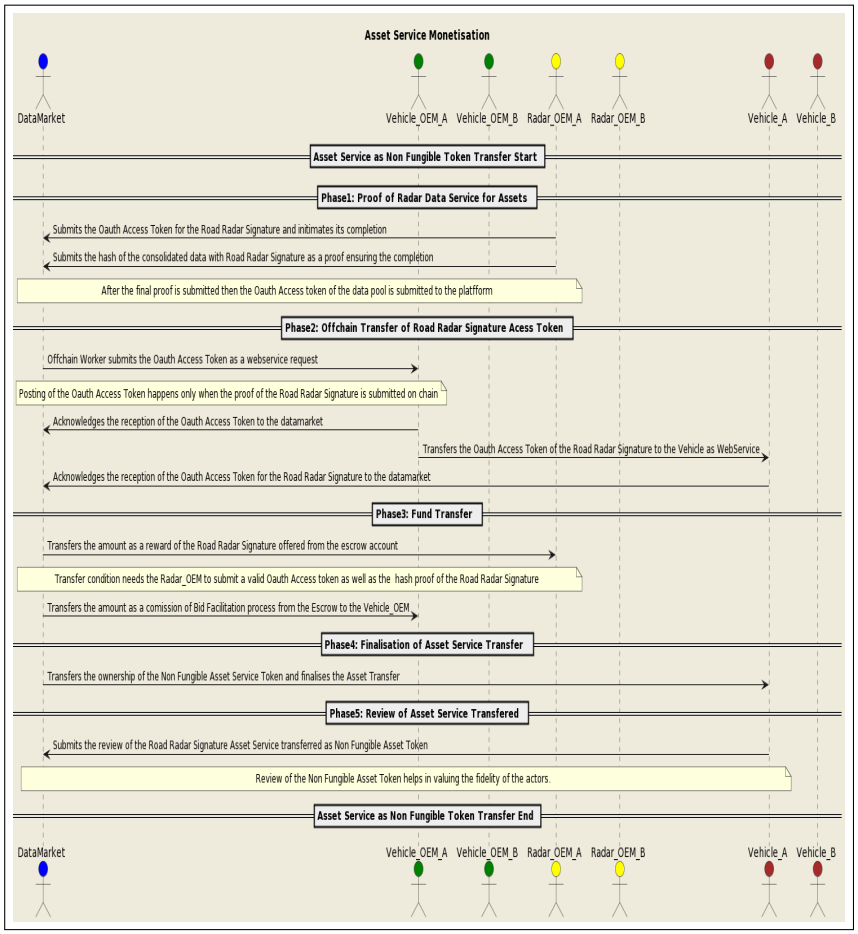

Figure 8.9: Tokenized Asset Data-Service Publish Monetisation Architecture

8.3 Simulated Byzantine Fault Tolerant Consensus Algorithms For Normalised Evaluation Discussion

Figure 8.10: API Module Package Architecture of Simulator Exhibit 1

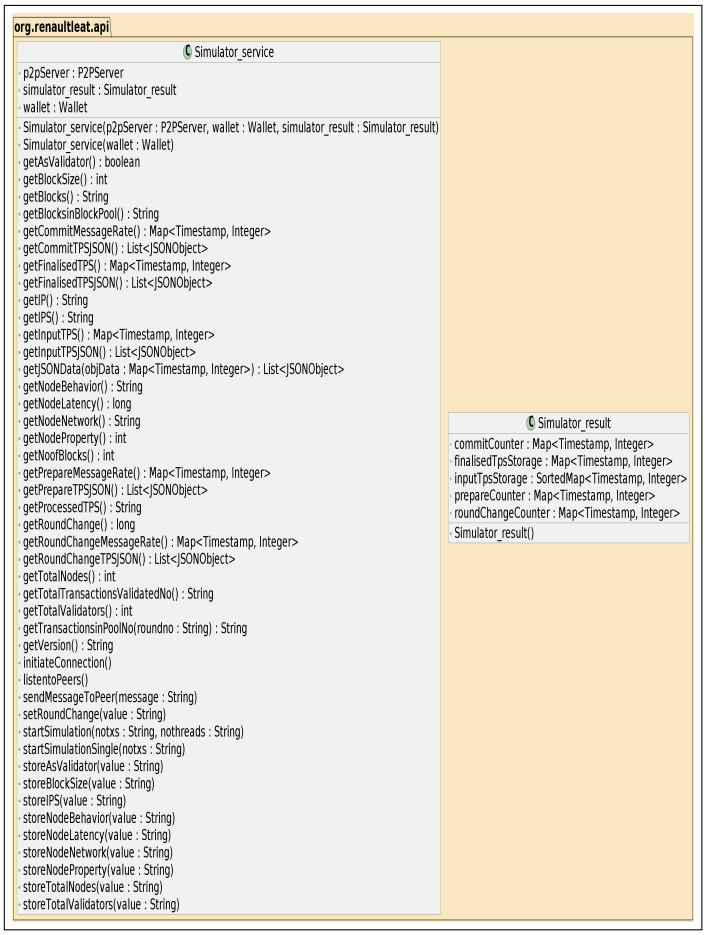

Figure 8.11: API Module Package Architecture of Simulator Exhibit 2

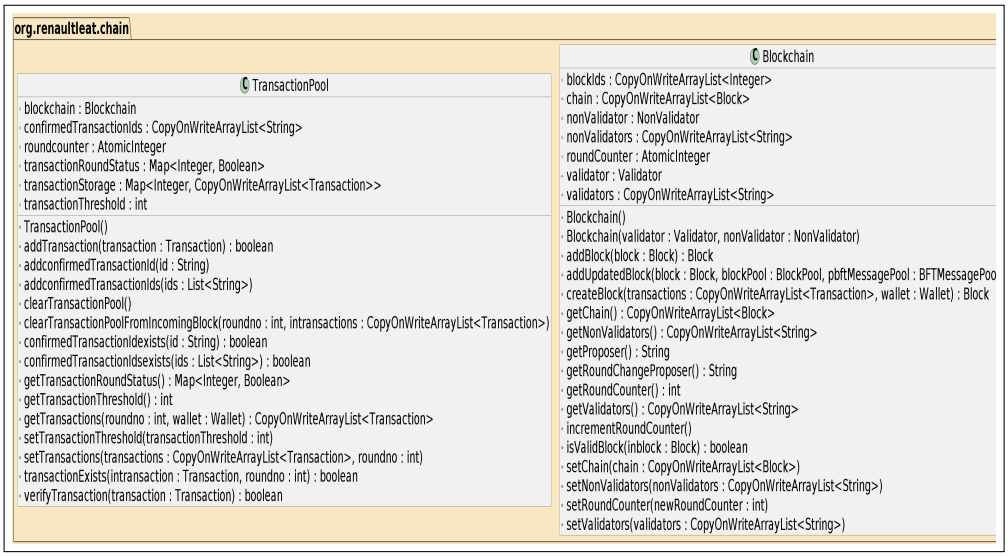

Figure 8.12: Chain Module Package Architecture of Simulator Exhibit 1

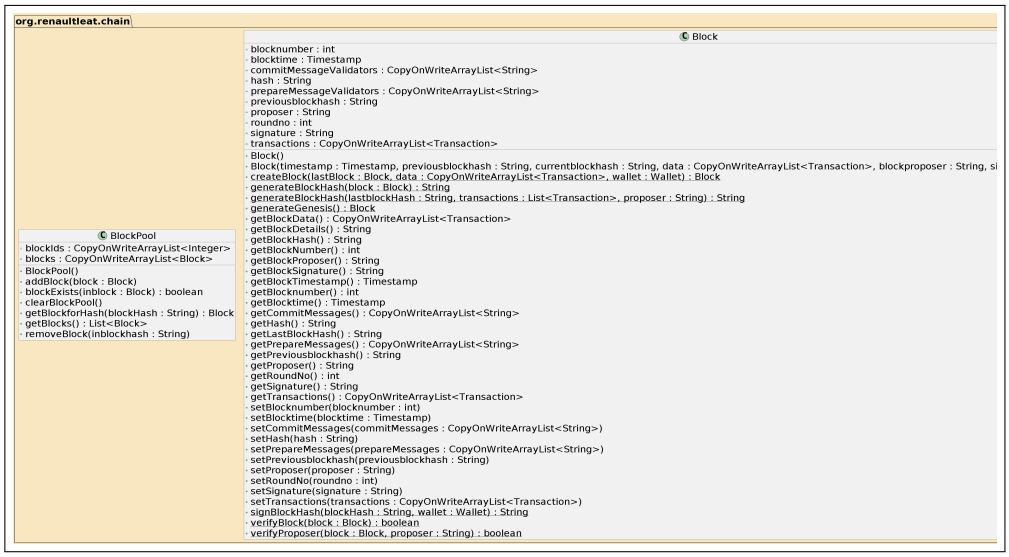

Figure 8.13: Chain Module Package Architecture of Simulator Exhibit 2

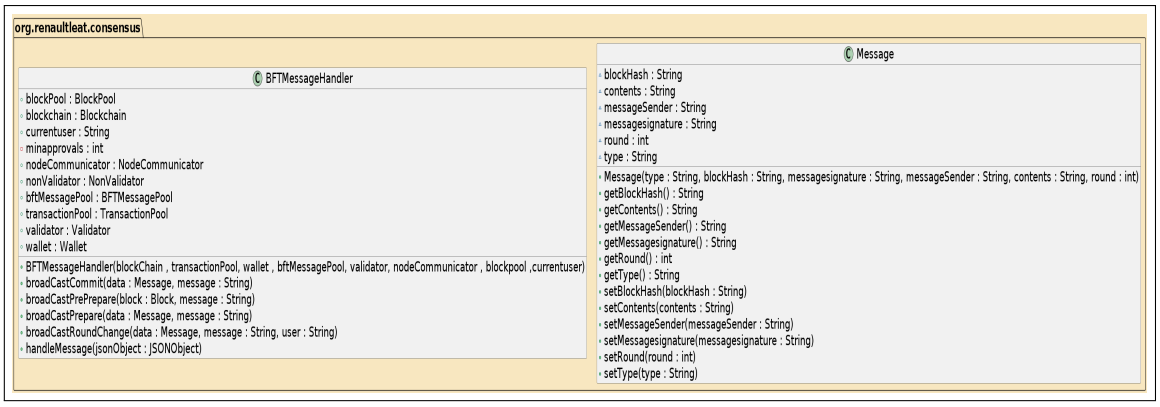

Figure 8.14: Consensus Module Package Architecture of Simulator Exhibit 1

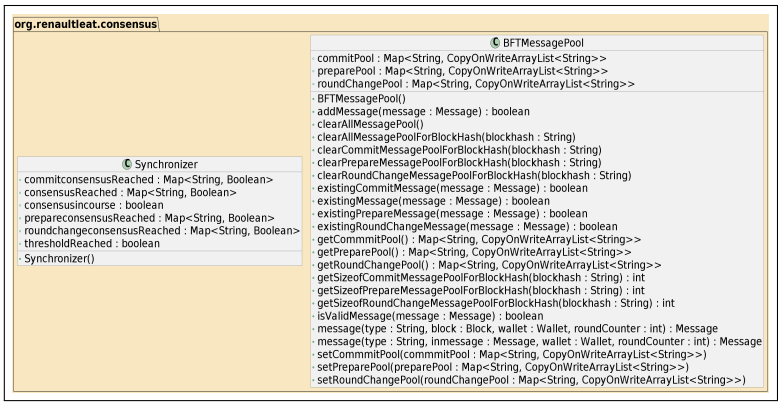

Figure 8.15: Consensus Module Package Architecture of Simulator Exhibit 2

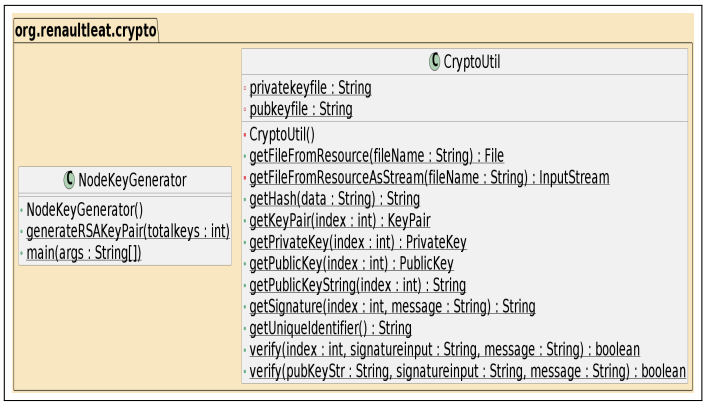

Figure 8.16: Cryptographic Module Package Architecture of Simulator

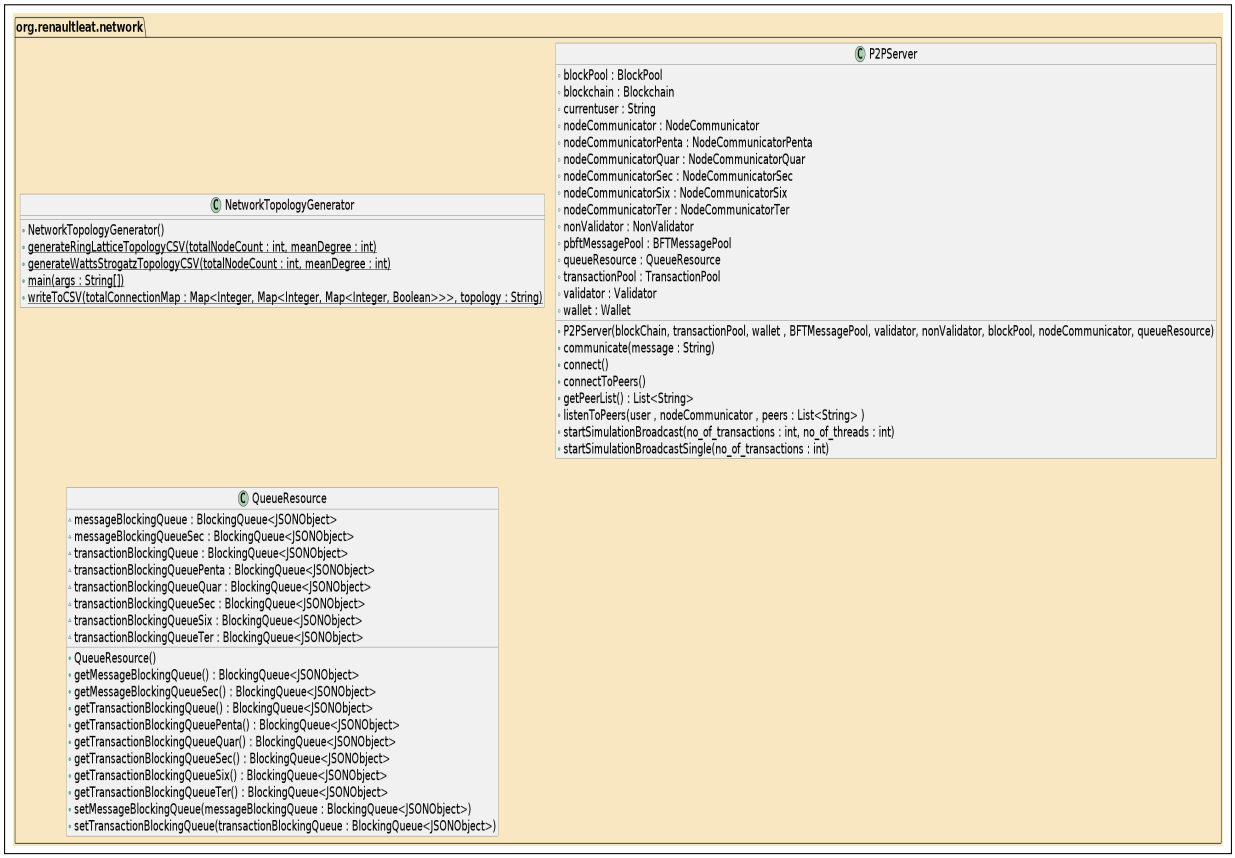

Figure 8.17: Network Module Package Architecture of Simulator

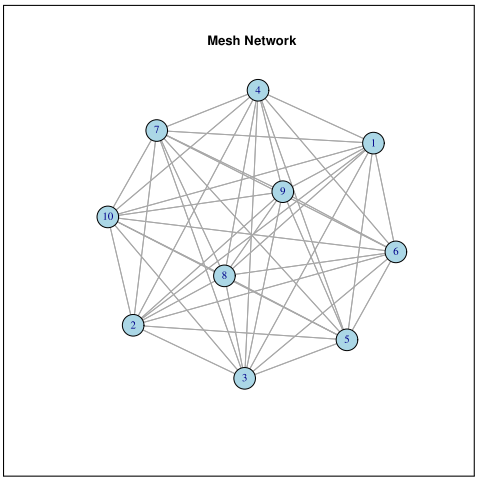

Figure 8.18: Fully Connected Mesh Network Topology of 10 Nodes

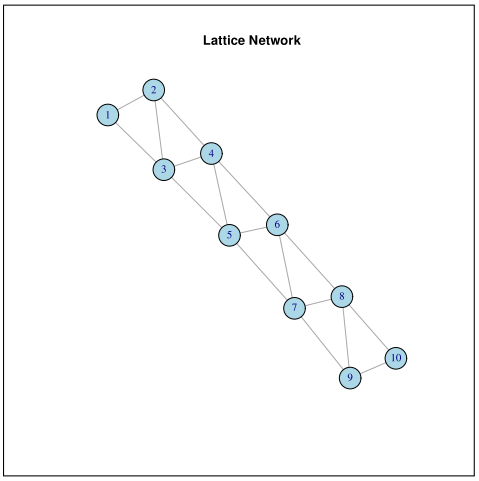

Figure 8.19: Lattice Network Topology of 10 Nodes

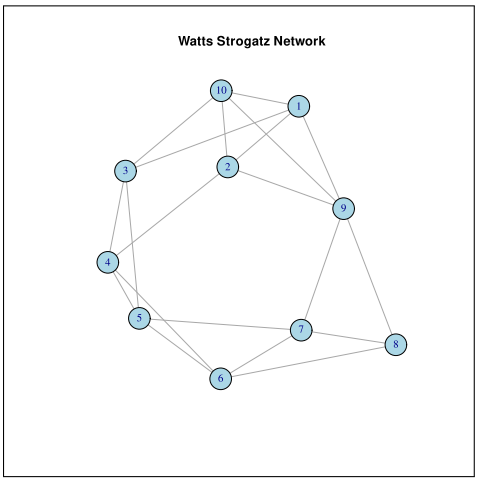

Figure 8.20: Watts Strogatz Network Topology of 10 Nodes

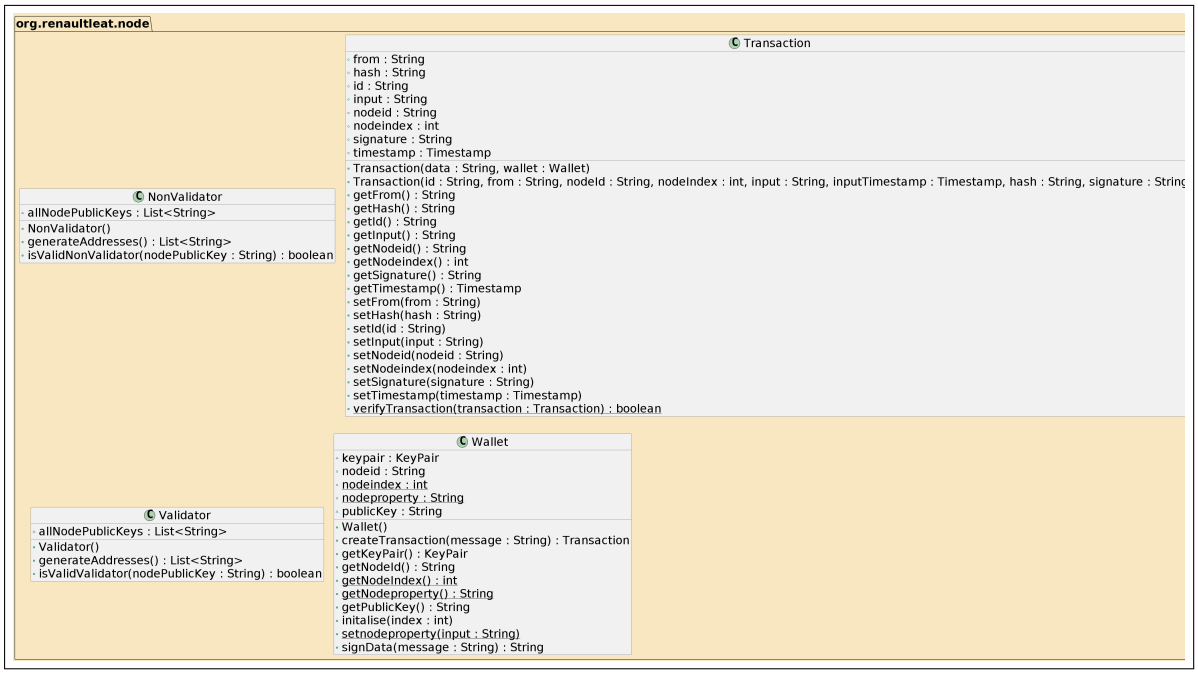

Figure 8.21: Node Module Package Architecture of Simulator

Figure 8.22: Simulation Property Module Package Architecture of Simulator

8.4 Data Monetisation Substrate Discussion

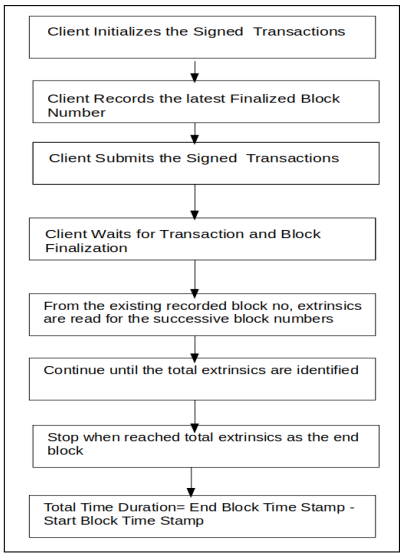

Figure 8.23: Data Monetisation Extrinsic Performance Calculation

8.4.1 BABE and GRANDPA

Block Period Influence RPC Response for stalled GRANDPA state. Votes for Pre-Commit is missing from the 5 validators which stalls the system for lesser block period times. It means there is no synchronicity between the decision of the final block for the 5 validators


```
{
  setId: 0
  best: {
    round: 1,309
    totalWeight: 5
    thresholdWeight: 4
    prevotes: {
      currentWeight: 5
      missing: []
    }
    precommits: {
      currentWeight: 0
      missing: [
        5EAM8S4LXvwQx66jx3WngV6FSDj38yv3h3YxJnpPCRLS3jb2
        5EAfTL69giARzZNQswFRUxbQw1ihzCSsXGkTxRwYvVsZZ7iG
        5EmYCT1kddcQPj9CUcswV2dr6dpzQYSBpdurTuEGHXMdC9K3
        5FXn1Gspo8AgczA2qx4QnkTANdsrVuY3NJNcNzEowisxUqWE
        5Hh1HqkGxH2m2EsBQUoXC7f4VbVsFuk39iPnXBxcycguFyU8
      ]
    }
  }
  background: []
}
```


Figure 8.24: BABE and GRANDPA Consensus Network Stalling

Figure 8.25: BABE and GRANDPA Consensus Network Stalling Epoch Error

8.5 CUBA Consensus Additional Discussion

In this section, we enlist the various algorithms which part of the Competing Utilitarian Byzantine Agreement

8.5.1 Transaction Processing

Algorithm 3: Transaction Processing and Partial Block Proposal

Data:
$\zeta \leftarrow$ Consortium Blockchain Network of K Nodes;
$N_{\mathrm{i,r}} \leftarrow$ Node i which belongs to a Quorum r;
$Q_{\mathrm{r}} \leftarrow$ Identifier for Quorum r containing its members;
$\rho \leftarrow$ Total Number of Quorums in the network;
R$\leftarrow$ Index of Quorum r among the ρ Quorums ;
$\sigma \leftarrow$ Total Number of Members inside a Quorum;
$\mu \leftarrow$ Index of $\mathrm{N_{i,r}}$ within Quorum r;
$\alpha_{\psi} \leftarrow$ Ephemeral Blockchain State;
$\beta_{\omega} \leftarrow$ Finalised Blockchain State;
$\Delta \leftarrow$ Ephemeral State Timeout;
$\eta \leftarrow$ Finalised State Timeout;
$\xi \leftarrow$ Round Change Timeout;
$\theta \leftarrow$ HeartBeat Frequency;
$T_{\chi} \leftarrow$ Incoming Transaction;
$H_{\mathrm{T}_{\chi}} \leftarrow$ Hash of the Transaction;
$P_{\mathrm{k,l}} \leftarrow$ New Partial Block to Form for k is the Block Height and l is the Quorum Index;
$\pi_{\mathrm{r}} \leftarrow$
Transaction Pool to stock a valid and eligible transaction by a Quorum Participant in $\mathrm{Q_r}$;
$\kappa \leftarrow$ Partial Block Size;
$\lambda \leftarrow$ Current Ephemeral Block Round;
$\Gamma \leftarrow$ Epoch Block Limit;
k$\leftarrow$ Current Block Round;
K$\leftarrow$ Total number of members in the Network;
c$\leftarrow$ Codepoint position for Transaction Assignation to a Quorum;
$\mathrm{c_1} \leftarrow$ Codepoint position for Partial Block Proposer;
$\mathrm{c_2} \leftarrow$ Codepoint position for Partial Block Competitor;
Result: $\mathrm{N_{i,r}}$ adds T_{χ} to its pool if condition satisfied and proposes Partial Block if eligible
while $T_{\chi} \neq 0$ ***and*** $H_{T_{\chi}} \notin \beta_{\omega}, \alpha_{\psi}$ **do**
  // Modulus Operation on Total Number of Quorums
  if *Modulus(CodePoint$_c$($H_{T_{\chi}}$),ρ) is R* **then**
    $\pi_{\mathrm{r}} \leftarrow \pi_{\mathrm{r}} + \mathrm{T}_{\chi}$;
    // Form partial block of pooled transactions if proposer or competitor and if size reached
    if *$N_{i,r}$ is [IsPartialBlockProposer($N_{i,r}$,k)* ***or*** *IsPartialBlockCompetitor($N_{i,r}$,k)]* ***and*** *[$|\pi_r|$ is κ]* **then**
      $P_{\mathrm{k,l}} \leftarrow \pi_{\mathrm{r}}$;
      // Reinitialise the Transaction Pool
      $\pi_{\mathrm{r}} \leftarrow 0$;
      Broadcast **PROPOSE_MESSAGE<$\mathbf{P_{k,l}}$>** to other Nodes inside Quorum r;
    end
  end
  Broadcast T_{χ} to other Nodes in Consortium ζ;
end

Algorithm 4: Macro Functions of Algorithm:3 for Transaction Processing and Partial Block Proposal

```
// Input: Node Identifier, Current Ephemeral Block Round
Function IsPartialBlockProposer(N_{i,r}, λ):
    if Modulus(Codepoint(Hash(λ),c1,σ))) is μ then
        return true;
    end
    return false;
// Input: Node Identifier, Current Ephemeral Block Round
Function IsPartialBlockCompetitor(N_{i,r}, λ):
    if Modulus(Codepoint(Hash(λ),c2,σ))) is μ then
        return true;
    end
    return false;
```

8.5.2 Intra-Quorum Consensus

Algorithm 5: Intra-Quorum Consensus

Data: Reference inherited from Algorithm:3

$c_3 \leftarrow$ Codepoint position for Partial Block Finaliser;

$c_4 \leftarrow$ Codepoint position for Partial Block Fulfiller;

$B_{c,k} \leftarrow$ Intermediate Block or Container to accommodate Partial Block $P_{k,l}$;

$T_{B,k} \leftarrow$ Temporal Hash State of the Block Container for round k ;

$P_{k,l} \leftarrow$

New Partial Block to finalise for k is the Block Height or round and l is the Quorum Index;

PROPOSE_MESSAGES ← Mapping<$P_{k,l}$,PROPOSE_MSG>;

COMMIT_MESSAGES ← Mapping<$P_{k,l}$,COMMIT_MSG>;

FINALISE_MESSAGES ← Mapping<$P_{k,l}$,FINALISE_MSG>;

Result: Finalise a Partial Block $P_{k,l}$ where k is the Block Height and l is the Quorum Index

upon receiving **PROPOSE_MESSAGE<$P_{k,l}$>do if** *[IsValidPartialBlock($P_{k,l}$)* ***and*** *IsValidMessage(PROPOSE_MESSAGE<$P_{k,l}$>) is true]* **then**

PROPOSE_MESSAGES = PROPOSE_MESSAGES + PROPOSE_MESSAGE<$P_{k,l}$>;

