Governing the commons in blockchains

Yackolley Amoussou-Guenou

THÈSE DE DOCTORAT
SORBONNE UNIVERSITÉ

Spécialité: Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)
ED130 — EDITE de Paris

Présentée par

Yackolley Amoussou-Guenou

Pour l'obtention du grade de docteur

Sujet de la thèse :

Gouvernance des Biens Communs dans les Blockchains

Governing the Commons in Blockchains

Composition du jury :

Rapporteurs

Emmanuelle Anceaume	Directrice de Recherche – CNRS, IRISA
Antonio Fernández Anta	Research Professor – IMDEA Networks

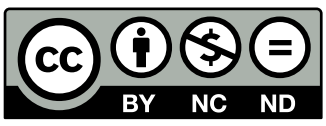
Examinateurs

Amal El Fallah Seghrouchni	Professeure – Sorbonne Université
Petr Kuznetsov	Professeur – Télécom Paris
Julien Prat	Directeur de Recherche – CNRS, CREST & École Polytechnique

Encadrement de la Thèse

Bruno Biais	Encadrant, Professeur – HEC
Maria Potop-Butucaru	Directrice, Professeure – Sorbonne Université
Sara Tucci-Piergiovanni	Encadrante, Cheffe de Laboratoire – CEA List

Thèse soutenue publiquement le *08 octobre 2020*



REMERCIEMENTS

Ces lignes seront sûrement (et statistiquement) les plus lues de ce document. Une explication possible, sans parler de techniques, est qu'elles sont marquées d'émotions, un aspect que nous essayons de supprimer de tout document scientifique afin d'être le plus objectif possible.

First and foremost, I would like to sincerely thank Emmanuelle Anceaume and Antonio Fernández Anta for accepting and taking on their precious time to review this manuscript and providing insightful comments. J'aimerais aussi en profiter pour remercier Amal El Fallah, Petr Kuznetsov et Julien Prat d'avoir accepté de faire partie de mon jury de thèse et de m'examiner.

Je ne peux trouver les mots pour remercier les personnes sans qui cette thèse n'aurait pas existé. Je remercie tout d'abord Maria Potop et Sara Tucci de m'avoir encadré durant ces dernières années. Je les remercie pour toutes les discussions que nous avons eues, autant scientifiques que moins scientifiques. Il y a quelques années, j'aurais sûrement dit ne pas vouloir faire de systèmes distribués ; grâce à elles, j'en fais, et je ne le regrette pas. J'aimerais remercier infiniment Bruno Biais, qui en plus de m'avoir fait connaître un peu plus la discipline des sciences économiques m'a permis d'en apprendre plus et de m'y intéresser, moi qui n'avais aucune connaissance dans le domaine. Je suis heureux et reconnaissant d'avoir effectué cette thèse avec votre encadrement et vos directions. J'ai appris énormément, et je vois qu'il me reste encore tant à apprendre. J'espère pouvoir continuer sur la voie que vous m'avez montrée.

Je tiens aussi à remercier les personnes qui n'ont pas ou très peu eu de liens directs avec ce travail. Pour éviter des remerciements fleuves et des oublis, je ne mentionnerai pas de noms.

J'aimerais remercier les institutions qui ont abrité cette thèse, le CEA List, et le LIP6, en particulier et respectivement, le laboratoire LICIA (plus généralement le DILS) et l'équipe NPA; j'aimerais remercier tous les membres passés et présents avec lesquels j'ai eu la chance de discuter, le tout dans une ambiance toujours agréable, sans oublier ceux du LINCS également.

Je tiens aussi à exprimer une grande gratitude aux institutions comme l'EDITE et ~~l'IFD~~ le Collège Doctoral de Sorbonne Université pour la formation et les expériences vécues durant cette thèse ; l'UFR d'Informatique ~~de Paris-Diderot~~ d'Université de Paris qui m'a permis d'avoir des expériences d'enseignement en tant que moniteur ; … Je remercie aussi les enseignants que j'ai eus et qui par leurs cours et conseils m'ont mené sur la voie de la recherche scientifique.

J'ai une pensée particulière à tous les doctorantes et doctorants avec qui j'ai pu échanger, que ce soit au DILS, au LIP6, avec le comité des doctorant.e.s, durant MT180SU, et surtout ceux et celles rencontrés durant divers évènements comme les conférences, séminaires, ateliers, … Au plaisir de vous retrouver dans un futur pas trop éloigné, et je vous souhaite le meilleur !

Merci à mes amis, qui me subissent depuis des années, et surtout à celles et ceux qui ont été là durant ces trois dernières années assez particulières. Merci enfin à ma famille qui m'a toujours soutenu, et en particulier à mes parents sans qui tout cela ne serait pas possible.

À toi aussi lecteur de ce manuscrit, j'espère qu'il te servira.

Akpé, Awanou, Barka, Bedankt, Choukrane, Danke, Dhanyavaad, Djarama, Gracias, Grazie, I ni ce, Merci, Mulțumesc, Spassiba, Teşekkürler, Thanks, Tôt fôfô, Xièxiè…

PREFACE

Blockchains are one of the most appealing technologies over the last years, both for scientists and the general public. Blockchains are distributed ledgers that aim to offer transparency, integrity and many more advantages over their centralised counterparts. Blockchains were "revealed" and became popular thanks to the creation and rise of the cryptocurrency Bitcoin. Over the years, blockchain technologies become more and more popular with an exceptional peak in 2017. Blockchains are becoming mainstream technologies, as there is an observatory for blockchains established by the European Commission, blockchain forums in many countries, blockchain start-ups are flourishing, scientific conferences are discussing the topic, and even some scientific conferences are now specifically dedicated to the technology, *etc.*

The blockchain technology promises, thanks to its integrity and transparency properties to be useful and interesting in various domains, and not only for financial systems. However, many questions and doubts float around it. Is it environmentally viable? Is the technology even ensuring its promises? Can blockchains be used in real-life settings, *etc.* For instance, Bitcoin, the most popular blockchain system is not environment friendly since it requires lot of computing powers. It does also not offer strong consistency, because participants may have different information at the same time, which is not desirable for critical systems.

Among Bitcoin's alternatives, an interesting class is the *committee-based blockchains*, in which updates are done by committees, *i.e.*, subsets of participants. Committee-based blockchains have many advantages, for example, they do not need huge computations to work, and they guarantee consistency under known conditions by relying on research from distributed systems. However, these blockchains were not formally studied. In this thesis, we defined the problem they solve, and as use case, we analyse the correctness of one of the most used committee-based blockchain; these contributions have been published in the proceedings of peer-reviewed conferences [15, 16, 17].

The distribution of rewards to a committee when some of the participants may be faulty is important but was not studied in the context of blockchains. In this thesis, we study that question from a fairness point of view and provide a formal *definition of the fairness* problem for committee-based blockchains. Furthermore, we study the impact of the communication on the existence of fair reward distributions. This has been published in the proceedings of the following peer-reviewed conferences [16, 18].

The costs and rewards also allow to exhibit rational and selfish behaviours in the system, participants who want to maximise their gain while incurring minimal (or even no) cost. It is important to ensure that the system is resilient to such participants. Our last and main contribution of this thesis relies on the combination of approaches from computer science and economics. We propose a model to *analyse the behaviour* of participants in committee-based blockchains when participants can have different interests; by using the model, we show that even though blockchain properties cannot be always guaranteed, under some conditions, there are situations where the blockchains' properties are satisfied. Part of these contributions is published in the proceedings of peer-reviewed conferences [12, 13, 14].

CONTENTS

CHAPTER 1

INTRODUCTION

Contents

Alice sells cards, and Bob wants to purchase them, but they do not live in the same place. Alice does not want to send the cards to Bob unless she has proofs that Bob indeed has enough funds and made the payment. In the same fashion, Bob does not trust Alice and wants to ensure that she is sending the cards. This is not an issue since there is a classical solution: using an intermediary. To achieve the exchange between Alice and Bob, Carl will serve as an intermediary and will verify if the operation can be done. Both Alice and Bob will send their products to Carl and pay intermediation fee to Carl. Once Carl receives the goods, he can send the cards to Bob, and the funds to Alice; the exchange is then over.

To make the operation going as smoothly as possible, Carl needs to be trusted by both Alice and Bob, and he should correctly do his tasks. We can ask ourselves what happens when Alice or Bob do not trust Carl. Such distrust can be explained by Carl's history of bad behaviours, or the distrust can be the result of fear over the big powers Carl has in hands; Carl can ask for too much intermediation fee, or he can take the cards/the funds and then disappear.

The fear of Alice and Bob can be easily explained because Carl can do whatever he wants once he receives the goods. The operation system is too *centralised*. To avoid that, one may propose to use a huge number of people as intermediary at the same time, *i.e.*, decentralising and distributing the validity verifications. That is basically the proposition of *Bitcoin* [128][1]: make financial exchanges in an open setting between any set of individuals, where failures can happen, participants can behave maliciously, and participants do not necessarily trust one another. In Bitcoin, everyone can see and verify the validity of all transactions.

The cryptocurrency Bitcoin, as well as most of the other cryptocurrencies, relies on the distributed ledger technology: *blockchain*. As of writing this manuscript, blockchain is one of the most appealing technologies since its introduction in the Bitcoin's white paper [128] in 2008. Blockchain systems, similar to peer-to-peer systems in the early 2000s (*e.g.*, Napster [109]), take their roots in the non-academic research. After the release of the most popular

[1] Actually, Bitcoin does only consider funds' validity and financial exchanges; not real life objects.

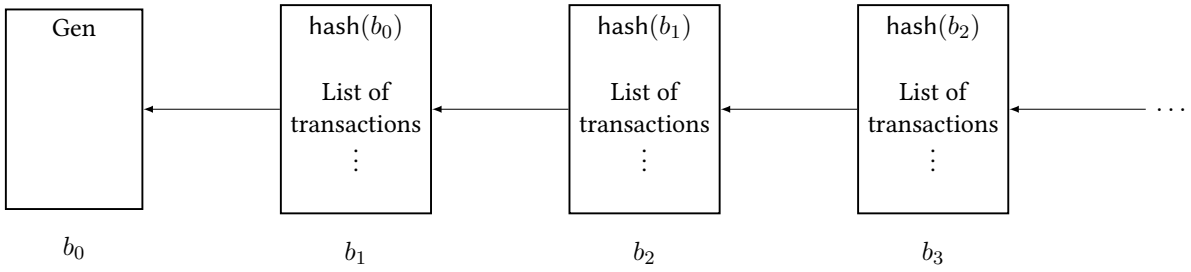


Figure 1.1: Example of an Ideal blockchain

blockchains (*e.g.*, Bitcoin, Ethereum [159], *etc.*) which have a specific focus on financial transactions, the huge potential of blockchains for various other applications ranging from notary to medical data recording became evident. In fact, blockchains offer, among other advantages, transparency, and the information once stored in the ledger cannot be removed nor corrupted.

Even with all these proclaimed or promised advantages, can we really trust the blockchain technology? To put it another way, do they work? Are they really useful? Do we have an interest in using them? These are somehow the questions that drove this thesis. The main questions addressed in this thesis, and presented in this manuscript are:

- whether blockchains are correct, *i.e.*, do they achieve what they are expected to do; and
- whether blockchains can be used in an environment where each user wants to maximise its personal gain.

In the following, we will explain what a blockchain is, the various interests in the technology, and our contributions.

1.1 Blockchains & Context

In blockchain systems, the only way participants communicate is by exchanging messages, and they aim to maintain a *distributed ledger* (fully decentralised, with no central authority), *i.e.*, all participants have locally the exact same ledger, and information added in the ledger of one participant should be added in the ledger of all the other participants in the same order. Blockchains should be resilient to malicious attacks. The distributed ledger should satisfy some guarantees:

- *Tamper-resistance*: modification of an information already in the ledger should be difficult, even impossible to achieve;
- *Append only* construction: new information can be added only at the end.

In particular, it should not be possible to modify (remove, change, *etc.*) information already in the blockchain, hence the advantages of the "append-only" guarantee.

To achieve the above properties, a blockchain consists of a continuously growing history of ordered information, encapsulated in blocks. Information (blocks of data) can be anything depending on the applications, for instance, it can be financial transactions as in Bitcoin, it can be information about documents for tracking modifications, it can also be programs called *smart contracts* [147] that can be executed by everyone locally, *etc.* Smart contracts may help to automatically execute some set of instructions whenever an event happens (*e.g.*, two conflictual information in the blockchain).

Each block (pack of information) in the blockchain is linked to the prior block, as presented in Figure 1.1. All the information in the blockchain is then always ordered. The blockchain itself is a distributed ledger replicated among the different participants.

To ensure that each block is uniquely identified, and is linked to the prior block, blockchains rely on mathematical functions providing the uniqueness of identification: *collision-resistant hash functions*, *i.e.*, each block contains the hash of the previous block. Hash functions basically allow producing a unique identifier from a sequence of data such that any (even slight) modification in the input data results in a completely different identification. To preserve the chain structure, the participants need to agree on the next block to append in order to avoid the so-called *forks*, where two participants or more do not have locally the same chain. When one participant adds some information in its ledger, all other participants should be aware of that, and add the same information in the same order. Many techniques exist and were proposed to select for each block one participant that will have the right to add the next block. These techniques are diverse, and many have some issues, either technical or societal. For example, Bitcoin, the most known blockchain application demands high energy consumption, which is not good, especially concerning to the ongoing climate change. Bitcoin and its variants do not provide strong consistency, *i.e.*, forks can happen. Forks are an important issue for the adoption of blockchain technologies in critical applications (*e.g.*, finance). Moreover, in most applications, forks resolution is done only with probabilistic guarantees, where deterministic guarantees are desirable in various applications.

In this thesis, we study techniques to add new information (blocks) in the blockchain by ensuring that forks will not happen under clear and deterministic assumptions.

Even if all participants have the ability to add new blocks in the blockchain, not necessarily everyone wants to do so, and not everyone is always selected to add the next block. Participants that want to participate in the addition of new blocks in the blockchain are said to *maintain* the blockchain, and we call them *block creators*. The creation of blocks and the procedure to make a block accepted by the others may be costly for a participant; therefore, to incentivise participants to serve as block creators, once a (set of) participant successfully adds a block to the blockchain, it is rewarded for its work. Rewards should be given fairly to the different participants by respecting their work and investments; however, how to distribute the rewards and whether the rewards are really an incentive are complicated questions that we studied and discussed in this manuscript.

1.1.1 Interdisciplinarity and Applications of Blockchains

Although the first known blockchain system (Bitcoin) is a financial application, the research questions blockchain systems raise span over multiple scientific fields.

Medicine and Health. In health, for instance, for the management of sensible (personal and health) related data, in particular, when such data should be used by different services, there is a need to protect both anonymity and integrity of the information [142]. Blockchains are especially built to protect that. More specifically, the use of smart contracts to give reading rights only to special users, for instance, only medical doctors from specific medical institutions.

The Internet-of-Things (IoT) which connects multiple devices using the Internet (or other communication protocols) is of interest in health [97]; it helps for tracking and tracing patients for therapy monitoring, treatment follow-up, *etc.* Some blockchains are especially dedicated to IoT devices (*e.g.*, [137]). Studies show the advantages and research questions raised by using blockchain systems for IoT in healthcare [67].

Economics & Finance. Economics is the social science that studies the behaviours and the different interactions of participants in given situations. Since human takes big roles in blockchain systems, and their behaviours may seem sometimes unexpected, it is important to study what they are exactly doing, and understand why they are doing so [31]; this is a line of research using tools from the field of *game theory*. Game theory is the branch of mathematics whose goal is to study the optimal decision-making process in presence of multiple decision-makers. Another topic of interest is the impact of blockchain systems in financial markets by proposing models to match and then predict the evolution of the systems (*e.g.*, [139]).

Finance is the topics where start-ups and most communication about blockchain are targeted to. Blockchains will change the financial system is the promise from many "gurus". Blockchains emphasize some advantages decentralised and distributed solutions may have over centralised solutions. The comparison between the advantages, drawbacks, and when one approach is better than the other is highly studied. In particular, blockchains may give rise to new financial applications that are impossible using traditional centralised approaches [119], since they provide new tools [138].

Law. The link between law and blockchains may seem far-fetched. Especially when the obvious link of *smart contracts* are neither smart nor are contracts. However, one can note that in blockchains, it is wanted that information should not be removed. This kind of property violates the *law to be forgotten*, and more generally the *General Data Protection Regulation* (GDPR) [71] of the European Union. The usability of such technology is then highly studied. We can however see many applications that can be managed more efficiently with blockchains. For example, notary or real estate, where information can be secretly kept in a non-violable fashion and the ownership can always be proved. In the same line, we can also name identity management. The question of *code is law, i.e.*, whether code can be regarded as law [114], or how code should be considered by law has taken a renewed interest with the rise of blockchains.

Computer science. Computer science is probably the science that has the most fields interested in blockchains. We can highlight for instance cryptography [100], which is at the core of blockchains, to guarantee the veracity of information sent, and of sharing securely information. Network science is also highly touched; participants in blockchain systems communicating to one another usually over the Internet, or other means that are studied by network scientist. Distributed systems [117] have a lot to say related to blockchains; multiple participants agreeing on common information is one of the most studied questions in the field. Formal methods [158], in particular formal language, is also interested in blockchains and more specifically in smart contracts [147]; smart contracts are written in languages, and having a good and useful language is important to guarantee that things can be effectively done as expected.

Video Games. Less of a science, the video game industry is also interested in the topic [125]. First, the existence of a currency existing only in their ecosystem is something they aim to do for a long time. Moreover, the fact that information can be stored and checked by anyone having the correct access, and that such information cannot be modified may be a good addition to ranking competitions in video games.

Traceability. We finish off this short listing with traceability. We can take the example of *supply chain* in the food industry [49], where food scandals are the main topics of the media every once in a while, mostly due to lack of frequent and systematic quality control. With the use of blockchains and smart contracts, we can foresee the automation of the verification processes [98]; all new information added can be certified and the verification of their control

validity can be done every time and automatically, allowing to give more confidence in the final product, and hence avoiding scandals.

Most of these research questions cannot be replied by one field alone, since none of them has the full legitimacy regarding blockchains, they need insights from one another. We note that blockchain systems can be a great pretext to promote and to achieve more interdisciplinary research.

There are many more topics that show interest around blockchains. In the above descriptions, we just highlight some of them. Blockchains are quite a nice and strong candidate to foster multidisciplinary work. We present the following example that drove this thesis.

In all applications targeting the use of blockchains, decentralisation is a key point of why they are interested in the technology. Being interested in studying the correctness of blockchain systems, because of their intrinsic decentralised nature, techniques from distributed systems should be considered, since they are perfect and made for those analyses. Being also interested in studying the behaviours of participants in blockchain systems, the available tools from distributed systems are not sufficient alone; techniques from other fields such as game theory should be considered. Therefore, during this thesis and in this manuscript, we studied blockchain systems by combining methodologies from the following two fields: *distributed systems* and *game theory* detailed below.

1.1.2 Distributed Systems

In distributed systems, many participants communicate to achieve common goals. Participants locally use the same algorithm, called a distributed algorithm [117]. The algorithm is said to be correct with respect to a given problem or a task when the task is completed successfully. Some participants may fail (computer bugs, malicious behaviours, *etc.*) and not execute the algorithm; therefore, a desirable property is to ensure that the algorithms designed are tolerant and resilient to such failures. The design and analyses of distributed algorithms are fundamental and complex, and is a research field on its own: *distributed computing*, which can broadly be defined as the study of distributed systems.

Distributed systems usually deal with two kinds of systems, (i) on the tiny or small scales, *shared memory* also called concurrent systems, where all participants have access to the same shared memory where they can read and write; and (ii) in large scales, *message-passing* distributed systems, where the only interactions between participants are done by sending messages to one another. Blockchain systems are exactly message-passing systems, and so the decades of research in the field of distributed systems are useful to study and analyse blockchains from a technical and correctness point of view.

1.1.3 Game Theory

Although works in game theory existed in the 1930s, the field of game theory was really introduced by the book *Theory of Games and Economic Behavior* from the mathematician John von Neumann and the economist Oskar Morgenstern in 1944 [156].

Even though game theory is mostly used in economics, it has multiple applications in many other sciences, whenever interactions may happen. That is exactly the case of distributed systems such as blockchains. Using game theory, we can represent the multiple interactions existing between participants, and make predictions about the rational behaviours of the participants in the blockchain, where rational is taken to mean, “does the best for oneself”. Analysing the behaviour of participants in a system can allow making predictions about

how participants may behave in the system; it also allows responding to the question: *can blockchains be really used in real-life scenarios?*

It is important to mention that game theory is studied in computer science. The field of *algorithmic game theory* [130] is a field at the interface of both computer science and economics. However, the works in algorithmic game theory are mostly algorithmic, *i.e.*, they study questions like (i) how complex is the computation of stable situations, (ii) does there exist optimal or stable situations given some settings, *etc.* They also use game-theoretical tools to design algorithms for given problems. We have to stress out that in (classical/economical) game theory, the study is about the participants' behaviours, whereas in algorithmic game theory the analyses use the behaviours of participants to design algorithms. Once these algorithms are designed, one can actually do a game-theoretical analysis of them. To understand and analyse the behaviours of participants in blockchain systems, as studied in this thesis, the economical approaches fit the best.

More generally, one can wish to see extended and more generalised collaborations between computer science and economics. Such collaborations will give the necessary insights between the behaviours of both computers and humans in a society where they are more and more intertwined. We take a step in that direction by analysing algorithms inspired by real life (currently in use) blockchain algorithms in the presence of rational decision-makers.

1.2 Contributions and Organisation

Recall that all participants in the blockchain systems although having the ability to add new blocks in the blockchain, do not necessarily want to do so; and not everyone is always selected to add the next block. The ones who add, try to add, or simply have the rights to add new blocks in the blockchains are the block creators.

Many mechanisms exist for selecting which participant(s) has/have the right to add the next blocks in the blockchain; similarly, different mechanisms and proposals are made on how rewards should be given to ensure some desirable properties. The key component to handle for using properly a distributed system such as a blockchain is agreeing on common and shared information. Participants need to have consensus on the blockchain. In this doctoral thesis, we study generally how to use distributed consensus, and specifically decades of work from the distributed systems community, to build a blockchain. We are moreover interested in the impact of selfish participants (wanting to maximise their gain with minimal effort) of the consensus.

