Individual Diploma Thesis

INTEGRATING BLOCKCHAIN IN IOT WITH ASYNCHRONOUS CONSENSUS PROTOCOL

PANTELIS MIKELLI

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2022

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Integrating Blockchain in IoT with Asynchronous Consensus Protocol

Pantelis Mikelli

Supervising Professor

Chrysanthou Yiorgos

The Individual Diploma Thesis was submitted for partial fulfillment of the requirements for obtaining the degree of Computer Science of the Department of Computer Science of the University of Cyprus

May 2022

Acknowledgments

Upon completing my individual diploma thesis, I would like to express my sincere gratitude to all those who contributed to its elaboration. Many thanks to all my supervising professors, for the trust they have shown me from the beginning and for the help and guidance they gave me in its elaboration.

Last but not least, I would like to express my sincere gratitude to my family for their support and understanding throughout my studies.

Abstract

The Internet of Things (IoT) is unquestionably here to stay as technology advances rapidly. Today, billions of physical devices are connected to the internet worldwide and send a massive amount of data (sensor measurements) daily. Often, the integrity of those data is crucial in real-time decision-making if for example, they are used in SCADA (Supervisory Control and Data Acquisition) components. Thus, the trust in this technology is often compromised by significant security and scalability issues. An excellent way to address some of those challenges and accelerate the adoption of IoT technology is to store those data in a blockchain.

In the past few years, Blockchain technology has attracted the eyes of millions of people mainly because of cryptocurrencies. Blockchain can alleviate the concerns associated with IoT by using methods like cryptographic algorithms and by building a tamper-proof distributed peer-to-peer system with multiple benefits.

One of the first and best Companies that came up with a blockchain-based solution for IoT is Helium, a startup known as the first decentralized machine network in the world. After collecting IoT measurements with the use of an innovative model called “Proof-of-Coverage”, Helium propagates the data to a consensus group to verify them and create a new block to the blockchain. To achieve a high rate of confirmed transactions, this consensus group employs an asynchronous variant of the Byzantine Fault Tolerant protocol known as "HoneyBadgerBFT."

This thesis could help the technology community better understand the benefits of a blockchain in the IoT world and, most importantly, understand what an asynchronous consensus protocol can offer. With the help of Amazon Web Services (AWS), I conducted an experiment across five different regions across the world with an asynchronous BFT protocol called “Dumbo” which is a better version of “HoneyBadgerBFT”.

Contents

Chapter 1

1 Introduction

Internet of Things is one of the most significant disruptive technologies of the last years and is recognized as the “the third wave in the development of the Internet” [39]. IoT is capable of interconnecting billions of devices over the internet, varying from the smallest sensor to a car. There are now more than 10 billion internet-connected devices, with that number predicted to rise to more than 50 billion by 2025 and the volume of data generated by them will reach 80ZB [40].

Since IoT can benefit everyone and improve everyone's lives it is rapidly infiltrating various fields, such as healthcare, transportation, smart homes/cities, water systems and many more. Every IoT device collects sensitive data, and its security is becoming more and more crucial. However, the existing centralized architecture of the Internet of Things exposes plenty of security risks and cannot be trusted.

A suitable solution to this problem could be blockchain. Blockchain is also one of the most talked about technologies the last years after the huge success of Bitcoin. Blockchain is essentially a growing list of blocks that is distributed to every member of the blockchain. Blockchain can overcome most of the current IoT challenges and eliminate the single point of failure as it is decentralized, immutable and no one can alter the blocks or add fake transactions. Aside from security features, IoT could benefit from this decentralized architecture, as it could manage billions of data from IoT devices without the need for massive and expensive data centers.

A consensus mechanism is required for blockchains to function, as it is used to generate new blocks and guarantee that each member of the blockchain has the same copy. The majority of

the blockchains use consensus protocols with synchrony since they deploy timing assumptions for the submission of transactions in the blockchain. These timing assumptions are proven to

be dangerous because if for any reason(e.g. Denial Of Service attack) the time threshold is exceeded, the performance of blockchain is greatly decreased.

To overcome this problem, asynchronous consensus mechanisms could be used. The reason why asynchronous protocols were not used as much before is because of the well-known FLP impossibility result which states that it is impossible to develop a deterministic algorithm for reaching consensus in an asynchronous network with faulty nodes. The only way to create an asynchronous consensus protocol is with a source of randomness. Researchers spent many years attempting to develop a randomized algorithm for such cases, but they only managed to develop impractical theories until Andrew Miller et al.[29] developed the first practical asynchronous consensus protocol in 2016 called “HoneyBadgerBFT”.

This thesis presents an overview of blockchain integration in the Internet of Things, as well as the benefits and challenges that must be addressed. The ultimate purpose of this thesis is to demonstrate that an asynchronous consensus protocol can outperform protocols with synchrony. I present "HoneyBadgerBFT" and "Dumbo," and then I demonstrate the results of the Amazon Web Services (AWS) experiments I conducted across five different regions.

Chapter 2

2 Background

2.1 Centralized vs Decentralized vs Distributed Systems

In a Centralized system, every user must be connected to a central server, which serves as a single point of contact and keeps all data and user information. Although this type of system is simple and affordable to set up, it has a major flaw: a single point of failure. If the server crashes, no one will be able to access it (availability concerns), and data may be altered or lost permanently.

A Decentralized system, on the other hand, instead of a single central server, it has multiple central servers, each of which holds a copy of the data and is able to make its own decision, with the final answer being an aggregate of all the decisions. In this manner, in the event of a failure, if at least one of the servers is available, the users can be served. Although such a network improves overall performance, cost is higher compared to a single server.

In the case of Distributed system, computation is distributed among several interconnected nodes located in different physical locations, while decision making can be either centralized or decentralized. In order for this kind of system to work a consensus mechanism must be in place, enabling nodes to agree on a common value.

2.2 Byzantine fault tolerance (BFT)

Achieving consensus in the presence of faulty nodes is the most crucial problem in distributed systems. Lamport, Shostak, and Pease formally introduced the consensus problem in such circumstances in 1982, as a story of Byzantine generals in the presence of unreliable generals(fault tolerance) trying to agree on a single attack plan via exchanging messages[6]. Since then BFT has become one of the most studied topics in distributed computing and multiple BFT-based consensus protocols have been developed.

In byzantine generals problem when an individual x makes a proposal y, an agreement between n generals can be achieved if:

I. All reliable generals agree on the same plan/value y.
II. The agreed upon value z is the same as the initial value the individual proposed y=z.

2.3 Synchronous vs Asynchronous

The consensus protocol can be considered in the case of differing assumptions of synchrony.

Till today most of the consensus protocols have been working with synchrony assumptions, meaning they assume that there is a priori known bounded delays(in seconds or rounds) on all messages. The delay a synchronous algorithm has is usually proportional to the number of faults in the system. Furthermore, adversaries capable of launching DoS attacks can take advantage of these timing assumptions.

On the contrary, in a fully asynchronous system there are no timing assumptions in place and messages might take an infinite amount of time to arrive at their destination or better arrive even faster than expected. Moreover, reaching consensus in such scenarios is much more difficult compared to scenarios with timing assumptions. The well-known FLP impossibility result demonstrates that developing a deterministic algorithm for reaching consensus in such a network with faulty nodes is impossible [26]. Given a specific input a deterministic algorithm will always have the same outcome.

The only method to achieve consensus in a fully asynchronous system is by randomization. This implies that the outcome may not necessarily be consistent across executions with the same inputs [27].

2.4 Blockchain Technology

In 2008, Satoshi Nakamoto, a mysterious figure, proposed the first decentralized blockchain with significant improvements to the architecture that keep the pace of block additions steady. A year after, the idea was successfully implemented as an essential part of the cryptocurrency bitcoin by the same figure(Nakamoto). Blockchain acts as the public ledger for all bitcoin network transactions. [1]

Blockchain is essentially a distributed growing sequence of records called “blocks”, connected to each other using cryptography. It is used to form a trustless digital ledger of transactions shared across an entire network of computer systems, often called ”nodes”. This method is called “Distributed Ledger Technology”, and each time new transactions occur, a new block containing those transactions must be created and added to every node in the network. In addition every user of the blockchain owns an address instead of personal information.

2.4.1 Block

As Figure 2.1 illustrates, each block in bitcoin is divided into two parts: the header, which summarizes the block, and the body [2]. The block body includes a group of valid transactions and their counter, and the block header is made up of the following six components:

I. **Version:** specifies the block validation rules that the block employs.
II. **Parent block hash:** a 32-byte hash value pointing to the preceding block.
III. **Merkle tree root:** a 32-byte hash value of all the transactions in the block.
IV. **Timestamp:** a 4-byte field representing the universal time as seconds since January 1970.
V. **Difficulty Target:** a value or a target threshold of a valid block hash.
VI. **Nonce:** a number that blockchain miners try to solve.

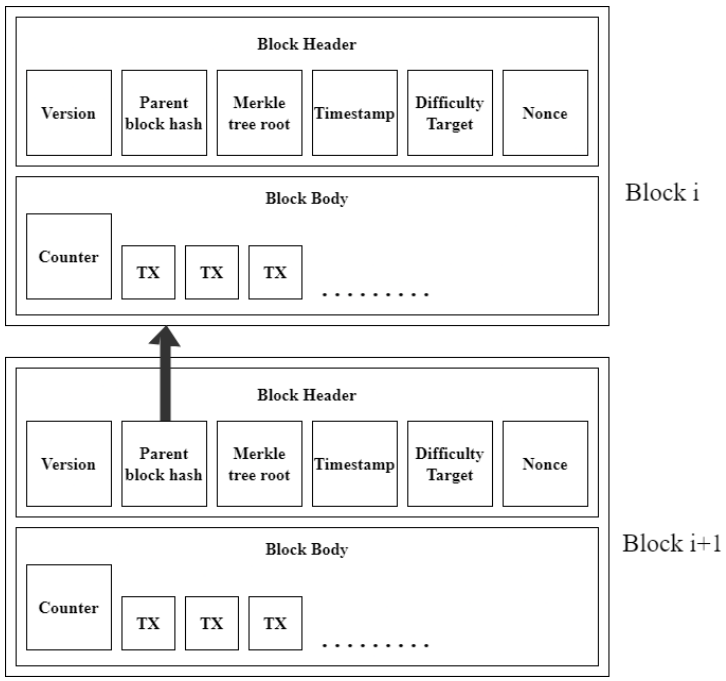

Figure 2.1: Block Structure

2.4.2 Digital signature

Every user except from the address he also owns a private and public key pair, and in order to send a transaction, it must be signed with the private key, which must be kept secret. Therefore, the signed value of the hashed transaction is broadcasted across all nodes of the network, and each node with the use of the known public key can verify that the transaction had not been manipulated. The elliptic curve digital signature algorithm is the most common digital signature algorithm used in blockchains(ECDSA).[3]

2.4.3 Smart Contracts

A smart contract is simply a set of contractual terms agreed upon from stakeholders that are encoded in computer programs and run automatically as soon as certain conditions are satisfied[38].

Smart contracts are built on top of blockchains and are saved and updated on every node of the blockchain to guarantee proper contract execution. This improves the conventional contracts where a trusted middleman who requires a fee is responsible for the execution of the contract[38].