Broadcast **PROPOSE_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;

Broadcast **COMMIT_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;

end

Algorithm 6: Algorithm continuation of 5

upon receiving **COMMIT_MESSAGE<$P_{k,l}$>** **do if** *[IsValidPartialBlock($P_{k,l}$)* ***and*** *IsValidMessage(COMMIT_MESSAGE<$P_{k,l}$>)] is true* **then**

// If two commits are received by PROPOSER AND COMPETITOR for same Partial Block

if *COMMIT_MESSAGES.contains(COMMIT_MESSAGE<$P_{k,l}$>) is true* **then**

// Comparing the timestamp of incoming message and the existing message

if *COMMIT_MESSAGE<$P_{k,l,incoming}$>.time* $<$ *COMMIT_MESSAGE<$P_{k,l,existing}$>.time is true* **then**

COMMIT_MESSAGES = COMMIT_MESSAGES + COMMIT_MESSAGE<$P_{k,l}$>;

Broadcast **COMMIT_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;

end

end

COMMIT_MESSAGES = COMMIT_MESSAGES + COMMIT_MESSAGE<$P_{k,l}$>;

Broadcast **COMMIT_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;

if *[COMMIT_MESSAGES.SIZE* $\geq 2/3(\sigma)$*+1* ***and*** *IsPartialBlockFinaliser($N_{i,r}$,k)] is true* **then**

Broadcast **FINALISE_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;

end

// In case of State Stalled and Timeout Tolerance attained

if *[COMMIT_MESSAGES.SIZE* $\geq 2/3(\sigma)$*+1* ***and*** *IsPartialBlockFulFiller($N_{i,r}$,k)* ***and*** *IsFulfillmentNeeded(α_ψ, Δ)] is true* **then**

Broadcast **FINALISE_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;

end

end

upon receiving **FINALISE_MESSAGE<$P_{k,l}$>do if** *[IsValidPartialBlock($P_{k,l}$)* ***and*** *IsValidMessage(FINALISE_MESSAGE<$P_{k,l}$>)] is true* **then**

FINALISE_MESSAGES = FINALISE_MESSAGES + FINALISE_MESSAGE<$P_{k,l}$>;

if α_ψ*.notcontains(k)* **then**

α_ψ.[K].BlockContainer = α_ψ.[K].BlockContainer + $P_{k,l}$;

α_ψ.[K].BlockContainer.TemporalHashing.Time $\leftarrow$ Current Time;

α_ψ.[K].BlockContainer.TemporalHashing.StateHash[l] $\leftarrow$ Hash(α_ψ.[K].BlockContainer);

else

$B_{c,k} \leftarrow \{\}$;

$B_{c,k} \leftarrow$ $BlockContainer_K$;

$B_{c,k} \leftarrow B_K$.BlockContainer+ $P_{k,l}$;

α_ψ[K].BlockContainer.TemporalHashing.Time $\leftarrow$ Current Time;

α_ψ[K].BlockContainer.TemporalHashing.StateHash[l] $\leftarrow$ Hash(α_ψ[K].BlockContainer);

end

Broadcast **FINALISE_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;

end

Algorithm 7: Macro Functions for Algorithm:5 of Intra-Quorum

```
        // Input: PartialBlock P_{k,l}
Function IsValidPartialBlock(P_{k,l}):
    if Signature(P_{k,l})) is Valid and [IsPartialBlockProposer(P_{k,l}) or
     IsPartialBlockCompetitor(P_{k,l})] then
        return true;
    end
    return false;
    // Input: Consensus Message MSG{k,l
// Checks Message Signature's validity and sender is part of its Quorum
Function IsValidMessage(MSG_{k,l}):
    if MSG.Signature(P_{k,l})) is Valid and [Q_r.contains(MSG.sender)] then
        return true;
    end
    return false;
    // Input: PartialBlock P_{k,l}
Function IsPartialBlockProposer(P_{k,l}):
    if Modulus(Codepoint(Hash(P_{k,l}.BlockRound),c_1),σ) is P_{k,l}.N_{i,r} then
        return true;
    end
    return false;
    // Input: PartialBlock P_{k,l}
Function IsPartialBlockCompetitor(P_{k,l}):
    if Modulus(Codepoint(Hash(P_{k,l}.BlockRound),c_2),σ) is P_{k,l}.N_{i,r} then
        return true;
    end
    return false;
    // Input: Node Identifier, Current Ephemeral Block Round
Function IsPartialBlockFinaliser(N_{i,r}, λ):
    if Modulus(Codepoint(Hash(λ),c_3),σ) is μ then
        return true;
    end
    return false;
    // Input: Node Identifier, Current Ephemeral Block Round
Function IsPartialBlockFulfiller(N_{i,r}, λ):
    if Modulus(Codepoint(Hash(λ),c_4),σ) is μ then
        return true;
    end
    return false;
// Checks if state of ephemeral blockchain is stalled and timeout tolerance reached
// Input:Ephemeral Blockchain State, Ephemeral State Timeout
Function IsFulfillmentNeeded(α_ψ, Δ):
    if Δ is reached and α_{ψ t-1} == α_{ψ t} then
        return true;
    end
    return false;
```

8.5.3 Inter-Quorum Consensus

Algorithm 8: Inter-Quorum Consensus

Data: Reference Data and Macros inherited from Algorithms:3 and 5
$c_5 \leftarrow$ Codepoint position for Block Proposer;
$c_6 \leftarrow$ Codepoint position for Block Fulfiller;
$c_7 \leftarrow$ Codepoint position for Quorum Proposer;
$B_{c,k} \leftarrow$ Intermediate Block or Container to accommodate Partial Block $P_{k,l}$;
$B_k \leftarrow$ To be finalised Block for round k;
FINALISE_MESSAGES $\leftarrow$ Mapping<$P_{k,l}$,FINALISE_MSG>;
BLOCK_MESSAGES $\leftarrow$ Mapping<$P_{k,l}$,BLOCK_MSG>;
Result: Finalise a Block B_k where k is the Block Height
upon receiving **FINALISE_MESSAGE<$P_{k,l}$>do if** *[IsValidPartialBlock($P_{k,l}$)* ***and*** *IsValidMessage(FINALISE_MESSAGE<$P_{k,l}$>) is true]* **then**
  FINALISE_MESSAGES = FINALISE_MESSAGES + FINALISE_MESSAGE<$P_{k,l}$>;
  if α_ψ*.notContains(k)* **then**
    α_ψ.[K].BlockContainer = α_ψ.[K].BlockContainer + $P_{k,l}$;
    α_ψ.[K].BlockContainer.TemporalHashing.Time = Current Time;
    α_ψ.[K].BlockContainer.TemporalHashing.StateHash[l] = Hash(α_ψ.[K].BlockContainer)
  else
    $B_K \leftarrow \{\}$;
    $B_K \leftarrow$ BlockContainer$_K$;
    $B_K = B_K.BlockContainer + P_{k,l}$;
    α_ψ = α_ψ+B_K; α_ψ.[K].BlockContainer.TemporalHashing.Time = Current Time;
    α_ψ.[K].BlockContainer.TemporalHashing.StateHash[l] = Hash(α_ψ.[K].BlockContainer);
  end
  Broadcast **FINALISE_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;
  if *IsBlockProposer(B_K)* ***or*** *IsBlockFulfiller(B_K)* **then**
    B_K = B_K + Hash(B_K.contents)+ Signature(B_K); // Calculate Utilitarian Score of Block and store
    UtilitarianScoreUpdate(B_k);
    Broadcast **BLOCK_MESSAGE<B_K>** to other Nodes inside Consortium ζ;
  end
end
upon receiving **BLOCK_MESSAGE<B_k>do**
if *[IsValidBlock(B_k)* ***and*** *IsValidMessage(BLOCK_MESSAGE<B_k>) is true]* **then**
  β_ω = β_ω + B_k; // Calculate Utilitarian Score of Block and store
  UtilitarianScoreUpdate(B_k);
  Broadcast **BLOCK_MESSAGE<B_k>** to other Nodes inside Consortium ζ;
end
// In case of Finalised Blockchain State Stalled and Timeout Tolerance attained
if *[IsBlockFulFiller($N_{i,r}$,k)* ***and*** *IsBlockFulfillmentNeeded(β_ω, η)] is true* **then**
  B_K = B_K + Hash(B_K.contents)+ Signature(B_K); // Calculate Utilitarian Score of Block and store
  UtilitarianScoreUpdate(B_K);
  Broadcast **BLOCK_MESSAGE<B_K>** to other Nodes inside Consortium ζ; **if** *[IsNewQuorumNeeded(β_ω)* ***and*** *IsQuorumProposer($N_{i,r}$) is true]* **then**
    // Re-Organise the Quorum based on Utilitarian Score
    QUORUM_MESSAGE<E_ε> $\leftarrow FormNewQuorum(NewEpochE_\varepsilon)$;
    Broadcast **QUORUM_MESSAGE<E_ε>** to Nodes in Consortium ζ for Epoch E ε;
  end
end