This manuscript is organised as follows. Chapter 2 presents the different models and notions used throughout the manuscript; Chapter 3 provides a state-of-the-art of the research related to the studies done in this thesis; the technical results of this thesis are presented in Chapters 4 - 6; Chapter 7 summarises the different results, and gives an overview of interesting research questions opened by or closely linked to the results from this thesis.

Before going to the next chapter, we first give a glance of our contributions.

Committee-based Blockchains. Building a blockchain is quite challenging, even if the number of new blockchains is following an exponential growth. To verify if a blockchain proposal is correct, one needs first to understand what a correct blockchain means. In this thesis, we focus on a specific type of blockchains called *committee-based blockchains*. In these blockchains, for each block, there is a subset of participants, a committee that produces that block, while in most other blockchains such as Bitcoin, the goal is to delegate that work to exactly one participant. Under some assumptions, they offer guarantees of consistency of

the blockchain, *i.e.*, the blockchain will always have the ideal chain structure (Figure 1.1). Moreover, committee-based blockchains rely on committees to maintain the blockchain, they use techniques from decades of research in the field of distributed systems, specifically on the problem of consensus. Committee-based blockchains were more often used in consortium settings, where the block creators are known and clearly defined.

In open settings where anyone can participate, committee-based blockchains were not studied. In this thesis, we analyse how committee-based blockchains can be used in open settings. In more details, we formally define the problem committee-based blockchains are solving: the *Byzantine repeated consensus problem*; we then examine a popular committee-based blockchain against this problem and study its correctness. Committee-based blockchains are now considered as an important class of blockchains that are used or plan to be used in many applications.

These contributions have been presented and published in the proceedings of peer-reviewed conferences [15, 16, 17].

Fairness Definition and Analysis. Participants in blockchain systems, and in particular block creators (those maintaining the system by adding new information) are rewarded for their work. Knowing that some participants can have unintended, malicious, or faulty behaviours; to prevent rewarding them more than their correct counter-parts, the designer of the protocol needs to guarantee a certain fairness in the reward distribution, in the sense that each participant has a share of the rewards proportional to its efforts.

The rewarding is a new aspect introduced by blockchain that was not really considered and not studied in distributed systems. More specifically, in committee-based blockchains, rewards need to be shared inside the committee, a fair repartition is a good property users may like. However, there was a lack of studies on the topic, and there was no clear definition of the problem. Our first contribution regarding the fairness of committee-based blockchains is to formally define the fairness of reward distribution in committee-based blockchains. Reflecting on when fairness can be achieved, our second contribution is to show the impact of the communication delays on fairness.

Our contributions on the fairness of committee-based blockchains have been presented and published in the proceedings of peer-reviewed conferences [17, 18].

In Chapter 4, we present both our contributions related to the correctness of committee-based blockchains, as well as analyses and discussions about fairness in those blockchains.

Game Theoretical Analysis of Committee-based Blockchains. As said above, block creators are rewarded for their work. However, it should be noted that executing the protocol might be costly. One may ask what are the behaviours of "selfish" participants, *i.e.*, participants that want to have more reward when at the same time paying little or no cost.

To understand the impact of these "selfish" participants on the system, we model their behaviours in committee-based blockchains. Using economical approaches and game theory, we study the outcome of systems where the participants are selfish. The system studied is inspired by real-life committee-based blockchain protocols. We theoretically analyse the behaviours of the participants under different reward schemes and show that there are situations where the blockchain properties are guaranteed, however, there are many other outcomes where the properties are violated; moreover, our analysis shows that rewarding all the committee once they produce a block seems to be a better scheme than rewarding only committee members whose works are visible. *These contributions are accepted and will be presented and published in the proceedings of the following peer-reviewed conference [13].*

Furthermore, as done in distributed systems, and since they are present in real settings, it is interesting and important to consider the presence of malicious participants in the system. To do such analysis, it is important to have the necessary tools. In fact, malicious behaviours and obedient participants are not often considered in economical analyses. Our first contribution on the question is to present a model that can encompass all three types of participants: (i) malicious who wants to prevent the blockchain properties to be achieved, (ii) obedient who always follows the prescribed algorithms, and (iii) strategic/selfish participants who want to increase their personal gain.

Using that model, we study outcomes resulting from the different types of behaviour in committee-based blockchain systems where both malicious and selfish participants are present. A major contribution is to show that under certain conditions, *i.e.*, bounds on the number of malicious participants, *etc.*, there are outcomes where the blockchain properties can be guaranteed, however, there are other outcomes where the properties are violated.

Part of these contributions have been presented and published in the proceedings of peer-reviewed conferences [12, 14], and has been awarded the *best student paper award* of [12].

Chapters 5 & 6 address the analyses of rational behaviours in committee-based blockchains.

CHAPTER 2

MODEL

Contents

John von Neumann said *"the sciences ... mainly make models"*. Models are an essential part in the scientific knowledge. A proved result is applicable only in the model that was used. If one wants its result to be applied in real environments, it has to use a model that reflects or captures the essence of what is studied and its environment. Once such a model is defined, some realistic interpretations of the results can be done, and some predictions can be proposed.

In this thesis, we did not intend to create new models for blockchains systems. Blockchain systems being in nature distributed systems, they mainly inherit models from the distributed systems literature. The literature of distributed systems is quite vast, and realistic models have been proposed. In this chapter, we introduce the different models that we will use. All following chapters will recall the model they use and for more in-depth details, the reader can come back to this chapter.

The models take into account the number of participants in the systems, their behaviour, how connected they are, how they can exchange information, *etc.* This chapter also aims at providing the basis and notations used in the following chapters.

2.1 Notations

We introduced in this section some notations we use throughout the manuscript.

Let S be a set, we denote by $|S|$ the cardinality of S; and 2^S or $\wp(S)$ is the power set of S, the set of all subsets of S.

Let $b \in \{\texttt{false}, \texttt{true}\}$ be a boolean value. $\mathbb{1}_b$ denotes the indicator function, which outputs the value 1 if $b = \texttt{true}$, and 0 if $b = \texttt{false}$. Let $b_1, b_2 \in \{\texttt{false}, \texttt{true}\}$ be two boolean variables. The formula $b_1 \wedge b_2$ is evaluated at $\texttt{true}$ if and only if $b_1 = \texttt{true}$ and $b_2 = \texttt{true}$, and at $\texttt{false}$ otherwise. The formula $b_1 \vee b_2$ is evaluated at $\texttt{true}$ if $b_1 = \texttt{true}$ or $b_2 = \texttt{true}$; if $b_1 = b_2 = \texttt{true}$, $b_1 \vee b_2$ is also evaluated at $\texttt{true}$, *i.e.*, $b_1 \vee b_2$ is evaluated at $\texttt{false}$ if and only if $b_1 = b_2 = \texttt{false}$. The formula $\neg b$ is evaluated at $\texttt{true}$ if and only if $b = \texttt{false}$.

Let A be a non-empty set. We denote by A^* the set of all finite concatenations of elements of A, *i.e.*, if $u \in A^*$, then either $u = \varepsilon$, or $\exists k \geq 1 : u = u_1 \cdot \ldots \cdot u_k$, where $\forall i \in \{1, \ldots, k\}, u_i \in A$. The operation $\cdot$ represents the concatenation, the empty word ε is such that $\forall a \in A, a \cdot \varepsilon = \varepsilon \cdot a = a$, and we write A^+ for the set $A^* \setminus \{\varepsilon\}$. Note the following exception, $\mathbb{N} = \{0, 1, \ldots\}$ is the set of natural numbers, and $\mathbb{N}^* = \mathbb{N} \setminus \{0\} = \{1, 2, \ldots\}$ is the set of all positive natural numbers.

Let A and B be two events of probability. We denote by $\neg A$ or $\bar{A}$ the complementary event of A. We denote by $\mathbb{P}[A|B]$ the probability of having event A assuming that event B happens. Let λ be a random variable, we write $\mathbb{E}[\lambda]$ for the expected value or the average value of λ, and we denote by $\mathbb{E}[\lambda|A]$ the expected value of λ assuming that event A happens.

2.2 System Model

2.2.1 Participants

A blockchain system is composed of a set Π of *participants* or players, or processes communicating by exchanging messages over a network. Messages represent (a batch of) transactions. We consider that messages have finite length and can be arbitrary sequences of information.

A participant models a computer program, the couple human/machine, *etc.*, that takes part in the blockchain system. In this manuscript, we will use the term participant to refer both to players and processes. The use of a single term throughout the manuscript is for the sake of consistency.

2.2.1.1 Execution Model

Each participant has an internal state and proceeds in rounds. At the end of each round, the participant goes to the next round and so on. A round is composed of three phases:

- A *send phase* at the beginning of the round. It is during the send phase that a participant can send messages to the other participants. The message to send should be prepared during the previous rounds, and as for the send phase of the first round, an initial (default) value may be sent. We consider that the send phase is executed atomically no matter the number of messages or the size of each message. Atomically means that if one message is sent then all messages prepared for this phase are also sent. After the send phase, the participant goes to the delivery phase.

- A *delivery phase*: During this phase, a participant collects from the network messages previously sent. These messages can be from the current round or the previous rounds.

We consider that participants have an unbounded memory, *i.e.*, they can store all messages they collect. When a participant effectively collects a whole message and can process it, we say that it has *delivered* the message, otherwise, even if part of the message is collected, it has not delivered the message. The delivery phase has a finite and positive duration. The duration represents the time the participant plans to wait for collecting messages sent. After its delivery phase, the participant goes to the compute phase.

- A *compute phase* at the end of the round. In compute phases, a participant uses all messages it delivered to update its internal state and to prepare the messages for the next round. After the compute phase, the participant goes to the send phase of the next round.

Hence, participants receive information from their environment by means of messages, they use the received information in their computations and they produce messages to the environment.

The duration of the send phase and the duration of the compute phase are negligible since they are absorbed by the duration of the messages exchanges. We then consider the duration of one round as being exactly the duration of its *delivery phase*. For the same or for different participants, different rounds can have different duration.

Formally, the execution of each participant can be viewed as an *input/output automaton* [117]. An input/output automaton $\mathcal{A}$ is a tuple $(\mathit{Signature}, \mathit{States}, \mathit{Init}, \mathit{Transition})$, where:

- *Signature* is the actions signature of the automaton. It is a non-empty set of internal, input, and output actions;
- *States* is a non-empty set of states;
- $\mathit{Init} \subseteq \mathit{States}$ is the set of initial states;
- $\mathit{Transition} \subseteq \mathit{States} \times \mathit{Signature} \times \mathit{States}$ is the transition relation.

We assume that participants execute their automata in a *sequential* way, *i.e.*, they execute one step of instruction (computational or communication steps) at a time. This does not prevent them from executing several threads of instructions with appropriate multiplexing.

Let $n > 0$ be a positive integer, and let $\mathcal{A}_i = (\mathit{Signature}_i, \mathit{States}_i, \mathit{Init}_i, \mathit{Transition}_i)$ be some input/output automata, $\forall i \in \{1, \ldots, n\}$. We say that the *composition* of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, is the automaton $\mathcal{A} = (\mathit{Signature}, \mathit{States}, \mathit{Init}, \mathit{Transition})$ if and only if:

- $\mathit{Signature} = \bigsqcup_{i=1}^{n} \mathit{Signature}_i$, where $\sqcup$ is the disjoint union;
- $\mathit{States} = \prod_{i=1}^{n} \mathit{States}_i$;
- $\mathit{Init} = \prod_{i=1}^{n} \mathit{Init}_i$;
- If $((s_1, \ldots, s_n), a, (d_1, \ldots, d_n)) \in \mathit{Transition}$, then $\exists i \in \{1, \ldots, n\} : a \in \mathit{Signature}_i$ and $(s_i, a, d_i) \in \mathit{Transition}_i$, and $\forall j \neq i, d_j = s_j$.

When the automaton of a participant runs, it generates *executions*, where an execution is a sequence of alternating states and actions. The execution of the composition of automata of all participants in the system is called the *global execution*, or just the *execution* of the system.

We consider, in Chapter 4 that participants are *asynchronous*, and in Chapters 5 & 6 that they are *synchronous*. The notion of participant synchrony refers to the speed of execution of the participants [62]. The participants are synchronous if they all take the same amount of time to execute an instruction. The participants are asynchronous if they are not synchronous; in more details, one or many participants can remain at the same instruction for an arbitrary long but finite amount of time, while the others continue their execution. In asynchronous systems, the speed of execution of one participant can differ from the speed of execution of another participant; the relative speeds of participants are not common knowledge and are unbounded.

We assume that there is a *fictional global clock* that takes values in $\mathbb{T}$, *i.e.*, where $\mathbb{T} = \mathbb{N}$, the system evolves in discrete time. When we refer to a *point in time* or to a *date*, it is with respect to the global clock. To represent the possible differences in participants' speed, we assume that each participant has a local clock. We note that participants do not have access to the global clock. When the participants are synchronous, their local clocks are synchronized among each other. When the participants are asynchronous on the other hand, their local clocks are not necessarily synchronized among the participants.

2.2.1.2 Merit of Participant

In blockchain systems, for a given time, some participants may have special roles. To have such role, some blockchain systems consider the computing power of participants, and other blockchain systems use other metrics. To generalise that idea, we introduced in this thesis the notion of *merit of a participant*.

When a participant is part of a blockchain system, it has a certain *merit* represented by a real number between 0 and 1. The merit of a participant is an abstraction of its effort in constructing and maintaining the blockchain. The merit of a participant is represented by the proportion of its effort over all efforts in the system. The merit of a participant may vary over time. At any time, the sum of the merit of all participants is equal to 1. $\forall t \in \mathbb{T}, \forall i \in \Pi$, and let $\mu(i,t)$ be the merit of participant i at time t, we have $\mu(i,t) \in [0,1]$ and $\sum_{i \in \Pi} \mu(i,t) = 1$.

2.2.2 Communication Network

In order to communicate, participants are linked to each other over a network. A network is a couple (Π, E) where Π is the set of participants in the system, and $E \subseteq \Pi \times \Pi$ is the set of bidirectional communication channels (edges) between participants, *i.e.*, $\forall i, j \in \Pi$, if $(i,j) \in E$, then $(j,i) \in E$. Let i, j be two participants, if $(i,j) \in E$, there exists a communication channel between them; we say that i is a neighbour of j and vice-versa. If $(i,j) \notin E$, there is no direct communication channel between i and j, they are not neighbours of each other.

We say that there is a *path* between two participants i and j if i and j are neighbours, or if there exists a finite sequence of participants $\pi_1, \pi_2, \ldots, \pi_n$, where n is the length of the path, such that the following conditions hold:

- $\pi_1 = i$, called the source;
- $\pi_n = j$, called the recipient; and
- $\forall k \in \{1, \ldots, n-1\}, (\pi_k, \pi_{k+1}) \in E$.

When between any two participants there exists a path, we say that the network is *connected*.

We say that two paths are *(node) disjoint* if they do not share any participant in common except the source and the recipient. A network is k-*connected* if and only if it contains at least k disjoint paths between any two participants.

We assume that the networks considered in this manuscript are always connected. We also consider that the participants do not know E: they do not know the exact topology of the network.

Participants can send messages to their neighbours through this network. We assume that the network does not create nor drop messages. A message sent cannot get lost.

The networks presented are called *static*. A more general class of network exists where the communication channel between participants can vary over time (being present or not). We say that the network is dynamic. Dynamic networks can be modelled using *Time Varying Graph*s as defined by Casteigts *et al.* [44]. A Time Varying Graph is a tuple (Π, E, ρ, ζ), where:

- (Π, E) is the underlying static graph. If $\exists i, j \in \Pi$ such that $(i, j) \in E$, it means that there is a point in time when i and j are neighbours.
- $\rho : E \times \mathbb{T}^+ \to \{0, 1\}$, is the *presence* function, such that $\rho(e, t) = 1$ if and only if the communication channel is active at time t.

  ρ indicates whether the channel given in parameters is active at the given time.
- $\zeta : E \times \mathbb{T} \to \mathbb{T}$, is the *latency* function. It indicates the time it takes to cross a given communication channel at a given time.

  We note that the latency of a communication channel can vary over time.

The above notion of path cannot be applied in Time Varying Graph. The notion of time should be taken into account. We say that $\pi_1, \ldots, \pi_n$ is a *dynamic path* between participant i and j if $\exists t_1, \ldots, t_n \in \mathbb{T}$ such that:

- $\pi_1 = i$, $\pi_n = j$, and $\forall k \in \{1, \ldots, n-1\}, (\pi_k, \pi_{k+1}) \in E$;
- $\forall k \in \{1, \ldots, n-1\}, \forall t \in [t_k, t_k + \zeta((\pi_k, \pi_{k+1}), t_k)], \rho((\pi_k, \pi_{k+1}), t) = 1$; for all $k \leq n-1$ at date t_k, there exists for a sufficiently long enough time a channel for π_k to send a message to π_{k+1};
- $\forall k \in \{1, \ldots, n-1\}, \zeta((\pi_k, \pi_{k+1}), t_k) \leq t_{k+1} - t_k$, for all $k \leq n-1$, messages sent by π_k to π_{k+1} are received before t_{k+1}.

A static network (Π, E) is a special class of a Time Varying Graphs (Π, E, ρ, ζ) where $\forall t \in \mathbb{T}, \forall e \in E, \rho(e, t) = 1$, the communication channels if they exist are always active. The latency or message transmission delay is discussed in Section 2.3.2.

2.2.3 Dynamicity of the System

The number of participants in a blockchain system can be known or not, and finite or not bounded. Blockchain systems can be classified into different classes, depending on who can access the system, and who can participate in maintaining it. In particular, blockchain systems are often open, *i.e.*, new participants can enter in the system at any point in time.

A dynamic distributed system is *a continuously running system in which an arbitrarily large number of participants are part of the system during each interval of time and, at any time, any participant can directly interact with only an arbitrary small part of the system* [25]. Dynamic distributed systems can be categorised by the number of participants that can arrive (enter)

in the system during its execution. That number is referred to as the *arrival model* [6] of the system. Participants can always leave the system. The arrival models are classified into three different categories:

- n-*arrival model*, where n is a positive integer: the system is composed of exactly n participants, and n is known by the system. Therefore, n can be used by the protocols.
- *Finite arrival model*: there is an infinite number of participants, but in each execution of the system, only a finite number of participants can arrive (enter in the system). The number of arrivals is however not known.
- *Infinite arrival model*: there is an infinite number of participants, and at each time in the execution of the system, new participants can always arrive in the system.

2.2.4 Failure Models

A distributed *protocol* or a sequence of instructions can be viewed as an assignment of automata to every participant in the system. When a participant executes its assigned automaton, we say that the participant follows the protocol; otherwise, we say that it deviates from the protocol. In this manuscript and in particular, in Chapter 4, we consider that participants can fail [140] at executing a given protocol. A participant can fail in different ways: for example, it can *crash*: from a point in time, it is not executing the protocol any more; it can fail by *omission*: it is not able to send or to receive messages any more; it can fail by sending contradictory messages, *etc.* More generally, a participant can fail in any arbitrary way. A participant that can fail in the system is called a *faulty* or a *Byzantine* participant [136]; Byzantine participants can control the network by modifying the order in which messages are received, but they cannot postpone forever message receptions. Moreover, Byzantine participants can collude to "pollute" the computation (*e.g.*, by sending messages with different contents, while they should send messages with the same content if they were non-faulty). Participants that always follow the protocol are called *correct*. In particular, a correct participant is always present in the system since its beginning, and for the whole execution.

Since Byzantine participants can behave arbitrarily from a prescribed protocol, we do not make any assumption on their behaviour in our analyses. Byzantine faults are the worst faults that can occur and the most general, hence an interesting class of study. In particular, when a protocol is tolerant to Byzantine participants, we have the assurance that no fault can break it.

Sometimes, as in Section 4.3.2 of Chapter 4, we are interested in participants that are correct during a special fragment of their execution, and not necessarily during the whole execution of the system. Let χ be a fragment of execution of a participant i. We say that i is χ-*correct* if at the beginning of the fragment χ the internal state of the participant is *valid* and during χ it follows the protocol. We say that a state q of a participant i is *valid* at a time t if there exists a correct participant i' such that replacing i by i' in the system, and all other participants behaving as before, i' is in state q at time t. If a participant is not χ-correct, we say that it is non χ-correct. Note that a participant is correct if $\forall\chi$, it is χ-correct.

2.2.5 Cryptographic Assumptions

In blockchain systems, cryptography is used at various places. Cryptography can be seen as *the scientific study of techniques for securing digital information, transactions, distributed computations* [100]. It is used, for instance, to ensure the integrity of the blockchain, the

authenticity and security of the transactions, to preserve the anonymity of the human behind each participant, to ensure secure communication between participants, *etc.*

Asymmetric Cryptography. In blockchain systems, the paradigm of asymmetric cryptography is often used; intuitively, each participant has a pair of keys:

- A *private key*: which is a string of alphanumerical characters that is known only by the participant.
- A *public key*: which is another string of alphanumerical characters, *different* from the private key of the participant. The public key of a participant may be shared and may be known by all other participants.

The pair private/public key has to satisfy some mathematical properties depending on the cryptosystem[1] used. All participants use the same cryptosystem. To compute the pair of keys, one may use a *one-way* function. A one-way function $f : X \to Y$ is a function such that: (i) $f(x)$ is "easy" to compute (*i.e.*, possible in polynomial time), and (ii) inverting f, *i.e.*, knowing $y = f(x)$, find x is "difficult" to compute (*i.e.*, is not possible in polynomial time). It is an open question to prove whether such functions exist or not, in fact, their existence will be a proof of $P \neq NP$. However, there exist some candidates for such functions, *e.g.*, prime factoring, discrete logarithm, *etc.* There is, in the state of the art, no proof that these candidate functions cannot be inverted in polynomial time on a classical computer.