Contractual terms are written in the form of "if-else-if" statements, and a contract execution is also recorded on the blockchain, providing the following advantages [38]:

I. Contracts cannot be altered.

II. Additional fees are not required.

III. A computer executes the contract fast which saves time.

IV. Guarantee that contract is executed correctly as soon as conditions are satisfied.

2.4.4 Advantages of Blockchain

Blockchain technology has multiple security benefits, gained from the following key characteristics [4]:

V. **Immutability:** It is impossible to change any block or transaction since it is distributed across every node in the network.

VI. **Decentralized:** A transaction can be completed between two peers without the use of an intermediary, which can be time-saving and offer discretion between the peers.

VII. **Anonymity:** Users hide their identity since they can produce many addresses and communicate with the blockchain network using just those addresses.

2.4.5 Types of blockchains and Applications

Until today there are three types of blockchain: public, private, and consortium[5].

I. **Public:** It is permissionless, which implies that anybody with an internet connection may join the network.

II. **Private:** It is permissioned, and a central authority governs who is permitted to join the network.

III. **Consortium:** It is permissioned, and more than one authority governs who is permitted to join the network.

Each category is employed in a distinct context and has its own set of benefits and drawbacks.

Blockchain may have begun as a distributed ledger technology for cryptocurrencies, but the more confidence it obtains, the more diversified its applications grow. It starts to integrate with fields except for Finance like Video Games, Entertainment, Supply chains, Security services, and IoT.

2.4.6 Consensus Algorithms

A consensus is required for all nodes to agree on a block of transactions, construct a block and finally make sure that every node keeps the same blockchain. Because nodes in a blockchain system are untrustworthy, the consensus mechanism is needed to tolerate Byzantine failures[6] and ensure that every node in the network has the same blockchain. Day by day, blockchain gets integrated into many different fields, and thus several types of consensus are developed with different approaches and intentions. Two of the most known consensus protocols that fit the requirements of a permissionless blockchain(scalable) are the following:

I. **Proof of Work (PoW):** Markus Jakobsson and Ari Juels initially developed and standardized the term "proof of work" in a 1999 publication[7]. Later, in 2009, the Bitcoin network implemented PoW as the permissionless decentralized network's consensus protocol. In each round of consensus, a cryptographic challenge is solved with the help of computer power in order to choose one node to build a new block. To solve the complex cryptographic puzzle, each node or “miner” must keep modifying the nonce value until the correct solution is obtained. PoW is extremely costly in terms of computing and power usage[8].
II. **Proof of Stake (PoS):** Due to the exorbitant cost of PoW, PPcoin[11] introduced a second consensus protocol, Proof-of-stake, which determines a node's capacity to create a new block based on its stake in the blockchain. To put it another way, if you own 1% of the currency, you will be responsible for 1% of all proof-of-stake blocks[8].

Permissionless protocols manage to scale well in terms of network size, but they have very low throughput of validated transactions. On the other hand, a permissioned network is capable of achieving very high throughput of validated transactions since all nodes are verified and obligated to operate properly and therefore, there is no need for an expensive consensus process. Furthermore, the majority of permissioned blockchains use BFT-based consensus protocols which have a high level of communication complexity, often tolerate 30% unreliable nodes, and function badly under adverse conditions. Due to the high communication complexity BFT-based consensus protocols also scale badly in terms of validator nodes but as long as the nodes are well validated and honest that shouldn’t be a problem.

Two popular examples of consensus protocols for permissioned blockchains are the following:

I. **Practical Byzantine Fault Tolerance (PBFT):** In distributed systems, PBFT is the first partially synchronous Byzantine Fault Tolerant protocol with a low algorithm complexity and great practicality. It employs a stable leadership model, in which a leader is only replaced when a problem shows up. It is also partially synchronous since it can function in an asynchronous network like the internet while yet having time assumptions(synchrony), ensuring both safety and liveness. Time assumptions ensure that clients will ultimately receive responses to their queries (liveness). In order for the protocol to work, messages must be broadcasted to everyone, creating a complexity $O(n^2)$ for the communication[9].

II. **Hotstuff:** Hotstuff, like PBFT, is partially synchronous and is one of the most recent BFT protocols that provide both linearity and responsiveness. It employs a rotating leadership model, in which a leader is rotated after a single epoch. For the protocol to work, messages do not have to be broadcasted to everyone and create a complexity O(n) for the communication. It also manages to reach high throughput compared to other BFT protocols[10].

Table 2.1 illustrates a small comparison of the Consensuses.

Property	PoW	PoS	PBFT	HotStuff
Application	Public	Public	Permissioned	Permissioned
Fault Tolerance	50%	50%	33%	33%
Examples	Bitcoin	Cardano	HL Fabric	FB LibraBFT
Synchrony	Synchronous	Synchronous	Partial Synchrony	Partial Synchrony

Table 2.1: Consensus comparison

2.4.7 Blockchain Challenges

Nobody can deny that blockchain is a brilliant technology, but as a new technology there are several significant issues that must be addressed before more people would trust and accept it. Below are some of the most important challenges:

I. **Scalability:** When a large number of transactions occur at once, blockchain becomes extremely sluggish.
II. **Energy Consumption:** Consensuses like PoW require significant amount of energy usage to operate miners.
III. **Privacy:** In a public blockchain, a user can maintain his anonymity, but transaction details and balances are publicly visible, which is a huge issue for some people/organizations.
IV. **Regulatory:** As a new technology there aren't any laws around it.

2.5 Internet of Things (IoT)

The Internet of Things (IoT) is essentially the collection of billions of interconnected devices or "things" over the internet. Sensors are built into each device, which gather data that may be saved and analyzed to display meaningful results, with the potential to enhance everyone's life.

2.5.1 History

British computer scientist Kevin Ashton officially introduced the Internet of Things term in 1999 and even said, "The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so." [13]

The world's first internet-connected refrigerator was introduced by LG Electronics in 2000, enabling users to complete their food shopping online. However, it was too costly for consumers to purchase, and so it failed. [14] After the term was referenced in many mainstream publications between 2003-2004, the International Telecommunication Union (ITU) in 2005 issued the first official report and hence definition of the IoT concept as "A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies".[16]

2.5.2 Present & Future of IoT

Currently a centralized server/client model is mainly used in order for the Internet of things to work. Such architecture requires every device to be authenticated through a central server. IoT devices are billions and can be found anywhere from the smallest things like light bulbs to the biggest like cars. By 2025, according to former Cisco CEO John Chambers, there will be 500 billion connected devices.

The advantages of such a concept are numerous for both individuals and businesses. Some of the capabilities could be:

I. Boost productivity in businesses by reducing human work and hence increase profitability.
II. Improve quality of life by monitoring pollution and air quality in real-time.
III. Save money by automating multiple devices.
IV. Assist in predicting anything from a lack of stock to any other kind of behavior.

Based on these facts the current architecture will start to fail and many challenges will start to rise as the number of IoT devices keep increasing and it will only be capable of supporting small-scale IoT networks. Despite the benefits of IoT, those challenges will manage to slow down the adoption in the world. Most of the challenges arise due to the centralized structure and the single point of failure of the architecture which causes the following concerns:

I. **Scalability:** As the number of devices grows rapidly, the volume of communication required by the central servers would become unmanageable.
II. **Security:** Centralized systems are known for their single point of failure that may bring the whole network down in case of an availability attack like a distributed denial-of-service attack (DDOS). Also, such systems are very vulnerable to data tampering by either the central owner or an adversary. Therefore Confidentiality, Integrity, and Availability may be compromised.

2.5.3 Integrating blockchain to IoT

A good solution which can address most of the current IoT issues could be the blockchain technology(section2.4). It is a strong and innovative technology that has the power to

decentralize the IoT system and address many of the ever-growing problems. Blockchain, with its peer-to-peer capabilities and decentralized structure, has the potential to benefit the IoT world by

distributing computational and storage demands among several nodes. Some of the benefits it could get from this integration are:

I. There will be no central owner who will be able to verify or tamper the transactions.
II. It will be nearly impossible for anyone to tamber the transactions as they are copied to every node and everyone would have to verify any change.
III. Minimize the operational expenses involved compared to centralized data centers.
IV. Using cryptographic techniques, transactions can be kept private.
V. Owners' identities can be protected since the blockchain is typically accessed using simply an address.
VI. It will be able to handle billions of IoT devices as the number of devices rises fast.

Integrating blockchain to IoT would undoubtedly provide several benefits, but being a relatively new technology, is not a flawless solution and has issues of its own. Most of these issues are mentioned in Blockchain section2.4.6. The main problem that must be addressed is the poor throughput of validated transactions, which occurs as a result of time-consuming consensus protocols.

2.6 LoRa & LoRaWAN

IoT devices are often battery-powered sensors used inside houses, industrial buildings, or even in the middle of a forest, so they must not only handle battery life carefully, but they must also be able to communicate from a long distance. An ideal solution for such cases is the LoRaWAN protocol which is developed and maintained by the LoRa Alliance. In 2015, the first LoRaWAN

standard was issued [18]. LoRa is a wireless radio modulation technique that encodes information on radio waves and its name derives from Long Range since it can provide long-distance communication links of up to 15 kilometers and is installed in a star topology network based on the open LoRaWAN protocol [17].

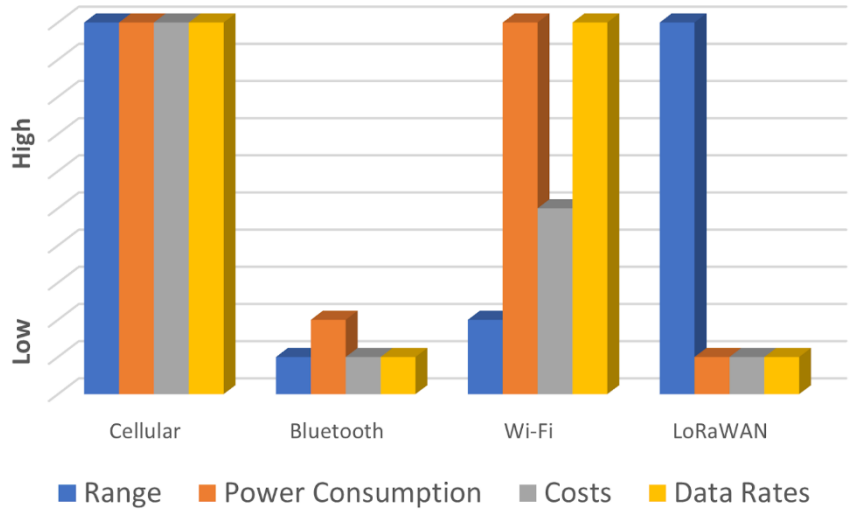

Figure 2.2: Range vs Power Consumption vs Costs vs Data Rates (Adapted from [17])

As we can observe from figure2.2, LoRaWAN excels in all areas over the other network technologies except data rate which is necessary to achieve the rest[17]:

I. It is capable of covering very long range distances
II. Optimizes the battery life since it has very low power consumption. (~10years lifetime)
III. The cost for deployment and maintenance is minimal compared to other network technologies.

Some additional advantages of this technology are:

I. **Accuracy:** accurate location without the need for GPS.
II. **Security:** end-to-end encryption.
III. **High Capacity:** it can support millions of messages.