Algorithm 9: Macro Functions for Algorithm:8 of Inter-Quorum

```
// Input: Block B_K
Function IsBlockProposer(B_K):
    if Modulus(Codepoint(Hash(B_K.BlockRound),c_5),K)) is N_{i,r} then
        return true;
    end
    return false;
// Input: Block B_K
Function IsBlockFulfiller(B_K):
    if [Modulus(Codepoint(Hash(B_K.BlockRound),c_6),K)) is N_{i,r} and
     IsBlockFulfillmentNeeded(BLOCK_MESSAGE<B_k>) is true ] then
        return true;
    end
    return false;
// Input: Proposed Epoch E_ε
Function IsQuorumProposer(E_ε):
    if [Modulus(Codepoint(Hash(E_ε),c_7),K)) is N_{i,r} ] then
        return true;
    end
    return false;
// Input: Block B_k
Function IsValidBlock(B_k):
    if Signature(B_k)) is Valid and [IsBlockProposer(B_k) or IsBlockFulfiller(B_k)] then
        return true;
    end
    return false;
// Checks if state of finalised blockchain is stalled and timeout tolerance reached
// Input:Ephemeral Blockchain State,Finalised Blockchain State, Finalised State Timeout
Function IsBlockFulfillmentNeeded(α_ψ, β_ω, η):
    if α_ψ.[K].B_{c,k}.SIZE == ρ and η is reached and β_{ω t-1} == β_{ω t} then
        return true;
    end
    return false;
// Checks if a new epoch is reached
// Input:Finalised Blockchain State
Function IsNewQuorumNeeded(β_ω):
    if MODULUS(β_ω.SIZE,Γ) == 0 then
        return true;
    end
    return false;
```

8.5.4 Round Change Algorithm

Algorithm 10: Round-Change Consensus

Data: Reference Data and Macros inherited from Algorithms:3, 5,8,10
ROUND_CHANGE_MESSAGES ←
Mapping<BLOCK HEIGHT K,ROUND_CHANGE_MSG>;
c_8← Codepoint position for Quorum Fulfiller in case of RoundChange;
Result: Propose a Round Change at K which is the Block Height

while *IsRoundChangeNeeded(K) is true* **do**
  Broadcast **ROUND_CHANGE_MESSAGE<K>** to other Nodes inside Consortium ζ;
end
upon receiving **ROUND_CHANGE_MESSAGE<K>do**
if *[IsRoundChangeNeeded(K)* ***and*** *IsValidMessage(ROUND_CHANGE_MESSAGE<K>) is true]* **then**
  ROUND_CHANGE_MESSAGES = ROUND_CHANGE_MESSAGES + ROUND_CHANGE_MESSAGE<K>;
  Broadcast **ROUND_CHANGE_MESSAGE<K>** to other Nodes inside Consortium ζ;
  if *[ROUND_CHANGE_MESSAGES.SIZE* $\geq 1/2(K)+1$ ***and*** $IsQuorumFullFiller(E_\varepsilon)$*]is true* **then**
    Broadcast ***QUORUM_MESSAGE<E_ε>*** *to other Nodes inside Consortium ζ for Epoch ε;*
    Broadcast ***ROUND_CHANGE_MESSAGE<K>*** *to other Nodes inside Consortium ζ;*
  end
end
  // Checks if state of ephemeral,finalised blockchain is stalled and timeout tolerance reached
// Input:Ephemeral,Finalised Blockchain State, Roundchange State Timeout
Function `IsRoundChangeNeeded(`$\alpha_\psi, \beta_\omega, \xi$`):`
  if $\alpha_{\psi\,t\text{-}1} == \alpha_{\psi\,t}$ ***and*** $\beta_{\omega\,t\text{-}1} == \beta_{\omega\,t}$ ***and*** ξ *is reached* **then**
    return true;
  end
  return false;
// Input: Proposed Epoch E_ε
Function `IsQuorumFullFiller(`E_ε`):`
  if *[Modulus(Codepoint(Hash(E_ε),c_8),K)) is* $N_{i,r}$ *]* **then**
    return true;
  end
  return false;

8.5.5 Heart Beat Protocol

Algorithm 11: Heart Beat Message Exchange

Data: Reference Data and Macros inherited from Algorithms:3, 5,8,10
HEART_BEAT_MESSAGES ← Mapping<TIME t ,HEART_BEAT_MSG>;
Result: Broadcast a HEART_BEAT message at time frequency t

while *IsHeartBeatNeeded() is true]* **do**
  Broadcast **HEART_BEAT_MESSAGE<t>** to other Nodes inside Consortium ζ at time t;
end
upon receiving **HEART_BEAT_MESSAGE<t>do**
if *[IsValidMessage(HEART_BEAT_MESSAGE<t>) is true]* **then**
  HEART_BEAT_MESSAGES = HEART_BEAT_MESSAGES + HEART_BEAT_MESSAGE<t>;
  Broadcast **HEART_BEAT_MESSAGE<t>** to other Nodes inside Consortium ζ;
end
  // Checks if frequency to broadcast Heart Beat is reached
// Input:HeartBeat Frequency
Function `IsHeartBeatNeeded(`θ`)`**:**
  if θ *is reached* **then**
    return true;
  end
  return false;

8.5.6 Utilitarian Score Processing

Algorithm 12: Macro Functions for UTILITARIAN SCORE UPDATE of Algorithm:8 of Inter-Quorum

Data: Reference Data and Macros inherited from Algorithms:3 and 5,8 10 11
// Utilitarian Unit Score
// Positive Unit Score
PartialBlock_Proposal_Win_ScoreUnit ← PB_Proposal_win;
PartialBlock_Commit_Win_ScoreUnit ← PB_Commit_win;
HeartBeat_Win_ScoreUnit ← HB_Score_win;
// Negative Unit Score
PartialBlock_Proposal_Loss_ScoreUnit ← PB_Proposal_loss;
PartialBlock_Commit_Loss_ScoreUnit ← PB_Commit_loss;
Malicious_ScoreUnit ← M_Score;
HeartBeat_Loss_ScoreUnit ← HB_Score_loss;
// Utilitarian Storage
Utilitarian_Proposal_Win ← Mapping<B_K,<Node Identity N_i,Score»;
Utilitarian_Proposal_Loss ← Mapping<B_K,<Node Identity N_i,Score»;
Utilitarian_Commit_Win ← Mapping<B_K,<Node Identity N_i,Score»;
Utilitarian_Commit_Loss ← Mapping<B_K,<Node Identity N_i,Score»;
Utilitarian_HeartBeat ← Mapping<B_K,<Node Identity N_i,Score»;
Utilitarian_HeartBeat_Missed ← Mapping<B_K,<Node Identity N_i,Score»;
Utilitarian_Malicious ← Mapping<B_K,<Node Identity N_i,Score»;

// Input: New Block B_K
Function `UtilitarianScoreUpdate(`B_K`):`
    // Inter-Block Time Coefficient
    Inter_Block_Time_Coefficient ← B_K.blocktime - B_{K-1}.blocktime;
    // Normalising to milliseconds
    Time_Coefficient ← Time_Coefficient/100000;
    for $EachPartialBlock$ ***In*** $B_K.PartialBlocks$ **do**
        Utilitarian_Proposal_Win ← Utilitarian_Proposal_Win +
          Mapping<B_K,<EachPartialBlock.PartialBlockProposer , PB_Proposal_win ∗
          Time_Coefficient + getFairnessScore(CurrentEpoch,Node Identifier) >>;
        Utilitarian_Proposal_Loss ← Utilitarian_Proposal_Loss +
          Mapping<B_K,<EachPartialBlock.PartialBlockCompetitor , PB_Proposal_loss ∗
          Time_Coefficient >>;
        Utilitarian_Commit_Win ← Utilitarian_Commit_Win +
          Mapping<B_K,<EachPartialBlock.PartialBlockCommiter , PB_Commit_win ∗
          Time_Coefficient + getFairnessScore(CurrentEpoch,Node Identifier) >>;
        Utilitarian_Commit_Loss ← Utilitarian_Commit_loss +
          Mapping<B_K,<EachPartialBlock.PartialBlockCommitLoser , PB_Commit_loss ∗
          Time_Coefficient >>;
    end

Algorithm 13: Macro Functions for UTILITARIAN SCORE UPDATE continuation of Algorithm:12 for Inter-Quorum

Data: Reference Data and Macros inherited from Algorithms:3 and 5,8, 10, 11, 12
// Utilitarian Members Storage
Ideal_Utilitarians_Storage ← Mapping<Epoch E,List<Node Identity N_i»;
Utilitarians_Storage ← Mapping<Epoch E,List<Node Identity N_i»;
Fair_Utilitarians_Storage ← Mapping<Epoch E,List<Node Identity N_i»;
Weak_Utilitarians_Storage ← Mapping<Epoch E,List<Node Identity N_i»;
// Utilitarian Score Storage
Ideal_Utilitarians_ScoreFairnessCoefficient ← IU_{sc};
Utilitarians_ScoreFairnessCoefficient ← U_{sc};
Fair_Utilitarians_ScoreFairnessCoefficient ← FU_{sc};
Weak_Utilitarians_ScoreFairnessCoefficient ← WU_{sc};
for $EachPartialBlock$ ***In*** $B_K.PartialBlocks$ **do**
  // If heartbeat sent by the node recently within frequency
  if *HEART_BEAT_MESSAGES.contains(EachPartialBlock.PartialBlockProposer)* **then**
    Utilitarian_HeartBeat ← Utilitarian_HeartBeat +
      Mapping<B_K,<EachPartialBlock.PartialBlockProposer ,HB_Score_win ∗
      Time_Coefficient + getFairnessScore(CurrentEpoch,Node Identifier) >>;
  else
    Utilitarian_HeartBeat_Missed ← Utilitarian_HeartBeat_Missed +
      Mapping<B_K,<EachPartialBlock.PartialBlockProposer ,HB_Score_loss ∗
      Time_Coefficient >>;
  end
  if ***IsInvalid****(EachPartialBlock)* **then**
    Utilitarian_Malicious ← Utilitarian_Malicious +
      Mapping<B_K,<EachPartialBlock.PartialBlockProposer ,M_Score ∗
      Time_Coefficient >>;
  end
end
// Input: Current Epoch E_ε, Node N_i
Function `getFairnessScore(`E_ε`):`
  // If an Ideal Utilitarian add inversely less coefficient
  if *Ideal_Utilitarians_Storage.contains(N_i)* **then**
    return Ideal_Utilitarians_ScoreCoefficient;
  end
  // If an Utilitarian add inversely less coefficient
  if *Utilitarians_Storage.contains(N_i)* **then**
    return Utilitarians_ScoreCoefficient;
  end
  // If a Fair Utilitarian add inversely less coefficient
  if *Fair_Utilitarians_Storage.contains(N_i)* **then**
    return Fair_Utilitarians_ScoreCoefficient;
  end
  // If a Weak Utilitarian add inversely less coefficient
  if *Weak_Utilitarians_Storage.contains(N_i)* **then**
    return Weak_Utilitarians_ScoreCoefficient;
  end

8.5.7 Quorum Reorganisation

Algorithm 14: Macro Functions for Quorum Preparation of Algorithm:8 of Inter-Quorum

Data: Reference Data and Macros inherited from Algorithms:3 and 5,8, 10, 11, 12
Effective_Score ← Mapping<Epoch E, <Node Identity N_i,EffectiveScore$_{N_i}$»;
Forgetting_Coefficient ← Mapping<Epoch E, ForgettingCoefficient$_E$>;
Final_Score ← Mapping<Epoch E, <Node Identity N_i,FinalScore$_{N_i}$»;
Final_Rank ← Mapping<Node Rank, Node Identity N_i>;
// Input: New Epoch E_ε
Function `FormNewQuorum(`E_ε`):`

- **for** $EachEpoch \leftarrow 0$ **to** $[CurrentEpoch - 1]$ **do**
  - **for** $EachMemberN_i$ ***In*** *Consortium* ζ **do**
    - *Utilitarian_Proposal_Win*$_{N_i}$ ← *Utilitarian_Proposal_Win*.$get[N_i]$;
    - *Utilitarian_Proposal_Loss*$_{N_i}$ ← *Utilitarian_Proposal_Loss*.$get[N_i]$;
    - *Utilitarian_Commit_Win*$_{N_i}$ ← *Utilitarian_Commit_Win*.$get[N_i]$;
    - *Utilitarian_Commit_Loss*$_{N_i}$ ← *Utilitarian_Commit_Loss*.$get[N_i]$;
    - *Utilitarian_HeartBeat*$_{N_i}$ ← *Utilitarian_HeartBeat*.$get[N_i]$;
    - *Utilitarian_HeartBeat_Missed*$_{N_i}$ ← *Utilitarian_HeartBeat_Missed*.$get[N_i]$;
    - *Utilitarian_Malicious*$_{N_i}$ ← *Utilitarian_Malicious*.$get[N_i]$;
    - // Positive Score Aggregation
    - *Positive_Score*$_{N_i}$ ← *Utilitarian_Proposal_Win*$_{N_i}$ + *Utilitarian_Commit_Win*$_{N_i}$ + *Utilitarian_HeartBeat*$_{N_i}$;
    - // Negative Score Aggregation
    - *Negative_Score*$_{N_i}$ ← *Utilitarian_Proposal_Loss*$_{N_i}$ + *Utilitarian_Commit_Loss*$_{N_i}$ + *Utilitarian_HeartBeat_Missed*$_{N_i}$ + *Utilitarian_Malicious*$_{N_i}$;
    - // Effective Score Aggregation
    - *Effective_Score*$_{N_i}$ ← *Positive_Score*$_{N_i}$ − *Negative_Score*$_{N_i}$;
    - *Effective_Score* ← *Effective_Score* + <*EachEpoch*,<N_i,*Effective_Score*$_{N_i}$»;
  - **end**
- **end**
- *FormForgettingCoefficient*(E_ε);
- // Apply the Forgetting Coefficient on the Final Score for previous epochs based on current epoch
- *Final_Score* ← $UpdateForgettingCoefficient(E_\varepsilon)$;
- // Classify the nodes based on Utilitarian Score and form the new Quorum
- *QUORUM_MESSAGE*<E_ε> ← *ClassifyandReorganiseQuorum*(E_ε);
- **return** *QUORUM_MESSAGE*<E_ε>;

Algorithm 15: Macro Functions for Quorum Preparation of Algorithm:8 of Inter-Quorum Continuation

Data: Reference Data and Macros inherited from Algorithms:3 and 5,8, 10, 11, 12

// Input: New Epoch E_{ε}

Function `FormForgettingCoefficient(`E_{ε}`):`

  for $EachEpoch \leftarrow$ [*CurrentEpoch-1*]**to**0 **by** -1 **do**

    // Calculate Forgetting Coefficient

    $\text{Forgetting\_Coefficient}_{\text{EachEpoch}} \leftarrow$ (1-(CurrentEpoch-EachEpoch)/CurrentEpoch);

    $\text{Forgetting\_Coefficient} \leftarrow \text{Forgetting\_Coefficient} + \text{Forgetting\_Coefficient}_{\text{EachEpoch}}$;

  end

// Input: New Epoch E_{ε}

Function `UpdateForgettingCoefficient(`E_{ε}`):`

  for $EachEpoch \leftarrow 0$ **to** *[*$CurrentEpoch - 1$*]* **do**

    for $EachMemberN_i$ ***In*** *Consortium* ζ **do**

      EachEpoch_Forgetting_Coefficient $\leftarrow$ *Forgetting_Coefficient.*$get[EachEpoch]$*;*

      // Apply the Forgetting Coefficient on Effective Score

      Final_Score $\leftarrow$ *Final_Score* $+$

        <EachEpoch,<N_i*,EffectiveScore.get(*$_{N_i}$*)*EachEpoch_Forgetting_Coefficient»;*

    end

  end

Algorithm 16: Macro Functions for Quorum Preparation of Algorithm:8 of Inter-Quorum Continuation