Using these keys (private/public), participants can have unique digital identities and can sign messages. The *identity* of a participant or its *address* depends only on its public key. In this manuscript, we consider the address as being exactly the public key, and we assume that each participant has a unique public key, *i.e.*, if two participants i and j have the same public key, then i and j are the same participant. We also consider in this manuscript that a participant can have only one pair of public/private keys.

When a participant sends a message, it sends it with a *digital signature.* A digital signature is a mathematical scheme to ensure the authenticity of a message and its sender. A participant i signs a message by using its private key and the content of the message it will send. When a participant j receives the message and the associated signature, j can verify that the message was created with the private key of i. Formally, a digital signature scheme is a triple (K, S, V), where:

- K, is the key generator function, the cryptosystem used;
- S, the signing function, takes as input a private key sk and a message m, and $S(sk, m)$ is a string, the signature s of m using the private key sk;
- V, the verifying function, takes as input a public key pk, a message m and a string s and returns `true` if s is a signature of m using the associated private key of pk, and returns `false` otherwise.

We assume that signatures *cannot be forged*, *i.e.*, if participant i is different from participant j, then j cannot impersonate i by using i's digital signature. This assumption is equivalent to saying that no subset of participants has enough power to find the private key of any other participant. That assumption implies that if participant j receives a message signed by participant i, it knows that i created and sent the message. i cannot deny the operations; no one can alter the message between sending and receiving it, because signatures are unforgeable.

[1] A cryptosystem describes how to compute the keys, and how to perform the cryptographic operations.

Hash Functions. Other cryptographic functions are heavily used in blockchain systems: the *cryptographic hash functions*, or simply the *hash functions* [141]. Let n be an integer, a function $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^n$ is a hash function if it satisfies the following properties:

- *Pre-image Resistance*: hash is a one-way function;
- *Second Pre-image Resistance*: given x_1, it should be "difficult" to find a $x_2 \neq x_1$ such that $\mathsf{hash}(x_1) = \mathsf{hash}(x_2)$;
- *Collision Resistance*: it should be "difficult" to find two values x_1 and x_2 such that they have the same image, *i.e.*, $\mathsf{hash}(x_1) = \mathsf{hash}(x_2)$.

Let $x \in \{0,1\}^*$ be a message, if $h = \mathsf{hash}(x)$, we say that h is the *hash* of x. Intuitively, a hash function is a function: (i) which maps data of arbitrary length to a fixed-sized length data, its *hash*, and (ii) where changing even a little bit the input gives a completely different output. Hash functions give some confidence that two data that have the same hash are actually the same.

2.3 Communication

2.3.1 Communication Primitives

One of the needs in blockchain systems is to ensure that all participants have the same information and messages. To this end, we consider a broadcast primitive. A broadcast is a technique to send a message to all other participants. If a participant broadcasts a message m, it wants all other participants to eventually deliver m.

It is often sufficient to have a *Byzantine reliable broadcast* [38]. We say that a broadcast is reliable if the following two conditions hold: (i) safety: every message delivered by a correct participant has been previously sent by a source, and (ii) liveness: every correct participant eventually delivers every message sent by a correct source, and (iii) all correct participants eventually share the same set of messages.

Formally, we say that a broadcast protocol is Byzantine reliable if it satisfies the following properties, assuming that at least one participant broadcasts a message.

- brb-Validity. If a correct participant delivers a message m from a correct participant i, then i broadcasted the message m;
- brb-Integrity. No correct participant delivers twice the same message from a participant;
- brb-Termination-1. If a correct participant broadcasts a message m, all correct participants eventually deliver m;
- brb-Termination-2. If a correct participant delivers a message m from a participant, then all correct participants eventually deliver m.

This means that all other correct participants eventually deliver all messages sent or delivered by at least one correct participant. It does not necessarily imply that the order in which messages are sent is the same order of messages delivered.

To disseminate a message to all participants, a *dissemination protocol* can be used. In such a protocol, if one participant receives a message for the first time, it sends the message to all its neighbours. Intuitively, if all participants are correct, eventually all participants will deliver the message. In contrast, when there are some Byzantine participants, such approach does not guarantee the property of reliable broadcast as shown by the following example.

Example 2.1. *Let the following dissemination protocol for broadcast: if one participant receives a message for the first time, it sends the message to all neighbours.* (Π, E) *is the network such that* $\Pi = \{i, j, k\}$ *and* $E = \{(i, j), (j, k)\}$*. Let* i *and* k *be correct, and let* j *be Byzantine.* i *and* j *are neighbours, and so are* j *and* k*, but* i *and* k *are not.*

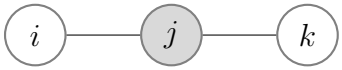


Figure 2.1: Communication Network of Example 2.1.

Participant i *wants to broadcast the message* m*, and by following the protocol, it sends it to* j*.* j *when receiving* m *deviates and does nothing. Here,* k *does not receive the message* m*.*

However, with some hypotheses on the communication network, it is possible to achieve a reliable broadcast in the presence of Byzantine participants as shown by the following remark.

Remark 2.2. *Let* n *be the total number of participants and* f *be the maximum number of Byzantine participants. In a static network with Byzantine participants, [61] shows that when the network is* $(2f + 1)$*-connected, it is possible to guarantee that all messages sent by correct sources are delivered by correct participants. Similarly, [124] shows that there exists a class of Time Varying Graphs where it is possible to guarantee that all messages sent by a correct source are eventually delivered by correct participants; the result of [124] also depends on the maximum number of Byzantine participants in the system. Both propositions however do not satisfy the property brb-Termination-2. If* $n = 3f + 1$*, [37, 140] show that there exist broadcast protocols where all participants share the same set of messages, satisfying brb-Termination-2. Hence, by combining the results [61, 124] and [37, 140], Byzantine reliable broadcast is possible.*

Intuitively, the results of [37, 61, 124, 140] require that the network is "sufficiently connected", such that there always exist enough (dynamic) paths with no Byzantine participants between any two correct participants. In the same spirit, [107] proposes a protocol for dynamic Byzantine reliable broadcast.

In this manuscript, we assume the existence of a reliable broadcast that is used by participants, and we do not discuss the network nor the broadcast protocols. However, as explained in Remark 2.2, the existence of such a primitive implies some restrictions on the connectivity of the communication network. To broadcast a message m, a participant uses the primitive broadcast(m) during one of its send phase m. During a delivery phase, a participant receives a broadcast of a message by executing the primitive delivery$()$.

2.3.2 Time Assumptions

When a participant sends a message, other participants do not deliver the message instantaneously, there is a *message (transmission) delay* or a *transmission latency*. Let $\Delta \geq 0$ be the maximum message delay between any two participants, *i.e.*, $\forall i, j \in \Pi$, if i sends a message to j at date (time) t, then at date $t + \Delta$, j has delivered the message. The communication can be classified into different categories based on the finiteness of Δ and whether the participants know its value or not.

- **Synchronous Communication**: In synchronous communication systems, Δ is finite and known by the participants. If a participant broadcasts a message at a time t, then at time $t + \Delta$, all participants have delivered the message.

Since the maximum delay Δ is known, participants can use that information in their execution; typically, they can match the duration of their delivery phase with Δ to ensure the reception of all messages sent at the beginning of the round.

- **Asynchronous Communication**: In asynchronous communication systems, Δ is infinite. That means that there is no upper bound on the message transmission delay. Messages sent are eventually delivered, since messages sent cannot be lost; but no one can predict an upper bound on when a message will be delivered.

- **Semi-Synchronous Communication**: Between synchronous and asynchronous communications, there exists a class of communication systems called semi-synchronous. In semi-synchronous systems, Δ is finite. We can further divide the class as follows:

  - *Partial Synchronous* Communication. As defined by [68], a communication system is partially synchronous if Δ is finite but unknown by the participants.

    Since Δ is not known, participants cannot use it directly, they may however try to guess it during their execution.

  - *Eventual Synchronous* Communication. A communication system is eventually synchronous if there exists an unknown date $t \in \mathbb{T}$, called the *Global Stabilisation Time* (GST) such that all messages sent after date t are received within the bound Δ. There is no guarantee on message transmission delay for the messages sent before GST; however, the message will be eventually delivered since messages cannot be lost. The period before GST is called the *asynchronous period*, and the time after GST is called the *synchronous period*. It is important to note that since participants do not know when GST will be, they can only try to guess if GST is already reached during their execution. That is the case whether Δ the message transmission delay (hereafter GST) is known or not.

    Most works in the literature assume that Δ is known, and only the GST is unknown. However, to the best of our knowledge, it is not known if that model is equivalent to having both Δ and GST unknown. If Δ is known, it seems that it can be used with a certain multiplicative factor to design the duration of the delivery phases of participants, while when Δ is unknown, it is not possible.

    In this work, to be the most generic as possible, when using an eventual synchronous system, we assume that both GST and Δ are finite but unknown.

An important remark is that the synchrony assumptions of the communication hold no matter the size (finite) of the messages, *i.e.*, no matter how long (but finite) a message is, it is delivered entirely with respect to the communication's time assumptions.

Remark 2.3. *[25] and [127] discussed and proved that in infinite arrival models, only asynchronous communication is possible, whereas in finite arrival models it is possible to achieve a semi-synchronous communication, and synchronous communication can only be achieved in n-arrival models.*

Intuitively, in finite arrival models, there is a time from when no new participants enter the system, hence the maximum number of communication channels between any two participants can be bounded, and so can be the message transmission delay.

2.4 Game Theory

One can defined *Game theory* as a mathematical formalisation of conflicts and cooperation between rational decision-makers in a given environment. Game theory has various domains of application as of computer science, economics, evolutionary biology, mathematics, political science, *etc.* It is mainly used to model and study rational behaviours between different participants. Using game theory appears to be useful according to our goal of studying rational behaviours in blockchain systems.

Formally, a game is a tuple $\mathcal{G} = (N, (A_i)_{i\in N}, (\theta_i)_{i\in N}, (g_i)_{i\in N})$, where:

- N is a non-empty set of participants;
- A_i is the set of actions of participant i;
- θ_i is the type of participant i;
- g_i is the gain function of participant i;

Static Games. At a beginning of a static game, all participants choose simultaneously the actions they will play, without knowing which actions the others will choose. The gain of participant i is of the form $g_i : A_1 \times \cdots \times A_n \to \mathbb{R}$. Static games are also called *simultaneous* game. The most popular simultaneous game is perhaps *rock-paper-scissors*.

Dynamic or Repeated Games. A repeated game is a sequence of static games where participants may have some information thanks to the previous static games played. We say that the first static game is the game at stage 1, the second is the game at stage 2, and so on. To accommodate the information each participant has, we denote by h_i^t the information set of participant i at time t. At each time of the game, each participant has an information set that contains information of what happened previously. Let $\mathcal{H}_i^t$ be the set of information sets participant i can have at time t.

A *pure strategy* or simply a *strategy* of a participant i at time t is a function $\sigma_i^t : \mathcal{H}_t \to A_i$. A strategy of the participant is a family $(\sigma_i^t)_{t\geq 1}$. However, in the rest of the manuscript, we will write σ_i to represent the strategy of participant i, and we will denote by $\mathcal{S}_i$ the set of strategies of participant i. A vector of strategies for each participant $\sigma = (\sigma_1, \ldots, \sigma_n)$ is called a *strategy profile*. In repeated games, the gain of participant i is of the form $g_i : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathbb{R}$.

2.4.1 Type of Participants

In a blockchain system, all participants do not necessarily have the same behaviour. Depending on their executions, we can classify participants into different types.

During the global execution of the system, participants do take *actions*, where an action can be doing an instruction or not. For example, sending a message is an action; not sending a message is also an action; doing another instruction instead of the prescribed one is also an action, *etc.* Actions may have costs, and different actions can have different costs; note that the cost of actions can be perceived differently from one type of participants to another.

We say that a participant *follows a sequence of instructions* (a protocol) if the actions the participant takes are in line with that sequence of instructions. If the participant does not follow the protocol, we say that it *deviates from* it. We can view a protocol as a sequence of prescribed actions.

2.4.1.1 Classical Participants

The first two types of participants, the only ones used in Chapter 4 are the classical types we can find in the distributed systems' literature, as described in Section 2.2.4: the *correct* and the *Byzantine* participants. Both correct and Byzantine participants do not care about the costs of their actions, nor the global execution of the system. In this manuscript, and for the sake of clarity and consistency, we use the terminology *obedient* instead of correct. The term *honest* can also be found in the blockchain literature to designate obedient participants.

Definition 2.4 (Obedient Participant)**.** *A participant is obedient with respect to a protocol $\mathcal{A}$ if it always follows protocol $\mathcal{A}$.*

When the protocol is clear from the context, we just say that the participant is obedient. In particular, the above definition of an obedient participant encapsulates the fact that the participant is always in the system and always follows the protocol.

Definition 2.5 (Byzantine Participant)**.** *A participant is* Byzantine *with respect to a protocol $\mathcal{A}$ if it can arbitrarily deviate from protocol $\mathcal{A}$.*

When the protocol is clear from the context, we just say that the participant is Byzantine. A Byzantine participant represents all kinds of (unfortunate) situations that can happen, *e.g.*, a failure, or doing others actions than the prescribed. Any behaviour can be seen as Byzantine.

2.4.1.2 Rational Participants

A third class of participants is considered in Chapters 5 & 6: the *rational* participants. Rational participants may care about the cost of their actions, and they have preferences over all the different global executions of the system. In more details, a rational participant assigns a value to each possible execution of the system; between any two executions of the system, a rational participant prefers the one having a higher value. The *gain* of a rational participant is a function of (i) the global execution of the system, and of (ii) the actions the participant takes. The gain has values in the set of real numbers, $\mathbb{R}$. We recall that the actions of one participant may have an effect on which global execution the system will have.

Definition 2.6 (Rational Participant)**.** *A participant is* rational *if it is self-interested. It does an action instead of another action if and only if doing so increases its expected gain.*

A rational participant has preferences and takes actions to have its most preferred execution of the system, taking into account what the other participants do. It does not necessarily want to hurt the system; it wants to benefit from it.

We now introduce two refinements of the rational participants: the *strategic*, and the *malicious* participants. Having a property on the system's executions, one can know whether a given execution satisfies that property. In particular, a property defines a partition on the executions of the system; in one hand, the executions that satisfy the property and on the other hand, the executions that do not satisfy the property.

Definition 2.7 (Strategic Participant)**.** *A participant is* strategic *with respect to a property $\mathcal{P}$ if it is rational, and it prefers executions that satisfy $\mathcal{P}$, i.e., its assigns a positive value to executions where $\mathcal{P}$ is satisfied, and a negative value to executions where $\mathcal{P}$ is not satisfied.*

A strategic participant prefers the executions where the system achieves its properties, but think about its own interest in the first place.

Definition 2.8 (Malicious Participant)**.** *A participant is* malicious *with respect to a property $\mathcal{P}$ if it is rational, and it prefers executions that do not satisfy $\mathcal{P}$, i.e., its assigns a positive value to executions where $\mathcal{P}$ is not satisfied.*

A malicious participant's objective is to hurt the system. A malicious participant takes all actions it can to increase the chance of having a global execution that does not satisfy the properties of the system.

The *Byzantine-Altruistic-Rational* (BAR) Model Aiyer *et al.* introduced the Byzantine-Altruistic-Rational model or BAR model in [7]. In the BAR model, given a protocol, there are three types of participants:

- *Byzantine*: Participants that can arbitrarily deviate from the protocol;
- *Altruistic*: Participants that always follow the protocol;
- *Rational*: Rational participants will deviate from the protocol if and only if doing so increases their net utility from participating in the system; in particular, rational participants receive a long-term benefit from participating in the protocol.

[7] makes the following assumption "rational participants are conservative when computing the impact of Byzantine participants on their gain", meaning, rational participants assume that the maximum number of Byzantine is present in the system and their goal is to minimise rationals' gain.

For completeness and correctness, we remove that assumption. Specifically, we introduce the type of *malicious participants* that represents the above assumption, and since they are rational, we consider them as such. The participants called rational by [7] are now called strategic. Our definition of rational allows removing the assumption that rational participants receive a long-term benefit from participating in the protocol; being rational have the classical definition of having preferences, and taking actions to have a better outcome, as usually defined in economics, philosophy, *etc.* [157]. Another difference is that we do not necessarily need not to assume the exact number of malicious participants.

2.4.2 A Comparison of Possible Executions from Participants' Type

Let $\mathcal{A}$ be a protocol. We can see from Definition 2.6 that a rational participant can be viewed as a special subclass of Byzantine participants, rational participants can deviate from the protocol, but only if that deviation is beneficial to them.

Let $\mathcal{C}$ be a class of rational participants such that their objective is to follow the protocol $\mathcal{A}$ no matter what is the global execution of the system and what the other participants do. The class $\mathcal{C}$ consists of exactly all the obedient participants with respect to $\mathcal{A}$, and only them. We can consider that some rational participants can behave as obedient, if their gain function is defined in that case. That exhibits that the behaviour of rational participants depends really on their gains. Therefore, the obedient participants are a subclass of rational participants where their objective is to follow the protocol.

Usually, a protocol is designed to solve a problem, or equivalently to satisfy a property. It can then happen for strategic participants to behave as obedient. However, if running the protocol is (too) costly, strategic participants may deviate to improve their gain. The class of obedient participants is not a subclass of strategic participants, but both can intersect.

As mentioned earlier, malicious and strategic participants are rational participants, but they have opposing objectives, so they do not intersect. A summary of the comparison of possible execution can be viewed in Figure 2.2.

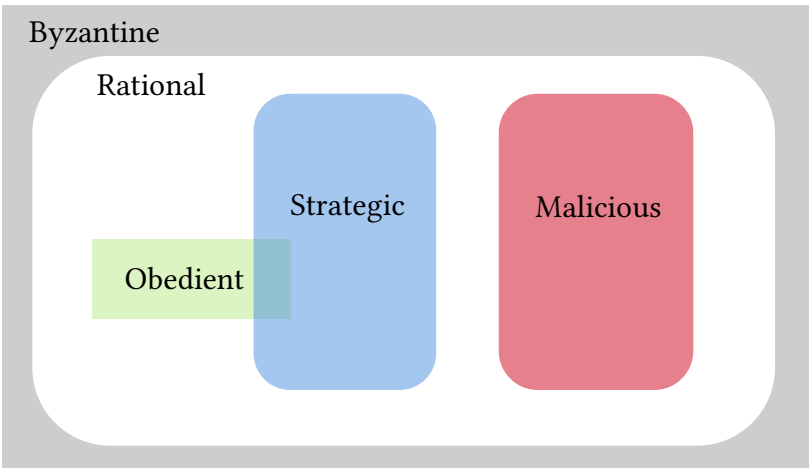


Figure 2.2: A Comparison of Participant's Types

2.4.3 Completeness of Information in Games

2.4.3.1 Complete Information Games

In complete information games, all participants know the other participants' information, their types, their gain function, their actions set, *etc.* At each time in the game, all participants know what actions the other participants did previously.

2.4.3.2 Incomplete Information Games or Bayesian Games

When a game is not with complete information, we say that it is an *incomplete information* game. In incomplete information games, at least one participant is not sure about the type of another participant. We can say that participants have private information that other participants do not know about. For example, it can be known that there is a malicious participant in the system, but except the malicious, no one else knows which participant it is.

An initial distribution of the type of participants is set at the beginning of the game, but participants do not know it exactly. Participants only have at each time a probability distribution over the other participants' type; the probability distribution is updated following *Bayes' rule* [27] and is kept in the information set of each participant. That is why incomplete information games are also called *Bayesian games*.

Let A and B be two events such that $\mathbb{P}[B] > 0$. *Bayes' theorem* also called *Bayes' rule* is the following equality:

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[B|A] \cdot \mathbb{P}[A]}{\mathbb{P}[B]}$$

The law of total probability states that $\mathbb{P}[B] = \mathbb{P}[B|A] \cdot \mathbb{P}[A] + \mathbb{P}[B|\neg A] \cdot \mathbb{P}[\neg A]$, we can then rewrite Bayes' theorem as:

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[B|A] \cdot \mathbb{P}[A]}{\mathbb{P}[B|A] \cdot \mathbb{P}[A] + \mathbb{P}[B|\neg A] \cdot \mathbb{P}[\neg A]}$$

Let us consider the following example to see how one updates its knowledge according to Bayes' rule.

Example 2.9 (An Application of Bayes' Rule). *Assume in a population of* $1,000,000$ *people,* 0.01% *(*100 *persons) are of type* I*, and the rest (so* $9,900$ *persons) are of type* H[2]*. Assume also*

[2] I refers to Infected and H to healthy

		Test says *yes*	Test says *no*
$\mathbb{P}[I] = 0.01\%$	Being type I	$\mathbb{P}[Yes\|I] = 100\%$	$\mathbb{P}[\neg Yes\|I] = 0\%$
$\mathbb{P}[\neg I] = 99.99\%$	Being type H	$\mathbb{P}[Yes\|\neg I] = 0.1\%$	$\mathbb{P}[\neg Yes\|\neg I] = 99.9\%$

Table 2.1: Summary of the Probabilities of Example 2.9

that people are evenly distributed across the population: by picking one person at random, its probability of being of type I is 1.10^{-4}.

Imagine there is a test for examining if people are of type I or not, but unfortunately, the test is not 100% accurate (specificity). In details:

- *If the person is of type I, the test says* yes *with probability 1;*
- *If the person is of type H, the test says* no *with a probability of 0.999. There is a probability of 0.001 of being a false positive, the test says* yes *for a person of type H.*

The probabilities are summarised in Table 2.1. One person is picked at random in the population, and its test says yes*. By following Bayes' rule, the probability of this person being of type I should be updated from 10^{-4} (0.01%) to approximately 0.09 (9%).*

In summary, even if the test is positive a person picked at random has less than 10% of being of type I. That may seem counter-intuitive, but it follows Bayes' rule as shown in the following.