2.7 Helium

Helium is the world's first peer-to-peer blockchain-based IoT solution, offering a safe and cost-effective mechanism for low-power smart devices to communicate with the Internet. It is open-source and it makes use of the largest secure decentralized worldwide network of LoRaWAN devices known as “Hotspots” or “Miners”. Each hotspot is owned and operated by individuals who are motivated by the rewards they get from expanding and securing the Network. Asides from that, each hotspot provides connectivity to nearby LoRaWAN compatible devices [19].

2.7.1 Helium History

Helium startup was founded in 2013 by Shawn Fanning, Amir Haleem, and Sean Carey. Later in July 2019, the helium network was launched with the goal of creating a blockchain powered global decentralized network of hotspots for IoT devices [20]. Helium has seen tremendous adoption from people the last 2 years and as of today there are more than 820000 hotspots all over the world [21]. It is becoming more powerful every day and has managed to form partnerships in a variety of fields, including fields like environment, agriculture, asset tracking and more [20]. Every hotspot can be seen in helium`s map explorer along with other information and statistics of the network. As we can see from figure2.3 the entire world is filled with hotspots including Cyprus.

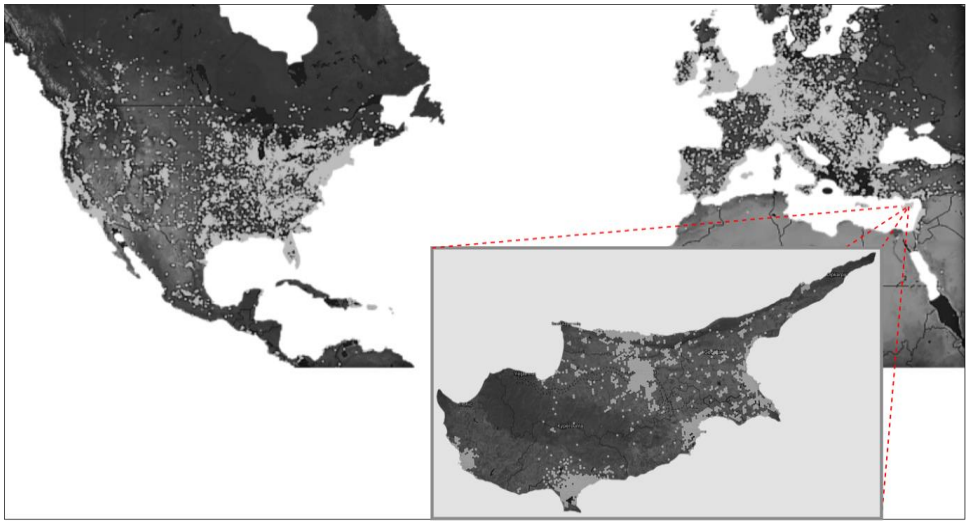

Figure 2.3: World – helium network coverage

2.7.2 Network Structure

Figure2.4 illustrates a simplified structure of the helium network where devices inside a miner coverage are capable of connecting to the miner if they are LoRaWAN compatible and other entities communicate with each other in a manner which will be explained below.

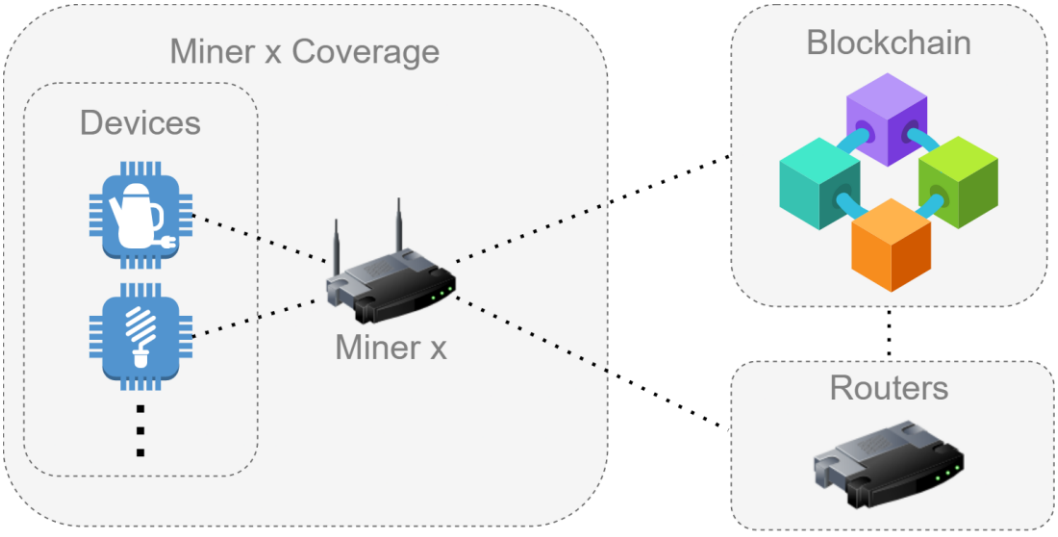

Figure 2.4: World – helium network coverage

2.7.2.1 Hotspots & IoT devices

Hotspots are very easy to deploy by anyone and are also called miners because each individual can earn Helium`s blockchain token “HNT“ as a reward simply by extending the coverage of the network. Hotspots are basically LoRaWAN gateways which by combining LoRaWAN features, and helium`s blockchain they form the so-called LongFi protocol created by Helium systems. They provide wireless network coverage to LoRa compatible devices.

IoT devices compatible with LoRa do not require any credentials like other networks do in order to make a transaction. They just transmit an electronic payment along with a data request to the nearest hotspot. In addition, each Hotspots and IoT device has a private key stored in its hardware as well as their public keys on the blockchain.

The absence of any battery consuming GPS in the hotspots and the deployment of the innovative PoW protocol called “Proof-Of-Coverage” results in the low energy consumption for each(5W of energy).

2.7.2.2 Routers - LoRaWan Network Server (LNS)

Helium treats the LoRaWAN Network Server(LNS) differently than traditional LoRaWAN Networks in which a central or regional LNS is controlled by a single authority. Helium allows multiple routers to exist in the network and be owned and operated by individuals. Routers(LNS) are basically internet applications that accept packets from IoT devices through hotspots and route them to appropriate endpoints like HTTP. Some of its functions are:

I. The Uplink and Downlink data from and to the IoT devices accordingly.
II. Third-party cloud services are provided with authentication and routing capabilities.
III. Have a local copy of the blockchain.

Each router is assigned to a unique OUI (Organization Unique Identifier) and is saved into the blockchain along with a list of IoT devices that it is responsible for. As a result, whenever a hotspot receives data from an IoT device for Uplink, it must first query the blockchain to discover which router to transfer the data to. This feature enables the secure delivery of data to a router of your choice that may even be yours [22].

2.7.3 Proof of Coverage

In order to guarantee the security and coverage of the network, Helium came up with a novel consensus algorithm called Proof-of-Coverage (PoC) that is built on top of the Helium`s blockchain consensus responsible for the creation of blocks. Proof-of-Coverage essentially proves that each hotspot is located where it claims it is and that it provides wireless network coverage to that location [23].

PoC uses RF(Radio Frequency) signals to determine the locations of hotspots. This is possible because the strength of an RF signal is proportional to the distance from the hotspot

transmitting. PoC frequently conducts checks called PoC challenges to assess network`s quality and stores the results into the blockchain. During a challenge hotspots divide in 3 roles:

I. **Challenger:** Hotspot conducting the challenge. Every hotspot conducts one approximately every 6 hours.
II. **Beaconer or "Challengee":** Hotspot that is being evaluated.
III. **Witness:** Hotspots in close proximity to the Beaconer that were able to "witness" the challenge packets sent by the Beaconer.

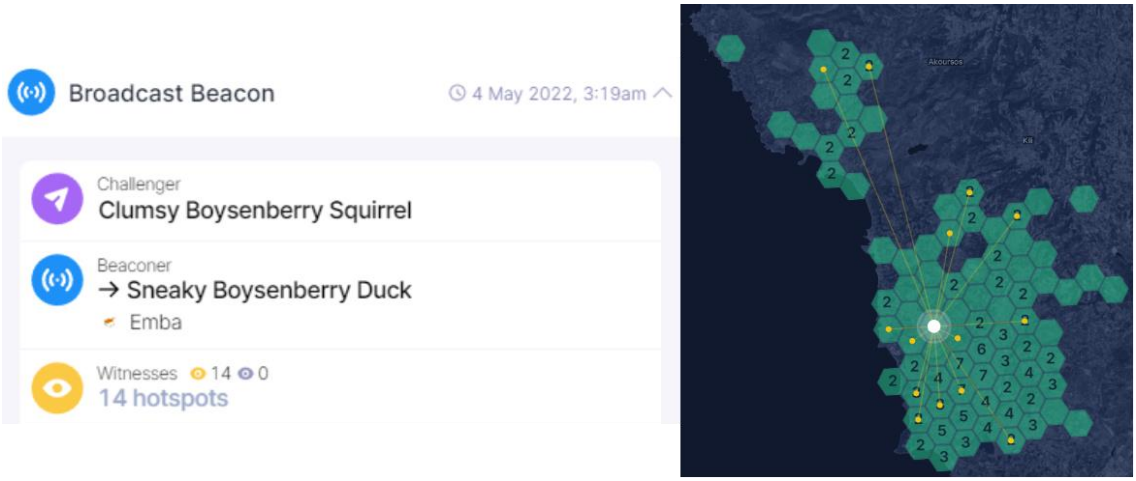

Figure 2.5: PoC Challenge example (Recovered from [21])

Figure2.5 illustrates a challenge that happened on 4th of May with challenger a hotspot in Brazil, beaconer a hotspot in Cyprus and 14 witnesses nearby beaconer.

2.7.4 Validators & Consensus protocol

Validators are hotspots responsible for the process and generation of blocks, and the group of validators is also called "consensus group" of the blockchain. To become a validator you must stake 10,000 HNT which is currently equivalent to 150000€. Every epoch(creation of 30 blocks) a new set of 40 validators is elected at random to form the consensus group and they get rewarded [24]. Anything that needs to be added to the blockchain is sent to the consensus group which is in charge of validating and ordering all the transactions, generating the block, and adding it to the blockchain [24].

Helium`s Consensus Protocol is permissionless because every hotspot following the rules of validators can join the consensus group and they will also be visible to all entities of the network. Due to the fact that the network is spread all over the world Helium chose an asynchronous Byzantine Fault Tolerant consensus protocol called HoneyBadgerBFT(HBBFT) that allows a group of known nodes to reach consensus over unreliable connectivity. HBBFT also achieves high throughput of validated transactions and censorship-resistance [25]. This protocol will be further analyzed in chapter 3.

2.7.5 Helium`s blockchain

Hotspots and Routers maintain a ledger of the blockchain and continuously sync their local blockchain by getting chain updates from other hotspots or routers in the network. This is achieved by propagating new blocks to the network as long as they are added to the blockchain. Helium`s blockchain stores data about the locations and coverage of the network and additional information like the OUI discussed in section2.7.2.2 and token transactions. However, the data from the IoT devices are not stored in the blockchain but either in the router or in a third-party cloud service.

The current latency of block generation is 60 seconds and inside each block there is a version, a height, the previous block hash, a Merkle hash of the transactions, and a threshold signature from the current consensus group [24]. Helium`s blockchain records everything from PoC challenges to current validators and a payment.