Data: Reference Data and Macros inherited from Algorithms:3 and 5,8, 10, 11, 12
Ideal_Utilitarian ← Mapping<Epoch E,List<Node Identity N_i»;
Utilitarian ← Mapping<Epoch E,List<Node Identity N_i»;
Fair_Utilitarian ← Mapping<Epoch E,List<Node Identity N_i»;
Weak_Utilitarian ← Mapping<Epoch E,List<Node Identity N_i»;
PotentialSuspended_Utilitarian ← Mapping<Epoch E,List<Node Identity N_i»;
Suspended_Utilitarian ← Mapping<Epoch E,List<Node Identity N_i»;
Z ← Epoch Suspension Threshold for Weak Utilitarians;
Quorum_Storage ← Mapping<Epoch E, <Quorum Index Q $_i$,Node Identity N_i»;
// Input: New Epoch E_ε, Final_Score Mapping<Epoch E, <Node Identity N_i,FinalScore$_{N_i}$»
Function `ClassifyandReorganiseQuorum(`E_ε`):`
    Final_RankMapping<Node Rank, Node Identity N_i> ←
     SortByFinalScore(Final_Score);
    Index_Split ←K/5;
    for $IdealUtilitarianIndex \leftarrow 0$ **to** *[Index_Split]* **do**
        Ideal_Utilitarian ← *Ideal_Utilitarian* + $Mapping <$
         $NewEpoch$, *Final_Rank.get*$(IdealUtilitarianIndex) >$;
        // Reorganise Nodes to the Quorum_n
        Quorum_n ← *Mapping <Quorum Index Q $_i$,Final_Rank.get(IdealUtilitarianIndex)>;*
    end
    QUORUM_MESSAGE<E_ε> ← *QUORUM_MESSAGE<E_ε>* + *Quorum_n;*
    Quorum_Storage ← *Quorum_Storage* + *Mapping<CurrentEpoch, Quorum_n>;*
    for $UtilitarianIndex \leftarrow$ *Index_Split* **to** *[Index_Split* $* 2$*]* **do**
        Utilitarian ← *Utilitarian* + $Mapping <$
         $NewEpoch$, *Final_Rank.get*$(UtilitarianIndex) >$;
        // Reorganise Nodes to the Quorum_n+1
        Quorum_n+1 ← *Mapping <Quorum Index Q $_i$,Final_Rank.get(UtilitarianIndex)>;*
    end
    QUORUM_MESSAGE<E_ε> ← *QUORUM_MESSAGE<E_ε>* + *Quorum_n+1;*
    Quorum_Storage ← *Quorum_Storage* + *Mapping<CurrentEpoch, Quorum_n+1>;*
    for $FairUtilitarianIndex \leftarrow$ *Index_Split* $* 2$ **to** *[Index_Split* $* 3$*]* **do**
        Fair_Utilitarian ← *Fair_Utilitarian* + $Mapping <$
         $NewEpoch$, *Final_Rank.get*$(FairUtilitarianIndex) >$;
        // Reorganise Nodes to the Quorum_n+2
        Quorum_n+2 ← *Mapping <Quorum Index Q $_i$,Final_Rank.get(FairUtilitarianIndex)>;*
    end
    QUORUM_MESSAGE<E_ε> ← *QUORUM_MESSAGE<E_ε>* + *Quorum_n+2;*
    Quorum_Storage ← *Quorum_Storage* + *Mapping<CurrentEpoch, Quorum_n+2>;*

Algorithm 17: Macro Functions for Quorum Preparation of Algorithm:8 of Inter-Quorum Continuation

for $WeakUtilitarianIndex \leftarrow$ *Index_Split* $* 3$ **to** *[Index_Split* $* 4$*]* **do**
  Weak_Utilitarian $\leftarrow$ *Weak_Utilitarian* $+ Mapping <$
  $NewEpoch$, *Final_Rank.get*$(WeakUtilitarianIndex) >$;
  // Reorganise Nodes to the Quorum_n+3
  Quorum_n+3 $\leftarrow$ *Mapping <Quorum Index Q* $_i$*,Final_Rank.get(WeakUtilitarianIndex)>;*
end
QUORUM_MESSAGE<E_ε*>* $\leftarrow$ *QUORUM_MESSAGE<*E_ε*>* $+$ *Quorum_n+3;*
Quorum_Storage $\leftarrow$ *Quorum_Storage* $+$ *Mapping<CurrentEpoch, Quorum_n+3>;*
for $VeryWeakUtilitarianIndex \leftarrow$ *Index_Split* $* 4$ **to** *[Index_Split* $* 5$*]* **do**
  // Allow weak utilitarian nodes if not present successively in this range for Z previous epochs
  for $PreviousEpoch \leftarrow [\, CurrentEpoch - Z\,]$**to** $CurrentEpoch - 1$ **do**
    if
    [Weak_Utilitarian.get(PreviousEpoch).notContains(Final_Rank.get(VeryWeakUtilitarianIndex))
    and *HEART_BEAT_MESSAGES.contains(VeryWeakUtilitarianIndex)]* ***or***
    Suspended_Utilitarian.get(PreviousEpoch).Contains(Final_Rank.get(VeryWeakUtilitarianIndex)
) **then**
      Weak_Utilitarian $\leftarrow$ *Weak_Utilitarian* $+ Mapping <$
      $NewEpoch$, *Final_Rank.get*$(WeakUtilitarianIndex) >$;
      // Reorganise Nodes to the Quorum_n+3
      Quorum_n+3 $\leftarrow$ *Mapping <Quorum Index Q* $_i$*,Final_Rank.get(WeakUtilitarianIndex)>;*
    else
      Suspended_Utilitarian $\leftarrow$ *Suspended_Utilitarian* $+ Mapping <$
      $NewEpoch$, *Final_Rank.get*$(VeryWeakUtilitarianIndex) >$;
      PotentialSuspended_Utilitarian $\leftarrow$ *PotentialSuspended_Utilitarian* $+ Mapping <$
      $NewEpoch$, *Final_Rank.get*$(VeryWeakUtilitarianIndex) >$;
      Weak_Utilitarian $\leftarrow$ *Weak_Utilitarian* $+ Mapping <$
      $NewEpoch$, *Final_Rank.get*$(VeryWeakUtilitarianIndex) >$;
    end
  end
  QUORUM_MESSAGE<E_ε> $\leftarrow$ QUORUM_MESSAGE<E_ε> $+$ Quorum_n+3;
  Quorum_Storage $\leftarrow$ Quorum_Storage $+$ Mapping<CurrentEpoch, Quorum_n+3>;
end
return QUORUM_MESSAGE<E_ε>;

8.5.8 CUBA: Intra-Quorum Implicit Conensus

Algorithm 18: Intra-Quorum Implicit Consensus

Data: Reference inherited from Algorithm:3
$c_3 \leftarrow$ Codepoint position for Partial Block Finaliser;
$c_4 \leftarrow$ Codepoint position for Partial Block Fulfiller;
$B_{c,k} \leftarrow$ Intermediate Block or Container to accommodate Partial Block $P_{k,l}$;
$T_{B,k} \leftarrow$ Temporal Hash State of the Block Container for round k ;
$P_{k,l} \leftarrow$
New Partial Block to finalize for k is the Block Height or round and l is the Quorum Index;

FINALISE_MESSAGES $\leftarrow$ Mapping<$P_{k,l}$,FINALISE_MSG>;
Result: Finalise a Partial Block $P_{k,l}$ where k is the Block Height, and l is the Quorum Index
if *[IsPartialBlockFinaliser($N_{i,r}$,k) **or** IsPartialBlockFulfiller($N_{i,r}$,k)] is true* **then**
  Broadcast **FINALISE_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;
end
upon receiving **FINALISE_MESSAGE<$P_{k,l}$>do**
if *[IsValidPartialBlock($P_{k,l}$) **and** IsValidMessage(FINALISE_MESSAGE<$P_{k,l}$>)] is true* **then**
  FINALISE_MESSAGES = FINALISE_MESSAGES + FINALISE_MESSAGE<$P_{k,l}$>;
  if α_ψ.*notcontains(k)* **then**
    α_ψ.[K].BlockContainer = α_ψ.[K].BlockContainer + $P_{k,l}$;
    α_ψ.[K].BlockContainer.TemporalHashing.Time $\leftarrow$ Current Time;
    α_ψ.[K].BlockContainer.TemporalHashing.StateHash[l] $\leftarrow$ Hash(α_ψ.[K].BlockContainer);
  else
    $B_{c,k} \leftarrow \{\}$;
    $B_{c,k} \leftarrow$ $BlockContainer_K$;
    $B_{c,k} \leftarrow$ B_K.BlockContainer+ $P_{k,l}$;
    α_ψ[K].BlockContainer.TemporalHashing.Time $\leftarrow$ Current Time;
    α_ψ[K].BlockContainer.TemporalHashing.StateHash[l] $\leftarrow$ Hash(α_ψ[K].BlockContainer);
  end
  Broadcast **FINALISE_MESSAGE<$P_{k,l}$>** to other Nodes inside Quorum r;
end

Bibliography

[1] Robert Paul Resch. *Utopia, Dystopia, and the Middle Class in George Orwell's Nineteen Eighty-Four*. Vol. 24. 1. Duke University Press, 1997, 137–176. URL: http://www.jstor.org/stable/303755 (visited on 04/26/2023) (page 2).

[2] Lewis, Michael. *Flash Boys: A Wall Street Revolt*. W. W. Norton Company, 2014. ISBN: 9780393244663. URL: https://www.simonandschuster.com/books/Flash-Boys/Michael-Lewis/9781442370289. (accessed: 20.04.2023) (page 2).

[3] Fontanel, Jacques. *GAFAM, a progress and a danger for civilization*. Saint-Petersbourg, Russia, Apr. 2019. URL: https://hal.univ-grenoble-alpes.fr/hal-02102188. (accessed: 20.04.2023) (page 2).

[4] "Holbein, Ambrosius and More, Thomas". *"Utopia"*. URL: https://library.princeton.edu/visual\%5Fmaterials/maps/websites/thematic-maps/theme-maps/utopia.html. (accessed: 26.05.2023) (page 2).

[5] Brown, Clare and Hardman, Michael and Davies, Nick and Armitage, Richard. "Mobility as a Service: Defining a Transport Utopia". In: *Future Transportation* 2.1 (2022), 300–309. ISSN: 2673-7590. DOI: 10.3390/futuretransp2010016. URL: https://www.mdpi.com/2673-7590/2/1/16. (accessed: 02.05.2023) (page 3).

[6] Wunderlich, Karl. *Operational and Financial Implications of Transactionalizing Multi-Machine Maneuvers in Self-Organizing Autonomous Systems*. Ed. by Volodymyr Babich, John R. Birge, and Gilles Hilary. Cham: Springer International Publishing, 2022, 23–42. ISBN: 978-3-030-75729-8. DOI: 10.1007/978-3-030-75729-8_2. URL: https://doi.org/10.1007/978-3-030-75729-8\%5F2. (accessed: 03.05.2023) (page 3).

[7] "EIT Urban Mobility". *"Mobility Data Decentralisation leveraging Galileo and Blockchain"*. URL: https://www.eiturbanmobility.eu/wp-content/uploads/2022/11/Factual\%5Fleaflet.pdf. (accessed: 26.05.2023) (page 3).

[8] Sophia Auer and Sophia Nagler and Somnath Mazumdar and Raghava Rao Mukkamala. "Towards blockchain-IoT based shared mobility: Car-sharing and leasing as a case study". In: *Journal of Network and Computer Applications* 200 (2022), p. 103316. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2021.103316. URL: https://www.sciencedirect.com/science/article/pii/S1084804521003015. (accessed: 03.05.2023) (page 3).

[9] T. Donna Chen and Kara M. Kockelman. "Carsharing's life-cycle impacts on energy use and greenhouse gas emissions". In: *Transportation Research Part D: Transport and Environment* 47 (2016), 276–284. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2016.05.012. URL: https://www.sciencedirect.com/science/article/pii/S1361920916303030. (accessed: 04.05.2023) (page 3).

[10] Karger, Erik and Jagals, Marvin and Ahlemann, Frederik. "Blockchain for Smart Mobility—Literature Review and Future Research Agenda". In: *Sustainability* 13 (Nov. 2021), p. 13268. DOI: 10.3390/su132313268. (accessed: 04.05.2023) (page 3).

[11] Ballandies, Mark and Dapp, Marcus and Pournaras, Evangelos. "Decrypting distributed ledger design—taxonomy, classification and blockchain community evaluation". In: *Cluster Computing* 25 (2022), 1–22. DOI: 10.1007/s10586-021-03256-w. (accessed: 06.05.2023) (page 3).

[12] Lopez Vivar, Antonio and Castedo, Alberto and Sandoval Orozco, Ana and Villalba, García. "Smart Contracts: A Review of Security Threats Alongside an Analysis of Existing Solutions". In: *Entropy* 22 (2020), p. 203. DOI: 10.3390/e22020203. (accessed: 07.05.2023) (page 5).

[13] El Ioini, Nabil and Pahl, Claus. "A Review of Distributed Ledger Technologies: Confederated International Conferences: CoopIS, CTC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II". In: (Oct. 2018), 277–288. DOI: 10.1007/978-3-030-02671-4_16. (accessed: 08.05.2023) (page 5).

[14] "Holochain". *"Holochain Architecture"*. URL: https://github.com/Holochain/holochain-proto/wiki. (accessed: 27.05.2023) (page 5).

[15] Thuat Do. *SoK on Blockchain Evolution and a Taxonomy for Public Blockchain Generations*. Cryptology ePrint Archive, Paper 2023/315. https://eprint.iacr.org/2023/315. 2023. URL: https://eprint.iacr.org/2023/315. (accessed: 09.05.2023) (page 6).

[16] Cai, Wei and Wang, Zehua and Ernst, Jason B. and Hong, Zhen and Feng, Chen and Leung, Victor C. M. "Decentralized Applications: The Blockchain-Empowered Software System". In: *IEEE Access* 6 (2018), 53019–53033. DOI: 10.1109/ACCESS.2018.2870644. (accessed: 09.05.2023) (page 7).

[17] Leiponen, Aija and Thomas, Llewellyn and Wang, Qian. "The dApp economy: A new platform for distributed innovation?" In: *Innovation: Organization Management* (2021), DOI: 10.1080/14479338.2021.1965887. (accessed: 09.05.2023) (page 7).

[18] Zheng, Peilin and Jiang, Zigui and Wu, Jiajing and Zheng, Zibin. "Blockchain-based Decentralized Application: A Survey". In: *IEEE Open Journal of the Computer Society* PP (2023), 1–12. DOI: 10.1109/OJCS.2023.3251854. (accessed: 09.05.2023) (page 7).

[19] Delmolino, Kevin and Arnett, Mitchell and Kosba, Ahmed and Miller, Andrew and Shi, Elaine. *Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab*. 2016. DOI: 10.1007/978-3-662-53357-4_6. (accessed: 10.05.2023) (page 7).

[20] "Szabo, Nick". *"Smart Contracts: Building Blocks for Digital Markets"*. URL: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart\%5Fcontracts\%5F2.html. (accessed: 27.05.2023) (page 7).

[21] "Massa Blockchain". *"Autonomous Smart Contracts (ASC)"*. URL: https://massa.net/autonomous-sc/. (accessed: 27.05.2023) (page 8).

[22] Cheng, Raymond and Zhang, Fan and Kos, Jernej and He, Warren and Hynes, Nicholas and Johnson, Noah and Juels, Ari and Miller, Andrew and Song, Dawn. "Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts". In: (2019), 185–200. DOI: 10.1109/EuroSP.2019.00023 (page 8).

[23] *2018 reform of EU data protection rules*. European Commission. May 25, 2018. URL: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN (visited on 06/17/2019). (accessed: 09.11.2022) (pages 8, 26, 103).

[24] Oyinloye, Damilare Peter and Teh, Je Sen and Jamil, Norziana and Alawida, Moatsum. "Blockchain Consensus: An Overview of Alternative Protocols". In: *Symmetry* 13.8 (2021). ISSN: 2073-8994. DOI: 10.3390/sym13081363. URL: https://www.mdpi.com/2073-8994/13/8/1363. (accessed: 14.05.2023) (page 9).

[25] Lampson, Butler W. "How to Build a Highly Available System Using Consensus". In: WDAG '96 (1996), 1–17. (accessed: 13.05.2023) (page 10).

[26] Fischer, Michael J. and Lynch, Nancy A. and Paterson, Michael S. "Impossibility of Distributed Consensus with One Faulty Process". In: *J. ACM* 32.2 (1985), 374–382. ISSN: 0004-5411. DOI: 10.1145/3149.214121. URL: https://doi.org/10.1145/3149.214121. (accessed: 11.11.2022) (pages 10, 30, 31, 143, 162).

[27] Verma, Neetu and Jain, Saurabh and Doriya, Rajesh. "Review on Consensus Protocols for Blockchain". In: (2021), 281–286. DOI: 10.1109/ICCCIS51004.2021.9397089. (accessed: 15.05.2023) (page 10).

[28] Varun Kohli and Sombuddha Chakravarty and Vinay Chamola and Kuldip Singh Sangwan and Sherali Zeadally. "An analysis of energy consumption and carbon footprints of cryptocurrencies and possible solutions". In: *Digital Communications and Networks* 9.1 (2023), 79–89. ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2022.06.017. URL: https://www.sciencedirect.com/science/article/pii/S2352864822001390. (accessed: 15.05.2023) (page 10).

[29] Bianchi Alves, Bianca and Wang, Winnie and Moody, Joanna and Waksberg Guerrini, Ana and Peralta Quiros, Tatiana and Velez, Jean Paul and Ochoa Sepulveda, Maria Catalina and Alonso Gonzalez, Maria Jesus. "Adapting Mobility-as-a-Service for Developing Cities : A Context-Sensitive Approach. Mobility and Transport Connectivity". In: (2021). DOI: https://openknowledge.worldbank.org/handle/10986/36787. (accessed: 05.11.2022) (page 17).

[30] Merkert, Rico and Bushell, James and Beck, Matthew. "Collaboration as a service (CaaS) to fully integrate public transportation – Lessons from long distance travel to reimagine mobility as a service". In: *Transportation Research Part A: Policy and Practice* 131 (2020). Developments in Mobility as a Service (MaaS) and Intelligent Mobility, 267–282. ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2019.09.

025. URL: https://www.sciencedirect.com/science/article/pii/S0965856418309716. (accessed: 14.12.2022) (page 17).

[31] de Wilde, Thijs. "Thesis Mobility as a Service (MaaS) meets Blockchain". PhD thesis. 2019, DOI: 10.13140/RG.2.2.25503.61608. (accessed: 05.11.2022) (page 17).

[32] Auer, Sophia and Nagler, Sophia and Mazumdar, Somnath and Rao Mukkamala, Raghava. "Towards blockchain-IoT based shared mobility: Car-sharing and leasing as a case study". In: *Journal of Network and Computer Applications* 200 (2022), p. 103316. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2021.103316. URL: https://www.sciencedirect.com/science/article/pii/S1084804521003015. (accessed: 05.11.2022) (pages 18, 80).