Let i be the person picked at random in the population. Let I be the event "i is of type I"; the event i is of type H is $\neg I$. We know that $\mathbb{P}[I] = 10^{-4}$. Let Yes *be the event "i's test says* yes*"; the event i's test says* no *is* $\neg$Yes*. What interest us is knowing what is the probability of i being of type I knowing its test says* Yes *($\mathbb{P}[I|\text{Yes}]$), which by Bayes' rule is:*

$$\begin{aligned}\mathbb{P}[I|\text{Yes}] &= \frac{\mathbb{P}[\text{Yes}|I] \cdot \mathbb{P}[I]}{\mathbb{P}[\text{Yes}|I] \cdot \mathbb{P}[I] + \mathbb{P}[\text{Yes}|\neg I] \cdot \mathbb{P}[\neg I]} \\ &= \frac{10^{-4}}{10^{-4} + 10^{-3} \cdot 0.9999} \\ \mathbb{P}[I|\text{Yes}] &\approx 0.09\end{aligned}$$

An intuition is that, since the population is big, with the accuracy (specificity) of the test, there will be on average a lot more false-positive (people of type H but test says yes*) than real positive (type I).*

If the probability of false positive was way lower, for example, $\mathbb{P}[\text{Yes}|\neg I] = 0.01\%$, in case of a positive test, the knowledge of i being of type I should be updated to approximately 50%. To update the knowledge to over 99%, the false positive rate should be less than 0.0001%.

2.4.4 Solution Concept

Now that the game is defined, we can try to predict the behaviour of rational participants. To do so, different concepts exist, the main one being the concept of *Nash equilibrium* [129].

Intuitively, a strategy profile is a Nash equilibrium when each participant has a strategy that maximises its gain with respect to the other participants' strategies in the strategy profile. A Nash equilibrium can also be seen as a strategy profile where no participant can increase its gain by deviating alone from the strategy profile.

Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ be a strategy profile, and let $\sigma'_i \in \mathcal{S}_i$ be a strategy of participant i. We denote the strategy profile $(\sigma_1, \ldots, \sigma_{i-1}, \sigma'_i, \sigma_{i+1}, \ldots, \sigma_n)$ by (σ_{-i}, σ'_i). (σ_{-i}, σ'_i) is the strategy profile where participant i deviates by doing σ'_i instead of σ_i, and all other participants continue with the same strategies. We can now formally define a Nash equilibrium.

Definition 2.10 (Nash Equilibrium). *A strategy profile σ is a Nash equilibrium if and only if*

$$\forall i \in N, g_i(\sigma) \geq g_i((\sigma_{-i}, \sigma'_i)).$$

The definition of a Nash equilibrium can be seen as too restrictive in some games, and the concept of *approximate Nash equilibrium* is used. An *ε-approximate Nash equilibrium* is a strategy profile where if one participant deviates, it can gain at most ε more than its gain at equilibrium. Formally, a strategy profile σ is an ε-approximate Nash equilibrium if and only if $\forall i \in N, g_i(\sigma) \geq g_i((\sigma_{-i}, \sigma'_i)) - \varepsilon$. Approximate Nash equilibria are not stable situations, since a participant can prefer to deviate, even if its gain is just ε, so we do not consider them in this manuscript.

On another hand, and especially in repeated games, the concept of Nash equilibrium allows too many behaviours, and some are not interesting or coherent. The concept of *subgame perfect equilibrium* [133] is defined to specifically capture equilibria that are coherent throughout the whole game.

At any history, the "remaining game" can be regarded as a game on its own. We call such a remaining game a *subgame* of the game. Note that the whole game is also its own subgame.

Definition 2.11 (Subgame Perfect Nash Equilibrium). *A strategy profile σ is a subgame perfect Nash equilibrium if in all subgames, σ restricted to the subgame is a Nash equilibrium.*

Bayesian Equilibria. In Bayesian games, the concept of Nash equilibrium is also not well suited. In Bayesian game, since at least one participant is unsure of the type of other participants, a strategy profile may give more than one execution, according to the type distributions. An analogous concept to Nash equilibrium, the *Bayesian equilibrium* [154] is defined for Bayesian games. Contrarily to Nash equilibrium, in a Bayesian equilibrium, each participant's goal is to maximise its expected gain, given its knowledge about the types' distribution; in particular, each participant's beliefs are consistent with Bayes' law when computing probabilities conditional on events that have positive probability on the equilibrium path.

In Bayesian games, the gain function g is a probability distribution over the gain of all different executions that correspond to the given strategy profile.

Definition 2.12 (Bayesian Equilibrium). *A strategy profile σ is a Bayesian equilibrium at time t if and only if*

$$\forall i \in N, \forall i \in N, \mathbb{E}[g_i(\sigma)|h_i^t] \geq \mathbb{E}[g_i(\sigma_{-i}, \sigma'_i)|h_i^t].$$

As for Nash equilibria, the concept of subgame perfection exists in Bayesian games, and such equilibrium is called a *perfect Bayesian equilibrium* [79].

Definition 2.13 (Perfect Bayesian Equilibrium). *A strategy profile σ is called a subgame perfect Bayesian equilibrium if in all subgames, σ restricted to the subgame is a Bayesian equilibrium.*

2.5 Conclusion

In this chapter, we broadly introduced the models and the different notions that we will use throughout this manuscript.

In the next chapter, we give a state of the art of the blockchain literature related to the construction of a blockchain and to the rational behaviours in blockchains, and we will highlight our contributions to that state of the art.

In Chapter 4, we discuss how to build blockchains in systems prone to failure and under different communication systems. That chapter leans on Sections 2.2 & 2.3.

In Chapters 5 & 6, we study rational behaviours and use notions defined in Section 2.4.

CHAPTER 3

STATE OF THE ART

Contents

A global trend in research is the fast-growing of scientific production in the past years. The literature in blockchains is no exception to this phenomenon; it seems to be even worse. The literature concerning blockchain systems is enormous, it concerns several fields, and is growing rapidly. Specifically, on blockchains, there are many preprints from academics that are not yet published; huge numbers of "white papers" from practitioners or non-academics that present their technologies whose novelties are often hard to capture; there are more and more peer-reviewed scientific papers, and conferences specifically dedicated to blockchains and distributed ledger technologies.

Blockchains span among various scientific topics. To name few of them, we have artificial intelligence, cryptography, data science, database systems, distributed systems, economics, finance, financial technology (fintech), formal language, identity management, law, multiagent systems, privacy, network, supply chain, traceability, *etc.* It is also considered in health science, and in voting to guarantee non-falsifiability. Please note that the above list is not exhaustive, and even new disciplines start to be interested in the topic.

During this thesis, and in this manuscript we focus only on a few of these topics; mainly, we use distributed systems, and game theory to study blockchains. More specifically, we study committee-based blockchains, and how blockchains are built (state of the art in Section 3.1); the fairness for the block creators in blockchains systems (state of the art in Section 3.2), and on selfish behaviours in blockchain systems (state of the art in Section 3.3).

3.1 Blockchains' Construction

One of the most studied axes of research on blockchains is without a doubt how blockchains are built if they are correct, and how they work exactly. The correctness of blockchain systems is well studied, and formal proofs start to be interested in the topic. For example, in [51], Chaudhary *et al.* formally analyse Bitcoin against double-spending attacks using formal methods, namely, model checking techniques; in [9], Alturki *et al.* proved the safety of the Algorand's blockchain (they did not prove the liveness) using a proof assistant; we can also cite Tezos [8] whose code has been written in a language known by formal verification specialists. The models used in their tools are much more simplistic than the real-world environment of these protocols. Blockchains are hence often "proved" correct using the classical "pen-and-papers" technique.

Before the presentation of some techniques to build a blockchain, let us recall a definition of blockchains. In a system where participants communicate only by exchanging messages with each other, a blockchain is basically a tamper-resistant distributed ledger built in an append-only manner. A blockchain is typically a chain of blocks, where a block is a collection of information, and where each block is linked to the prior one by its hash [141].

To build a blockchain, the ideal will be to select one participant at random which will have the task to add new information to the chain and then share it with the rest of the participants, then a new (maybe the same) participant will be selected, and so on. However, that is complicated since participants are not known in advance, the selected participants can be offline, it can add "invalid" information, *etc.* Note that participants that add blocks to the blockchains are rewarded as an incentive.

In this section, we present a glance at the state-of-the-art of techniques/algorithms used to build blockchains.

Since Bitcoin is the most known and is the origin of all the hype and of the studies of blockchain systems, we first describe how its blockchain is constructed.

3.1.1 Bitcoin & Proof-of-Work Blockchains

Bitcoin [128], the most known, and to the best of our knowledge, the first financial application based on blockchain proceeds by using a technique called *proof-of-work*, which was introduced by Dwork and Naor in 1992 [69]. Using this technique, Bitcoin's participants aim to reach some sort of agreement in an open and asynchronous system. In Bitcoin, some participants, called *miners* want to add blocks to the blockchains. Miners are block creators. To add a new block to the chain, each miner needs to prove that it worked to have the right to add a new block. Proving that a miner worked is represented by solving first among the miners a cryptographic puzzle. The best-known way to solve such a problem is by repeated trials, hence the more computing power a miner has, the higher are its chances to solve the puzzle first.

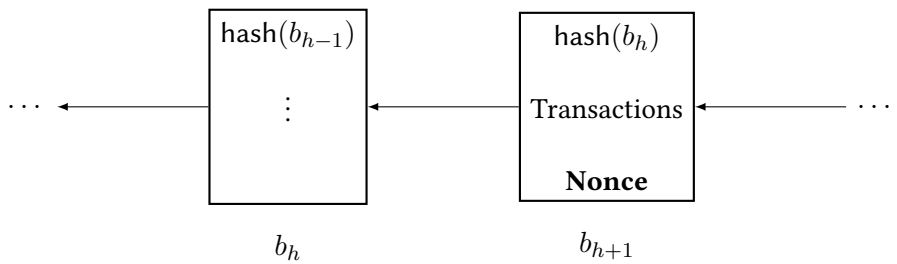


Figure 3.1: Content of a block in a proof-of-work blockchain

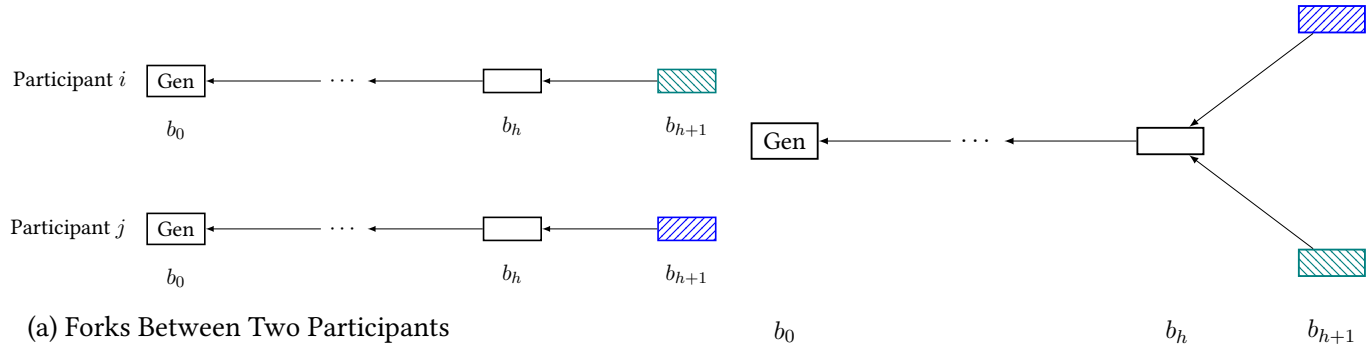
In proof-of-work, a participant needs to prove that it worked to have the right to add the next block. In more details, for a participant to be selected to add a block in the blockchain, it has to be the first to resolve a crypto-puzzle: the more computing power, the higher the chances are to win. When a miner wins, it can send its block to the other participants, along with the proof of its work. The other participants can then check if the proof is valid, then if the block is valid, they will add the block in their chain, and the scenario restarts for the next block.

More specifically, once a block, say b_h is added to the blockchain, a miner creates directly a new block and plans to add it to the blockchain. That new block contains besides of the classical information of a block (hash to the previous block b_h, list of transactions, *etc.*) a *nonce* (Figure 3.1). The nonce is a value the miner should add such that the hash of its proposed block b_{h+1} is lower than a given value, or equivalently, the hash of the new block should start with a given number of zeroes. We can write $b_{h+1} = \mathsf{hash}(b_h) \cdot \text{Transactions} \cdot \textit{Nonce}$, where the symbol $\cdot$ represents the concatenation. The goal for each participant is then to find a value *Nonce* such that $\mathsf{hash}(b_{h+1}) = \mathsf{hash}[\mathsf{hash}(b_h) \cdot \text{Transactions} \cdot \textit{Nonce}] \leq D$, where D is called the difficulty parameter. Note that the set of transactions a miner takes is not necessarily the same set of transactions another miner takes (order, number, *etc.*). Additionally, each miner adds its public key to its blocks; no two miners have the exact same blocks created. Therefore, each miner tries to find a nonce corresponding to its block. The first miner to find the appropriate nonce for its block is the "winner". Note that finding a nonce value such that the block satisfies a given property resembles the inversion of hash, which by definition of a hash function (Pre-image resistance property, Section 2.2.5) should be difficult. The best-known way to tackle such problem, and find an appropriate nonce is by repeated trials. Due to the "random characteristics" of hash functions, if a miner has more computing power than another one, the former has more chances to find its nonce first. That leads to miners paying more specialised hardware [148]. The race to be the fastest and having more powerful machines leads to a *high energy-consumption*; Bitcoin is not environment friendly, which is one of the first and main problems of proof-of-work.

Another issue of proof-of-work closely related to the computing power is the creation and existence of *pool*s of miners. Specifically, to increase their gains and rewards over time, miners gather in pools. All miners in a pool try to find the nonce for the exact same block, and so there are more chances that a pool adds a block compared to a single miner. The computation power of a pool is about the sum of the computing power of each of its members. Once a block is found by a pool, the reward is redistributed among the pool members according to specific and internal rules. The miners of Bitcoin tend to be more and more part of pools since pools ensure more frequent payments than the current extremely low probability of winning at the proof-of-work alone [135]. Pools of miners concentrate powers and make Bitcoin more centralised than what was envisioned by its founder(s). The presence of pools and the more and more centralised system is the second main critique of Bitcoin linked to the proof-of-work.

Note that more powerful machines lead to finding the appropriate nonce value faster. To approximately ensure that the interval between the addition of two consecutive blocks is constant (10min in Bitcoin), the value of the difficulty parameter D changes over time to take into account the changing computation power of all miners. The sum can be approximated by carefully monitoring how fast blocks are produced. In particular, if new blocks are found faster than expected, then the difficulty increases; and if new blocks are found at a speed slower than the expected time interval, the difficulty decreases. The difficulty management is even more subtle since it is designed such that with high probability, there is only one winner, *i.e.*, the probability of having two miners that find first and relatively at the same time a proper nonce for their block is close to 0. However, even if extremely low, such a scenario is possible. When

that happens, we say that there is a *fork* in the system. Participants do not necessarily know which block to add to their blockchain. Depending on how connected the participants are, some can receive the block of one miner first, others will receive the block of another first. An example of fork is depicted on Figure 3.2; participant i receives the green block first, and participant j receives the blue block first, as a global view, there is a fork for the block b_{h+1}, and how to solve the fork and select only one block should be clearly defined.



(a) Forks Between Two Participants

(b) Global View of a Fork

Figure 3.2: Example of a fork

Forks. The presence and management of forks are discussed in the original Bitcoin's paper [128]. The author, Satoshi Nakamoto, proposes to follow the *Longest Chain's Rule* (Figure 3.3). The longest chain's rule says the following. When there is a fork, each miner should take at random one block of the forks (*e.g.*, between the blue and the green block on Figure 3.2) and tries to create a new block on top of it, by computing a nonce, *etc.* The first block between the forked that is extended (added to the chain) becomes the new block on which all miners should work. Basically, the block that became the longest first, is the one that resolves the fork, and is the branch of the chain on top of which miners should continue to work. It was then added that if a block has 6 subsequent blocks, *i.e.*, there is a depth of at least 6 after a block, the former is considered *finalised* and we say that it is *on the main chain*; the other block(s) is called orphan. The reasons and origin of why 6 blocks are not clear and seems to be an arbitrary value[1]. However, knowing when a block is finalised is important for the end-user. For a financial application for instance, the finalisation of a block means the confirmation of payment, so the seller can send the good. If the good is sent but the block in which the transaction becomes orphan, the sender loses the object and is not paid. Forks are another big issue of Bitcoin's proof-of-work.

Scientific approaches. While the "first" and most popular blockchain Bitcoin came in 2008, the scientific community started to be interested around the mid-2010s.

It was only recently that distributed computing academics focus their attention on the theoretical aspects of blockchains, motivated mainly by the intriguing claim of popular blockchains that they implement consensus in an asynchronous dynamic and open system prone to failures and to the presence of adversaries. Such a claim must not be true, since it is refuted by the famous impossibility result in distributing computing [77]. In [77], Ficher *et al.* prove that it is impossible to achieve consensus in an asynchronous environment prone to failures (even with only 1 crash).

[1] What if there is a fork of depth more than 6 for instance? Such case already happened, and the fork was solved over an online forum; which contradict with the decentralised nature of the blockchain system.

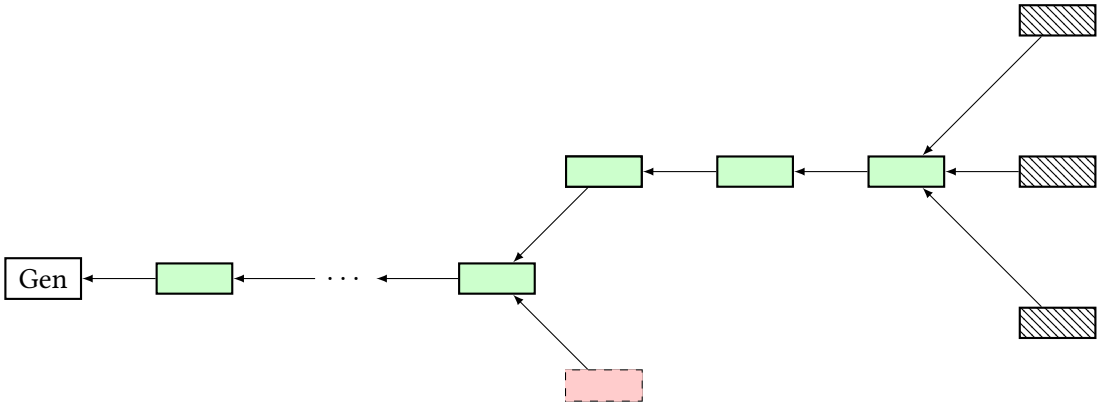


Figure 3.3: Longest Chain's Rule

The theoretical studies of Bitcoin and the *proof-of-work* protocols have been pioneered by Garay *et al.* in [83]. Garay *et al.* study the pseudo-code of Bitcoin and analyse its agreement aspects considering a synchronous communication model. This study was later extended by Pass *et al.* in [134] to systems where messages sent in a round can be received later. [83] and [134] among other things prove that with high probability, all participants of a proof-of-work blockchain will share a growing common chain. Basically, the forks will be resolved with high probability.

Bitcoin-NG. To overcome some of the drawbacks of Bitcoin, Eyal *et al.* propose in [73] *Bitcoin-NG*, NG for next-generation. The idea of Bitcoin-NG is that the execution of the system is organised in epochs. In each epoch, a leader elected via a proof-of-work mechanism will decide the order of transactions that will be committed in the blockchain until the next epoch. Although more scalable than Bitcoin, Bitcoin-NG, however, inherits the drawbacks of Bitcoin: costly proof-of-work process, presence of forks, no guarantee that a leader in an epoch is unique; Bitcoin-NG actually introduces a new drawback, which is that the leader can change the history at will if it is corrupted.

Some Proof-of-Work based Protocols. On another line of research, Decker *et al.* propose in [59] the *PeerCensus* system that targets linearisability of transactions. PeerCensus combines the proof-of-work blockchain and the classical results in Byzantine Fault Tolerant (BFT) agreement area. However, being built on the Bitcoin's blockchain, PeerCensus suffers from the same drawbacks as Bitcoin.

In the same spirit as PeerCensus, and decouple to Bitcoin, in [106], Kokoris-Kogias *et al.* propose *ByzCoin*. Byzcoin is based on a leader-based consensus over a group of members chosen based on a proof-of-membership mechanism. As in PeerCensus, when a miner succeeds to mine a block, the former is included in the set of voting members. ByzCoin enhanced with a scalable collective signing process the classical consensus algorithm PBFT (*Practical Byzantine Fault-Tolerant*) [45]. ByzCoin also inherits some of the Bitcoin problems and vulnerabilities. It has to be noted that the distributed implementation of the core idea, the collective signing, is still an open problem.

3.1.2 Proof-of-* – Alternatives to Proof-of-Work

To solve the high energy-consumption issue of proof-of-work, many proposals were made to select the next participant that should add a new block to the chain. Specifically, most of the

proposals replace the need for computing power by other things (such as memory, wealth, *etc.*) keeping the main idea of proof-of-work, namely the random selection.

In this section, we give an overview of some of the proposals. Note that there are always new mechanisms presented, therefore giving a complete and total snapshot of the proof-of-work alternatives is not feasible.

Proof-of-stake. The main candidate alternative of proof-of-work is *proof-of-stake*. Even the second most known blockchain system, Ethereum [159], plans to switch their proof-of-work mechanism to proof-of-stake.

In proof-of-stake, the role of computing power is replaced by the number of stakes possessed by participants in the system. There exist many different ways of using stakes. For example, in *PeerCoin* [105] the first blockchain proposing proof-of-stake, the information considered is the "age" of the coin. If a participant holds a coin for a long time, it has more chances to be the next to add a block relative to another participant that just received a new coin. Other blockchain systems consider the number of coins a participant has, its stake, such as Ouroboros [102], *i.e.*, the higher stake a participant has, the higher are its chances to be selected to add a block.

Note that usually in proof-of-stake, the selection of the next participant to add a block is also random (to calque the behaviour of Bitcoin), but is proportional to the stake, *i.e.*, it may happen that multiple participants are selected. Forks do still exist in proof-of-stake.

There exist many more specific proof-of-stakes, for example, *delegated proof-of-stake* or *DPoS* in which participants can delegate part of their stakes to other participants that have more chances to be selected, the rewards are then shared among the winner and its delegators. DPoS is used for instance in Tezos [8].