Chapter 3

3 HoneyBadgerBFT

The requirements that arose as a result of the cryptocurrency's popularity inspired Andrew Miller and his colleagues to develop HoneyBadger BFT (HBBFT) at the University of Illinois. HBBFT is the first practical asynchronous BFT protocol [29].

3.1 HoneyBadger`s solution

As explained in Section2.3 in an asynchronous system where no timing assumptions are made, it's impossible to develop a deterministic consensus algorithm. As a solution, HBBFT employs a randomization source(cryptography) to circumvent the FLP impossibility. Although randomness almost always guarantees correctness, the performance of such protocols has been a concern for many years compared to synchronous protocols [28].

HoneyBadger manages to overcome this issue by cherry-picking better methods from the literature that have never been present together. It's the first BFT atomic broadcast protocol(“consensus”) that achieved optimal asymptotic efficiency in an asynchronous network.

3.2 Issues with timing assumptions

Miller and his colleagues claim that timing assumptions are harmful in an unstable network with physical or adversarial delays. If any timing assumptions exist and aren't met due to faulty nodes or network delays the consensus has to make certain corrective actions to recover. As a result throughput is significantly decreased. Furthermore, setting the time boundaries is challenging since an improper value may need continuous corrective actions [29].

Asynchronous protocols, on the other hand, impose no time assumptions and even under unstable network conditions, progress is made whenever messages are delivered.

3.3 HoneyBadger`s System Model

Their system model consists of a purely asynchronous network of N known nodes(P_0,...,P_{N-1}). The network employs a leaderless consensus in which every node is a proposer rather than only the leader, as most known BFT consensuses do.

Transactions can be generated and sent by any client to all of the nodes of the network and are considered committed after the client gathers from the majority of the nodes signatures as shown in figure3.1 Each transaction committed by a client is a unique string.

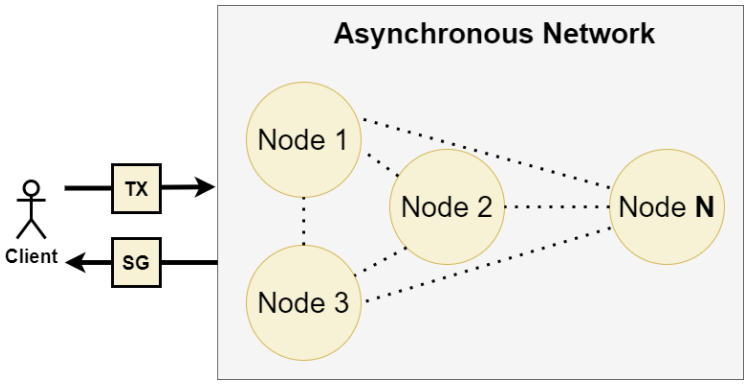

Figure 3.1: HoneyBadger`s System Model

Each network node gets transactions as input and attempts to achieve consensus on a transaction order.

To be able able to function correctly, their system model makes the following assumptions:

I. Each connection between nodes uses a reliable channel that delivers every message.
II. Nodes have unbounded buffers for queuing transactions regardless of the potential delay.
III. Consensus can withstand up to **f** faulty nodes, where $3f+1 \leq N$ (N=number of Nodes)
IV. A distributed key generation protocol must be deployed to distribute keys during setup.

3.4 HoneyBadger`s Consensus Mechanism

HoneyBadger has a modular approach and is composed of the following modules that will be explained in this section:

I. HoneyBadgerBFT
II. **A**synchronous **C**ommon **S**ubset (**ACS**) - Ben-Or et al. [32]
III. **R**eliable **B**road**c**ast (**RBC**) - Bracha [33] & Cachin and Tessaro[34]
IV. Asynchronous **B**inary **A**greement (**BA**) - Moustefaoui et al [35]
V. CommonCoin - Moustefaoui et al [35]

Consensus occurs in epochs, with each epoch allowing up to B transactions (batch size) to be committed. At a high level, incoming transactions in each node are kept in an unbounded buffer (FIFO), and B/N transactions are chosen at random from the buffer at each epoch. They are chosen randomly in order to propose as many different transactions as possible. The selected transactions are then given as input to the protocol where the final set of transactions is chosen.

The random selection of transactions from the buffer may jeopardize censorship resistance since an adversary might choose a transaction that does not wish to be submitted, and prevent it by eliminating whatever node proposes it. This problem is addressed by threshold cryptography, which conceals which transactions are suggested by which nodes until after consensus has been reached.

3.4.1 Threshold Encryption

Threshold cryptography allows nodes to encrypt a set of transactions with a master public key M. We can split M to multiple secret key shares and distribute one to each node. The network of nodes then have to collaborate to decrypt it. The number of shares required to decrypt the ciphertext is f+1 which means we need at least one correct node to share its decryption share. Until that happens an adversary does not know anything about the set of transactions [29].

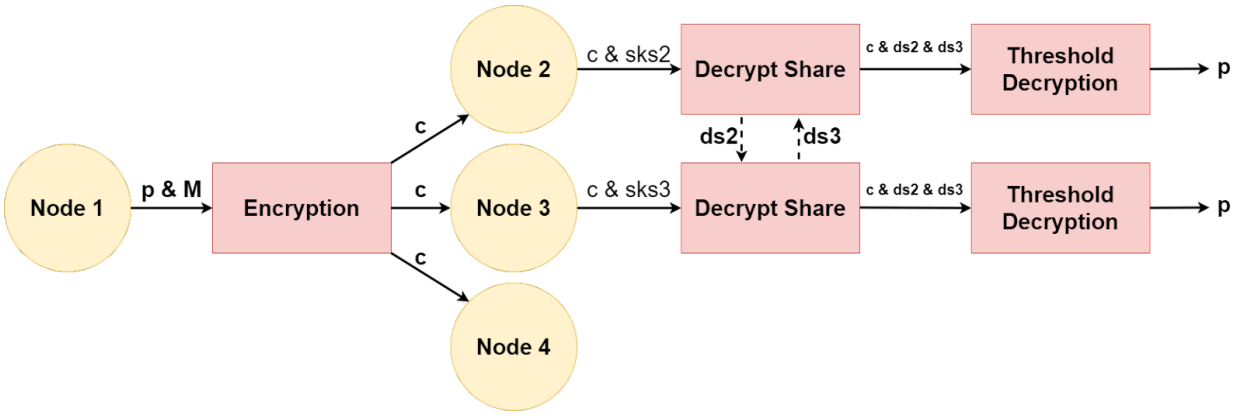

Figure 3.2: Threshold Encryption

Figure3.2 illustrates an execution of an example of threshold cryptography with 4 Nodes and 1 faulty node which means that at least 2 nodes must collaborate to decrypt the ciphertext [31]:

I. Node 1 encrypts p (set of transactions) with M (master public key)
II. Node 1 sends c (ciphertext) to every node.
III. Node 2 & 3 collaborate to decrypt it by inserting c and their sks (secret key share) into the Decrypt Share function.
IV. Decrypt Share function outputs a decryption share which is afterwards exchanged between nodes.
V. Each node then can insert c and the decryption shares collected into the Threshold Decryption function that outputs the original set of transactions.

3.4.2 Detailed Explanation

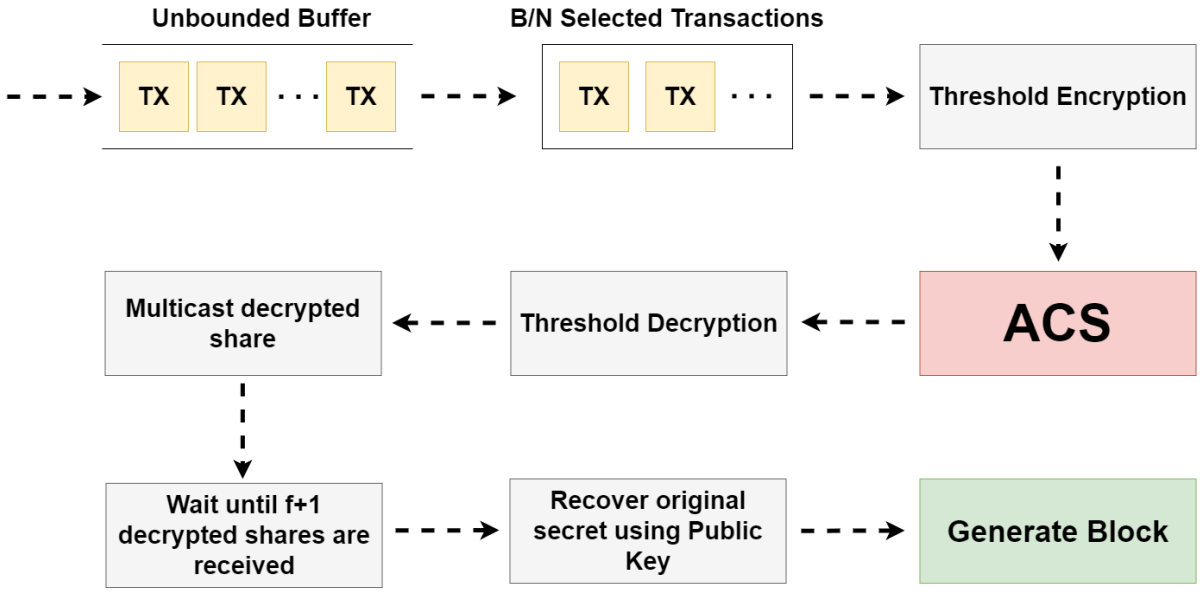

Figure 3.3: HoneyBadgerBFT Algorithm

In each epoch HoneyBadgerBFT is executed as shown in figure3.3 [30]:

I. Each node selects B/N transactions from their buffer to submit to the consensus.
II. After the set of transactions are selected they are encrypted using threshold cryptography and the master public key.
III. Encrypted sets of transactions are afterwards given to the ACS module(Asynchronous Common Subset).
IV. The ACS module enables the network to agree on a subset of the encrypted set of transactions from at least N-f nodes and outputs it.
V. Each node uses Decrypt Share function to get a decryption share and then multicasts decrypted share to every node (Block transactions determined before the adversary is able to see the proposed transactions).
VI. If a node wants to decrypt the original set of transactions agreed upon it has to receive more than a threshold share of decryptions.
VII. Original set of transactions agreed upon is afterwards decrypted and this epoch`s block is generated.

3.4.3 Asynchronous Common Subset (ACS)

After each node proposes a set of transactions and gets encrypted, inputs encrypted transactions to ACS where the network of nodes agrees on a subset from all the proposed encrypted set of transactions. ACS as shown in figure3.4 consists of 2 modules executed in order:

I. **R**eliable **B**road**c**ast (**RBC**)

II. **A**synchronous **B**inary **A**greement (**ABA**)

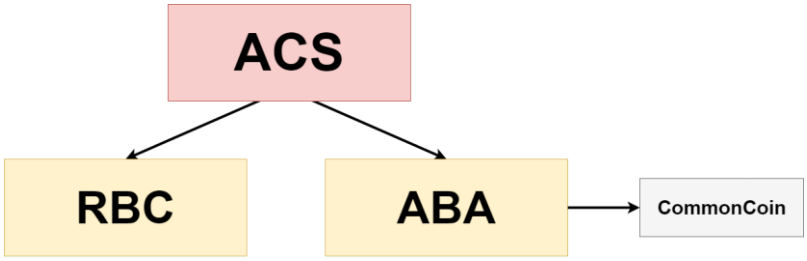

Figure 3.4: ACS decomposition

Reliable Broadcast is responsible for broadcasting the proposed values to every node and to do so N concurrent RB instances are run in each node. For it to work and reduce the communication complexity, erasure coding is used. With erasure coding proposed values are split into N pieces and then passed along. Furthermore, RBC ensures that every honest node receives the same output if one honest node manages to receive it.