[33] Mallah, Ranwa Al and López, David and Farooq, Bilal. "Cyber-Security Risk Assessment Framework for Blockchains in Smart Mobility". In: *IEEE Open Journal of Intelligent Transportation Systems* 2 (2021), 294–311. DOI: 10.1109/OJITS.2021.3106863. (accessed: 05.11.2022) (page 18).

[34] de Oliveira, I. Romani and Matsumoto, T. and Neto, E. C. Pinto. "Blockchain-based traffic management for Advanced Air Mobility". In: (2022). DOI: 10.48550/ARXIV.2208.09312. URL: https://arxiv.org/abs/2208.09312. (accessed: 05.11.2022) (page 19).

[35] Ayaz, Ferheen and Sheng, Zhengguo and Tian, Daxin and Nekovee, Maziar and Saeed, Nagham. "Blockchain-Empowered AI for 6G-Enabled Internet of Vehicles". In: *Electronics* 11.20 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11203339. URL: https://www.mdpi.com/2079-9292/11/20/3339. (accessed: 05.11.2022) (page 19).

[36] Nguyen, Tri Hong and Partala, Juha and Pirttikangas, Susanna. "Blockchain-Based Mobility-as-a-Service". In: (2019), 1–6. DOI: 10.1109/ICCCN.2019.8847027. (accessed: 05.11.2022) (page 19).

[37] Gong, Shuangqing and Hossein Chinaei, Mohammad and Luo, Fengji and Hossein Rashidi, Taha. "The distributed ownership of on-demand mobility service". In: *International Journal of Transportation Science and Technology* (2022). ISSN: 2046-0430. DOI: https://doi.org/10.1016/j.ijtst.2022.06.003. URL: https://www.sciencedirect.com/science/article/pii/S2046043022000569. (accessed: 05.11.2022) (page 19).

[38] Ongaro, Diego and Ousterhout, John. "In Search of an Understandable Consensus Algorithm". In: USENIX ATC'14 (2014), 305–320. (accessed: 13.12.2022) (pages 19, 145).

[39] López, David and Farooq, Bilal. "A multi-layered blockchain framework for smart mobility data-markets". In: *Transportation Research Part C: Emerging Technologies* 111 (2020), 588–615. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.01.002. URL: https://www.sciencedirect.com/science/article/pii/S0968090X19300361. (accessed: 05.11.2022) (page 19).

[40] Wu, Junhua and Jin, Zhenyu and Li, Guangshun and Xu, Zhuqing and Fan, Cang and Zheng, Yuanwang. "Design of Vehicle Certification Schemes in IoV Based on Blockchain". In: *World Wide Web* 25.5 (2022), 2241–2263. ISSN: 1386-145X. DOI: 10.1007/s11280-022-01078-3. URL: https://doi.org/10.1007/s11280-022-01078-3. (accessed: 05.11.2022) (page 20).

[41] Gîrbacia, F and Voinea, Gheorghe and Boboc, Razvan and Duguleană, M and Postelnicu, Cristian. "Toward blockchain adoption for the automotive industry". In: *IOP Conference Series: Materials Science and Engineering* 1220 (2022), 012026. DOI: 10.1088/1757-899X/1220/1/012026. (accessed: 05.11.2022) (page 21).

[42] McPhie, Tim and Crespo-Parrondo, Ana. *Zero emission vehicles: first 'Fit for 55' deal will end the sale of new CO2 emitting cars in Europe by 2035*. Oct. 2022. (accessed: 05.11.2022) (page 21).

[43] Mohamed, Nader and Al-Jaroodi, Jameela. "Applying Blockchain in Industry 4.0 Applications". In: (2019), 0852–0858. DOI: 10.1109/CCWC.2019.8666558. (accessed: 06.11.2022) (page 21).

[44] Jadoon, Gullelala and Ud Din, Ikram and Almogren, Ahmad and Almajed, Hisham. "Smart and Agile Manufacturing Framework, A Case Study for Automotive Industry". In: *Energies* 13.21 (2020). ISSN: 1996-1073. DOI: 10.3390/en13215766. URL: https://www.mdpi.com/1996-1073/13/21/5766. (accessed: 06.11.2022) (page 21).

[45] F Gîrbacia and D Voinea and R Boboc and M Duguleană and C C Postelnicu. "Toward blockchain adoption for the automotive industry". In: *IOP Conference Series: Materials Science and Engineering* 1220.1 (2022), 012026. DOI: 10.1088/1757-899X/1220/1/012026. URL: https://dx.doi.org/10.1088/1757-899X/1220/1/012026 (page 21).

[46] Das, Moonmoon and Azad, Rahat Uddin and Efat, Md. Iftekharul Alam. "Blockchain aided Vehicle Certification (BVC): A Secured E-Governance Framework for Transport Stakeholders". In: (2020), 1–6. DOI: 10.1109/ICCIT51783.2020.9392725. (accessed: 06.11.2022) (page 21).

[47] Bannister, Frank and Connolly, Regina. "The future ain't what it used to be: Forecasting the impact of ICT on the public sphere". In: *Government Information Quarterly* 37.1 (2020), p. 101410. ISSN: 0740-624X. DOI: https://doi.org/10.1016/j.giq.2019.101410. URL: https://www.sciencedirect.com/science/article/pii/S0740624X19301248. (accessed: 06.11.2022) (page 21).

[48] Swiderski, Frank and Snyder, Window. *Threat Modeling*. USA: Microsoft Press, 2004. ISBN: 0735619913. (accessed: 06.11.2022) (page 21).

[49] Chanson, Mathieu and Bogner, Andreas and Wortmann, Felix and Fleisch, Elgar. "Blockchain As a Privacy Enabler: An Odometer Fraud Prevention System". In: UbiComp '17 (2017), 13–16. DOI: 10.1145/3123024.3123078. URL: http://doi.acm.org/10.1145/3123024.3123078. (accessed: 06.11.2022) (pages 21, 22).

[50] Brousmiche, Kei-Léo and Heno, Thomas and Poulain, Christian and Dalmieres, Antoine and Ben Hamida, Elyes. "Digitizing, Securing and Sharing Vehicles Life-cycle over a Consortium Blockchain: Lessons Learned". In: (2018), 1–5. DOI: 10.1109/NTMS.2018.8328733. (accessed: 06.11.2022) (page 22).

[51] Gerrits, Luc and Kromes, Roland and Verdier, François. "A True Decentralized Implementation Based on IoT and Blockchain: a Vehicle Accident Use Case". In: (2020), 1–6. DOI: 10.1109/COINS49042.2020.9191405. (accessed: 06.11.2022) (page 23).

[52] Avyukt, Anusha and Ramachandran, Gowri and Krishnamachari, Bhaskar. "A Decentralized Review System for Data Marketplaces". In: (2021), 1–9. DOI: 10.1109/ICBC51069.2021.9461149. (accessed: 06.11.2022) (page 23).

[53] Meijers, James and Dharma Putra, Guntur and Kotsialou, Grammateia and Kanhere, Salil S. and Veneris, Andreas. "Cost-Effective Blockchain-based IoT Data Marketplaces with a Credit Invariant". In: (2021), 1–9. DOI: 10.1109/ICBC51069.2021.9461127. (accessed: 07.11.2022) (page 24).

[54] Al-Sada, Bader and Lasla, Noureddine and Abdallah, Mohamed. "Secure Scalable Blockchain for Sealed-Bid Auction in Energy Trading". In: (2021), 1–3. DOI: 10.1109/ICBC51069.2021.9461071. (accessed: 07.11.2022) (page 24).

[55] Banerjee, Prabal and Ruj, Sushmita. "Blockchain Enabled Data Marketplace – Design and Challenges". In: *arXiv e-prints*, arXiv:1811.11462 (Nov. 2018), arXiv:1811.11462. arXiv: 1811.11462. (accessed: 07.11.2022) (pages 24, 102).

[56] Dziembowski, Stefan and Eckey, Lisa and Faust, Sebastian. "FairSwap: How To Fairly Exchange Digital Goods". In: CCS '18 (2018), 967–984. DOI: 10.1145/3243734.3243857. URL: https://doi.org/10.1145/3243734.3243857. (accessed: 07.11.2022) (pages 24, 102).

[57] Koutsos, Vlasis and Papadopoulos, Dimitrios and Chatzopoulos, Dimitris and Tarkoma, Sasu and Hui, Pan. "Agora: A Privacy-aware Data Marketplace". In: (2020), 1211–1212. DOI: 10.1109/ICDCS47774.2020.00156. (accessed: 07.11.2022) (page 24).

[58] Chotard, Jérémy et al. "Decentralized Multi-Client Functional Encryption for Inner Product". In: () (page 25).

[59] Jeong, Byeong-Gyu and Youn, Taek-Young and Jho, Nam-Su and Shin, Sang Uk. "Blockchain-Based Data Sharing and Trading Model for the Connected Car". In: *Sensors* 20.11 (2020). ISSN: 1424-8220. DOI: 10.3390/s20113141. URL: https://www.mdpi.com/1424-8220/20/11/3141. (accessed: 08.11.2022) (page 25).

[60] Zhang, Jindan and Wang, Xu An and Ma, Jianfeng. "Data Owner Based Attribute Based Encryption". In: (2015), 144–148. DOI: 10.1109/INCoS.2015.42. (accessed: 08.11.2022) (page 25).

[61] Faroukhi, Abou Zakaria and El Alaoui, Imane and Gahi, Youssef and Amine, Aouatif. "An Adaptable Big Data Value Chain Framework for End-to-End Big Data Monetization". In: *Big Data and Cognitive Computing* 4.4 (2020). ISSN: 2504-2289. DOI: 10.3390/bdcc4040034. URL: https://www.mdpi.com/2504-2289/4/4/34. (accessed: 09.11.2022) (page 25).

[62] K. Ozyilmaz and M. Dogan and A. Yurdakul. "IDMoB: IoT Data Marketplace on Blockchain". In: (2018), 11–19. DOI: 10.1109/CVCBT.2018.00007. URL: https://doi.ieeecomputersociety.org/10.1109/CVCBT.2018.00007. (accessed: 09.11.2022) (page 26).

[63] Anderson, Ross J. *Security Engineering: A Guide to Building Dependable Distributed Systems*. 2nd ed. Wiley Publishing, 2008. ISBN: 9780470068526. (accessed: 09.11.2022) (pages 26, 56).

[64] Schwartz, Mischa and Kleinrock, L. "An early history of the internet [History of Communications". In: *IEEE Communications Magazine* 48 (2010), 26–36. DOI: 10.1109/MCOM.2010.5534584. (accessed: 09.11.2022) (page 26).

[65] Wensley, J.H. and Lamport, L. and Goldberg, J. and Green, M.W. and Levitt, K.N. and Melliar-Smith, P.M. and Shostak, R.E. and Weinstock, C.B. "SIFT: Design and analysis of a fault-tolerant computer for aircraft control". In: *Proceedings of the IEEE* 66.10 (1978), 1240–1255. DOI: 10.1109/PROC.1978.11114. (accessed: 09.11.2022) (page 27).

[66] Rotem-Gal-Oz, Arnon. *Fallacies of Distributed Computing Explained*. 2008. (accessed: 09.11.2022) (page 27).

[67] World Economic Forum and Marsh McLennan and SK Group and Zurich Insurance Group. *The Global Risks Report*. Jan. 2022. (accessed: 09.11.2022) (page 27).

[68] Lamport, Leslie and Shostak, Robert and Pease, Marshall. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (1982), 382–401. ISSN: 0164-0925. DOI: 10.1145/357172.357176. URL: https://doi.org/10.1145/357172.357176. (accessed: 10.11.2022) (pages 27–29, 58).

[69] Pease, M. and Shostak, R. and Lamport, L. "Reaching Agreement in the Presence of Faults". In: *J. ACM* 27.2 (1980), 228–234. ISSN: 0004-5411. DOI: 10.1145/322186.322188. URL: https://doi.org/10.1145/322186.322188. (accessed: 10.11.2022) (page 27).

[70] Akkoyunlu, E. A. and Ekanadham, K. and Huber, R. V. "Some Constraints and Tradeoffs in the Design of Network Communications". In: *SIGOPS Oper. Syst. Rev.* 9.5 (1975), 67–74. ISSN: 0163-5980. DOI: 10.1145/1067629.806523. URL: https://doi.org/10.1145/1067629.806523. (accessed: 11.11.2022) (page 27).

[71] Bayer, Rudolf and Graham, Robert M. and Saltzer, Jerome H. and Seegmüller, Gerhard. *Operating Systems, An Advanced Course - Introduction*. Berlin, Heidelberg: Springer-Verlag, 1978, 1–6. ISBN: 3540087559. (accessed: 11.11.2022) (page 27).

[72] Dwork, Cynthia and Lynch, Nancy and Stockmeyer, Larry. "Consensus in the Presence of Partial Synchrony". In: *J. ACM* 35.2 (1988), 288–323. ISSN: 0004-5411. DOI: 10.1145/42282.42283. URL: https://doi.org/10.1145/42282.42283. (accessed: 11.11.2022) (pages 30, 143, 162).

[73] Castro, Miguel and Liskov, Barbara. "Practical Byzantine Fault Tolerance". In: OSDI '99 (1999), 173–186. (accessed: 12.11.2022) (pages 32, 35, 40, 49, 50, 59, 121, 125, 135, 147, 148, 189).

[74] Nguyen, Thanh Son Lam and Jourjon, Guillaume and Potop-Butucaru, Maria and Thai, Kim Loan. "Impact of network delays on Hyperledger Fabric". In: (2019), 222–227. DOI: 10.1109/INFCOMW.2019.8845168. (accessed: 12.11.2022) (page 33).

[75] Hyperledger Foundation. *Sawtooth PBFT Request for Comments*. URL: https://github.com/hyperledger/sawtooth-rfcs/blob/main/text/0019-pbft-consensus.md. (accessed: 08.11.2022) (pages 33, 133).

[76] Kotla, Ramakrishna and Alvisi, Lorenzo and Dahlin, Mike and Clement, Allen and Wong, Edmund. "Zyzzyva: Speculative Byzantine Fault Tolerance". In: *ACM Trans. Comput. Syst.* 27.4 (2010). ISSN: 0734-2071. DOI: 10.1145/1658357.1658358. URL: https://doi.org/10.1145/1658357.1658358. (accessed: 12.11.2022) (pages 33–35, 145, 147, 148).

[77] Abraham, Ittai and Gueta, Guy and Malkhi, Dahlia and Alvisi, Lorenzo and Kotla, Rama and Martin, Jean-Philippe. "Revisiting Fast Practical Byzantine Fault Tolerance". In: (2017). DOI: 10.48550/ARXIV.1712.01367. URL: https://arxiv.org/abs/1712.01367. (accessed: 14.11.2022) (pages 34, 35).

[78] Clement, Allen and Kapritsos, Manos and Lee, Sangmin and Wang, Yang and Alvisi, Lorenzo and Dahlin, Mike and Riche, Taylor. "Upright Cluster Services". In: SOSP '09 (2009), 277–290. DOI: 10.1145/1629575.1629602. URL: https://doi.org/10.1145/1629575.1629602. (accessed: 14.11.2022) (page 34).

[79] Kursawe, K. "Optimistic Byzantine agreement". In: (2002), 262–267. DOI: 10.1109/RELDIS.2002.1180196. (accessed: 14.11.2022) (page 34).

[80] Martin, Jean-Philippe and Alvisi, Lorenzo. "Fast Byzantine Consensus". In: *IEEE Trans. Dependable Secur. Comput.* 3.3 (2006), 202–215. ISSN: 1545-5971. DOI: 10.1109/TDSC.2006.35. URL: https://doi.org/10.1109/TDSC.2006.35. (accessed: 16.11.2022) (page 34).

[81] Aublin, Pierre-Louis and Guerraoui, Rachid and Knežević, Nikola and Quéma, Vivien and Vukolić, Marko. "The Next 700 BFT Protocols". In: *ACM Trans. Comput. Syst.* 32.4 (2015). ISSN: 0734-2071. DOI: 10.1145/2658994. URL: https://doi.org/10.1145/2658994. (accessed: 16.11.2022) (page 34).

[82] Amiri, Mohammad Javad and Agrawal, Divyakant and Abbadi, Amr El. "Modern Large-Scale Data Management Systems after 40 Years of Consensus". In: (2020), 1794–1797. DOI: 10.1109/ICDE48307.2020.00172. (accessed: 13.11.2022) (pages 34, 35).

[83] Yin, Maofan and Malkhi, Dahlia and Reiter, Michael K. and Gueta, Guy Golan and Abraham, Ittai. "HotStuff: BFT Consensus with Linearity and Responsiveness". In: PODC '19 (2019), 347–356. DOI: 10.1145/3293611.3331591. URL: https://doi.org/10.1145/3293611.3331591. (accessed: 17.11.2022) (pages 35, 36, 121, 149, 215, 218).

[84] Diem Team. *DiemBFT v4: State Machine Replication in the Diem Blockchain*. Diem (Libra,Novi a Facebook Project. 2021. URL: https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2021-08-17.pdf. (accessed: 18.11.2022) (pages 35, 121).

[85] Alqahtani, Salem and Demirbas, Murat. "Bottlenecks in Blockchain Consensus Protocols". In: (2021), 1–8. DOI: 10.1109/COINS51742.2021.9524210. (accessed: 18.11.2022) (pages 35, 139).

[86] Vitalik Buterin. *Casper FFG with one message type, and simpler fork choice rule*. Ethereum Research. 2017. URL: https://ethresear.ch/t/casper-ffg-with-one-message-type-and-simpler-fork-choice-rule/103. (accessed: 18.11.2022) (page 36).

[87] Buchman, Ethan and Kwon, Jae and Milosevic, Zarko. "The latest gossip on BFT consensus". In: (2018). DOI: 10.48550/ARXIV.1807.04938. URL: https://arxiv.org/abs/1807.04938. (accessed: 18.11.2022) (page 37).

[88] Lagaillardie, Nicolas and Djari, Mohamed Aimen and Gürcan, Önder. "A Computational Study on Fairness of the Tendermint Blockchain Protocol". In: *Information* 10.12 (2019). ISSN: 2078-2489. DOI: 10.3390/info10120378. URL: https://www.mdpi.com/2078-2489/10/12/378. (accessed: 18.11.2022) (page 38).