Proof-of-burn can also be considered as a variant of proof-of-stake. The main ingredient of proof-of-burn is that the more stakes a participant burns (stakes that cannot be used any more), the higher the chances of the participants to be selected.

Several academic works address *proof-of-stake* based blockchains. To the best of our knowledge, the first on this line is *Snow White* [58] authored by Daian *et al.*. Daian *et al.* propose a protocol for semi-synchronous systems. The execution of the protocol is organised in epochs. Similar to Bitcoin-NG [73] in each epoch a different committee is elected, and inside the elected committee a leader will be chosen. The leader is allowed to extend the blockchain by adding blocks. The protocol in [58] is validated via simulations and only partial proofs of correctness are provided. More recently, *Kiayias et al.* propose [102] Ouroboros. Ouroboros uses a sortition based proof-of-stake protocol and the article addresses mainly the security aspects of the proposed protocol. Algorand [85] also uses proof-of-stake for the blockchain. In Algorand, participants are selected to maintain the blockchain base only on their stakes and by relying on cryptographic techniques.

Other proof-of-*. We quickly present some other alternatives.

In *proof-of-Elapsed-Time* or *PoET*, (i) each participant has to wait a random amount of time, and (ii) the first participant to finish waiting its time gets to add the next block. To ensure that a participant indeed waits its given duration, PoET considers the use of special CPU instructions (called *Trusted Execution Environment* or *TEE*) like the Intel SGX. Intel SGX allows applications to run trusted code in a protected environment, and a participant can prove that it really executed some instructions (*e.g.*, waiting). PoET was introduced by Intel with the blockchain system Hyperledger Sawtooth [132]. Since PoET relies almost entirely on SGXs and the most known SGX is Intel's, such an approach defies the decentralised nature of blockchains. It also requires all participants to be equipped with SGXs.

Proof-of-space [70] is similar to proof-of-work, but instead of computing power, participants have to provide the evidence that a certain amount of storage is available; *proof-of-authority* or its improved variant *proof-of-reputation* [80], in which the more authority or the more reputation a participant has in the system, the higher are its chances to be selected as the next to add a block.

All these proof-of-* alternatives have the same fork issue as in proof-of-work, in part because the selection relies heavily on random functions weighted with the *-specific component. They did not receive much attention in the academic research. Among all these alternatives, *proof-of-stake* protocols and specifically those using variants of PBFT [45] became recently popular not only for in-chain transaction systems but also in systems that provide transactions between different blockchain systems. We review some in the next section.

3.1.3 Committee-based

As we have already seen quickly with PeerCensus or Byzcoin, some blockchain systems use committee and variants of Byzantine fault-tolerant (BFT) algorithms. These blockchains are called *committee-based blockchains*. In these blockchains, for each block, there is a subset of participants, a committee that produces that block, while in other blockchains, the goal is to delegate that work to exactly one participant.

The problem of agreeing over a distributed system is called the *consensus problem*. Formally, we say that an algorithm implements the consensus, or the algorithm is a consensus algorithm if the following properties hold:

- *Termination.* Every obedient participant decides on a value;
- *Agreement.* If two obedient participants decide respectively on values v and v', then $v = v'$;
- *Validity.* If all obedient participants have the same input v, then if an obedient participant decides, it must decide on value v.

With the above definition of validity, if all participants do not have the same input, always deciding a default value (not even in the input value) is a consensus algorithm. To avoid such a scenario, variants of the validity definition were proposed, we can cite the following:

- a value decided by an obedient participant should be in the input of an obedient participant (sometimes called *strong validity*).
- a value decided by an obedient participant should be in the input of a participant (called *uniform validity* in [48]).

More variants of the validity property exist as well, for instance, *vector validity* defined in [65], where each participant has as input a vector, and the decision of each participant must be a vector instead of a single value. An algorithm satisfies the vector validity property if the decided vector contains values from the vectors of enough obedient participants.
Instead of relying on the input values of the participants, depending on the application, the concept of *external validity* may be of interest. *External validity* was defined by [41], and requires that a value decided should be valid with respect to a given and known predicate. For example, all financial transactions contained in blocks should be valid and maintain positive balance considering the whole history of the blockchain up to that respective block.

When a consensus algorithm tolerates Byzantine participants, the algorithm is said to be *Byzantine fault-tolerant* or *BFT* for short. In the blockchain realm, there exist several BFT consensus-based blockchain proposals (*e.g.*, [4, 21]).

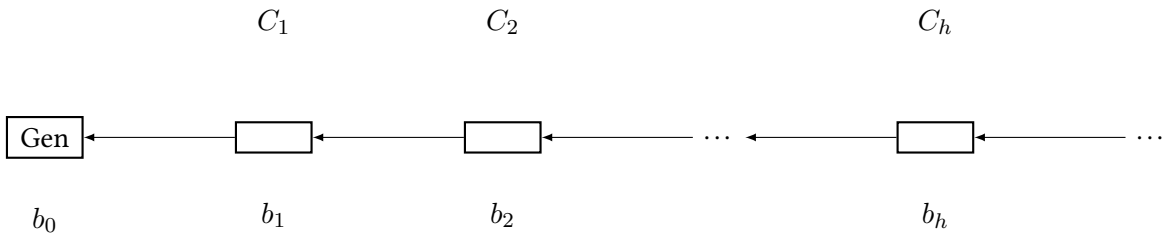


Figure 3.4: Committee-based Blockchain

Byzantine Agreement Protocols The Byzantine Agreement problem or BFT consensus problem has been introduced in [113] where it has been proved that, in the presence of f Byzantine participants, such problem cannot be solved with less than $3f + 1$ participants in a synchronous message-passing system (where the message delivery delay is upper bounded). The consensus problem, as proved in the seminal FLP paper [77], cannot be solved in an asynchronous message-passing system (when there are no upper bounds on the message delivery delay) in the presence of at least one faulty participant.

In between those impossibility results, it is still possible to solve consensus in an asynchronous setting, either adding randomness [30] (which also proved the impossibility result for $n \leq 3f$ for any asynchronous solution) or partial synchrony as in Dwork et al. [68] (DLS) where BFT consensus is solved in an eventual synchronous message-passing system (there is a time τ after which there is an upper bound on the message delivery delay). DLS preserves safety during the asynchronous period and the termination only after τ when the message transfer delay becomes bounded. Finally, Castro and Liskov proposed PBFT [45], a leader-based protocol that optimises the performances of the previous solutions.

In [76], Fisher and Lynch prove that it is not possible, in the worst case, to solve consensus in less than $f + 1$ rounds. Those protocols (as their improvements [52]) work in eventual synchronous systems evolving in a sequence of "views" or "epochs", and in each of them, one participant, the leader for that epoch, is responsible for the evolution of the execution. What is important for safety is how the "view change" is managed, since between different views participants may deliver different messages, which can lead to violation of the agreement property.

In order to use BFT protocols in an open setting, recent research has been devoted to either find secure mechanisms to select committees of fixed size over time (*e.g.*, [85]) and/or to propose incentives to promote participation (*e.g.*, [4]). These researches revive the study of the distributed problem of *group membership* [56], which is the task of creating a (dynamic) subset of participants in a distributed fashion.

A way to use BFT protocols is to select whenever needed a committee that will run a BFT consensus algorithm and produce a block. An example is shown in Figure 3.4, where committee C_1 produces the block b_1, the committee C_2 produces the block b_2, *etc.* Each committee produces a block by using a Byzantine agreement protocol.

Such approach can guarantee the absence of fork under deterministic and known conditions. Moreover, if the selection of the committee is done by using energy-efficient techniques, then the high energy-consumption problem is also avoided.

BFT-inspired Blockchain Protocols Several blockchains are based on Byzantine agreement protocols: Algorand [85], ByzCoin [106], HotStuff [160], RedBelly [54], Solidus [4], Tendermint [39], *etc.*

It is interesting to note that a leader-based approach is not necessary to implement con-

sensus. In [36], Borran and Schiper implement a leader-free algorithm keeping an optimal number of replicas. In the same spirit, Redbelly [54] proposes for the first time a leader-free algorithm solving consensus among participants in a blockchain system. Interestingly the consensus specifications have been adapted to the blockchain scenario. That is, a decided value has to be valid with respect with the blockchain level rather than being one of the values proposed at the beginning. Such specification is then considered in DBFT [55], an evolution of the consensus algorithm in RedBelly (now at the heart of the RedBelly blockchain), and in the Tendermint consensus algorithm. DBFT works with a weak coordinator, meaning that the execution can terminate even in presence of a faulty coordinator, while Tendermint and HotStuff are changing the coordinator at each view as in [155]. In the same blockchain context, other consensus algorithm thought to be scalable, are SBFT [86] which optimises the performance of PBFT, and HotStuff [160] (inspired by Tendermint) which uses in addition threshold signatures.

On a probabilistic side, Algorand uses *sortition* algorithms. The work of Gilad *et al.* in [85] focuses mainly on the agreement aspects of blockchains using probabilistic ingredients and providing probabilistic guarantees. More specifically, the set of participants that are allowed to produce and validate blocks is randomly chosen and changes over time.

In any way, the objective of blockchain consensus algorithms is to ensure that all participants agree on the same (ever-growing) blockchain as if they repeatedly do consensus on each new block of the chain. This can be called a *repeated consensus*, and we define it as follows. We say that an algorithm implements a Byzantine repeated consensus if and only if it satisfies the following properties:

- *brc-Termination.* Every obedient participant has an infinite blockchain;
- *brc-Agreement.* If the i^{th} block of the blockchain of an obedient participant is B, and the i^{th} block of the output of another obedient participant is B', then $B = B'$;
- *brc-Validity.* Each block in the blockchain of any obedient participant is valid; it satisfies a predefined predicate.

It was shown by [91] and [89] that consensus is not needed for transferring financial assets, and specifically, consensus is not needed to prevent double-spending. Intuitively, when there are no relations between two transfers, there is no need to agree on their order. However, for more general blockchain applications, for instance, when smart contracts are considered, consensus is needed to guarantee consistency.

Complexity There exist many different consensus algorithms, and many more are continuously proposed. They usually have different efficiency or are more or less adapted to some specific applications. To compare different consensus algorithms, many factors can be taken into account; the most popular comparison tool is the "complexity" of the algorithms. The most famous for Byzantine agreement algorithms are *bit*, *message*, and *round* complexity.

Bit complexity refers to the number of information exchanged by participants in the agreement process, for simplicity here, we refer to it as the number of information each participant must store; message complexity is the worst-case number of messages sent during the synchronous period before reaching an agreement; and the round complexity is the maximum number of rounds, during the synchronous period, needed to reach an agreement.

Note that in PBFT in synchronous rounds, when the leader is obedient, the complexity of the protocol is $O(n^2)$. Otherwise, a view change mechanism takes place to change the leader and resume the computation. The view-change is used to avoid that, in case of a faulty leader,

if some obedient participant decides on a value v, the other obedient participants cannot decide on a value $v' \neq v$ when the new leader proposes a new value.

The view-change mechanism in classical consensus algorithms implies that when a leader is suspected to be faulty, all participants have to collect enough evidence for the view-change. That is, the view-change message contains at least $2f+1$ signed messages and these messages are sent from at least $2f+1$ participants which yield a message complexity of $O(n^2)$. These messages are then sent to all participants, the view-change has then $O(n^3)$ message complexity. Since the protocol terminates when there is an obedient leader, which may happen for the first time in epoch $f+1$, then in the worst-case scenario it has a message complexity of $O(n^4)$.

Interestingly, Tendermint, as well as similar recent approaches, uses an alternative mechanism for leader replacement that allows dropping message complexity to $O(n^3)$, and even $O(n^2)$ when using threshold or collective signatures. Participants, instead of exchanging all the messages they already delivered (used previously to trigger a view change), locally keep track of potentially decided values.

Additionally, in these algorithms, each view (an attempt to have an agreement on a value) is composed of multiple *phases*, where a phase is a sequence of send, delivery and compute.

In [108], Kowalski and Mostéfaoui propose an algorithm for consensus in synchronous setting with a bit complexity of $O(n^3 \log n)$ for an optimal round complexity of $f+1$. There is a trade-off between the bit complexity and the round complexity of the Byzantine agreement.

Protocol	Message complexity	Rounds complexity	Bit complexity	Phases per view
DBFT [55]	$O(n^3)$	-	?	-
PBFT [45]	$O(n^4)$	$f+1$	*poly*	3
HotStuff [160]	$O(n^2)$	$f+1$	$O(n^2 \log n)*$ threshold signature	4
Tendermint [39]	$O(n^3)$	n	$O(n^2 \log n)$	3

Table 3.1: Comparison of BFT Consensus algorithms in the Eventual Synchronous System.

We present in Table 3.1 the complexity of some consensus algorithms that are developed for eventual synchronous systems.

3.1.4 Our contributions to Blockchain's Construction

In this thesis, we study how committee-based blockchains are built in a deterministic manner. Specifically, we study the problem they are trying to solve in the sense of a distributed system abstraction, *i.e.*, we define the *Byzantine repeated consensus* abstraction in Byzantine environment, which represents the construction of a blockchain. Additionally, we formalise for the very first time the Tendermint algorithm, and prove its correctness; moreover, we compute its complexity and compare it to state-of-the-art algorithms.

3.2 Fairness of Blockchains

To motivate participants to add and maintain the blockchain correctly, rewarding mechanisms are in place. Rewards are given to block creators for each block successfully added to the blockchain. In blockchains such as Bitcoin, a reward is given to the block creator that successfully adds its block to the blockchain, *i.e.*, exactly one participant is rewarded at the time.

Since only correct blocks are added to the blockchain, participants that are rewarded behave correctly for that block.

In committee-based blockchains, rewards should be given to committees. In each committee, because only some committee members can be faulty but not all, rewarding mechanisms are inherently more complex to handle than in other blockchains. The properties of reward mechanisms for committee-based blockchains must be studied. Ad minimum, the rewarding mechanism must be fair, *i.e.*, distributing the rewards in proportion to the merit of participants, where *merit* abstracts the notion of effort participants invest for the construction of the blockchain.

A blockchain protocol is said to be *fair* if any *obedient participant* (a participant that followed the protocol) that has a fraction α of the total merit in the system will get at least α fraction of the total reward that is given in the system.

In the blockchain literature, chain-quality has been defined by [83] and was later on extended by [135]. In [83], Garay *et al.* define the notion of *chain-quality* as the proportion of blocks mined by obedient miners (block creators) in any given window; Garay *et al.* study the conditions under which during a given window of time, there is a bounded ratio of blocks in the chain that non-obedient participants produced, over the total blocks in the blockchain. Chain-quality can be seen as a definition of fairness in Bitcoin-like systems, and even more generally, in most forkable (proof-of-* based) blockchains. A blockchain system has the property of chain-quality if the proportion of blocks produced by obedient participants in any given window is proportional to their relative mining power. Intuitively, chain-quality ensures that non-obedient participants do not produce more blocks than their proportion of mining power.

In [102], Kiayias *et al.* propose Ourobouros [102] and analyse its chain-quality property. In [135], Pass and Shi propose a notion of *fairness*, which is an extension of the chain-quality property still dedicated to Bitcoin-like blockchains; they address one of the vulnerabilities of Bitcoin studied formally in [74]. In [74], Eyal and Sirer prove that if an adversary controls a coalition of miners holding even a minority fraction of the total computing power, this coalition can gain twice its share. Fruitchain [135] overcomes this problem by ensuring that no coalition controlling less than a majority of the computing power can gain more than a factor $1 + 3\delta$ by not respecting the protocol, where δ is a parameter of the protocol. We note that in the model of Fruitchain, only one participant creates a block in the blockchain and that a participant has a reward for the created block. In [90], Guerraoui and Wang study the effect of the delays of message propagation in Bitcoin, and they show that in a system of two miners, one can take advantage of the delays and be rewarded exponentially more than its share.

In [92, 93], Gürcan *et al.* study the fairness from the point of view of the participants that do not participate in the construction of the blockchain but create transactions. Herlihy and Moir do a similar work in [96] where the authors study users' fairness and consider as an example the Tendermint blockchain. Herlihy and Moir discussed how participants with malicious behaviour could violate fairness by choosing transactions, and then they propose modifications to the original Tendermint to make some violations detectable and accountable. In [115], Lev-Ari *et al.* study the fairness of transactions in committee-based blockchains with synchronous assumptions by using a detectable communication abstraction allowing them to identify faulty and malicious participants.

Recent works consider the distribution of rewards in proof-of-stake based blockchains. In [112], Lagaillardie *et al.* show that even if Tendermint is unfair, the presence of delegators helps the growth of the system. In [75], Fanti *et al.* define equitability, which represents the evolution of the fraction of total stakes of participants, in particular, Fanti *et al.* compute the so-called *compounding effect* where rewards are directly re-invested in stakes. In [99], Karakostas *et al.* define egalitarianism. Egalitarianism means that each participant, no matter its stake

proportion, wins the election and is the participant to append a new block the same amount of time as everyone else, no matter their stake proportion.

One of the main differences between Bitcoin-like systems and committee-based blockchains is that in the former, one participant produces a block, whereas, in committee-based blockchain, a committee (a set) of participants produces a block. A committee is not necessarily composed only of obedient participants but can contain a mix of Byzantine and obedient participants (with a correctness hypothesis of having some majority of obedient members). From the point of view of block creators, the definition of chain-quality cannot be applied to committee-based blockchains. The fairness should not be defined with respect to blocks produced, but rather, it should be relative to the proportion of total rewards each participant gets.

Our Contributions to Fairness of Blockchains. To study the fairness of committee-based blockchain protocols, we extend the definition of [135] for systems where each block is produced by a subset of participants. This is the case of Hyperledger Fabric or Tendermint for example, where for each block there is a subset of participants, *a committee* that produces that block. We define the fairness in two mechanisms: the *selection mechanism* and the *reward mechanism*. Each participant has a given merit, which represents the effort it is putting to maintain the blockchain, for instance, the merit can represent the mining power of a participant in proof-of-work blockchains, or the stakes in proof-of-stake blockchains, *etc.* The *selection mechanism* selects for each new height the committee members (the participants that will run the consensus instance) for that height. The *reward mechanism* is in charge to distribute rewards to committee members that produce a new block. Intuitively, if the selection mechanism is fair, then each participant will become committee member proportionally to its merit, and if the reward mechanism is fair then for each height, only the obedient committee members get a reward. By combining the two mechanisms, an obedient participant is rewarded at least a number proportional to its merit parameter over the infinite execution of the system. In this thesis, we focus on fairness, where participants with higher merit should get more rewards. Our notion of fairness is different from egalitarianism since the goal of egalitarianism is to have all participants rewarded the same, no matter their merit. We discuss how fair are few selection mechanisms, and we show the impact of time assumptions of the communication on the fairness of reward mechanisms.

3.3 Rational Behaviours in Blockchain Systems

3.3.1 Proof-of-* Blockchains & Rational Participants

It is famously quoted that Bitcoin's proof-of-work resists and works normally even if up to 50% of the computing power is held by malicious participants, and particularly also expected that all obedient participants will be rewarded a share proportional to their computing power. Note that when having more than 50% of control of the system, no distributed system can offer guarantees any more. The attacks that can happen in Bitcoin when a pool controls more than 50% of the total computing power are called the 51% *attacks*. That claim of resilience up to 50% was actually provided without any proof.

However, when considering rational arguments, it is possible to show that the above claim does not hold. Many articles [74, 101, 144] prove that thresholds on the fraction of obedient participants needed to guarantee security properties is lower than the 50% initially though, and is more about 25%; in particular, they show that rational (selfish) participants can stra-

tegically deviate from the proof-of-work protocol to gain more than their expected reward share by following the protocol. In an earlier version of [74], Eyal and Sirer presented the *selfish mining attack*. A selfish or rational participant does a selfish mining attack if instead of presenting the blocks it succeeded to mine (winner of the proof-of-work), keeps them secret to have an advance against the other participants, such that disclosing all blocks at the same time yields the attacker more rewards. Eyal and Sirer presented their attack in a system composed of 2 participants. They show that such an attack is always possible, but to be beneficial, the attacker needs to have at least a quarter of the total mining power of the system. In [101], Kiayias *et al.* extend the study of Eyal and Sirer. Kiayias *et al.* present tighter bounds on computing power where a selfish participant might do a selfish mining attack against an obedient participant, and show when it is better to follow the prescribed protocol; specifically, Kiayias *et al.* present the best response for a selfish participant against an obedient participant. In [144], Sapirshtein *et al.* present multiple variants of selfish mining attacks, and by comparing them, show that the initial selfish mining proposal of Eyal and Sirer is not the one giving the highest outcome to the attacker. Moreover, Sapirshtein *et al.* consider instead of only 2 participants, the more general case of multiple participants.

Although the works presented above, and many more, establish very pessimistic thresholds on Bitcoin, in practice Bitcoin works, even when the obedient majority bounds proposed do not hold. Following this observation, in [24], Badertscher *et al.* propose a rational analysis of Bitcoin based on the rational protocol design framework [81]. The proposed framework can be seen as a game; it is at an upper level of abstraction, proposing a two-player game between the protocol designer and the adversary. The rational protocol design also models the behaviour of (some of) the protocol participants as rational and studies the problem of *secure multi-party computation* [43]. Secure multi-party computation is broadly the study of problems where multiple participants aim to cooperate to solve given tasks without revealing their private information.

Considering only rational participants, in [31] Biais *et al.* model Bitcoin as a coordination game. Biais *et al.* prove that although playing the proof-of-work and following the longest chain's rule is an equilibrium, there exist multiple other equilibria where forks may persist.

Note that some results in these analyses may seem to be contradictory. This is mainly due to the models considered. In [103], Kiayias and Stouka propose a framework in which most prior results are presented, and in which the different prior results may be compared.

Concerning behaviours inside or between different pools of miners, in [72], Eyal provides insight about the miners infiltrating other pools such that the participants of targeted pool earn less, and in [29], Belotti *et al.* study the behaviours of Bitcoin's miners and show when they are better off switching to different pools or staying in the current one.