After RBC finishes execution, ABA decides on a bit vector that indicates which set of encrypted transactions were successfully broadcasted by RBC. It is calculated by combining the votes of all network nodes. N concurrent ABA instances are run in each node in order to vote (1 or 0) for each RBC and agree on a bit for each. ABA terminates as soon as more than 2/3 of the network nodes agree that the set of transactions should be included in the block.

```
If same votes collected for RBCi > 2/3:
    Agree on value
Else:
    Agree on the random coin
```

Figure 3.5: ABA pseudocode

As pseudocode in figure3.5 illustrates, if the ⅔ threshold is not met, ABA in order to properly terminate deploys a source of randomness using CommonCoin module which generates a random coin shared to every node that determines if a specific set of transactions is going to be included or not. CommonCoin also uses threshold cryptography too to ensure that coins cannot be manipulated by any adversary [36].

Chapter 4

4 Dumbo

As mentioned in section2.3 no deterministic algorithm can be developed for asynchronous consensus protocol. Despite the efforts of many researchers to find a practical solution to overcome this problem with various methods none of them actually succeeded and most of them were mostly theoretical. Following the promising performance of Andrew Miller and his colleagues' recent work (described in chapter3), people began to wonder if such protocols could somehow be practical.

Figure 4.1: Asynchronous Protocols history

Due to this development, a joint work of researchers at New Jersey Institute of Technology and Chinese academy of sciences in 2020 proposed a new protocol based on HoneyBadgerBFT called Dumbo. Since then, they published 3 more papers improving Dumbo more and they are still working on it to improve it even more. This new protocol outperforms HoneyBadgerBFT in terms of both latency and throughput [41][42][43].

4.1 How Dumbo improved HoneyBadger

After running experiments they identified the main bottleneck of HoneyBadger was the execution of a large number of asynchronous binary agreements (ABA) in each node. Due to this really slow agreement, the latency was very high.

To address this bottleneck, Dumbo instead of using N binary agreements (ABA) in each node, uses a single Multi-valued Validated Byzantine Agreement (MVBA) in the ACS component. The structure of MVBA is very complicated and a large number of rounds is required in order to complete. It is basically constructed from two or three ABA instances and essentially outputs a set of transactions from a single node as long as the node's input matches a pre-defined global predicate [41][42].

Because of its extensive communication requirements, MVBA was not previously used but Dumbo shows that if it is used correctly and with a small input size it performs better than HoneyBadger`s ACS.
Except for throughput Dumbo also improved the latency approximately by 90-95% but despite these improvements because it is a randomized protocol it was still quite slow compared to some synchronous protocols like HotStuff [41][42].

4.2 Dumbo Performance bottlenecks

At this point Dumbo improved the agreement process but apart from still being the major percentage of the whole latency, the reliable broadcast (RBC) started to take a significant percentage of the whole latency(Approximately 20% in contrast with 0.5% that had before the improvements).
Thus, the protocol still had the following bottlenecks:

I. High message complexity in RBC module execution.
II. Improved but still high latency in MVBA.

4.3 Speeding Dumbo (sDumbo)

They later(in 2021) went ahead and implemented Speeding-Dumbo to overcome the previous bottlenecks [43].
To do so they had to develop a new ACS framework where Reliable Broadcast (RBC) was replaced by a Provable Broadcast (PB) which was defined by Abraham et al. [37]. PB manages to minimize the communication but to do so it loses the guarantee that RBC had which ensured that if one honest node received the message, eventually every honest node received it. PB only manages to guarantee that f+1 nodes receive the same message. As a result of this compromise, a recovery component was introduced at the end of the protocol to ensure that at the end of the consensus, every node had the messages [43].

This method improved the communication complexity bottleneck in the broadcast phase but the latency of MVBA was still dominant. To overcome this issue, they designed their own MVBA protocol called Speeding-MVBA, which is a more compact and efficient MVBA protocol than any other MVBAs. It minimizes the required rounds to come to consensus to more than 50% than other known MVBA protocols.

4.4 Speeding Dumbo Performance and future

Provable broadcast managed to improve latency and nearly double the throughput and Speeding-MVBA is 2 times faster than other MVBA protocols. In general Speeding Dumbo using the cheaper broadcast component and the compact MVBA protocol managed to double its throughput(figure4.2) and almost became 2 times faster than before.

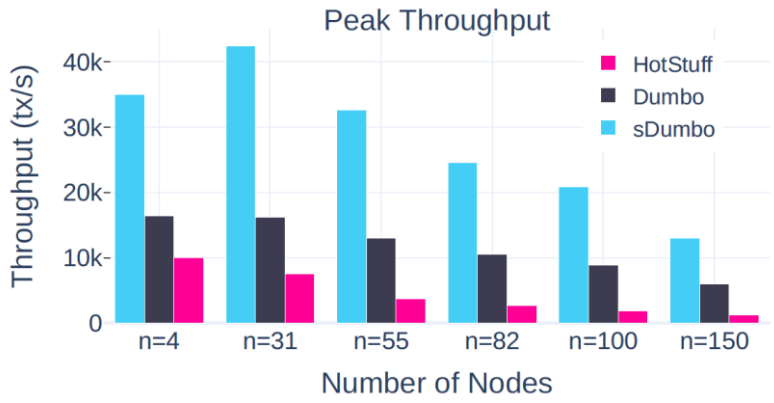

Figure 4.2: Throughput Comparison(HotStuff vs Dumbo vs sDumbo) [43]

Despite the very good performance in terms of throughput compared to prior Dumbo and HotSuff(synchrony), in terms of latency protocols like HotStuff are still considerably superior and for that the researchers are working on an even better solution.

Chapter 5

5 Experimental Results on Speeding Dumbo

5.1 Amazon Web Services(AWS)

To conduct my experiments, I used Amazon web services (aws) and more specifically the T2.medium ec2 instances which consist of 4GB RAM and 2vCpus each. EC2 instances provide a balance of computation, memory, and networking resources. Each instance was a Speeding Dumbo consensus protocol node, and I ran the experiment with several numbers of nodes and Batch sizes to see how they performed.

Each amazon instance was setup with an Ubuntu 20.04 server (focal). In addition each instance had a firewall which had to be modified to accept ssh access and enable communication with other instances.

The instances in the experiment were distributed across the world in 5 different regions as the figure5.1 shows.

Figure 5.1: Instances in 5 different regions

5.2 Libraries Used

To complete the experiment the following python libraries were used:

I. Boto3:

Amazon Web Services SDK for python that enable us to setup, launch and manage ec2 instances.

II. Fabric:

Enables us to execute multiple shell commands remotely over shh in order to:

i. Install dependencies on every instance
ii. Clone the GitHub repository
iii. Launch the experiments

III. Dash:

Low code framework written on top of Plotly.js and React.js with which interactive data apps can be rapidly developed in Python. An example of the data app created is shown in figure5.2 .

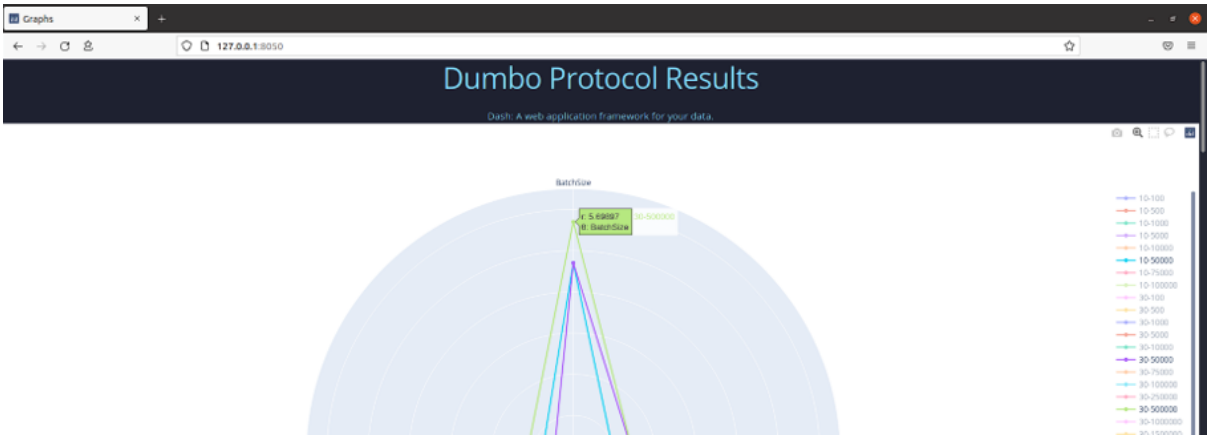

Figure 5.2: Data app written with dash for results.

5.3 Results

5.3.1 Varying Batch Sizes

In the following graphs the latency and throughput is compared based on the number of transactions proposed in each epoch. In figure5.3 x-axis is the batch size(number of transactions proposed) and y-axis is the latency(seconds needed for 1 block to be generated). Each coloured line indicates the number of nodes in the consensus.

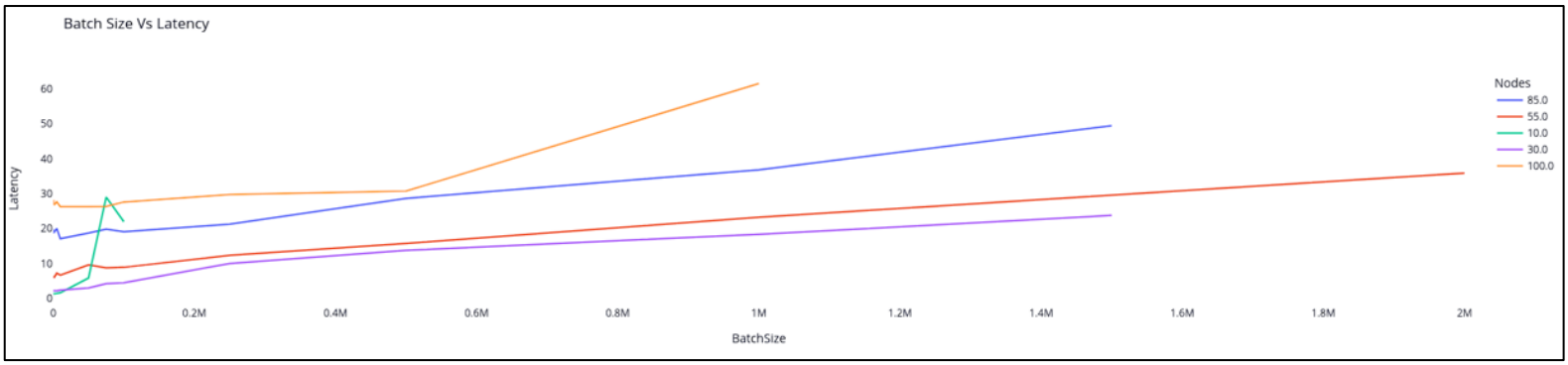

Figure 5.3: Latency(s) vs BatchSize(# proposed txs)

We can easily see that while the batch size increases in every case the latency increases too. This occurs because as the Batch size increases the required communication between the nodes increases too. Consensus group with 30 nodes performs better than all the others latency from 2 seconds at B=100 to 23 seconds at B=1500000.