[89] Buterin, Vitalik and Hernandez, Diego and Kamphefner, Thor and Pham, Khiem and Qiao, Zhi and Ryan, Danny and Sin, Juhyeok and Wang, Ying and Zhang, Yan X. *Combining GHOST and Casper*. 2020. DOI: 10.48550/ARXIV.2003.03052. URL: https://arxiv.org/abs/2003.03052. (accessed: 19.11.2022) (pages 38, 218).

[90] Wackerow, Paul. *Gasper*. Ethereum Foundation. 2022. URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/gasper/. (accessed: 20.11.2022) (pages 38–40).

[91] Buterin, Vitalik. *A Proof of Stake Design Philosophy*. Ethereum Foundation. 2016. URL: https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51. (accessed: 20.11.2022) (page 39).

[92] Buterin, Vitalik and Griffith, Virgil. *Casper the Friendly Finality Gadget*. 2017. DOI: 10.48550/ARXIV.1710.09437. URL: https://arxiv.org/abs/1710.09437. (accessed: 20.11.2022) (page 39).

[93] Sompolinsky, Yonatan and Zohar, Aviv. "Secure High-Rate Transaction Processing in Bitcoin". In: (2015). (accessed: 21.11.2022) (pages 39, 48).

[94] Stathakopoulou, Chrysoula and David, Tudor and Pavlovic, Matej and Vukolić, Marko. *Mir-BFT: High-Throughput Robust BFT for Decentralized Networks*. 2019. DOI: 10.48550/ARXIV.1906.05552. URL: https://arxiv.org/abs/1906.05552. (accessed: 21.11.2022) (pages 40, 41, 215).

[95] Neiheiser, Ray and Matos, Miguel and Rodrigues, Luís. "Kauri: Scalable BFT Consensus with Pipelined Tree-Based Dissemination and Aggregation". In: () (pages 40, 195).

[96] Milosevic, Zarko and Biely, Martin and Schiper, André. "Bounded Delay in Byzantine-Tolerant State Machine Replication". In: (2013), 61–70. DOI: 10.1109/SRDS.2013.15. (accessed: 21.11.2022) (page 40).

[97] Crain, Tyler and Natoli, Christopher and Gramoli, Vincent. "Red Belly: A Secure, Fair and Scalable Open Blockchain". In: (2021), 466–483. DOI: 10.1109/SP40001.2021.00087. (accessed: 22.11.2022) (page 40).

[98] Baird, Leemon and Luykx, Atul. "The Hashgraph Protocol: Efficient Asynchronous BFT for High-Throughput Distributed Ledgers". In: (2020), 1–7. DOI: 10.1109/COINS49042.2020.9191430. (accessed: 22.11.2022) (page 40).

[99] Miller, Andrew and Xia, Yu and Croman, Kyle and Shi, Elaine and Song, Dawn. "The Honey Badger of BFT Protocols". In: CCS '16 (2016), 31–42. DOI: 10.1145/2976749.2978399. URL: https://doi.org/10.1145/2976749.2978399. (accessed: 22.11.2022) (pages 40, 149).

[100] Duan, Sisi and Reiter, Michael K. and Zhang, Haibin. "BEAT: Asynchronous BFT Made Practical". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, 2028–2041. ISBN: 9781450356930. DOI: 10.1145/3243734.3243812. URL: https://doi.org/10.1145/3243734.3243812. (accessed: 22.11.2022) (page 40).

[101] Rebello, Gabriel Antonio F. and Camilo, Gustavo F. and Guimarães, Lucas C. B. and de Souza, Lucas Airam C. and Duarte, Otto Carlos M. B. "Security and Performance Analysis of Quorum-based Blockchain Consensus Protocols". In: (2022), 1–7. DOI: 10.1109/CSNet56116.2022.9955597. (accessed: 23.11.2022) (pages 42–45).

[102] Aublin, Pierre-Louis and Mokhtar, Sonia Ben and Quéma, Vivien. "RBFT: Redundant Byzantine Fault Tolerance". In: (2013), 297–306. DOI: 10.1109/ICDCS.2013.53. (accessed: 23.11.2022) (page 42).

[103] Chase, Brad and MacBrough, Ethan. *Analysis of the XRP Ledger Consensus Protocol*. 2018. DOI: 10.48550/ARXIV.1802.07242. URL: https://arxiv.org/abs/1802.07242. (accessed: 23.11.2022) (page 42).

[104] Kim, Minjeong and Kwon, Yujin and Kim, Yongdae. "Is Stellar As Secure As You Think?" In: (2019), 377–385. DOI: 10.1109/EuroSPW.2019.00048. (accessed: 23.11.2022) (page 43).

[105] Wang, Qin and Li, Rujia and Chen, Shiping and Xiang, Yang. "Formal Security Analysis on DBFT Protocol of NEO". In: *Distrib. Ledger Technol.* (2022). ISSN: 2769-6472. DOI: 10.1145/3568314. URL: https://doi.org/10.1145/3568314. (accessed: 24.11.2022) (page 43).

[106] EOS.IO Team. *EOS.IO Technical White Paper*. Block.One. 2018. URL: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. (accessed: 24.11.2022) (page 44).

[107] Zhao, Yijing and Liu, Jieli and Han, Qing and Zheng, Weilin and Wu, Jiajing. "Exploring EOSIO via Graph Characterization". In: (2020). Ed. by Zibin Zheng et al., 475–488. (accessed: 24.11.2022) (page 45).

[108] Veronese, Giuliana Santos and Correia, Miguel and Bessani, Alysson Neves and Lung, Lau Cheuk. "Spin One's Wheels? Byzantine Fault Tolerance with a Spinning Primary". In: (2009), 135–144. DOI: 10.1109/SRDS.2009.36. (accessed: 23.11.2022) (page 45).

[109] Bentov, Iddo and Gabizon, Ariel and Mizrahi, Alex. "Cryptocurrencies Without Proof of Work". In: (2016). Ed. by Jeremy Clark et al., 142–157. (accessed: 25.11.2022) (page 45).

[110] Kiayias, Aggelos and Russell, Alexander and David, Bernardo and Oliynykov, Roman. "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol". In: (2017). Ed. by Jonathan Katz and Hovav Shacham, 357–388. (accessed: 25.11.2022) (pages 45, 46).

[111] David, Bernardo and Gaži, Peter and Kiayias, Aggelos and Russell, Alexander. "Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain". In: (2018). Ed. by Jesper Buus Nielsen and Vincent Rijmen, 66–98. (accessed: 25.11.2022) (pages 45, 46, 51, 90).

[112] Daian, Phil and Pass, Rafael and Shi, Elaine. "Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake". In: (2019). Ed. by Ian Goldberg and Tyler Moore, 23–41. (accessed: 26.11.2022) (pages 45, 47, 51).

[113] Rabin, T. and Ben-Or, M. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority". In: STOC '89 (1989), 73–85. DOI: 10.1145/73007.73014. URL: https://doi.org/10.1145/73007.73014. (accessed: 25.11.2022) (page 45).

[114] Bentov, Iddo and Lee, Charles and Mizrahi, Alex and Rosenfeld, Meni. "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake [Extended Abstract]y". In: *SIGMETRICS Perform. Eval. Rev.* 42.3 (2014), 34–37. ISSN: 0163-5999. DOI: 10.1145/2695533.2695545. URL: https://doi.org/10.1145/2695533.2695545. (accessed: 26.11.2022) (pages 45, 51, 52).

[115] Xiao, Yang and Zhang, Ning and Lou, Wenjing and Hou, Y. Thomas. "A Survey of Distributed Consensus Protocols for Blockchain Networks". In: *IEEE Communications Surveys Tutorials* 22.2 (2020), 1432–1465. DOI: 10.1109/COMST.2020.2969706. (accessed: 24.11.2022) (pages 46, 47, 51).

[116] Pass, Rafael and Shi, Elaine. “The Sleepy Model of Consensus”. In: (2017). Ed. by Tsuyoshi Takagi and Thomas Peyrin, 380–409. (accessed: 26.11.2022) (page 47).

[117] De Angelis, Stefano and Aniello, Leonardo and Lombardi, Federico and Margheri, Andrea and Sassone, V. “PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain”. In: (2018). (accessed: 09.12.2022) (pages 47–49, 81, 133).

[118] Gerrits, Luc and Samuel, Cyril Naves and Kromes, Roland and Verdier, François and Glock, Severine and Guitton-Ouhamou, Patricia. “Experimental Scalability Study of Consortium Blockchains with BFT Consensus for IoT Automotive Use Case”. In: SenSys ’21 (2021), 492–498. DOI: 10.1145/3485730.3493374. URL: https://doi.org/10.1145/3485730.3493374. (accessed: 12.12.2022) (pages 48, 87, 179).

[119] Yu-Te Lin. *Istanbul Byzantine Fault Tolerance*. URL: https://github.com/ethereum/EIPs/issues/650. (accessed: 01.11.2022) (pages 49, 72, 150).

[120] Saltini, Roberto. *Formal Verification of the safety of QBFT*. URL: https://drive.google.com/file/d/1KmDrqxxFlbQGjyzsD\%5FQTTPgONDtg7euB/view. (accessed: 01.11.2022) (page 50).

[121] Henrique Moniz. *The Istanbul BFT Consensus Algorithm*. 2020. arXiv: 2002.03613. URL: https://arxiv.org/abs/2002.03613. (accessed: 11.12.2022) (pages 50, 125, 137, 150).

[122] Dwork, Cynthia and Naor, Moni. “Pricing via Processing or Combatting Junk Mail”. In: (1993). Ed. by Ernest F. Brickell, 139–147. (accessed: 27.11.2022) (page 51).

[123] Buterin, Vitalik and Griffith, Virgil. *Casper the Friendly Finality Gadget*. 2017. arXiv: 1710.09437. URL: http://arxiv.org/abs/1710.09437. (accessed: 29.11.2022) (page 51).

[124] Buterin, Vitalik. *Long-Range Attacks: The Serious Problem With Adaptive Proof of Work*. 2014. URL: https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work. (accessed: 29.11.2022) (page 51).

[125] Gaži, Peter and Kiayias, Aggelos and Russell, Alexander. “Stake-Bleeding Attacks on Proof-of-Stake Blockchains”. In: (2018), 85–92. DOI: 10.1109/CVCBT.2018.00015. (accessed: 30.11.2022) (page 51).

[126] Dunford, Rosie and Su, Quanrong and Tamang, Ekraj. “The Pareto Principle”. In: *The Plymouth Student Scientist* 7 (2014), 140–148. (accessed: 01.12.2022) (page 52).

[127] Chen, Lin and Xu, Lei and Shah, Nolan and Gao, Zhimin and Lu, Yang and Shi, Weidong. “On Security Analysis of Proof-of-Elapsed-Time (PoET)”. In: (2017). Ed. by Paul Spirakis and Philippas Tsigas, 282–297. (accessed: 01.12.2022) (page 52).

[128] Bernstein, Daniel J. *The Salsa20 Family of Stream Ciphers*. Ed. by Matthew Robshaw and Olivier Billet. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, 84–97. ISBN: 978-3-540-68351-3. DOI: 10.1007/978-3-540-68351-3_8. URL: https://doi.org/10.1007/978-3-540-68351-3\%5F8. (accessed: 03.12.2022) (page 53).

[129] Swamynathan, Gayatri and Almeroth, Kevin and Zhao, Ben. "The design of a reliable reputation system". In: *Electronic Commerce Research* 10 (Dec. 2010), 239–270. DOI: 10.1007/s10660-010-9064-y. (accessed: 03.12.2022) (page 53).

[130] Anton Kolonin and Ben Goertzel and Deborah Duong and Matt Ikle. *A Reputation System for Artificial Societies*. 2018. arXiv: 1806.07342 [cs.AI]. (accessed: 04.12.2022) (page 53).

[131] Oladotun Aluko and Anton Kolonin. "Proof-of-Reputation: An Alternative Consensus Mechanism for Blockchain Systems". In: *CoRR* abs/2108.03542 (2021). arXiv: 2108.03542. URL: https://arxiv.org/abs/2108.03542. (accessed: 06.12.2022) (pages 54, 55).

[132] Yu, Jiangshan and Kozhaya, David and Decouchant, Jeremie and Esteves-Verissimo, Paulo. "RepuCoin: Your Reputation Is Your Power". In: *IEEE Transactions on Computers* 68.8 (2019), 1225–1237. DOI: 10.1109/TC.2019.2900648. (accessed: 06.12.2022) (page 54).

[133] Kleinrock, Leonard and Ostrovsky, Rafail and Zikas, Vassilis. "Proof-of-Reputation Blockchain with Nakamoto Fallback". In: (2020), 16–38. DOI: 10.1007/978-3-030-65277-7_2. URL: https://doi.org/10.1007/978-3-030-65277-7\%5F2. (accessed: 06.12.2022) (page 55).

[134] Asharov, Gilad and Lindell, Yehuda and Zarosim, Hila. "Fair and Efficient Secure Multiparty Computation with Reputation Systems". In: (2013). Ed. by Kazue Sako and Palash Sarkar, 201–220. (accessed: 06.12.2022) (page 55).

[135] Gai, Fangyu and Wang, Baosheng and Deng, Wenping and Peng, Wei. "Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network". In: (2018). Ed. by Jian Pei et al., 666–681. (accessed: 06.12.2022) (pages 56, 57).

[136] H. Li and A. Clement and E. Wong and J. Napper and I. Roy and L. Alvisi and M. Dahlin. "BAR Gossip". In: (Nov. 2006). (accessed: 08.12.2022) (pages 57, 60).

[137] Wong, Edmund L. and Leners, Joshua B. and Alvisi, Lorenzo. "It's on Me! The Benefit of Altruism in BAR Environment". In: DISC'10 (2010), 406–420. (accessed: 08.12.2022) (page 57).

[138] Aiyer, Amitanand S. and Alvisi, Lorenzo and Clement, Allen and Dahlin, Mike and Martin, Jean-Philippe and Porth, Carl. "BAR Fault Tolerance for Cooperative Services". In: SOSP '05 (2005), 45–58. DOI: 10.1145/1095810.1095816. URL: https://doi.org/10.1145/1095810.1095816. (accessed: 08.12.2022) (pages 57–59).

[139] Joã Vilaça, Xavier and Leitão. "N-party BAR Transfer". In: () (page 57).

[140] Groupe Renault. *The blockchain, transformation vector for the future of the automotive industry*. URL: https://www.renaultgroup.com/en/news-on-air/news/the-blockchain-transformation-vector-for-the-future-of-the-automotive-industry/. (accessed: 24.03.2023) (page 64).

[141] Groupe Renault. *XCEED: a new blockchain solution for Renault plants in Europe.* URL: https://www.renaultgroup.com/en/news-on-air/news/xceed-a-new-blockchain-solution-for-renault-plants-in-europe/. (accessed: 24.03.2023) (page 64).

[142] Bernhart, Wolfgang and Baum, Markus and Meissner, Falk and Shirokinskiy, Konstantin. *Roland Berger Report: The future of the automotive software industry: Spend, trends and how to transform.* URL: https://content.rolandberger.com/hubfs/07\%5Fpresse/Roland\%5FBerger\%5FArticle\%5FComputer\%5Fon\%5FWheels\%5F4\%5F2022.pdf. (accessed: 24.03.2023) (page 65).

[143] International Standards Organization. *ISO/TC 307 Blockchain and distributed ledger technologies.* URL: https://www.iso.org/home.isoDocumentsDownload.do?t=FfjDIuj1cCQFy5qRq3trl4VYO8gQI5az388\%5FAqybXhJIImrBTF0-C5uxwIJQtjT7&CSRFTOKEN=QUFB-08O4-7YSX-540R-QUCP-FQ86-Q47Q-CS25. (accessed: 27.03.2023) (pages 65, 86).

[144] Volkwsagen Group. *Protecting people and the environment with blockchain.* URL: https://www.volkswagenag.com/en/news/stories/2019/04/protecting-people-and-the-environment-with-blockchain.html#. (accessed: 27.03.2023) (page 65).

[145] Volkwsagen Group. *From mine to factory: Volkswagen makes supply chain transparent with blockchain.* URL: https://www.volkswagenag.com/en/news/2019/04/volkswagen\%5Fblockchain\%5Fminespider.html. (accessed: 27.03.2023) (page 65).

[146] Dargahi, Tooska and Ahmadvand, Hossein and Alraja, Mansour Naser and Yu, Chia-Mu. "Integration of Blockchain with Connected and Autonomous Vehicles: Vision and Challenge". In: *J. Data and Information Quality* 14.1 (2021). ISSN: 1936-1955. DOI: 10.1145/3460003. URL: https://doi.org/10.1145/3460003. (accessed: 28.03.2023) (page 65).

[147] Fraga-Lamas, Paula and Fernández-Caramés, Tiago. "A Review on Blockchain Technologies for an Advanced and Cyber-Resilient Automotive Industry". In: *IEEE Access* 7 (2019), 17578–17598. DOI: 10.1109/ACCESS.2019.2895302. (accessed: 29.03.2023) (page 66).

[148] Besançon, Léo and Silva, Catarina Ferreira Da and Ghodous, Parisa. "Towards Blockchain Interoperability: Improving Video Games Data Exchange". In: (2019), 81–85. DOI: 10.1109/BLOC.2019.8751347. (accessed: 29.03.2023) (page 66).

[149] Dib, Omar and Brousmiche, Kei-Léo et al. "Consortium blockchains: Overview, applications and challenges". In: () (page 67).

[150] Gavin Wood. *Ethereum: A secure decentralised generalised transaction ledger EIP-150 REVISION (759dccd - 2017-08-07).* 2017. URL: https://ethereum.github.io/yellowpaper/paper.pdf. (accessed: 29.03.2023) (page 68).

[151] "Multichain Coin Sciences Limited". *"Multichain Permissions consensus"*. URL: https://www.multichain.com/developers/permissions-consensus/. (accessed: 20.05.2023) (page 68).

[152] Universite Cote D'Azur. *SIM - Smart IoT for Mobility*. URL: https://univ-cotedazur.eu/sim. (accessed: 20.01.2023) (page 72).

[153] M. Chuan. *Istanbul BFT's design cannot successfully tolerate fail-stop failures*. URL: https://github.com/ConsenSys/quorum/issues/305. (accessed: 30.03.2023) (page 72).