As we sketched, many analyses have been made on strategic behaviours in blockchains. However, they mainly focus on forkable (proof-of-* based) systems and even more specifically on proof-of-work. Some works have studied some alternatives such as the work of Saleh in [143] about proof-of-stake. Saleh studies *proof-of-stake* blockchains and shows that the *nothing-at-stake* problem (where participants may extend every fork) is mitigated because participants with large stakes on the main chain prefer not to add blocks on forking branches since doing so must reduce the value of their stakes. About committee-based blockchains, and to the best of our knowledge, very few works have been dedicated to analyse or discuss the presence of rational behaviours among participants. We present them in the next section.

3.3.2 Committee-based Blockchains & Rational Participants

Before presenting the state-of-the-art of studies related to rational behaviours in committee-based blockchains, we first present the studies of rational behaviours in distributed systems.

In Distributed Systems. Rational participants have been considered in various works in distributed computing, *e.g.*, [1, 2, 3, 5, 7, 81, 88, 95, 118]. [1] shows some advantages of combining game theory and distributed computing and presents some challenges. As for Byzantine consensus, the utilities of rational participants take into account whether or not a decision is reached. For the problem of Byzantine agreement for instance, Groce *et al.* in [88] consider an environment with rational and obedient participants where they provide protocols that tolerate rational adversaries and proved lower bounds. In [95], Halpern and Vilaça prove that in a full rational setting, if participants can fail by crashing, then there is no ex-post Nash equilibrium solving the fair consensus problem (where fair means that the input of every participant is decided with equal probability), even with only one crash. They also present a protocol satisfying fair consensus under some assumptions over the failures patterns. An ex-post Nash equilibrium is a situation where the behaviours of the rational participants are the same, regardless of their distribution in the set of participants.

In [5], Afek *et al.* propose building blocks for distributed algorithms and propose protocols solving the problems of consensus and renaming in presence of rational participants. For leader election, Abraham *et al.* in [3] show that in systems with only rational participants, under certain conditions, it is possible to obtain a k-resilient equilibrium (resistant to a coalition of up to k participants deviating).

In [118], Lysyanskaya and Triandopoulos consider rational and Byzantine participants while studying the problems of secret sharing and multi-party computation. Lysyanskaya and Triandopoulos propose an incentive-compatible protocol resistant to a coalition of up to f faulty participants, where the utilities of participants take into account whether a decision is reached or not, and on which value the decision is made; they also analysed the case where the utilities of Byzantine participants may be unknown. Concurrently to [118], Abraham *et al.* propose in [2] an incentive-compatible protocol for secret sharing with rational participants where some utilities can be unknown.

We can now focus on the works done regarding rational behaviours in committee-based blockchains.

In Committee-based Blockchains. In most analyses concerning committee-based blockchains, it is assumed that participants are either obedient or Byzantine. These analyses fail to thoroughly explore the effect of rational participants. In this line of work, the work of Abraham *et al.* in [4] is probably, and to the best of our knowledge, the first to consider rational participants in a committee-based blockchain. Abraham *et al.* have introduced in Solidus interesting incentive mechanisms; however, they neither provide a formal framework for their analysis nor consider the cost of the actions.

In the protocol proposed by Manshaei *et al.* in [121], multiple committees run in parallel to validate a non-intersecting set of transactions (a shard). A static game approach for the intra-committee protocol is taken, leading to the result that rational participants can free ride when rewards are equally shared.

In [78], Fooladgar *et al.* propose an analysis showing that the proposed reward distribution in Algorand [85] is not an equilibrium, *i.e.*, some participants may deviate from the protocol to increase their expected gain. Fooladgar *et al.* consider the cost of actions of the participants and propose a better reward scheme for Algorand.

3.3.3 Our Contributions to the Study of Rational Behaviours in Blockchain Systems

Often as in [95], the studies from the distributed system's literature focus on proposing protocols that are Nash equilibria, which guarantee consensus if and only if the rational participants do not deviate from the prescribed protocol. They do not fully analyse the system to find if there exist other equilibria. Most of the studies also considered only two actions for the participants, either following the protocol or not following it.

In this thesis, we specifically study these cases, *i.e.*, we study different equilibria in our setting and check the conditions under which the consensus is possible, even in the presence of rational participants. Moreover, we give to the participants fine-grained actions such that they can deviate at specific key points of the protocol. Since blockchains highlight the costs and rewards, we consider them in our analyses.

3.4 Conclusion

Before concluding, we would like to acknowledge that we did not talk about all distributed ledger technologies (DLTs); *e.g.*, the *directed acyclic graph based*, or simply *DAG-based* DLTs such as IOTA [137], where there are no blocks, but only links between transactions (specially designed for IoT devices); or Sycomore [20], built upon proof-of-work, which adapts to user transactions throughput by increasing the number of children a block can have.

In blockchain systems, many approaches exist to build a blockchain; in particular, the *proof-of-work* which is the most popular mechanism. Although quite efficient, and having some advantages, proof-of-work has some (serious) drawbacks. To name a few, the existence of pools make the system more centralised, which defies the idea behind blockchains; the huge energy consumption; and the presence of forks which lead to some (temporary) inconsistency in the blockchain. To fix or to avoid the drawbacks, many mechanisms were invented and are considered; however, most of them resolve only few of the problems. Interestingly, committee-based blockchains, which use results from decades of research from the distributed systems community, are recently considered and studied for blockchains. Such blockchains allow avoiding most of the problems inherent to proof-of-work. However, research is still needed to answer many questions such as the fairness of these systems relative to the block creators, and the behaviours of the participants.

In this thesis, we study and analyse committee-based blockchains. We provide a distributed system abstraction, which captures the construction of a blockchain, and we study the correctness of one of the most popular committee-based blockchain, Tendermint. This analysis is presented in Chapter 4. Moreover, in Chapter 4, we study these blockchains from a fairness aspect. We define the notion of fairness for these blockchains; we discuss the selection of the committees, and the impact of synchrony of the communication on fairness.

To understand the behaviours of rational participants, we provide a framework to analyse such behaviours in committee-based blockchains. We study the different equilibria against the consensus properties. Specifically, in Chapter 5, we provide the different equilibria in committee-based blockchains where all participants want to increase their expected gain; and in Chapter 6, we do similar analyses considering the presence of malicious participants inside the committees.

Note that many surveys, state-of-knowledge, and when-to-use articles are available on many blockchain-related topics, *e.g.*, [28, 34, 82, 116, 126, 145, 161]. They offer more in-depth descriptions and details of blockchain protocols, blockchain studies, and of their applications.

CHAPTER 4

CORRECTNESS AND FAIRNESS OF COMMITTEE-BASED BLOCKCHAINS

Contents

In a nutshell, blockchain systems maintain a continuously-growing history of ordered information, encapsulated in blocks. Blocks are linked to each other by relying on collision-resistant hash functions ([141]), *i.e.*, each block contains the hash of the previous block. The blockchain itself is a distributed data structure (distributed ledger) replicated among different peers. To preserve the chain structure, those peers need to agree on the next block to append in order to avoid forks. The most popular technique to decide which block will be appended is the *proof-of-work* mechanism of Dwork and Naor [69]. The block that will be appended to the blockchain is owned by the participant (miner) having enough computing power to solve a cryptographic puzzle first. The best-known way to solve this puzzle is by repeated trials. The major criticism for the *proof-of-work* approach is the following: the generation of a block is energetically costly, which yields to the creation of mining pools and finally, multiple blockchains might coexist in the system due to accidental or intentional forks. Many alternatives

in the same spirit of drawing one participant at random exist. One can cite *proof-of-activity*, *proof-of-burn*, *proof-of-elapsed time*, *proof-of-space*, *proof-of-stake*, *etc.* All these alternatives, although they reduce the energy consumption problem, suffer from the fork issue.

Other classes of building blockchains exist. Instead of drawing one participant at random, for each new block to be added, a committee is selected and is in charge of agreeing on which block to append next. Among all these alternatives, those using variants of *Practical Byzantine Fault-Tolerant* consensus [45] became recently popular not only for in-chain transaction systems but also in systems that provide cross-chain transactions. We can name the following blockchain systems as example: Algorand [85], HotStuff [160], RedBelly [54], Tendermint [39], *etc.* They have the purpose of avoiding forks by relying on a committee that has to agree on the next block to add. The committees run blockchain consensus algorithms. Those algorithms are inspired by well-known algorithmic techniques such as the ones from classical consensus, *e.g.*, [45, 68, 113, 122, 153]. Committee-based blockchains can guarantee the absence of fork. They seem promising, but many questions are open and need to be addressed.

In this chapter, we focus on the study of committee-based blockchains. We first define the problem committee-based blockchain try to solve in a deterministic fashion: the *repeated consensus*, by extending the existing definition to cover Byzantine failures. Additionally, we study these blockchains from a fairness point of view.

4.1 System Model

In this section, the reader may refer to Chapter 2 for more in-depth details about the model.

The system is composed of an infinite set $\Pi = \{1, 2, \dots i, \dots\}$ of sequential participants; i is the *index* of the participant i. *Sequential* means that a participant executes one step at a time. This does not prevent the participants from executing several threads with an appropriate multiplexing. As local processing time is negligible with respect to message transfer delays, we consider it as being equal to zero.

Arrival model. We assume a *finite arrival model*, *i.e.*, the system has infinitely many participants but each run has only finitely many. The size of the set $\Pi_\rho \subset \Pi$ of participants in each system run is not a priori-known.

Communication In the following, we assume the presence of a reliable byzantine broadcast, *i.e.*, the broadcast protocol satisfies the following conditions:

- Validity. If an obedient participant delivers a message m from an obedient participant i, then i broadcasted the message m;
- Integrity. No obedient participant delivers twice the same message from a participant;
- Termination-1. If an obedient participant broadcasts a message m, all obedient participants eventually deliver m.
- Termination-2. If an obedient participant delivers a message m from a participant, then all obedient participants eventually deliver m.

Messages are created with a digital signature, and we assume that digital signatures are unforgeable.

Time assumptions on communication. The participants communicate by exchanging messages through an eventually synchronous network. *Eventually Synchronous* means that after a finite unknown time τ there is an a priori unknown bound δ on the message transfer delay. When $\tau = 0$ and δ is known the network is *synchronous*.

Failure model. Some participants can exhibit a Byzantine behaviour in the system. A Byzantine participant is a participant that can deviate arbitrarily from the given protocol. We do not assume any bound on the number of Byzantine participants in the system, but inside each committee, the number of Byzantine participants is upper bounded. A participant that exhibits a Byzantine behaviour is called a Byzantine or a *faulty* participant, and a participant that follows the given protocol is called *obedient*.

Let i be a participant, and let T be a fragment of i's execution. If at the beginning of the fragment T, the internal state of i is correct and i follows the given protocol during the fragment T, then we say that i is T*-obedient*. A correct internal state is a state of the participant that can be the result of i following the protocol. An obedient participant is T-obedient $\forall T$.

4.2 Consensus & Blockchains

4.2.1 Distributed Consensus

Recall that blockchain systems are distributed ledgers, where all participants should locally have the same sequence of blocks. Blocks being added in an append-only manner, all participants should agree on the next block to add. In the most popular blockchain system, Bitcoin [128], it happens that participants "temporarily" disagree on which block is the next. When two or more participants disagree on the next block, we say that there is a *fork*.

Agreeing in a distributed fashion among a set of participants has been intensively studied in distributed systems with the abstraction of the *deterministic distributed consensus* [113], or simply *consensus* defined in the following.

Definition 4.1 (Consensus)**.** *An algorithm implements the consensus in presence of Byzantine participants if and only if it satisfies the following properties:*

- Termination. *Every obedient participant decides on a value (a block);*
- Agreement. *If two obedient participants decide respectively on values B and B', then $B = B'$;*
- Validity. *A decided value by any obedient participant is valid; it satisfies a predefined predicate.*

We use the concept of *external validity* introduced by [41]. The validity predicate must be known by all participants and is defined by the given application. External validity was later adapted by [54] as a well-suited validity concept for blockchains.

Note that it is impossible to solve consensus in asynchronous systems when there is at least one failure [77]. To solve consensus, it is necessary to have $f \leq \lfloor n/3 \rfloor$ participants that exhibit a Byzantine behaviour, where n is the total number of participants; equivalently, $n \geq 3f + 1$.

The claim that Bitcoin achieves consensus in an asynchronous system and is resilient to up to $n/2$ Byzantine participants cannot be true. The presence of forks in the lifetime of Bitcoin shows that consensus is not always guaranteed. For example, a participant does not know

when a block is really considered decided. Bitcoin implements in some sense a probabilistic consensus.

Using a consensus algorithm to reach decisions in blockchain systems is not simple. Often, we cannot directly use consensus algorithms to build blockchains. Blockchain systems may be open, so new participants can always enter the system, all participants may not be known in advance, and the number of participants may vary during the execution. Consensus algorithms, however, assume the number of participants to be known and fixed. Another important fact is that blockchain systems are designed to have a lot of participants, but consensus algorithms require a lot of exchanged messages to reach an agreement. Consensus is not really scalable in the number of participants.

A way to circumvent the problem is to select a known and finite subset of participants that will run the consensus. We call that subset of participants a *committee*. The committee may change during the system run, but its size is a priori known and fixed.

Committee-based blockchains (that are described in the next section) delegate the task of consensus to selected committees.

4.2.2 Committee-based Blockchains in a Nutshell

We denote by $\mathbb{B}$ the set of blocks. A block contains, among other things, a header and a list of transactions. The header contains the hash ([141]) of the previous block, the time at which the block was created, some application dependant information, *etc.* Note that we use the broad term "transactions" to designate the data inside the blockchain; transactions are application dependant.

We denote by $\mathbb{B}^*$ the set of all finite sequences of blocks. Let $bc \in \mathbb{B}^*$, be a finite sequence of blocks. $|bc|$ is the length (the number of blocks) of bc. We say that bc is a *blockchain* if $\forall k \in \mathbb{N}^* : 0 < k \leq |bc|$, in the header of the block at position k in bc there is the hash of the block at position $k-1$. The position of a block is also called its height. If additionally, the list of transactions in each block in the blockchain is valid with respect to the given validity predicate, we say that bc is a *valid* blockchain. The block at height 0 is the *genesis block*.

The genesis block initialises the blockchain, *i.e.*, it defines the *committee* in charge of producing the block at position 1 (a committee is a subset of participants), describes how rewards will be distributed among committee members (which we call the *reward mechanism*), describes how participants will be selected for being part of committees with respect to the state of the blockchain (the *selection mechanism*), *etc.* Ideally, all this information should be public and known by all participants. In fact, with the history of the blockchain, all participants should be able to compute and/or know the sets of committee members selected.

When a participant starts a new height, it computes the committee for that height. For a height, participants that are not members of the corresponding committee just wait for the decision from its committee members. The committee members for that height execute an agreement procedure (*e.g.*, a consensus algorithm, Definition 4.1) to determine the block they want to add for that height.

Once the committee members reach a decision, they each send the block decided to the whole network, and move to the next height. Participants that were not part of the committee, the non-committee members, wait to collect the decided block from "enough" different committee members; waiting for sufficient period of time the decision allows to be tolerant to failures. Once a non-committee member collects enough times the same decided block, it considers that block as its decision for that height and then moves to the next height. When moving to the next height, participants may wait a certain amount of time to collect more messages from committee members. The decision messages from committee members are the

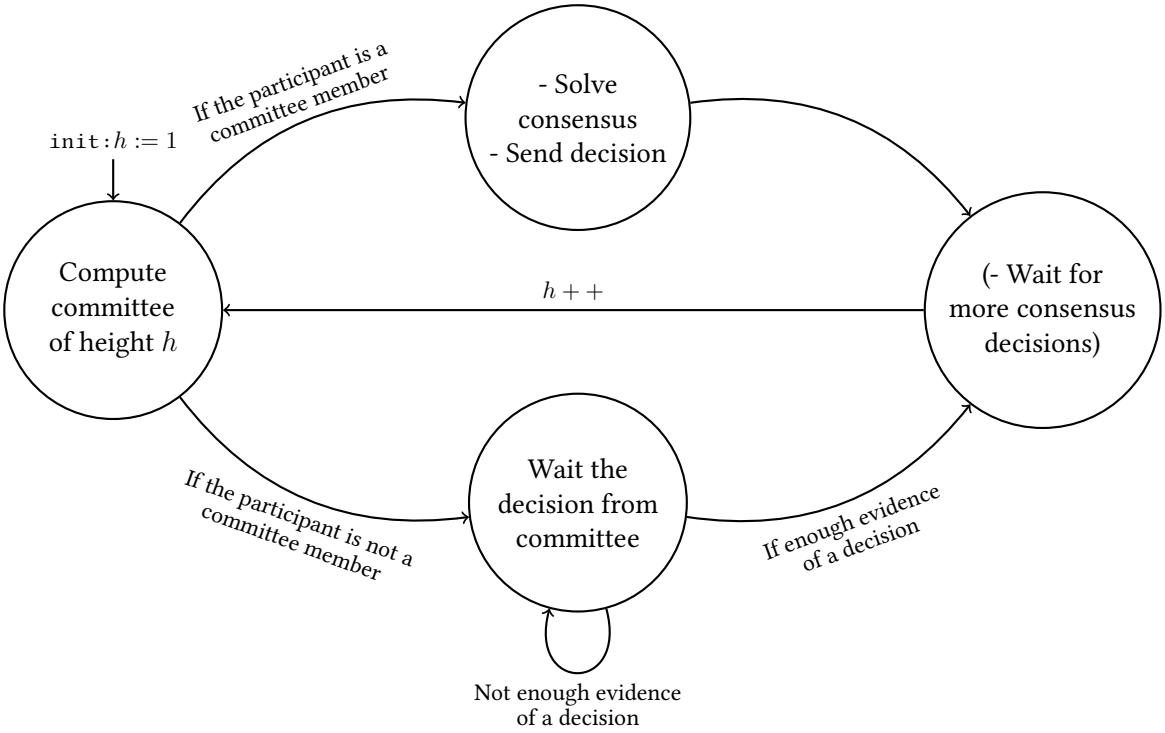


Figure 4.1: State Machine of a Repeated Consensus Algorithm to Build a Committee-based Blockchain.

ones used to reward previous committees. Intuitively, if a participant receives a decision message for the block decided by a committee member, then probably that committee member followed the protocol during that height[1] and can be rewarded. A state machine describing the generic execution is given in Figure 4.1.

We can note that using the consensus at each height allows having exactly one corresponding block, hence the absence of forks. Many blockchains such as Algorand, HotStuff, RedBelly, Tendermint, *etc.*, use at their core such approach; sometimes by using variants of the consensus abstraction; interestingly, the structure of the execution is the same. Note that Algorand uses a probabilistic variant of the consensus, leading to the existence of forks with very low probability.

4.3 Problem Definition

In this section, we introduce and define two abstractions studied during this thesis: the *repeated consensus*, which is the problem solved by committee-based blockchains; and the *fairness* of the rewards distribution.

4.3.1 Blockchain Repeated Consensus

The repeated production of blocks by the different committees can be seen as multiple instances of agreement, where at each height, exactly one block should be decided.

Many abstractions relative to multiple consensus have been studied. The concept of *multi-consensus* is presented in [26], where the authors assume that only the faulty participants can

[1] That is not true in general, since Byzantine participants, for instance, can send the decided value at the end without doing anything during their protocol execution.

postpone the decision of obedient participants; additionally, the consensus is made a finite number of times. The *long-lived consensus* presented in [64] studies the consensus when the inputs are changing over time, their specification aims at studying in which conditions the decisions of obedient participant do not change over time. However, none of the specifications previously cited is appropriate for blockchain systems. In [60], Delporte-Gallet *et al.* defined the *repeated consensus* as an infinite sequence of consensus instances, where the inputs values may be completely different from one instance to another; but all obedient participants have the same infinite sequence of decisions.

Blockchain consensus can be viewed as a repeated consensus problem. However, in [60] only crash failures were considered. We propose in Definition 4.2 a generalisation of the *repeated consensus* resilient to Byzantine failures. The main difference is that we do not predicate on the faulty participants, Byzantine included. Each obedient participant outputs an infinite sequence of decisions; each decision corresponds to a block with the blockchain analogy. We call that sequence of decisions the *output* of the participant.

Definition 4.2 (Repeated Consensus)**.** *An algorithm implements a Byzantine repeated consensus if and only if it satisfies the following properties:*

- brc-Termination. *Every obedient participant has an infinite output;*
- brc-Agreement. *If the i^{th} value of the output of an obedient participant is B, and the i^{th} value of the output of another obedient participant is B', then $B = B'$;*
- brc-Validity. *Each value in the output of any obedient participant is valid; it satisfies a predefined predicate.*

Alternatively, brc-Termination can be stated as follows: "$\forall k \in \mathbb{N}$, every obedient participant eventually outputs the k^{th} value". However, we use the definition above to stick with the definition of [60].

If clear from the context, we simply write Termination, Agreement and Validity instead of respectively brc-Termination, brc-Agreement and brc-Validity.

If an algorithm implements the repeated consensus then each obedient participant will have an infinite sequence of decisions (blocks); any two obedient participants will have the exact same sequence (the same blockchain), and all blocks in the sequence will be valid with respect to the application dependant predicate. This abstraction corresponds to the idea of a blockchain, as the output of each participant is its local sequence of blocks.

We now understand how committee-based blockchains work, and specifically which problem they solve. Blockchain systems usually give rewards to block creators, here the committees. These rewards serve to give incentives to maintain and to build the blockchain. Rewards are on top of the block agreement procedure: to know which participants to reward, the blocks of these participants should be in the blockchain, as well as the reward. The question of fairness is of interest in blockchain systems. In the next section, we propose a definition of fairness for committee-based blockchains.

4.3.2 Fairness of Committee-based Blockchains

In the blockchain literature, a blockchain protocol is considered *fair* if "any obedient participant having a fraction α of the total merit in the system gets at least α fraction of the total rewards" that are given in the execution of the system or over a sufficiently long period of time. Where intuitively, the merit is the effort the participant is putting to maintain the blockchain.

Classical notions of fairness in blockchains such as *chain-quality* [83] do not apply for committee-based blockchains. The chain-quality definition of fairness considers the number of blocks an obedient participant adds to the blockchain over a long period of time; however, since in committee-based blockchains, one block is produced by more than one participant, such definition should be extended. Instead of defining the fairness by the number of blocks a participant produces, we define a notion of fairness with respect to the number of rewards a participant gets over the execution of the system.