In addition, as the number of nodes increased the latency was increased and this is also due to the fact that a larger consensus group requires higher communication complexity in both provable broadcast phase and speeding multi-value validated byzantine agreement(MVBA) phase.

In figure5.4 x-axis is again the batch size(number of transactions proposed) and y-axis is the throughput(number of transactions submitted to block per second). Each coloured line indicates the number of nodes in the consensus.

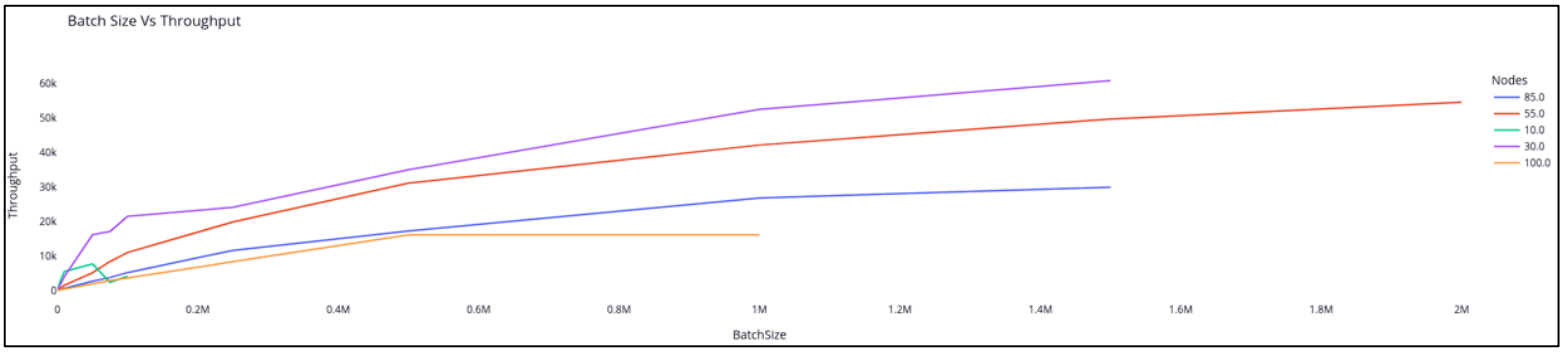

Figure 5.4: Throughput(txs/s) vs BatchSize(# proposed txs)

Here we can observe that as the batch size increases in each consensus group, so does the throughput, which makes sense given the increased number of proposed transactions in each epoch. Furthermore, the rate of increase of throughput decreases as soon as the batch size gets bigger than an instance can handle.

The best performance is again in consensus group with 30 nodes with peak throughput at 61000txs/s and this happens because as mentioned before the larger the number of nodes participating gets the communication required gets higher which results in higher latency and thus the lower throughput.

5.3.2 Stable Batch Size with Varying number of Nodes Participating

The following graphs compare the latency and throughput dependent on the number of nodes participating in the consensus group with a stable batch size.

In figure5.5 x-axis is the number of participating nodes and y-axis is the latency(seconds needed for 1 block to be generated). The stable Batch size used is 100000.

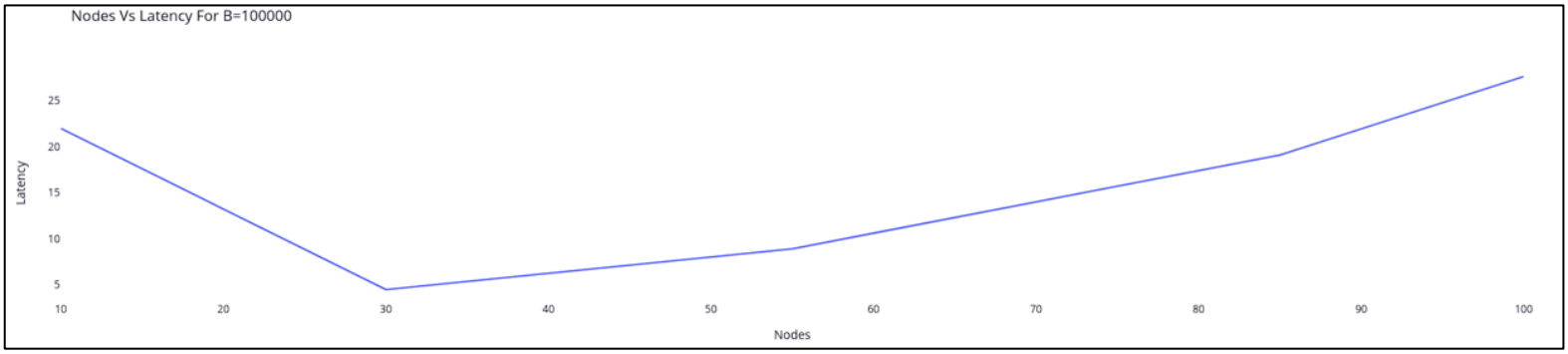

Figure 5.5: Latency(s) vs Nodes(# participating nodes) B=100000

At the beginning of the graph, we can see an anomaly where latency is higher at 10 nodes than 30 nodes and then it starts to increase. This occurs due to the fact that the 10 nodes cannot handle the communication complexity created by the B=100000. After the 30 nodes the latency starts to increase while the number of nodes rises because as stated in section5.3.1 the higher the consensus group the higher the required communication it gets.

In figure5.6 where the Batch size is smaller(B=5000) than figure5.5 we can observe than every consensus group can handle the batch size and the latency always increases with the size of the consensus group.

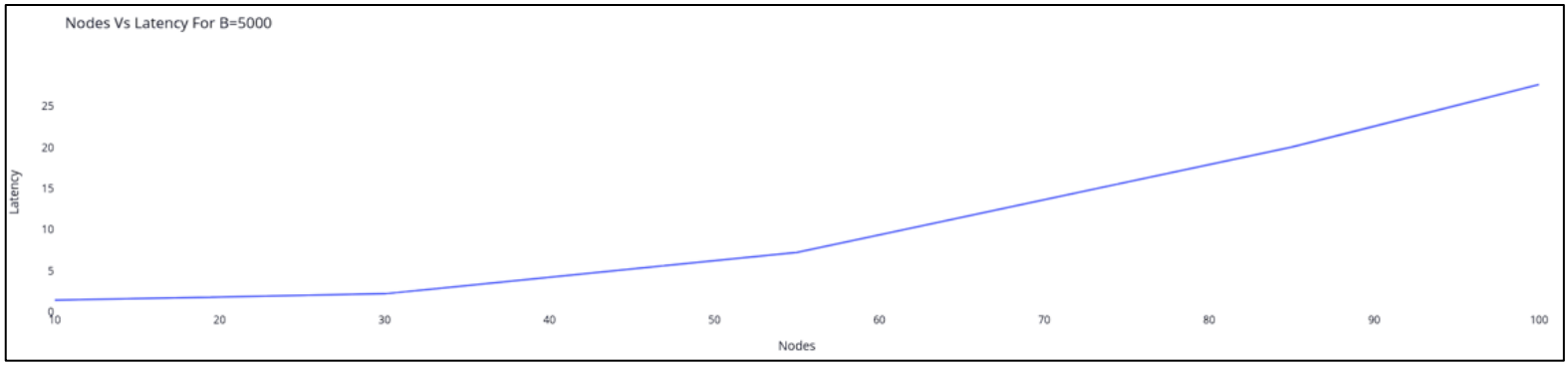

Figure 5.6: Latency(s) vs Nodes(# participating nodes) B=5000

In figure5.7 x-axis is again the number of participating nodes and y-axis is the throughput(number of transactions submitted to block per second). The stable Batch size used is 100000.

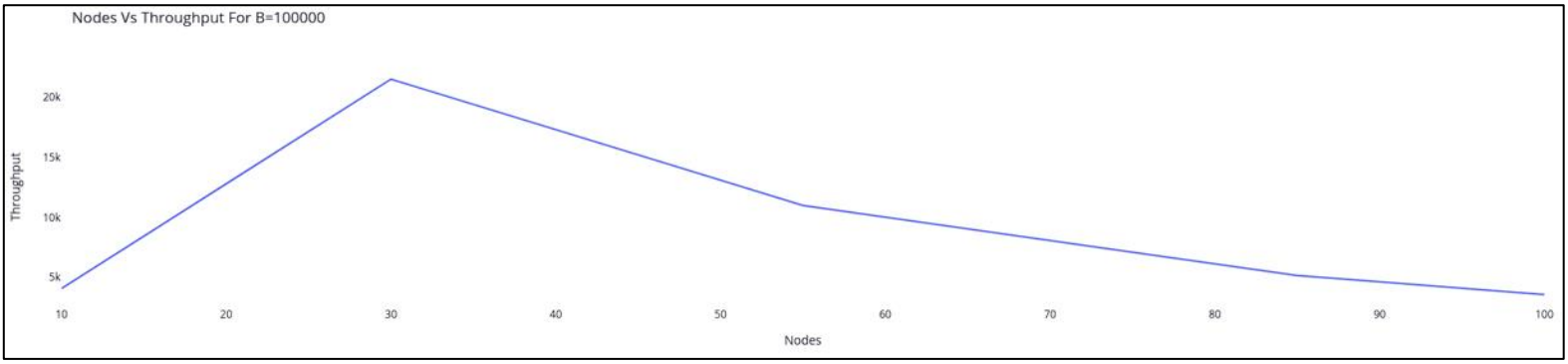

Figure 5.7: Throughput(txs/s) vs Nodes(# participating nodes) B=100000

For the same reason here as figure5.5 throughput is lower at 10 nodes since they cannot handle 100000 batch size and the latency gets higher. As the latency increases after 30 nodes due to the communication complexity the throughput decreases. In addition, we can also see that consensus group with 30 nodes performs better than everyone else with throughput at more than 20000 at B=100000.

5.3.3 Comparing Latency to Throughput with a Stable Number of Nodes Participating

The following graph(figure5.8) compares the latency to the throughput with a stable number of participating nodes and dynamic Batch size.

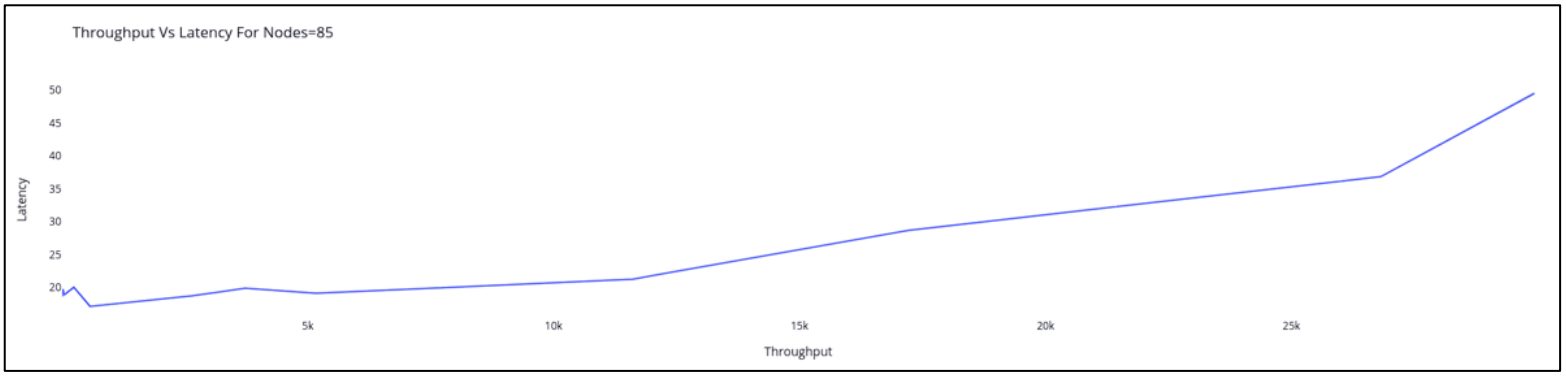

Figure 5.8: Latency(s) vs Throughput(txs/s) N=85

We can easily see that while the latency increases because the batch size gets bigger(communication complexity) the throughput increases too which happens because each block is delayed more and more to be generated and thus the number of transactions per second decreases too.