[154] Roberto Saltini. *Correctness Analysis of IBFT*. 2019. arXiv: 1901.07160. URL: http://arxiv.org/abs/1901.07160. (accessed: 31.03.2023) (page 72).

[155] Castro, Miguel and Liskov, Barbara. "Practical Byzantine Fault Tolerance and Proactive Recovery". In: *ACM Trans. Comput. Syst.* 20.4 (2002), 398–461. ISSN: 0734-2071. DOI: 10.1145/571637.571640. URL: https://doi.org/10.1145/571637.571640. (accessed: 26.02.2023) (pages 73, 172, 179).

[156] Gerrits, Luc and Kilimou, Edouard and Kromes, Roland and Faure, Lionel and Verdier, François. "A Blockchain cloud architecture deployment for an industrial IoT use case". In: (2021), 1–6. DOI: 10.1109/COINS51742.2021.9524264. (accessed: 01.04.2023) (pages 76, 77).

[157] Gerrits, Luc and Samuel, Cyril Naves. *Experimental Scalability Study of Consortium Blockchains with BFT Consensus for IoT Automotive Use Case*. Sept. 2021. URL: https://github.com/projet-SIM/acm-blocksys-2021. (accessed: 30.03.2023) (page 77).

[158] Samuel, Cyril Naves and Glock, Severine and Verdier, François and Guitton-Ouhamou, Patricia. "Choice of Ethereum Clients for Private Blockchain: Assessment from Proof of Authority Perspective". In: (2021), 1–5. DOI: 10.1109/ICBC51069.2021.9461085. (accessed: 06.03.2023) (pages 78, 87, 212).

[159] Samuel, Cyril Naves. *Geth-Node Stalled at EVM Layer*. 2020. URL: https://github.com/ethereum/go-ethereum/issues/21158. (accessed: 01.04.2023) (pages 80, 232).

[160] Samuel, Cyril Naves. *Geth-Block Period 1 Fork*. 2020. URL: https://github.com/ethereum/go-ethereum/issues/21191. (accessed: 01.04.2023) (pages 80, 232, 234).

[161] Samuel, Cyril Naves. *Geth-Clique PoA Issues*. 2020. URL: https://github.com/ethereum/go-ethereum/issues/18402#issuecomment-637328489. (accessed: 02.04.2023) (pages 80, 81, 232).

[162] Baecker, Julius and Engert, Martin and Pfaff, Matthias and Krcmar, Helmut. "Business Strategies for Data Monetization: Deriving Insights from Practice". In: (Mar. 2020). DOI: https://doi.org/10.30844/wi_2020_j3-baecker. URL: https://library.gito.de/oa\%5Fwi2020-j3.html. (accessed: 05.04.2023) (page 84).

[163] Ofulue, Joan and Benyoucef, Morad. "Data monetization: insights from a technology-enabled literature review and research agenda". In: *Management Review Quarterly* (Nov. 2022), DOI: 10.1007/s11301-022-00309-1. (accessed: 05.04.2023) (page 85).

[164] "MHP and Riddle&CODE". *"The Automotive Sector and Blockchain"*. URL: "https://uploads-ssl.webflow.com/612c9eb00ca5ae6d6482c315/62097b9e82558919ad527e77\%5FRIDDLE%26CODE-MHP-The-Automotive-Sector-Blockchain.pdf". (accessed: 04.04.2023) (page 86).

[165] European Union Blockchain Observatory Forum. *Blockchain Applications in the Automotive Sector*. URL: https://www.eublockchainforum.eu/sites/default/files/reports/eubof\%5Fautomotive\%5F2022\%5FFINAL.pdf. (accessed: 27.03.2023) (page 87).

[166] "Continental Automotive". *"Software-Defined Vehicle"*. URL: "https://www.continental-automotive.com/en-gl/Passenger-Cars/Technology-Trends/software-defined-vehicles". (accessed: 04.04.2023) (page 87).

[167] Samuel, Cyril Naves and Severine, Glock and David, Bercovitz and Verdier, François and Patricia, Guitton-Ouhamou. "Automotive Data Certification Problem: A View on Effective Blockchain Architectural Solutions". In: (2020), 0167–0173. DOI: 10.1109/IEMCON51383.2020.9284886. (accessed: 10.03.2023) (page 87).

[168] "Blair, Tesler (Insight)". *"What is Privacy by Design and by Default?"*. URL: https://www.morganlewis.com/pubs/2019/03/the-edata-guide-to-gdpr-what-is-privacy-by-design-and-by-default. (accessed: 07.04.2023) (page 88).

[169] "European Commission". *"What does data protection 'by design' and 'by default' mean?"*. URL: https://commission.europa.eu/law/law-topic/data-protection/reform/rules-business-and-organisations/obligations/what-does-data-protection-design-and-default-mean\%5Fen. (accessed: 07.04.2023) (page 88).

[170] Khizar Hameed and Mutaz Barika and Saurabh Garg and Muhammad Bilal Amin and Byeong Kang. "A taxonomy study on securing Blockchain-based Industrial applications: An overview, application perspectives, requirements, attacks, countermeasures, and open issues". In: *Journal of Industrial Information Integration* 26 (2022), p. 100312. ISSN: 2452-414X. DOI: https://doi.org/10.1016/j.jii.2021.100312. URL: https://www.sciencedirect.com/science/article/pii/S2452414X21001060. (accessed: 08.04.2023) (page 88).

[171] "Parity Technologies". *"Substrate Technology"*. URL: https://substrate.io/technology/. (accessed: 07.04.2023) (page 88).

[172] "Parity Technologies". *"Authority Round"*. URL: https://paritytech.github.io/substrate/master/sc\%5Fconsensus\%5Faura/index.html. (accessed: 07.04.2023) (page 89).

[173] "Kilinc Alper, Handan". *"Blind Assignment for Blockchain Extension (BABE)"*. URL: https://research.web3.foundation/en/latest/polkadot/block-production/Babe.html. (accessed: 07.04.2023) (page 90).

[174] "Petrowski,Joe (Parity Technologies)". *"Blind Assignment for Blockchain Extension (BABE)"*. URL: https://polkadot.network/blog/polkadot-consensus-part-3-babe?ref=cms.polkadot.network. (accessed: 07.04.2023) (page 90).

[175] "Stewart, Alistair". *"GRANDPA FINALITY"*. URL: https://research.web3.foundation/en/latest/polkadot/finality.html. (accessed: 07.04.2023) (pages 90, 92).

[176] "Bosch". *"Road Signature"*. URL: https://www.bosch-mobility.com/en/solutions/automated-driving/road-signature/. (accessed: 07.04.2023) (page 93).

[177] "Blockchain pour l'Internet des véhicules : une solution IoT dé". PhD thesis (pages 103, 110).

[178] Jabbar, Rateb and Kharbeche, Mohamed and Al-Khalifa, Khalifa and Krichen, Moez and Barkaoui, Kamel. "Blockchain for the Internet of Vehicles: A Decentralized IoT Solution for Vehicles Communication Using Ethereum". In: *Sensors* 20.14 (2020). ISSN: 1424-8220. DOI: 10.3390/s20143928. URL: https://www.mdpi.com/1424-8220/20/14/3928 (page 103).

[179] "Stack Exchange". *"BABE GRANDPA Stalled"*. URL: https://substrate.stackexchange.com/questions/214/recovering-from-stalled-finality-babe-grandpa. (accessed: 17.04.2023) (pages 106, 109).

[180] Wang, Yongge. "The Adversary Capabilities Innbsp;Practical Byzantine Fault Tolerance". In: (2021), 20–39. DOI: 10.1007/978-3-030-91859-0_2. URL: https://doi.org/10.1007/978-3-030-91859-0\%5F2. (accessed: 18.04.2023) (pages 106, 109).

[181] Damien Desfontaines and Balázs Pejó. "SoK: Differential privacies". In: () (page 110).

[182] Buterin, Vitalik and Weiss, Yoav and Gazso, Kristof and Patel, Namra and Tirosh, Dror and Nacson, Shahaf and Hess, Tjaden. *ERC-4337: Account Abstraction Using Alt Mempool*. URL: https://eips.ethereum.org/EIPS/eip-4337. (accessed: 17.03.2023) (page 110).

[183] H. Abbas and M. Caprolu and R. Di Pietro. "Analysis of Polkadot: Architecture, Internals, and Contradictions". In: (2022), 61–70. DOI: 10.1109/Blockchain55522.2022.00019. URL: https://doi.ieeecomputersociety.org/10.1109/Blockchain55522.2022.00019. (accessed: 18.04.2023) (page 110).

[184] A. Albshri and A. Alzubaidi and B. Awaji and E. Solaiman. "Blockchain Simulators: A Systematic Mapping Study". In: (2022), 284–294. DOI: 10.1109/SCC55611.2022.00049. URL: https://doi.ieeecomputersociety.org/10.1109/SCC55611.2022.00049. (accessed: 14.12.2022) (pages 115, 117).

[185] Faria, Carlos and Correia, Miguel. "BlockSim: Blockchain Simulator". In: (2019), 439–446. DOI: 10.1109/Blockchain.2019.00067. (accessed: 15.12.2022) (pages 118–120).

[186] Foytik, Peter and Shetty, Sachin and Gochhayat, Sarada Prasad and Herath, Eranga and Tosh, Deepak and Njilla, Laurent. "A Blockchain Simulator for Evaluating Consensus Algorithms in Diverse Networking Environments". In: (2020), 1–12. DOI: 10.22360/SpringSim.2020.CSE.003. (accessed: 15.12.2022) (page 118).

[187] Ongaro, Diego and Ousterhout, John. "In Search of an Understandable Consensus Algorithm". In: USENIX ATC'14 (2014), 305–320. (accessed: 15.12.2022) (page 118).

[188] Polge, Julien and Ghatpande, Sankalp and Kubler, Sylvain and Robert, Jérémy and Le Traon, Yves. "BlockPerf: A Hybrid Blockchain Emulator/Simulator Framework". In: *IEEE Access* 9 (2021), 107858–107872. DOI: 10.1109/ACCESS.2021.3101044. (accessed: 16.12.2022) (pages 119, 120).

[189] Wang, Bozhi and Chen, Shiping and Yao, Lina and Wang, Qin. "ChainSim: A P2P Blockchain Simulation Framework". In: (2021). Ed. by Ke Xu et al., 1–16. (accessed: 16.12.2022) (page 120).

[190] Wang, Ping-Lun and Chao, Tzu-Wei and Wu, Chia-Chien and Hsiao, Hsu-Chun. "Tool: An Efficient and Flexible Simulator for Byzantine Fault-Tolerant Protocols". In: (2022), 287–294. DOI: 10.1109/DSN53405.2022.00038. (accessed: 22.01.2023) (pages 120, 121, 139).

[191] Abraham, Ittai and Devadas, Srinivas and Dolev, Danny and Nayak, Kartik and Ren, Ling. "Synchronous Byzantine Agreement with Expected O(1) Rounds, Expected Communication, and Optimal Resilience". In: (2019), 320–334. DOI: 10.1007/978-3-030-32101-7_20. URL: https://doi.org/10.1007/978-3-030-32101-7\%5F20. (accessed: 22.01.2023) (page 121).

[192] Chen, Jing and Gorbunov, Sergey and Micali, Silvio and Vlachos, Georgios. *ALGORAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement*. Cryptology ePrint Archive, Paper 2018/377. https://eprint.iacr.org/2018/377. 2018. URL: https://eprint.iacr.org/2018/377. (accessed: 23.01.2023) (pages 121, 148, 149).

[193] Bracha, Gabriel. "Asynchronous Byzantine Agreement Protocols". In: *Inf. Comput.* 75.2 (1987), 130–143. ISSN: 0890-5401. DOI: 10.1016/0890-5401(87)90054-X. URL: https://doi.org/10.1016/0890-5401(87)90054-X. (accessed: 22.01.2023) (page 121).

[194] Casanova, Henri and Giersch, Arnaud and Legrand, Arnaud and Quinson, Martin and Suter, Frédéric. "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms". In: () (page 122).

[195] Stewart, Alistair. "Poster: GRANDPA Finality Gadget". In: CCS '19 (2019), 2649–2651. DOI: 10.1145/3319535.3363278. URL: https://doi.org/10.1145/3319535.3363278. (accessed: 23.01.2023) (page 133).

[196] Alper, Handan Kılınç. *BABE Consensus*. URL: https://research.web3.foundation/en/latest/polkadot/block-production/Babe.html#. (accessed: 20.01.2023) (page 133).

[197] Shi, Elaine. "Analysis of Deterministic Longest-Chain Protocols". In: (2019), 122–12213. DOI: 10.1109/CSF.2019.00016. (accessed: 24.01.2023) (page 133).

[198] Tang, Song and Wang, Zhiqiang and Jiang, Jian and Ge, Suli and Tan, GaiFang. "Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain". In: *Scientific Reports* 12.1 (2022), p. 4426. ISSN: 2045-2322. DOI: 10.1038/s41598-022-08587-1. URL: https://doi.org/10.1038/s41598-022-08587-1. (accessed: 24.01.2023) (pages 135–137, 139).

[199] Wang, Yong and Zhong, Meiling and Cheng, Tong. "Research on PBFT consensus algorithm for grouping based on feature trust". In: *Scientific Reports* 12 (2022), p. 12515. DOI: 10.1038/s41598-022-15282-8. (accessed: 12.03.2023) (pages 135–137).

[200] Messadi, Ines and Becker, Markus Horst and Bleeke, Kai and Jehl, Leander and Mokhtar, Sonia Ben and Kapitza, Rüdiger. "SplitBFT: Improving Byzantine Fault Tolerance Safety Using Trusted Compartments". In: Middleware '22 (2022), 56–68. DOI: 10.1145/3528535.3531516. URL: https://doi.org/10.1145/3528535.3531516. (accessed: 26.01.2023) (page 139).

[201] Sit, Man-Kit and Bravo, Manuel and István, Zsolt. "An Experimental Framework for Improving the Performance of BFT Consensus for Future Permissioned Blockchains". In: DEBS '21 (2021), 55–65. DOI: 10.1145/3465480.3466922. URL: https://doi.org/10.1145/3465480.3466922. (accessed: 25.01.2023) (page 139).

[202] Cristian, F. and Aghili, H. and Strong, R. and Volev, D. "ATOMIC BROADCAST: FROM SIMPLE MESSAGE DIFFUSION TO BYZANTINE AGREEMENT". In: (1995), 431–. DOI: 10.1109/FTCSH.1995.532668. (accessed: 12.03.2023) (page 143).

[203] Sedlmeier, Philipp and Schleger,Johannes and Helm, Max. "Atomic Broadcasts and Consensus: A Survey". In: 12 (Nov. 2020). URL: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2020-11-1/NET-2020-11-1\%5F19.pdf. (accessed: 12.03.2023) (page 143).

[204] Cachin, Christian and Kursawe, Klaus and Petzold, Frank and Shoup, Victor. "Secure and Efficient Asynchronous Broadcast Protocols". In: CRYPTO '01 (2001), 524–541. (accessed: 12.03.2023) (page 143).

[205] Miller, Andrew and Xia, Yu and Croman, Kyle and Shi, Elaine and Song, Dawn. "The Honey Badger of BFT Protocols". In: CCS '16 (2016), 31–42. DOI: 10.1145/2976749.2978399. URL: https://doi.org/10.1145/2976749.2978399. (accessed: 11.03.2023) (page 143).

[206] Ariely, Dan and Davis, Michael. *The (honest) truth about dishonesty : how we lie to everyone–especially ourselves*. Vol. 6. 2012. (accessed: 07.02.2023) (page 144).

[207] Buterin, Vitalik. *A Proof of Stake Design Philosophy*. URL: https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51. (accessed: 06.02.2023) (page 144).

[208] Vukolić, Marko. "The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication". In: (2015), 112–125. DOI: 10.1007/978-3-319-39028-4_9. URL: https://doi.org/10.1007/978-3-319-39028-4\%5F9. (accessed: 09.03.2023) (pages 144–146, 148, 151).

[209] Lone, Auqib and Mir, Roohie. "Evaluating Quality-of-Service in Blockchains Using Modelling and Simulation Tools". In: *International Journal of Computing and Digital Systems* 10 (2020), DOI: 10.12785/ijcds/100103. (accessed: 07.02.2023) (page 145).

[210] Hafid, Abdelatif and Hafid, Abdelhakim Senhaji and Samih, Mustapha. "Scaling Blockchains: A Comprehensive Survey". In: *IEEE Access* 8 (2020), 125244–125262. DOI: 10.1109/ACCESS.2020.3007251. (accessed: 20.02.2023) (page 145).

[211] Aublin, Pierre-Louis and Guerraoui, Rachid and Knežević, Nikola and Quéma, Vivien and Vukolić, Marko. "The Next 700 BFT Protocols". In: *ACM Trans. Comput. Syst.* 32.4 (2015). ISSN: 0734-2071. DOI: 10.1145/2658994. URL: https://doi.org/10.1145/2658994. (accessed: 09.03.2023) (page 145).

[212] Golan Gueta, Guy and Abraham, Ittai and Grossman, Shelly and Malkhi, Dahlia and Pinkas, Benny and Reiter, Michael and Seredinschi, Dragos-Adrian and Tamir, Orr and Tomescu, Alin. "SBFT: A Scalable and Decentralized Trust Infrastructure". In: (2019), 568–580. DOI: 10.1109/DSN.2019.00063. (accessed: 17.03.2023) (pages 145, 149).

[213] OpenEthereum. *Aura - Authority Round*. URL: https://openethereum.github.io/Aura. (accessed: 06.03.2023) (pages 146, 150, 189).

[214] Dang, Hung and Dinh, Tien Tuan Anh and Loghin, Dumitrel and Chang, Ee-Chien and Lin, Qian and Ooi, Beng Chin. "Towards Scaling Blockchain Systems via Sharding". In: SIGMOD '19 (2019), 123–140. DOI: 10.1145/3299869.3319889. URL: https://doi.org/10.1145/3299869.3319889. (accessed: 07.03.2023) (page 146).