To tackle the fairness of committee-based blockchain protocols such as Algorand, HotStuff, RedBelly, Tendermint, *etc.*, we split the mechanism in two: the *selection mechanism* and the *reward mechanism*. We say that each participant has a given merit, which represents the effort the participant is putting to maintain the blockchain, for instance, it can represent the mining power of a participant in proof-of-work blockchains, or the stakes in proof-of-stake blockchains, *etc.* The *selection mechanism* selects for each new height the committee members (the participants that will run the consensus instance) for that height. The *reward mechanism* is in charge of distributing rewards to committee members that produce a new block. Informally, if the selection mechanism is fair, each participant will become committee member proportionally to its merit; and if the reward mechanism is fair, for each height, only the obedient committee members get a reward. By combining the two mechanisms, an obedient participant is rewarded at least a number proportional to its merit parameter over the infinite execution of the system.

4.3.2.1 Selection Mechanism

In a system where the size of the committees is strictly lower than the number of participants in the system, there should be a way to select the members of the committees. Always selecting the same participants leads to the centralisation of the system. The participants that are *always selected* can exercise a power of oligarchy, and add in the blockchain only transactions they want.

Note that the task of creating a subset of participants in a distributed fashion has been studied in distributed systems under the name of *group membership problem* [56]; group membership is also about agreeing on the sequence of membership changes. The group membership was proved not to be always possible; in particular, it is impossible in asynchronous systems with crash failures [47]. However, the problem here seems slightly different since new groups should be formed for each height, and one can use a deterministic algorithm thanks to the blockchain that serves as a share and common view.

$\forall h > 0$, let C_h be the set of committee members for the height h. $\forall h > 0$, we assume that $|C_h| = n$, the size of the committees is fixed and equal to n. When the height is clear from the context, we simply write C for the set of committee members.

To analyse different mechanisms of selection, we first need to define it properly. Formally, a selection mechanism is a function $\text{selection} : \mathbb{B}^+ \times \mathbb{N} \to \Pi^n \cup \emptyset$; where n is the size of committees, Π the set of participants, and $\mathbb{B}^+$ represents the set of non-empty blockchains. If *bc* is a non-empty blockchain, then:

$$\text{selection}(bc, h) = \begin{cases} C_h, & \text{if } |bc| \geq h - 1 \\ \emptyset, & \text{otherwise} \end{cases}.$$

Recall that $|bc|$ is the length of the blockchain *bc*.

Some information can be computed from and/or stored in the blockchain, *e.g.*, the number of time each participant has been committee member, the wealth of each participant, *etc.* Based on this information, the selection mechanism can select according to some order, for instance,

wealthiest participants with the highest stakes, participants that were committee members less often, always the same set of participants, *etc.*

To abstract the notion of the effort of a participant in the system to maintain the blockchain, we denote by $\alpha_i(t) \in [0,1]$ the merit parameter of participant i at time t proportionally to the total merit at time t, such that $\forall t, \sum_{i \in \Pi_\rho} \alpha_i(t) = 1$.

If $\forall t \in \mathbb{T}, \forall i, \alpha_i(t) = \alpha_i(t_0)$, we simply denote by α_i the merit of the participant i. In that case, the merits do not depend on the evolution of the blockchain nor its contents.

Let v_i be the number of times i becomes committee member proportionally to the number of blocks. This value is computed at the limit of the blockchain construction. Therefore, if a participant j leaves the system at a point in time, then $v_j = 0$. We have that $v_i \in [0,1]$. We propose the following definition of the fairness of selection mechanisms where merits are fixed and do not change with time. The definition allows all participants with positive merit to be a member of committees infinitely often with respect to their merit.

Definition 4.3 (Fairness of Selection Mechanism)**.** *Assume that the blockchain is built infinitely, so $\forall h \geq 0$, there is a block at position h. We say that a selection mechanism is fair if it respects the following properties:*

1. *If $\alpha_i \neq 0$ then $v_i \neq 0$; or equivalently, $\alpha_i \neq 0 \implies \forall h \geq 0, \exists h' \geq h : i \in C_{h'}$;*
2. *If $\alpha_i \geq \alpha_j$ then $v_i \geq v_j$.*

Informally, Condition 1 means that each participant with a positive merit parameter should become a committee member infinitely often. Condition 2 means that a participant with low merit cannot be selected more than a participant with higher merit. Note that this definition depends only on the merit and not on the behaviour of the participants (obedient or Byzantine).

A fairness definition of a generic selection mechanism (where merits do change over time) is still an open question; however, such definition should encapsulate Definition 4.3 as a special case.

Remark 4.4. *If the total number of participants in the system is equal to the size of committees, then all participants are always selected, therefore, the selection mechanism, in that case, is trivially fair, although asking a huge set of participants to run the consensus is not scalable.*

4.3.2.2 Reward Mechanism

In blockchain systems, participants that produce and add blocks to the blockchain are rewarded. In committee-based blockchains, a committee is the producer of a block. Within that committee, some participants may not behave as prescribed, therefore, rewarding all participants may not be fair to those who followed the prescribed protocol. In particular, there may be different ways of rewarding members of committees. To do so, we define the *reward mechanism*. A reward mechanism consists of a function $\text{reward} : \mathbb{B}^+ \times \wp(\mathcal{O}(\textit{Information}) \times \mathbb{N} \to \mathbb{R}^{|\Pi_\rho|} \cup (\bot, \ldots, \bot)$; where $\wp(\mathcal{O}(\textit{Information})$ is the power set of all messages, and $\mathbb{B}^+$ represents the set of non-empty blockchains.

If *bc* is a blockchain and $|bc| < h$, $\text{reward}(bc, M, h) = (\bot, \ldots, \bot)$; we do not reward committee members when there is no block produced for their corresponding height. Otherwise, it assigns to each participant a given reward. In $\text{reward}(bc, M, h)$, M represents the set of messages used to compute the rewards, h the height of the blockchain *bc* where the reward of committee members is computed.

A reward is considered *allocated* if it is written in the blockchain. The second part of the reward mechanism is choosing when to allocate the reward corresponding to a given height.

If a reward has been allocated at a height h, the participant can use it after a certain number of blocks defined in the genesis block (*e.g.*, [74, 83]). We consider that for each block production, its rewards are allocated in one block and not split over different blocks, such that after the allocation of rewards a participant knows if it has been rewarded or not. Note that once rewards are allocated, their values cannot change any more.

It is interesting to note that some blockchain systems consider punishment mechanism, called *slashing*, to afflict some costs to a participant if there is a proof of misbehaviour, as described in [131]. We do not consider slashing in our mechanisms.

We recall the definition of T-obedient, with T being a fragment of execution of a participant. Let i be a participant, and let T be a fragment of i's execution. If at the beginning of the fragment T, the internal state of i is correct and i follows the given protocol during the fragment T, then we say that i is T-obedient. A correct internal state is a state of the participant that can be the result of i following the protocol. An obedient participant is T-obedient $\forall T$. For example, i is considered h-obedient if during its execution of height h it followed the protocol.

We define the following properties for characterising the fairness of a reward mechanism. Let h be a height. Each committee member has a boolean variable r_i^h, which we call *reward parameter* defined as follows:

f1 – If i is a not a committee member for h, then $r_i^h = 0$;

f2 – h-**completeness.** If i is a committee member for h and i is h-obedient, then $r_i^h = 1$;

f3 – h-**accuracy.** If i is a committee member for h and i is not h-obedient, then $r_i^h = 0$.

If $r_i^h = 0$, it means that i is not rewarded for height h, and if $r_i^h = 1$, i has been rewarded for h. The properties are inspired by the classical properties of failure detectors [48].

Remark 4.5. *We do not reward non-committee members. Many committee-based blockchains consider delegation, i.e., a participant delegates to a committee member, and once the committee member is rewarded, all of its delegates are rewarded proportionally to what they delegated. We do not consider delegations, but it can be nice to enrich our model with delegations. For rewards in blockchains with delegations, the reward parameter for each participant r_i^h must contain more information and not just be a boolean variable.*

Definition 4.6 (Complete Fairness of a Reward Mechanism)**.** *Let $\mathcal{R}$ be a reward mechanism. If $\forall h > 0$, $\mathcal{R}$ satisfies Conditions f1 and h-completeness (Condition f2), we say that $\mathcal{R}$ satisfies complete fairness.*

If a reward mechanism satisfies complete fairness, it means that for all height $h > 0$, all h-obedient committee members are rewarded, and non-committee members are not.

Definition 4.7 (Accurate Fairness of a Reward Mechanism)**.** *Let $\mathcal{R}$ be a reward mechanism. If $\forall h > 0$, $\mathcal{R}$ satisfies Conditions f1 and h-accuracy (Condition f3), we say that $\mathcal{R}$ satisfies accurate fairness.*

If a reward mechanism satisfies accurate fairness, it means that for all height $h > 0$, no h-obedient committee member is rewarded.

Definition 4.8 (Fairness of a Reward Mechanism)**.** *Let $\mathcal{R}$ be a reward mechanism. If $\forall h > 0$, $\mathcal{R}$ satisfies Conditions f1, h-completeness (Condition f2) and h-accuracy (Condition f3), we say that $\mathcal{R}$ is fair.*

We say that a reward mechanism is fair when at each height h, all and only h-obedient committee members are rewarded.

Our definition of fairness states that for any height Conditions *f1* - *f3* are satisfied. Participants should always receive all rewards they deserve. This definition can be weakened.

Definition 4.9 (eventually fairness of a Reward Mechanism)**.** *Let $\mathcal{R}$ be a reward mechanism. If $\exists h_0 > 0 : \forall h \geq h_0$, $\mathcal{R}$ satisfies Conditions f1, h-completeness (Condition f2) and h-accuracy (Condition f3), we say that $\mathcal{R}$ is eventually fair.*

A reward mechanism is eventually fair if after an a priori unknown but finite time, the rewards are allocated to and only to obedient committee members.

When a reward mechanism is fair, it is also eventually fair but the reverse (reciprocal) is not necessarily true.

4.4 Fairness Analysis of Committee-based Blockchains

4.4.1 Examples of Selection Mechanisms

First, we quickly review from a fairness point of view two selections mechanisms based on the wealth of participants. Let us assume that there are $N > n$ participants during the whole execution of the system, we also assume that no new participants can enter, and participants cannot exit. Let us also assume that for all participants, merits do not change over time, and all participants have the same merit: $\forall i, j \in \Pi_\rho, \alpha_i = \alpha_j > 0$. Recall that merits are not necessarily stakes.

In the examples below, we consider selection mechanisms that depend on the stakes of participants, while the merit is fixed and does not depends on the (evolution of) stakes. All participants are obedient, and a committee member is rewarded when the committee it is part of produces a block. When a participant is rewarded, its stakes increase, but its merit remains the same.

For our analysis, we further consider that participants are ordered by their stake and their id (public key). Let us assume that at the beginning of the execution, all participants have the same amount of stakes. Without loss of generality, and up to renaming, we consider at any point of the execution that if $\exists i, j : i < j$ such that i and j have the same amount of stakes, then i is selected before j.

Select the participants with the highest stake. This selection mechanism works as follows: for any height h, with respect to the blockchain up to height $h - 1$, the n participants having the biggest amount of stakes are selected to be part of the committee.

This mechanism leads to a situation where only the n participants selected first will always be selected, and the other participants will never be. This mechanism is not fair, since Condition 1 of Definition 4.3 is not satisfied. In fact, all not selected participants have positive merit, and they should be selected infinitely often according to the fairness definition.

Note that if we consider that the n selected participants are the only one with positive merit, and the others have a merit equal to 0, then the selection is fair.

Select the participants with the lowest stake. This selection mechanism works as follows: for any height h, with respect to the blockchain up to height $h - 1$, the n participants having the lowest amount of stakes are selected to be part of the committee.

Recall that N is the total number of participants; the number of times each participant has been selected after l blocks is on average $l \times n/N$ selections. This mechanism is fair according to Definition 4.3. In fact, all participants have the same positive merit, and they are all selected infinitely often and they are all selected about the same number of time.

Let us remark that this mechanism is fair in the model considered in this example since participants cannot exit nor enter after the beginning of the execution. If that assumption is removed, *i.e.*, if participants could enter or leave, the following can happen. Once a participant is selected and rewarded, it knows that it would not be selected before a long period of time, on average after n/N blocks; one incentive could be to create new sub addresses (new participants) such that they will have low stakes and will be selected faster and more often. Another similar scenario is that if a participant has a big amount of stakes, it might not be allowed to participate in committees for a long time until all the participants with small stakes caught up. The participant might want to split into a lot of stakeholders with small stakes. In such a way, the new stakeholders will always have the smallest stakes and be selected, if it continues to do so, it might block other participants to be committee members. Although fair, selecting the participants with the lowest stakes does not seem stable in an open setting.

Remark 4.10. *An unfair selection mechanism can lead to a centralisation of the system, by always letting the same participants decide on the blocks to add in the blockchain. Although the assumption on the bound of Byzantine participants does not depend on the selection mechanism, we note that when a selection mechanism selects the participants with the lowest amount of stakes, and only obedient participants in committees are rewarded, at some point participants that were not obedient will have the lowest stake, thus will be selected.*

The existence of a fair and "good" selection mechanism is still an open question.

4.4.2 Analysis of Reward Mechanism's Fairness

In this section, we review the time assumptions on communication needed to achieve fairness in committee-based blockchains.

Complete Fairness. If a reward mechanism satisfies complete fairness, it means that for all height $h > 0$, all h-obedient committee members are rewarded, and non-committee members are not.

Proposition 4.11. *There exists at least one reward mechanism satisfying complete fairness.*

Once a block is in the chain, rewarding all committee members, in the next block, for that block and only them satisfy Conditions *f1* and *f2*. Condition *f1* is satisfied since, for all height, non-committee members are not rewarded. Condition *f2* also holds, for any given height h, all committee members of h are rewarded, in particular all h-obedient committee members.

Accurate Fairness. If a reward mechanism satisfies accurate fairness, it means that for all height $h > 0$, all non h-obedient committee members are not rewarded.

Proposition 4.12. *There exists at least one reward mechanism satisfying accurate fairness.*

Never allocating rewards satisfies Conditions *f1* and *f3*. Condition *f1* is satisfied since non-committee members are not rewarded. Condition *f3* holds since no participant is rewarded; in particular, for any given height $h > 0$, no non h-obedient committee members is rewarded.

Fairness. Although simple and trivial to satisfy either complete fairness or accurate fairness, a mechanism satisfying both at the same time is more complex and not always possible.

First, about the fairness of committee-based blockchains, we can highlight the following.

Remark 4.13. *Recall that C_h is the set of committee members for height h. $\forall h > 0$, if $|C_h| > 1$, then for a reward mechanism to be (eventually) fair, rewards cannot be allocated directly in the corresponding block. For any height $h > 0$, the set of h-obedient committee members cannot be known in advance. If $\forall h > 0, |C_h| = 1$, the reward can be directly allocated only to the committee member, so in the block at height h; such is the case of Bitcoin-like blockchains.*

Theorem 4.14. *There exists a fair reward mechanism in a committee-based blockchain protocol if and only if the system is synchronous.*

Proof

We prove this theorem by double implication.

- If there exists a fair reward mechanism, then the system is synchronous.

  Let $\mathcal{R}$ be a reward mechanism. By contradiction, we assume that $\mathcal{R}$ is fair and that the system is not synchronous.

  C_h is the set of committee members for the height h. Let $k > 0$ be the fixed number of blocks to wait before distributing the rewards for C_h. The reward is allocated by the committee $C_{h'}$, where $h' = h + k$. Recall that k is defined in the genesis block. Since the system is not synchronous, the committee members of height h', $C_{h'}$, may not receive all messages from C_h before allocating the rewards. Reward allocation for height h should be done at height h' and not after.

  Since the reward mechanism is fair, by Conditions *f1* - *f3*, all and only the h-obedient committee members of the height h have a reward parameter equal to 1. That means that the h'-obedient committee members of $C_{h'}$ know exactly who were the h-obedient committee members in C_h, so they got all the consensus messages of height h before distributing the rewards. Contradiction, therefore the system is synchronous.

- If the system is synchronous, then there exists a fair reward mechanism.

  We assume that the system is synchronous and $\forall h > 1$, all messages sent by h-obedient participants at height h are delivered by all other obedient participants before the block at height $h + 1$. Let $\mathcal{R}$ be the following reward mechanism: let h be a height, rewards for a block at height h are allocated at height $h + 1$ by the committee C_{h+1}.

  - If a participant is not a committee member for height h, it sets its reward parameter to 0, this is known since participants are already at height $h + 1$.
  - By combining the messages from committee members of h participants, since the communication system is synchronous, it is possible to differentiate between h-obedient and non h-obedient committee members, then it sets the reward parameter of h-obedient committee members of h to 1 (this is possible since the system is synchronous, therefore obedient participants of C_{h+1} delivered the messages from the consensus instance of height h); and it sets the reward parameter of non h-obedient committee members of h to 0.

  By construction, the committee members in $h + 1$ allocates rewards to all and only h-obedient committee members, so $\mathcal{R}$ is fair, it satisfies all fairness Conditions *f1* - *f3*.

□*Theorem 4.14*

If there is no synchrony, there cannot be a fair reward mechanism for committee-based blockchains.

eventually fairness. Before discussing eventually fairness, we introduce detectable Byzantine. In fact, we need to detect the behaviour of participants to have a chance to be fair. The problem of detecting participants' behaviours is however still an open problem. In the following paragraph, we briefly discuss some existing works on the topic.

Detecting Byzantine Failures. In synchronous systems, it is always possible to detect Byzantine participants, for example using the broadcast abstraction detectable all-to-all (DA2A) defined in [115]. When Byzantine can be detected, it is possible to not reward them, therefore, being able to satisfy Condition *f3*. If we cannot detect them, then it will be difficult, if not impossible, to distinguish between participants that should be rewarded and those that should not be rewarded. In eventually synchronous systems, the problem of detecting Byzantine participants is complex. For instance, Kihlstrom *et al.* in [104], propose a failure detector to solve consensus in presence of detectable Byzantine. However, they distinguish between detectable and non-detectable Byzantine; detectable Byzantine are the participants whose behaviours can be detected, for instance by doing omission or commissions failures. Non-detectable Byzantine are Byzantine participants whose faults cannot be detected, for example participants that alter their internal state. The basic idea is that all Byzantine faults cannot be detected. Therefore, the focus can only be on detectable Byzantine. In [94], among other things, Haeberlen and Kuznetsov provide a formal framework for fault detections, and provide a formal classification of the various failures that can occur. In [87], Greve *et al.* extend the approach and propose a failure detector for detectable Byzantine in dynamic networks.

Although Kihlstrom *et al.* proposed a failure detector for solving consensus, our problem is not the same, and we cannot apply their failure detector as it is. In [104], once a Byzantine behaviour is detected, the Byzantine participant should be suspected forever. In blockchain systems, we do not want to punish indefinitely participants who failed at one point, but behave correctly after and forever. In more details, we want for any height h to not reward only participants that were not h-obedient. For example, let i be a participant such that it is part of committees h and h', such that $h' > h$. Suppose also that during height h, i sent unintentionally contradictory messages (and so is Byzantine), but then i recovered before the beginning of height h' and follows the protocol during h' and after. Even if i is not h-obedient, it recovered before h' and is h'-obedient. If i has been detected and not rewarded for height h, that should not prevent it to be rewarded for its work during height h', and since it follows the protocol, it should be rewarded for height h'. The failure detector proposed by Kihlstrom *et al.* is not appropriate for us.

Theorem 4.15. *There exists an eventually fair reward mechanism in a committee-based blockchain protocol if and only if the system is (eventually) synchronous and Byzantine participants are detectable.*

Proof

We prove this theorem by double implication.

- If there exists an eventually fair reward mechanism, then the system is eventually synchronous or synchronous and Byzantine participants are detectable.

  Let $\mathcal{R}$ be a reward mechanism. We assume that $\mathcal{R}$ is eventually fair.

If $\mathcal{R}$ is fair, by Theorem 4.14, the communication is synchronous and we can use the DA2A abstraction [115] to detect the Byzantine participants, which ends the proof. Otherwise, since $\mathcal{R}$ is eventually fair, that means that there is a point in time h from which all the rewards are correctly allocated, so for any height $h' \geq h$, h'-obedient committee members of committees at height h' are able to distinguish between non-obedient participants during the height they are distributing the rewards, the Byzantine are then detectable. If we consider h as the beginning of the execution, then we have that $\mathcal{R}$ is fair, and by Theorem 4.14, the communication from height h is synchronous, so the message delay is upper bounded. We have that after h, the message delay is upper bounded, so the communication is eventually synchronous. Therefore, the Byzantine are detectable and the communication is synchronous or eventually synchronous.

- If the system is eventually synchronous or synchronous, and Byzantine participants are detectable, then there exists an eventually fair reward mechanism.

  If the system is synchronous, the proof follows directly from Theorem 4.14. Consider that the system is eventually synchronous, but not synchronous. Let $\mathcal{R}$ be the following mechanism: Let h be a height, rewards for a block at height h are allocated at height $h+1$ by the committee C_{h+1}.

  - If a participant is not a committee member for height h, set its reward parameter to 0, this is known since participants are already at height $h+1$.
  - By combining the messages from committee members of h, if there is not sufficient information to detect the behaviour of participants, reward only those detected as h-obedient that are in C_h, and the participants proposing the distribution of reward increases their duration to wait (for more decisions) before starting the next height. If there is enough information to detect the behaviour of all participants in C_h, then we reward the h-obedient participants in C_h, and we do not reward non h-obedient participants in C_h.

  $\mathcal{R}$ is eventually fair.

□$_{Theorem\ 4.15}$

Corollary 4.16. *In an asynchronous system, there is no (eventual) fair reward mechanism in a committee-based blockchain tolerating Byzantine participants.*

Proof

Assume that the system is asynchronous, where there are good periods such that consensus can be reached. By contradiction, let $\mathcal{R}$ be an eventually fair reward mechanism.

- If there are non-detectable Byzantine participants in the system, $\mathcal{R}$ is not fair (Theorem 4.15);
- If all Byzantine participants are detectable, then by Theorem 4.15, the system must be synchronous, or eventually synchronous.