5.3.4 Stable B and N with Varying number of faulty nodes

In figure5.9 we see the effect that the number of faulty nodes has on both latency and throughput. In both graphs the x-axis is the number of faulty nodes, the Batch size is 50000 and the number of participating nodes is 30.

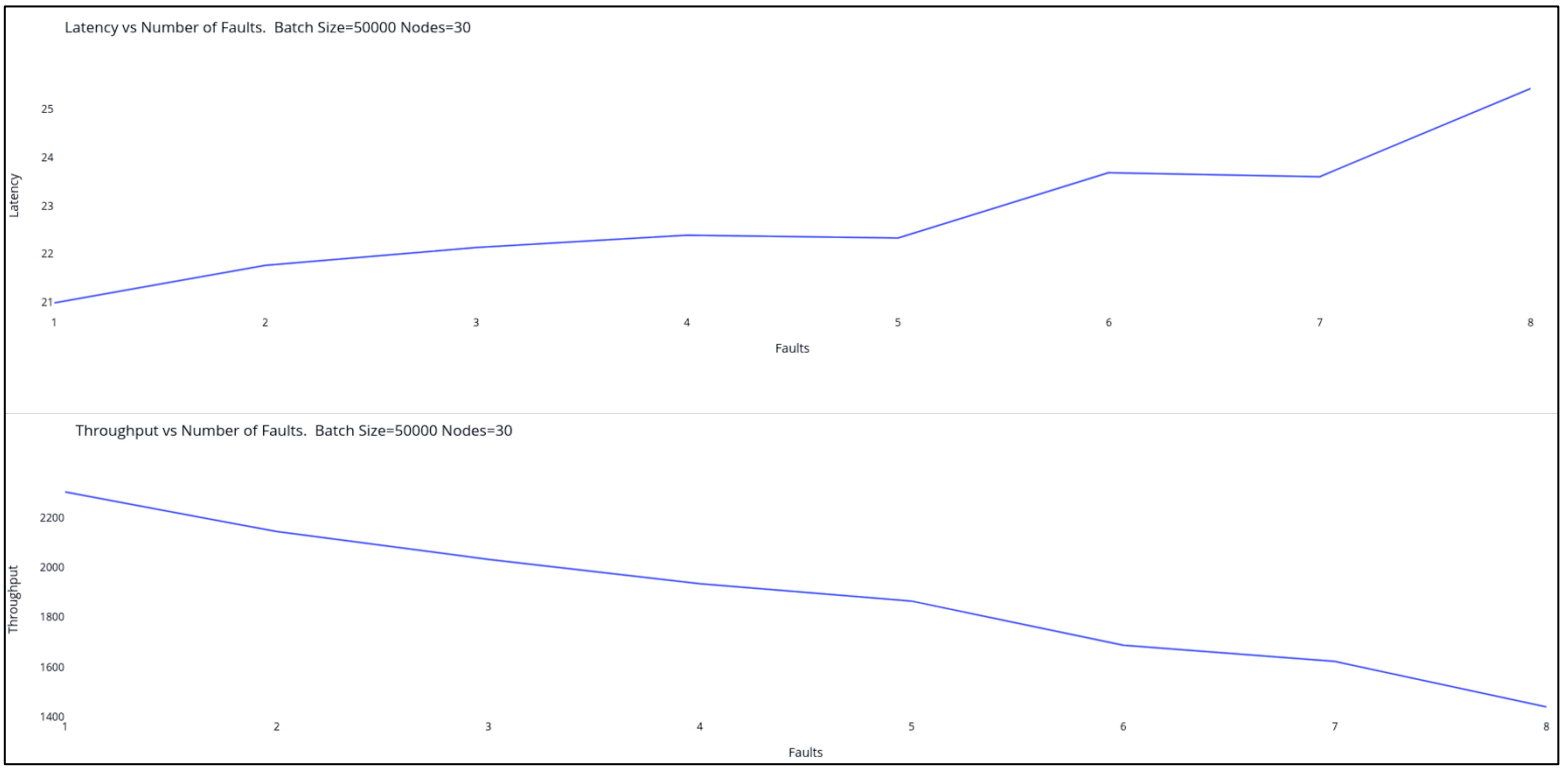

Figure 5.9: (Latency(s) & Throughput(txs/s)) vs Faulty nodes B=50000 & N=30

It is obvious that as the number of faulty nodes increases the latency increases too and thus the throughput decreases since:

I. In Provable Broadcast(PB) the algorithm waits for at least one honest node(f+1) to receive the set of transactions.

II. In Speeding-MVBA the protocol achieves consensus as soon as 2f+1 nodes agree on the set of transactions to supply the block.

Therefore, the bigger the number of faulty nodes the bigger the effort of the involved nodes gets and the latency increases.

5.3.5 Scheduler randomness

HoneyBadger and Dumbo are leaderless consensus protocols which means that every epoch every node proposes a number of transactions from its buffer and the end of the epoch the overlapping transactions are eliminated.

As stated in every paper of both HoneyBadger and Dumbo the selection of transactions from the unbounded buffer should be random in order to select as much distinct transactions as possible and thus increase the number of committed transactions in blocks per second(throughput).

However, the selection from the buffer in the open source code of Speeding Dumbo was First In First Out (FIFO) and thus I replaced it with random selection and compared the latencies.

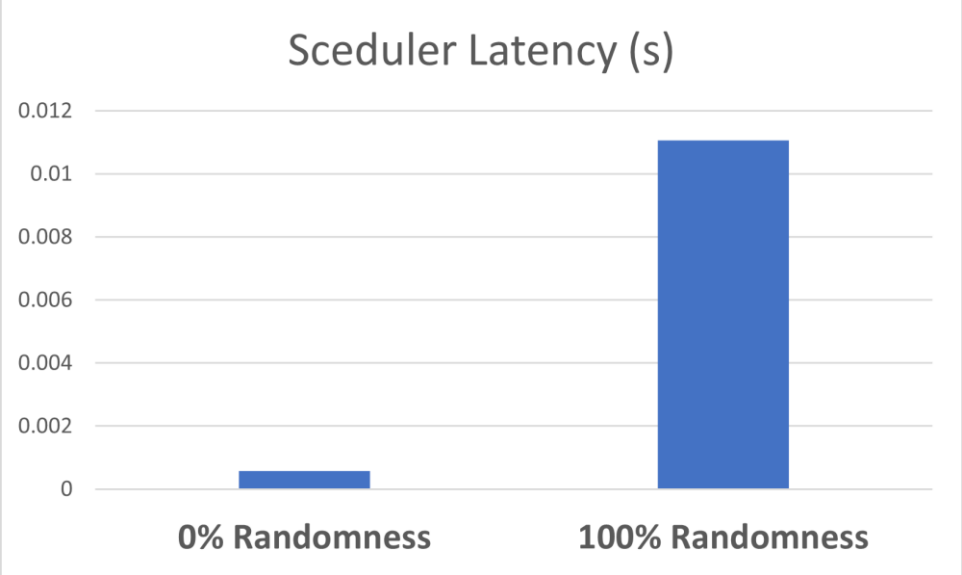

Figure 5.10: Latency(s) vs Randomness

The experiment was setup with Batch size = 10000 and N=10 and after checking the latency of the consensus without including the time needed for selection, I noticed that there was no alteration in the latency. Although as figure 5.10 illustrates, when I compared the latency of the selection phase only, I noticed that the time needed from the random selection commands was making a big difference(from 0.0005s to 0.01s) but not enough to affect the overall latency.

5.4 Results Conclusion

Taking into account each graph, we may conclude that Dumbo performance is very promising and proves that an asynchronous consensus protocol can be practical and as shown in figure5.11 the peak throughput of speeding Dumbo outperforms every consensus protocol with timing assumptions.

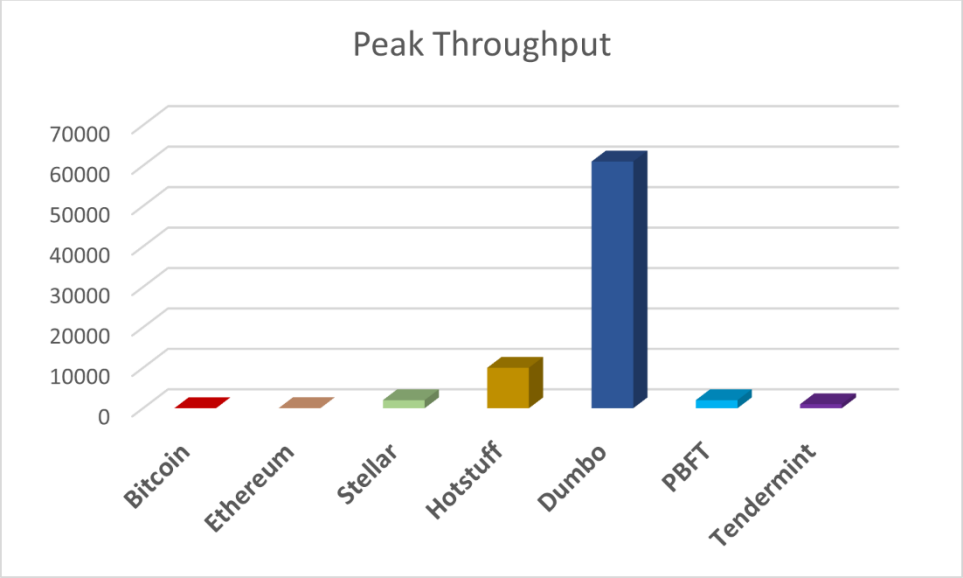

Figure 5.11: Peak throughput(txs/s) vs (Consensus protocols & blockchains)

However, except from throughput which has always room for improvement, the latency still needs large improvement to reach the performance (in terms of latency) of some consensus protocols with synchrony like Hotstuff that have a deterministic algorithm and thus latency lower than 1 second.

Every modification I made in the source code can be found in my GitHub repository [44].

Chapter 6

6 Future Work

Nobody can argue that the performance of speeding Dumbo is highly promising, and more research should be dedicated in the asynchronous consensus protocols to improve even more the performance and exploit the advanced security features it offers compared to synchronous consensus protocols.

In addition, further experiments should be conducted to verify the security of such protocols in an adversarial environment make improvements to withstand the volume of data that will probably be generated in a few years by IoT devices.