[215] Jalili Marandi,Parisa and Eduardo Benevides Bezerra, Carlos and Pedone, Fernando. "Rethinking State-Machine Replication for Parallelism". In: *2014 IEEE 34th International Conference on Distributed Computing Systems* (2013), 368–377. (accessed: 10.03.2023) (page 146).

[216] Kapritsos, Manos and Wang, Yang and Quema, Vivien and Clement, Allen and Alvisi, Lorenzo and Dahlin, Mike. "All about Eve: Execute-Verify Replication for Multi-Core Servers". In: OSDI'12 (2012), 237–250. (accessed: 09.03.2023) (page 146).

[217] Bano, Shehar and Sonnino, Alberto and Al-Bassam, Mustafa and Azouvi, Sarah and McCorry, Patrick and Meiklejohn, Sarah and Danezis, George. "SoK: Consensus in the Age of Blockchains". In: AFT '19 (2019), 183–198. DOI: 10.1145/3318041.3355458. URL: https://doi.org/10.1145/3318041.3355458. (accessed: 19.03.2023) (pages 146, 147).

[218] Luu, Loi and Narayanan, Viswesh and Zheng, Chaodong and Baweja, Kunal and Gilbert, Seth and Saxena, Prateek. "A Secure Sharding Protocol For Open Blockchains". In: CCS '16 (2016), 17–30. DOI: 10.1145/2976749.2978389. URL: https://doi.org/10.1145/2976749.2978389. (accessed: 19.03.2023) (page 146).

[219] Gilad, Yossi and Hemo, Rotem and Micali, Silvio and Vlachos, Georgios and Zeldovich, Nickolai. "Algorand: Scaling Byzantine Agreements for Cryptocurrencies". In: SOSP '17 (2017), 51–68. DOI: 10.1145/3132747.3132757. URL: https://doi.org/10.1145/3132747.3132757. (accessed: 08.03.2023) (page 146).

[220] Jalalzai, Mohammad M. and Busch, Costas and Richard, Golden G. "Proteus: A Scalable BFT Consensus Protocol for Blockchains". In: (2019), 308–313. DOI: 10.1109/Blockchain.2019.00048. (accessed: 11.03.2023) (page 146).

[221] Conti, Mauro and Gangwal, Ankit and Todero, Michele. "Blockchain Trilemma Solver Algorand Has Dilemma over Undecidable Messages". In: ARES '19 (2019). DOI: 10.1145/3339252.3339255. URL: https://doi.org/10.1145/3339252.3339255. (accessed: 08.03.2023) (page 146).

[222] Liu, Yu. *Investigating Byzantine Agreement Consensus Algorithm of Algorand*. 2020. URL: https://opus.lib.uts.edu.au/bitstream/10453/140228/2/02whole.pdf. (accessed: 09.03.2023) (page 146).

[223] Ben-Or, Michael. "Another Advantage of Free Choice (Extended Abstract): Completely Asynchronous Agreement Protocols". In: PODC '83 (1983), 27–30. DOI: 10.1145/800221.806707. URL: https://doi.org/10.1145/800221.806707. (accessed: 19.03.2023) (page 147).

[224] István, Zsolt and Sidler, David and Alonso, Gustavo and Vukolic, Marko. "Consensus in a Box: Inexpensive Coordination in Hardware". In: NSDI'16 (2016), 425–438. (accessed: 16.03.2023) (page 147).

[225] Poke, Marius and Hoefler, Torsten. "DARE: High-Performance State Machine Replication on RDMA Networks". In: HPDC '15 (2015), 107–118. DOI: 10.1145/2749246.2749267. URL: https://doi.org/10.1145/2749246.2749267. (accessed: 16.03.2023) (page 147).

[226] Veronese, Giuliana Santos and Correia, Miguel and Bessani, Alysson Neves and Lung, Lau Cheuk. "Spin One's Wheels? Byzantine Fault Tolerance with a Spinning Primary". In: (2009), 135–144. DOI: 10.1109/SRDS.2009.36. (accessed: 17.03.2023) (page 147).

[227] Arun, Balaji and Peluso, Sebastiano and Ravindran, Binoy. "ezBFT: Decentralizing Byzantine Fault-Tolerant State Machine Replication". In: (2019), 565–577. DOI: 10.1109/ICDCS.2019.00063. (accessed: 17.03.2023) (page 147).

[228] Rocket, Team and Yin, Maofan and Sekniqi, Kevin and van Renesse, Robbert and Sirer, Emin Gün. *Scalable and Probabilistic Leaderless BFT Consensus through Metastability*. 2019. DOI: 10.48550/ARXIV.1906.08936. URL: https://arxiv.org/abs/1906.08936. (accessed: 18.03.2023) (page 148).

[229] Berger, Christian and Reiser, Hans P. "Scaling Byzantine Consensus: A Broad Analysis". In: SERIAL'18 (2018), 13–18. DOI: 10.1145/3284764.3284767. URL: https://doi.org/10.1145/3284764.3284767. (accessed: 18.03.2023) (pages 148, 149).

[230] Kokoris-Kogias, Eleftherios and Jovanovic, Philipp and Gailly, Nicolas and Khoffi, Ismail and Gasser, Linus and Ford, Bryan. "Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing". In: SEC'16 (2016), 279–296. (accessed: 18.03.2023) (page 148).

[231] Li, Peilun and Wang, Guosai and Chen, Xiaoqi and Long, Fan and Xu, Wei. "Gosig: A Scalable and High-Performance Byzantine Consensus for Consortium Blockchains". In: SoCC '20 (2020), 223–237. DOI: 10.1145/3419111.3421272. URL: https://doi.org/10.1145/3419111.3421272. (accessed: 19.03.2023) (pages 148, 216).

[232] E. Syta and I. Tamas and D. Visher and D. Wolinsky and P. Jovanovic and L. Gasser and N. Gailly and I. Khoffi and B. Ford. "Keeping Authorities quot;Honest or Bustquot; with Decentralized Witness Cosigning". In: (2016), 526–545. ISSN: 2375-1207. DOI: 10.1109/SP.2016.38. URL: https://doi.ieeecomputersociety.org/10.1109/SP.2016.38. (accessed: 18.03.2023) (page 149).

[233] Schnorr, C. P. "Efficient Signature Generation by Smart Cards". In: *J. Cryptol.* 4.3 (1991), 161–174. ISSN: 0933-2790. DOI: 10.1007/BF00196725. URL: https://doi.org/10.1007/BF00196725. (accessed: 19.03.2023) (page 149).

[234] Emin Croman, Kyle and Decker, Christian and Eyal, Ittay and Gencer, Adem Efe and Juels, Ari and Kosba, Ahmed and Miller, Andrew and Saxena, Prateek and Shi, Elaine and Gün Sirer, Dawn Song, and Roger Wattenhofer. "On Scaling Decentralized Blockchains". In: () (page 151).

[235] David, Jacques Louis. *The Death of Socrates.* URL: https://www.metmuseum.org/art/collection/search/436105. (accessed: 16.02.2023) (page 152).

[236] Koneru, Anuradha. *A brief notes on Utilitarianism: A study on Bentham and J.S.Mill views.* URL: https://www.legalserviceindia.com/legal/article-3093-a-brief-notes-on-utilitarianism-a-study-on-bentham-and-j-s-mill-views.html. (accessed: 20.02.2023) (page 153).

[237] Delacroix, Eugène. *The Good Samaritan.* URL: https://commons.wikimedia.org/wiki/File:The\%5FGood\%5FSamaritan\%5F%28Delacroix\%5F1849%29.jpg. (accessed: 16.02.2023) (page 153).

[238] Simpson, Adam. *Surveillance State.* URL: https://www.nytimes.com/2013/07/21/books/review/the-panopticon-by-jenni-fagan.html. (accessed: 16.02.2023) (page 155).

[239] Mathisen, Ruben B. "Affluence and Influence in a Social Democracy". In: *American Political Science Review* 117.2 (2023), 751–758. DOI: 10.1017/S0003055422000739. (accessed: 12.04.2023) (page 156).

[240] Vecellio, Tiziano. *Sisyphus.* URL: https://www.museodelprado.es/coleccion/obra-de-arte/sisifo/bb56eb47-052f-4e15-8e46-75a3f18b13ad. (accessed: 16.02.2023) (page 156).

[241] Caredda, Sergio. *The Magic of Murmuration and Self-Management in Organizations.* URL: https://sergiocaredda.eu/organisation/organisation-design/the-magic-of-murmuration-and-self-management-in-organizations/. (accessed: 21.02.2023) (page 158).

[242] Young, George and Scardovi, Luca and Cavagna, Andrea and Giardina, Irene and Leonard, Naomi. "Starling Flock Networks Manage Uncertainty in Consensus at Low Cost". In: *PLoS computational biology* 9 (2013), e1002894. DOI: 10.1371/journal.pcbi.1002894. (accessed: 23.02.2023) (page 158).

[243] Dolgin, Elie. *The secret social lives of viruses*. 2019. DOI: 10.1038/d41586-019-01880-6. (accessed: 26.03.2023) (page 158).

[244] Rosenberg, Louis and Willcox, Gregg. "Artificial Swarms find Social Optima : (Late Breaking Report)". In: (2018), 174–178. DOI: 10.1109/COGSIMA.2018.8423987. (accessed: 23.02.2023) (page 158).

[245] Rosenberg, Louis and Baltaxe, David. "Setting group priorities — Swarms vs votes". In: (2016), 1–4. DOI: 10.1109/SHBI.2016.7780279. (accessed: 23.02.2023) (page 158).

[246] Kaur, Komalpreet and Kumar, Yogesh. "Swarm Intelligence and its applications towards Various Computing: A Systematic Review". In: (2020), 57–62. DOI: 10.1109/ICIEM48762.2020.9160177. (accessed: 23.02.2023) (page 158).

[247] Hildenbrandt, H. and Carere, C. and Hemelrijk, C.K. "Self-organized aerial displays of thousands of starlings: a model". In: *Behavioral Ecology* 21.6 (Oct. 2010), 1349–1359. ISSN: 1045-2249. DOI: 10.1093/beheco/arq149. eprint: https://academic.oup.com/beheco/article-pdf/21/6/1349/13894081/arq149.pdf. URL: https://doi.org/10.1093/beheco/arq149. (accessed: 23.02.2023) (page 158).

[248] Cooper, Ashley. *Starlings flying to roost near Kendal in Cumbria*. URL: https://cosmosmagazine.com/nature/modelling-birds-flock-speed/. (accessed: 16.02.2023) (page 158).

[249] Elleithy, Khaled and Blagovic, Drazen and Cheng, Wang and Sideleau, Paul. "Denial of Service Attack Techniques: Analysis, Implementation and Comparison". In: *Journal of Systemics, Cybernetics and Informatics* 3 (2006), 66–71. (accessed: 25.02.2023) (page 166).

[250] StackOverflow. *PBFT: Why cant the replicas perform the request after 2/3 have prepared? why do we need commit phase?* 2018. URL: https://stackoverflow.com/questions/51125238/pbft-why-cant-the-replicas-perform-the-request-after-2-3-have-prepared-why-do. (accessed: 26.02.2023) (page 172).

[251] Attiya, Hagit and Welch, Jennifer. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. Hoboken, NJ, USA: John Wiley amp; Sons, Inc., 2004. ISBN: 0471453242. (accessed: 27.02.2023) (page 185).

[252] Lynch, Nancy A. *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. ISBN: 9780080504704. (accessed: 27.02.2023) (page 185).

[253] Brewer, Eric. *CAP twelve years later: How the "rules" have changed*. 2012. DOI: 10.1109/MC.2012.37. (accessed: 13.12.2022) (page 188).

[254] Longchamp, Yves and Deshpande, Saurabh and Mehra, Ujjwal. *The Blockchain Trilemma*. 2020. URL: https://theblockchaintest.com/uploads/resources/SEBA%20-%20The%20Blockchain%20Trilema%20-%202020%20-%20Oct.pdf. (accessed: 04.03.2023) (page 190).

[255] Francez, Nissim. *Fairness by Nissim Francez*. eng. Texts and Monographs in Computer Science. New York, NY: Springer US, 1986. ISBN: 1-4612-9347-2. (accessed: 05.03.2023) (page 190).

[256] Sebastian Müller et al. “Fast Probabilistic Consensus with Weighted Votes”. In: () (page 195).

[257] Rajasekhar Chaganti and Bharat Bhushan and Vinayakumar Ravi. “A survey on Blockchain solutions in DDoS attacks mitigation: Techniques, open challenges and future directions”. In: *Computer Communications* 197 (2023), pp. 96–112. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2022.10.026. URL: {https://www.sciencedirect.com/science/article/pii/S0140366422004145} (page 209).

[258] Ekparinya, Parinya and Gramoli, Vincent and Jourjon, Guillaume. “The Attack of the Clones against Proof-of-Authority”. In: *CoRR* abs/1902.10244 (2019). arXiv: 1902.10244. URL: http://arxiv.org/abs/1902.10244. (accessed: 09.12.2022) (page 211).

[259] Saltini, Roberto and Hyland-Wood, David. *IBFT 2.0: A Safe and Live Variation of the IBFT Blockchain Consensus Protocol for Eventually Synchronous Networks*. 2019. arXiv: 1909.10194 [cs.DC]. (accessed: 06.03.2023) (page 212).

[260] Gai, Fangyu and Farahbakhsh,Ali and Niu, Jianyu and Feng, Chen and Beschastnikh, Ivan and Duan, Hao. “Dissecting the Performance of Chained-BFT”. In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)* (2021), 595–606. (accessed: 08.03.2023) (page 216).

[261] Gągol, Adam and Leundefinedniak, Damian and Straszak, Damian and undefinedwiundefinedtek, Michał. “Aleph: Efficient Atomic Broadcast in Asynchronous Networks with Byzantine Nodes”. In: AFT ’19 (2019), 214–228. DOI: 10.1145/3318041.3355467. URL: https://doi.org/10.1145/3318041.3355467. (accessed: 08.03.2023) (page 216).

[262] Danezis, George and Kokoris-Kogias, Lefteris and Sonnino, Alberto and Spiegelman, Alexander. “Narwhal and Tusk: A DAG-Based Mempool and Efficient BFT Consensus”. In: EuroSys ’22 (2022), 34–50. DOI: 10.1145/3492321.3519594. URL: https://doi.org/10.1145/3492321.3519594. (accessed: 09.03.2023) (page 216).

[263] Spiegelman, Alexander and Giridharan, Neil and Sonnino, Alberto and Kokoris-Kogias, Lefteris. “Bullshark: DAG BFT Protocols Made Practical”. In: CCS ’22 (2022), 2705–2718. DOI: 10.1145/3548606.3559361. URL: https://doi.org/10.1145/3548606.3559361. (accessed: 09.03.2023) (page 216).

[264] Cohen, Shir and Gelashvili, Rati and Kogias, Lefteris Kokoris and Li, Zekun and Malkhi, Dahlia and Sonnino, Alberto and Spiegelman, Alexander. “Be Aware of Your Leaders”. In: (2022). Ed. by Ittay Eyal and Juan Garay, 279–295. (accessed: 09.03.2023) (page 217).

[265] Cohen, Shir and Keidar, Idit and Naor, Oded. “Byzantine Agreement with Less Communication: Recent Advances”. In: *SIGACT News* 52.1 (2021), 71–80. ISSN: 0163-5700. DOI: 10.1145/3457588.3457600. URL: https://doi.org/10.1145/3457588.3457600. (accessed: 08.03.2023) (page 217).

[266] Zhang, Yupu and Dragga, Chris and Arpaci-Dusseau, Andrea and Arpaci-Dusseau, Remzi. "*-Box: Towards Reliability and Consistency in Dropbox-like File Synchronization Services". In: HotStorage'13 (2013), p. 2. (accessed: 04.03.2023) (page 218).

[267] Giridharan, Neil and Howard, Heidi and Abraham, Ittai and Crooks, Natacha and Tomescu, Alin. *No-Commit Proofs: Defeating Livelock in BFT*. Cryptology ePrint Archive, Paper 2021/1308. https://eprint.iacr.org/2021/1308. 2021. URL: https://eprint.iacr.org/2021/1308. (accessed: 06.03.2023) (page 218).

[268] Boneh, Dan and Drijvers, Manu and Neven, Gregory. "Compact Multi-Signatures for Smaller Blockchains". In: (2018), 435–464. DOI: 10.1007/978-3-030-03329-3_15. URL: https://doi.org/10.1007/978-3-030-03329-3\%5F15. (accessed: 06.03.2023) (page 218).

[269] Kelkar, Mahimna and Deb, Soubhik and Long, Sishan and Juels, Ari and Kannan, Sreeram. *Themis: Fast, Strong Order-Fairness in Byzantine Consensus*. Cryptology ePrint Archive, Paper 2021/1465. https://eprint.iacr.org/2021/1465. 2021. URL: https://eprint.iacr.org/2021/1465. (accessed: 07.03.2023) (page 218).

[270] Peter Kurrild-Klitgaard. "An Empirical Example of the Condorcet Paradox of Voting in a Large Electorate". In: *Public Choice* 107.1/2 (2001), 135–145. ISSN: 00485829, 15737101. URL: http://www.jstor.org/stable/30026259. (accessed: 07.03.2023) (page 218).

[271] Xiao, Yao and Xu, Lei and Zhang, Can and Zhu, Liehuang and Zhang, Yan. "Blockchain-Empowered Privacy-Preserving Digital Object Trading in the Metaverse". In: *IEEE MultiMedia* 30.2 (2023), pp. 81–90. DOI: 10.1109/MMUL.2023.3246528 (page 224).

[272] "Radix DLT". *"Why ERC-4337 "Account Abstraction" Falls Short of Radix Smart Accounts"*. URL: https://www.radixdlt.com/blog/comparing-account-abstraction-and-radix-smart-accounts. (accessed: 29.05.2023) (page 224).

[273] "Radix DLT". *"Why DAGs Don't Scale Without Centralization"*. URL: https://www.radixdlt.com/blog/dags-dont-scale-without-centralization. (accessed: 29.05.2023) (page 224).

[274] Dinh, Tien Tuan Anh and Wang, Ji and Chen, Gang and Liu, Rui and Ooi, Beng Chin and Tan, Kian-Lee. "Blockbench: A framework for analyzing private blockchains". In: (2017), 1085–1100. (accessed: 04.04.2023) (page 229).