6.1 High Level Proposed Model

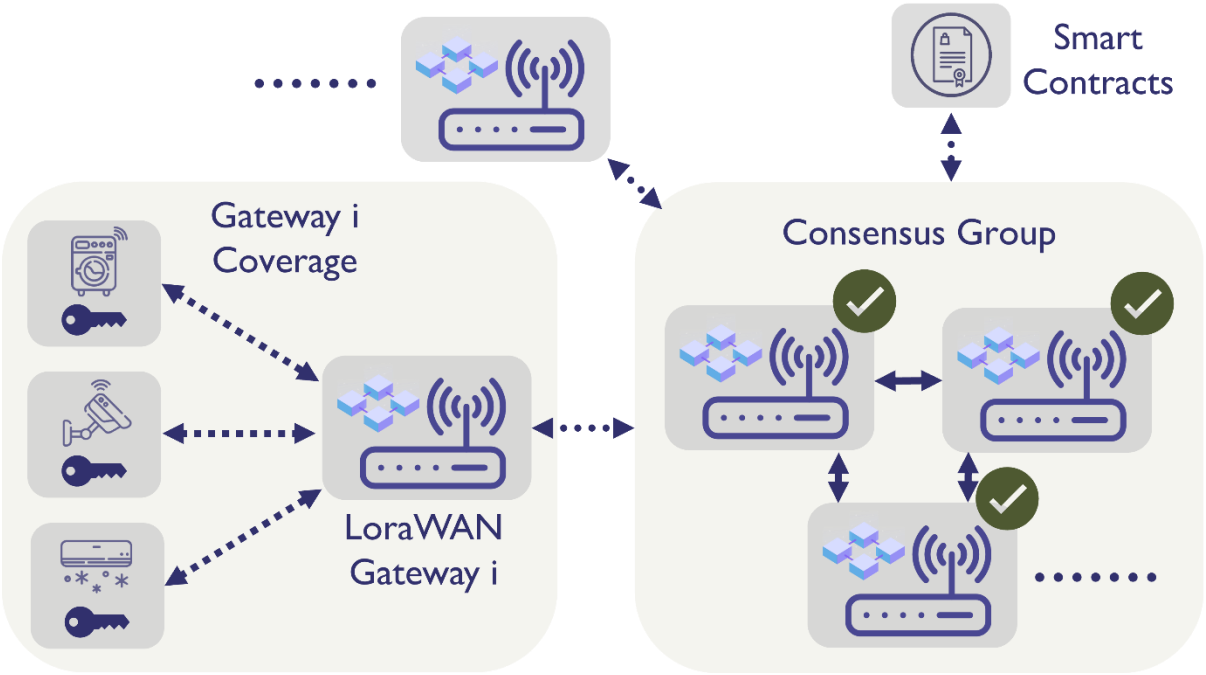

Figure 6.1: Proposed model

A possible architecture of the blockchain integrated in Internet of Things is illustrated in figure6.1. There could be LoraWAN gateways with processing power like in Helium network and each IoT device could send data to the closest gateway i.

Any gateway could be participating in the consensus group if he stakes a predefined amount of tokens, and each epoch(creation of 30-60 blocks) a new consensus group will be voted randomly. To perform well both in terms of throughput and security an asynchronous BFT protocol should be used like speeding Dumbo. The number of participating gateways should be kept around 30 which is a small number it terms of security(speeding Dumbo performs better) but should be okay if every member is staked and most probably honest. If for any reason a gateway is deemed unreliable it should be removed from the consensus group immediately.

Furthermore, smart contracts should be deployed to handle with safety triggers for IoT devices. For example, if a temperature sensor in the forest sends an unusually high temperature the smart contract should inform the closest fire station or if a sensor sense smoke in a building the smart contracts should send a downlink to start the water sprinklers.

An IoT device should submit a transaction in the following order:

I. IoT device sends data packets to LoRaWAN gateway encrypted with its private key.
II. LoRaWAN gateway sends data packets to consensus group.
III. Consensus group verifies the authenticity of transactions IoT device`s public key and starts the consensus protocol.
IV. Consensus group generates a block with the transaction included.

Chapter 7

7 Conclusions

IoT technology has been widely adopted and has reached nearly every household on the planet. By connecting any device with sensors to the Internet a massive amount of raw data is collected and by using analysis tools many useful statistics (related to environment, device state, etc.) can be gathered and improve everyone's life.

The security of such technology is critical since the data gathered and processed is usually sensitive or personal information. However, existing IoT architecture is based on a centralized approach that has a lot of security and scalability issues that must be resolved in order for IoT to be adopted even more widely and to harvest all of its benefits.

Blockchain is a suitable technology that could address the majority of the current issues and enable the IoT world to expand even further. Blockchain is a relatively new technology that still has a lot of legal and technical questions unanswered and challenges that need to be addressed but it has already demonstrated and proved that it can offer many solutions to the present problems.

The performance and security of any blockchain are mainly controlled by its deployed consensus mechanism, which is responsible for the generation and acceptance of a block of transactions in the presence of malfunctioning nodes. The importance of an asynchronous consensus in such environments is explained in this thesis in contrast to the consensus with any timing assumptions which are prone to any delays.

However, deterministic asynchronous consensus protocols are impossible to implement without a source of randomness due to the FLP-impossibility. The first practical asynchronous protocol (HoneyBadgerBFT) was implemented in 2016 and since then, scientists have been looking for ways to improve it or experimenting with alternative randomization methods to implement a practical asynchronous protocol.

This thesis describes a promising asynchronous byzantine fault tolerant consensus called speeding Dumbo. It demonstrates exceptional throughput performance while providing security and censorship resilience compared to consensus protocols with any sense of timing assumptions. Unfortunately, there is still room for improvement in terms of latency, which is why experts are continuously working on it.

To conclude, in general we are at a very good point but a deeper investigation in integrating blockchain in IoT is required. Furthermore, research should be conducted on how to increase the efficiency of blockchain while preserving all the security benefits to satisfy the high demands of IoT technology.

Bibliography

[1] Wikipedia Contributors (2019). Blockchain. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Blockchain.

[2] (Zheng, 2017) An Overview of Blockchain Technology.

[3] (Johnson, 2001) The elliptic curve digital signature algorithm (ecdsa).

[4] (Atlam, 2018) Blockchain with Internet of Things: Benefits, Challenges and Future Directions.

[5] Sharma, T.K. (2020). Types of Blockchains Explained- Public Vs. Private Vs. Consortium. [online] Blockchain council. Available at: https://www.blockchain-council.org/blockchain/types-of-blockchains-explained-public-vs-private-vs-consortium/.

[6] (Lamport, 1982) The byzantine generals problem.

[7] (Jakobsson, 1999) Proofs of Work and Bread Pudding Protocols.

[8] (Zhanga, 2020) Analysis of the main consensus protocols of blockchain.

[9] (Castro, 1999) Practical Byzantine Fault Tolerance.

[10] (Yin, 2018) HotStuff: BFT Consensus in the Lens of Blockchain.

[11] (King, 2012) Ppcoin: Peer-to-peer crypto-currency with proofof-stake

[12] Touron, M. (2019). Centralized vs Decentralized vs Distributed Systems · Berty Technologies. [online] Berty Technologies. Available at: https://berty.tech/blog/decentralized-distributed-centralized.

[13] (Ashton, 2009) That ‘Internet of Things’ Thing.

[14] Wikipedia. (2020). Smart refrigerator. [online] Available at: https://en.wikipedia.org/wiki/Smart_refrigerator.

[15] Cisco and ITU (2016). Harnessing IoT Global Development. [online] Available at: https://www.itu.int/en/action/broadband/Documents/Harnessing-IoT-Global-Development.pdf.

[16] (ITU & Cisco, 2016) Harnessing the Internet of Things for Global Development, 2016

[17] (LoRa Alliance, 2019) A Technical Overview Semtech Corporation December.

[18] The Things Network. (n.d.). What are LoRa and LoRaWAN? [online] Available at: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/.

[19] Gemini. (n.d.). Helium Network: Proof of Coverage & Helium Hotspots. [online] Available at: https://www.gemini.com/cryptopedia/helium-network-token-map-helium-hotspot-hnt-coin#section-introduction-to-the-helium-network.

[20] Martinez, A. (2021). What is Helium Network? [online] Coinmonks. Available at: https://medium.com/coinmonks/what-is-helium-network-fad1dc1e09c2.

[21] Helium (n.d.). Helium Explorer. [online] Helium Explorer. Available at: https://explorer.helium.com/hotspots.

[22] (Helium, 2018) Helium: A Decentralized Wireless Network whitepaper

[23] Helium (n.d.). proof-of-coverage | Helium Documentation. [online] docs.helium.com. Available at: https://docs.helium.com/blockchain/proof-of-coverage/.

[24] Helium (n.d.). proof-of-coverage | Helium Documentation. [online] docs.helium.com. Available at: https://docs.helium.com/blockchain/blockchain-primitives /.

[25] Helium (n.d.). proof-of-coverage | Helium Documentation. [online] docs.helium.com. Available at: https://docs.helium.com/blockchain/consensus-protocol /.

[26] (Fischer, 1985) Impossibility of Distributed Consensus with One Faulty Process.

[27] (Chondros, 2014) Practical Asynchronous Interactive Consistency.

[28] (Vukolic, 2015) The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication.

[29] (Miller, 2016) The Honey Badger of BFT Protocols.

[30] YongraeJo (2019). Honeybadger of BFT Protocols. [online] Available at: https://www.slideshare.net/YongraeJo/honeybadger-of-bft-protocols.

[31] https://www.xdaichain.com/for-validators/consensus/honeybadger-bft-consensus/honey-badger-bft-and-threshold-cryptography-part-3#threshold-cryptography

[32] (Ben-Or, 1994) Asynchronous secure computations with optimal resilience.

[33] (Bracha, 1987) Asynchronous byzantine agreement protocols. Information and Computation.

[34] (Cachin, 2005) Asynchronous verifiable information dispersal. In Reliable Distributed Systems.

[35] (Mostefaoui, 2014) Signature-free asynchronous byzantine consensus with t< n/3 and o (n 2) messages.

[36] Network, P.O.A. (2018). POA Network: How Honey Badger BFT Consensus Works. [online] POA Network. Available at: https://medium.com/poa-network/poa-network-how-honey-badger-bft-consensus-works-4b16c0f1ff94.

[37] (Abraham, 2019) Asymptotically optimal validated asynchronous byzantine agreement.
[38] (Zheng, 2020) An Overview on Smart Contracts: Challenges, Advances and Platforms.
[39] Banafa, A. (2015). Internet of Things (IoT): The Third Wave. [online] OpenMind. Available at: https://www.bbvaopenmind.com/en/technology/digital-world/internet-of-things-iot-the-third-wave/.
[40] Hojlo, J. (2021). Future of Industry Ecosystems: Shared Insights & Data | IDC Blog. [online] blogs.idc.com. Available at: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/#:~:text=IDC%20estimates%20there%20will%20be.
[41] (Lu, 2020) Dumbo-MVBA: Optimal Multi-Valued Validated Asynchronous Byzantine Agreement.
[42] (Lu, 2020) Dumbo: Faster Asynchronous BFT Protocols.
[43] (Lu, 2022) Speeding Dumbo: Pushing Asynchronous BFT Closer to Practice.
[44] https://github.com/pmikel01/Dumbo_UCY