We have a contradiction, since the system is asynchronous. It is not possible to have an (eventual) fair reward mechanism in an asynchronous system. □$_{Corollary\ 4.16}$

Note that if the protocol tolerates Byzantine faults, even if all the participants are obedient but that is not known in advance, Corollary 4.16 holds. It is different from the FLP impossibility result of consensus in an asynchronous system with one faulty participant [77].

4.4.3 Numerical Examples of Reward Allocation

In this section, we examine the impact of different communication models on the fairness of reward mechanisms through several numerical examples that confirm the results on the fairness of reward mechanisms from Section 4.4.2.

Execution. In our analyses, participants run a committee-based blockchain protocol as described in Section 4.5.1; and rewards for a block produced at a height h are allocated in the block at height $h+1$. Note that the consensus module is Byzantine fault tolerant. We highlight the environment's important characteristics: the communication system, the total number of participants in the system, the size of each committee, the different type of participants and their number at a given height, the rewarding mechanism, and the selection mechanism. We must choose the value of these parameters before launching the execution. We consider different communication systems, and rewards are allocated by the next committee by using messages they delivered from the previous height – they use the combination of all messages and check if they correspond to the correct time and a possible value to send according to their current state.

We consider a system where all participants are part of all committees. For clarity, and without loss of generality, we consider a system with $n \geq 4$ participants where they are all selected. As stated in Remark 4.4, selecting all participants is a fair selection mechanism. We can then focus on the impact of the network on rewards. For any height h, there can be at most $\lfloor (n-1)/3 \rfloor$ non h-obedient participants in each committee. For a committee, a quorum of $\lceil 2n/3 \rceil$ is needed for any decision. In the case where there are for any height h some non h-obedient participants, we assume that participants have enough information to detect them when allocating rewards. In particular, and for the experiments, the Byzantine participants are specially tagged, and that tag is used only for allocating rewards. When an h-obedient participant receives a message from a non h-obedient (which sends non correct information), it suspects it, and broadcasts the information. When an h-obedient participant delivers at least $2\lfloor n/3 \rfloor + 1$ suspicions for a participant, it considers it as non h-obedient, and does not propose to reward it.

We use MATLAB [123] for the analyses. We analyse three different communication models. First, a synchronous communication, where there is no delay. Then we consider the two following semi-synchronous communication models: (i) the system alternates between good and bad periods, where during good periods, message delays are upper bounded, and (ii) from an unknown time, message delays are upper bounded (eventually synchronous model). We note that in all these models, consensus can be reached. In the good/bad model, progress for consensus instances is guaranteed during the good periods. Note that in the eventually synchronous model, once the global stabilisation time (GST) happens all message delays are upper bounded. If for a participant, the GST happens during height h, then for all height $h' > h$, the message delays are upper bounded, and participants do not know when GST occurs nor the upper bound.

In each configuration of the communication model, we ran the experiment 50 times and took the mean. 0 represents if a participant did not receive a reward, and 1 if the participant received a reward for the corresponding height. We only present the experiment of the eventually synchronous model.

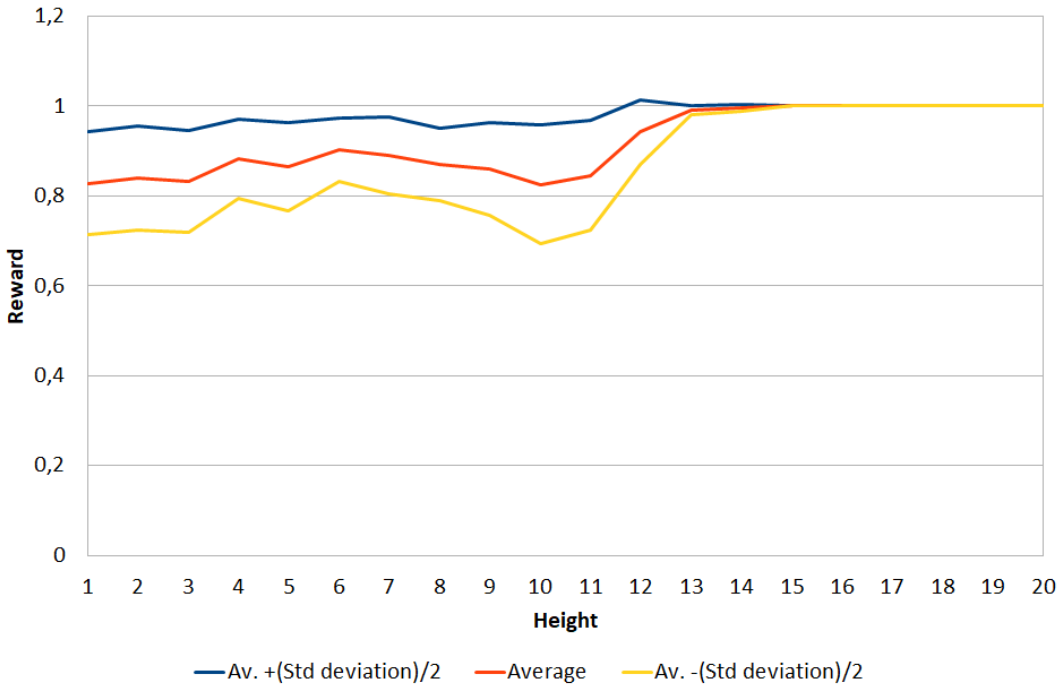


Figure 4.2: Evolution of Rewards in an eventually synchronous System, where GST happens during height 10.

Eventually Synchronous Model. We consider an eventually synchronous model where all participants are obedient, and all participants are part of each committee instances. Recall that a system is eventually synchronous when after a finite but unknown time (the GST), message delay is upper bounded for the rest of the execution. In our examples, we consider that the GST happens during height 10.

On Figure 4.2, we present the evolution of reward for each height. We draw the mean of the average reward of each participant (red curve). The top (blue) and bottom (yellow) curves represent the standard deviation. We can see the set in which the participants are rewarded. When the blue and yellow curves converge, it means that all participants have on average the same reward. We notice that from height 10, the evolution of the reward is increasing. Approximately, from height 14, all participants are rewarded. Before height 10, there is a fluctuation in the evolution of rewards allocated because of the asynchronous period: participants are not necessarily rewarded even if they participate. Their messages were not received on time. Once the GST happens, message delay becomes upper-bounded but some participants still have a timeout shorter than the bound. These participants still increase their timeout until they receive all messages or detect non-obedient behaviours. When all participants deliver messages during their corresponding rounds, they allocate the rewards to all and only obedient participants; it means that the reward mechanism is eventually fair.

4.5 Case Study: Analysis of Tendermint

In this section, we formalise and analyse the correctness of one of the most popular and used committee-based blockchain against the consensus abstraction (Definition 4.1), the repeated consensus abstraction (Definition 4.2), and we analyse its reward mechanism (Definition 4.9).

Tendermint [110] was the first having the merit to link the *Practical Byzantine Fault-Tolerant* consensus to the proof-of-stake technique and to propose a blockchain where a dy-

namic set of committee members (subset of the participants) decide on the next block to be appended to the blockchain.

Tendermint is the most used underlying protocol for committee-based blockchains; and as they called themselves on their website [151], as of June 2020, "*The leading BFT engine for building blockchains*" or "*The world's most popular blockchain framework*". As of writing this manuscript, more than 100 projects are using as core the Tendermint blockchain [53], including Binance Chain [32], a software system developed by Binance, the world largest cryptocurrency exchanges in terms of trading volume. Some techniques from Tendermint were inspirations of other protocols such as HotStuff [160], which itself is what LibraBFT[2] [149] is based on.

Tendermint has been intensively discussed (*e.g.*, [40, 42, 96, 120]) and was (and still is) at the core of many systems [53]; however, at the beginning of this thesis it was not formalised, and its correctness not proven. During this thesis, we formalise for the first time Tendermint, and we analyse the Tendermint consensus protocols. In our study, we found that the implemented version of Tendermint (up to December 2018) was not correct [15]. After the release of our report, the Tendermint team proposed a new version of their algorithm (that can be found in [39]) that was eventually implemented; however, detailed proofs of the algorithm was not provided. We proved correct the last version of Tendermint, showing that previous bugs have been resolved. In this section, we present our works on the formalisation and the correctness of Tendermint, as well as our study of its fairness.

4.5.1 Tendermint Consensus Algorithm

Although there exist many well-known and studied consensus algorithms such as PBFT [45], or DLS [68], for the core layer of their blockchain, Tendermint proposes a new consensus algorithm [39, 150]. We call this algorithm TendermintBFT. TendermintBFT is inspired by classical consensus algorithms such as PBFT but is a bit more adapted for blockchains applications. TendermintBFT as the ones that followed (*e.g.*, HotStuff) have smaller message complexity, as it will be discussed in Section 4.5.1.4; intuitively, once the synchronous period is reached, participants send fewer messages to reach consensus than in PBFT or DLS.

We note that we focus on a single committee at a time. TendermintBFT follows the rotating coordinator paradigm, *i.e.*, for each new block to be appended there is a proposer (chosen among the committee members) that proposes the block. If the block is not decided then a new proposer is chosen (in a round-robin fashion) and so on, until a block is decided by obedient committee members. In the following, we present variants of TendermintBFT in *synchronous* and *eventually synchronous* communication models. Our presentation focuses on the important and key aspects of the algorithm; for clarity, we did not include some optimisations (for example skipping some rounds if sufficiently many participants are in a higher round, *etc*). Recall that the number of participants in each committee is $n = 3f + 1$, where f is the maximum number of Byzantine participants in each committee.

Messages syntax. A committee member i broadcasts a message $\langle TAG, h, e, m \rangle$, where m contains a value v, we say that i pre-proposes, proposes, or votes v if *TAG*=PRE-PROPOSE, *TAG*=PROPOSE, *TAG*=VOTE, respectively. h represents the height or the consensus instance, and e, the *epoch*, intuitively represents the number of rounds for that height since the beginning of the consensus instance.

[2] LibraBFT is the consensus protocol used by committees of the (in)famous Facebook's Libra project. The authors of LibraBFT work at Novi Financial (ex-Calibra) an independent subsidiary of Facebook, Inc.

Algorithm 1 Messages management for participant i

```
1: upon reception of ⟨TYPE, h, e, message⟩ from participant j do
2:   if ∄c : (⟨TYPE, h, e, c⟩, j) ∈ messagesSet then
3:     messagesSet ← messagesSet ∪ (⟨TYPE, h, e, message⟩, j)
```

Basic principles of the protocol. Recall that each block in the blockchain is characterised by its height h, which is the distance in terms of blocks from the genesis block, which is at height 0. For each new height, the two algorithms (Algorithm 2 for the synchronous case and Algorithm 3 & 4 for the eventual synchronous case) share a common algorithmic structure, they proceed in *epoch*s; and each epoch e consists in three rounds: the *PRE-PROPOSE* round; the *PROPOSE* round; and the *VOTE* round. During the PRE-PROPOSE round, the proposer of the round pre-proposes a value v to all the other committee members. During the PROPOSE round, if a committee member accepts v then it proposes such value. If a committee member receives *enough* proposals for the same value v, then it votes for v during the VOTE round. Finally, if a committee member receives *enough* votes for v, it decides on v. In this case, *enough* means at least $2f+1$ occurrences of the same value from $2f+1$ different committee members; from each committee member, only the first value delivered for each round is considered (cf. Algorithm 1).

If the proposer is obedient, it pre-proposes the same value to all the obedient committee members (at least $2f+1$). If all the obedient committee members propose such value, it follows that all the obedient committee members will vote for such value and decide for it. If the proposer is Byzantine it can pre-propose different values to different obedient committee members, creating a partition in the pre-proposal value set collected by committee members. Depending on what the remaining Byzantine committee members do, some obedient committee members may decide on a value v and some other may not (since there are $3f+1$ committee members, there cannot be two different values that collect $2f+1$ distinct votes in the same epoch), then a new epoch starts. To not violate the Agreement property of the consensus (Definition 4.1), obedient committee members that have not decided yet in the previous epoch must only decide for v, for this reason, committee members, before they vote for some value v, *lock* on that value, *i.e.*, they will refuse to propose a further pre-proposed value different than v.

Information from one epoch to the next. The two variables *lockedValue* and *validValue*[3] carry the potentially decided value from one epoch to the next one. The variable *lockedValue* represents the following. If one obedient committee member decides on v, it means that it collected at least $2f+1$ votes for v during the VOTE phase, since there are at most f Byzantine participants among the committee members, thus there are at least $f+1$ obedient committee members that voted for v, and those committee members must not vote for any other value different than v. For this reason, when a committee member delivers at least $2f+1$ proposals for a value v during the PROPOSE round, it sets its *lockedValue* to v, and we say that the committee member is locked on the value v for the corresponding epoch. When an obedient committee member is locked on a value, it can only vote for that value. It can eventually change its locked value only if it delivers enough proposes for a new value more recent than its lock. Hence, if $f+1$ obedient committee members are locked on the same value v, they can only propose and vote for v; any other value $v' \neq v$ can only receive at most $2f$ proposals, which is not enough for both locking and voting, *i.e.*, it is not possible to decide any value

[3] *validValue* was not present in earlier versions of TendermintBFT [111], that was suffering from the Live Lock bug described in Appendix A.1.

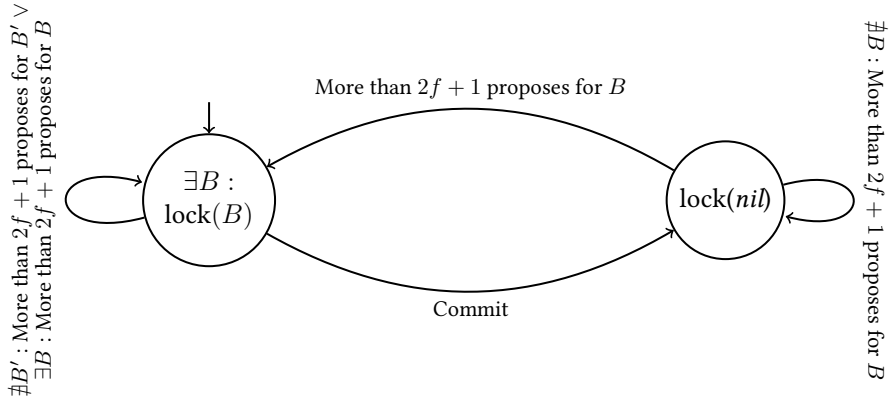


Figure 4.3: State machine Lock/Unlock

different than v.

We depict in Figure 4.3 how a committee member updates its lock value.

On the other side, if no obedient committee members have decided yet, Byzantine committee members may force them to lock on different values. Let us consider a scenario where the proposer is Byzantine and pre-proposes v to $f + 1$ obedient committee members, such that f Byzantine committee members make $x_v \leq f$ of them lock on v; a similar scenario can happen with another value v' so that we can have different obedient committee members, let us say $x_{v'} \leq f$ locked on a different value v'. If any new pre-proposal is checked only against the *lockedValue* then an obedient committee member locked on a value v will refuse (does not propose) any value different from v, and the same for v'. When some obedient committee members are locked, the proposer needs to propose a value on which obedient committee members are locked on; but to be accepted, such value cannot be checked against the *lockedValue* only, because we may never have enough obedient committee members proposing it. For example, if $x_v + x_{v'} \geq f + 1$, no value can get the minimal $2f + 1$ votes. For this reason, committee members keep track of a variable *validValue*, which by construction of the algorithms is such that all obedient committee members have the same *validValue* at the end of the epoch (in the synchronous period). *validValue* represents the latest value on which an obedient committee member may have locked on. Such value is then used to set the value to pre-propose and it is further used along with *lockedValue* to accept or not a pre-proposed value.

Variables and data structures. h is an integer representing the consensus instance the committee member is currently executing. e_i is an integer representing the epoch where the committee member i is in; we note that for each height, a committee member may have multiple epochs. $decision_i$ will stock the decision of committee member i for the current consensus instance. $proposal_i$ is the value the committee member i proposes. $vote_i$ is the value the committee member i votes. $lockedValue_i$ stores a value that is potentially decided by some other committee member. If committee member i delivers more than $2f + 1$ proposes for the same value v during its PROPOSE round, it sets $lockedValue_i$ to v. $validValue_i$ stores a value that is potentially decided by some other committee member. If the committee member i delivers at least $2f + 1$ proposes for the same value v (from different committee members) whether, during its PROPOSE round or its VOTE round, it sets $validValue_i$ to v. $validValue_i$ is the last value that a committee member delivered at least $2f + 1$ times, it can be different from $lockedValue_i$. The latter two variables are used as follows: if i is the proposer then i pre-proposes $validValue_i$ if different from *nil*. Otherwise, if i is a committee member, it checks the new pre-proposal against $lockedValue_i$ and $validValue_i$ if those are different from *nil*.

Functions. We denote as *Value* the set containing all values (blocks). Let $Height = \mathbb{N}^*$ be the set of all heights and let $Epoch = \mathbb{N}$ be the set of all epochs for a given height.

- proposer : $Height \times Epoch \rightarrow C \subseteq \Pi_\rho$ is a deterministic function which gives the proposer out of the set of committee members for a given epoch at a given height in a round-robin fashion.
- $\texttt{isValid} : Value \rightarrow \{\texttt{false}, \texttt{true}\}$ is an application-dependent predicate that is satisfied if the given value is valid with respect to the blockchain. If there is a value v such that $\texttt{isValid}(v) = \texttt{true}$, we say that v is valid. Note that we set $\texttt{isValid}(\textit{nil}) = \texttt{false}$. `isValid` is known and computable by all the committee members and depends on the blockchain history and blockchain application.
- getValue() $\in$ *Value* return a valid value.
- sentByProposer : $Height \times Epoch \times Value \rightarrow \{\texttt{false}, \texttt{true}\}$ is a predicate that gives `true` if the given value has been pre-proposed by the proposer of the given height during the given epoch, and `false` otherwise.
- Let $\mathcal{X}$ be the set either of possible proposals or of possible votes. The function $2\text{f} + 1 : 2^{\mathcal{X}} \rightarrow \{\texttt{false}, \texttt{true}\}$: checks if there are at least $2f + 1$ proposals (resp. votes) in the given set.

Everything defined above is common to the two algorithms. In each section, we specify the data structures relative to a specific version of the algorithm.

4.5.1.1 TendermintBFT for Byzantine Synchronous System

In Algorithms 1 & 2 we describe the algorithm to solve consensus in a synchronous system in the presence of Byzantine committee members. Recall that in synchronous systems, the upper bound of message transfer delay is fixed and known by all participants. The algorithm proceeds in 3 rounds for any given epoch at height h:

- Round PRE-PROPOSE (Lines 8 - 26 of Algorithm 2): If the committee member i is the proposer of the epoch, it pre-proposes its proposal value, otherwise, it waits for the pre-proposal from the proposer. The proposal value of the proposer is its $validValue_i$ if $validValue_i \neq nil$. If a committee member j delivers the pre-proposal from the proposer of the epoch, j checks the validity of the pre-proposal and if valid to accept it with respect to the values in $validValue_j$ and $lockedValue_j$. If the pre-proposal is accepted and valid, j sets its proposal $proposal_j$ to the pre-proposal, otherwise it sets it to *nil*.
- Round PROPOSE (Lines 27 - 39 of Algorithm 2): During the PROPOSE round, each committee member broadcasts its proposal, and collects the proposals sent by the other committee members. After the Delivery phase, committee member i has a set of proposals, and checks if v, pre-proposed by the proposer, was proposed by at least $2f + 1$ different committee members, if it is the case, and the value is valid, then i sets $vote_i$, $validValue_i$ and $lockedValue_i$ to v, otherwise, it sets $vote_i$ to *nil*.
- Round VOTE (Lines 40 - 57 of Algorithm 2): In the round VOTE, an obedient committee member i votes $vote_i$, then i collects all the messages that were broadcasted. First i checks if it has delivered at least $2f + 1$ of proposal for a value v' pre-proposed by the proposer of the epoch, in that case, it sets $validValue_i$ to that value then it checks if a

Algorithm 2 Synchronous TendermintBFT for height h executed by i

```
1:  Initialisation:
2:    e_i := 0                                                  /* This current epoch number */
3:    decision_i := nil                                         /* This variable stocks the decision of the committee member i */
4:    lockedValue_i := nil; validValue_i := nil
5:    proposal_i := getValue()                                  /* This variable stocks the value the committee member will (pre-)propose */
6:    v_i := nil                                                /* Local variable stocking the pre-proposal if delivered */
7:    vote_i := nil

8:  Round PRE-PROPOSE(e_i):
9:    Send phase:
10:     if decision_i ≠ nil then
11:       ∀v, j : (⟨VOTE, h, e_i, v⟩, j) ∈ messagesSet_i, broadcast⟨VOTE, h, e_i, v⟩
12:       return
13:     if proposer(h, e_i) = i then
14:       broadcast ⟨PRE − PROPOSE, h, e_i, proposal_i⟩ to all committee members
15:   Delivery phase:
16:     while (timerPrePropose not expired) do
17:       if ∃v : sentByProposer(h, e_i, v) then
18:         v_i ← v                                             /* v is the value sent by the proposer */
19:   Compute phase:
20:     if !isValid(v_i) then
21:       proposal_i ← nil                                      /* Note that isValid(nil) is set to false */
22:     else
23:       if lockedValue_i = nil ∨ v_i ∈ {lockedValue_i, validValue_i} then
24:         proposal_i ← v_i
25:       else
26:         proposal_i ← nil

27: Round PROPOSE(e_i):
28:   Send phase:
29:     if proposal_i ≠ nil then
30:       broadcast ⟨PROPOSE, h, e_i, proposal_i⟩ to all committee members
31:   Delivery phase:
32:     while (timerPropose not expires) do{}                   /* Collect messages */
33:   Compute phase:
34:     if ∃v : 2f+1(⟨PROPOSE, h, e_i, v⟩) ∧ isValid(v) ∧ sentByProposer(h, e_i, v) then
35:       lockedValue_i ← v
36:       validValue_i ← v
37:       vote_i ← v
38:     else
39:       vote_i ← nil

40: Round VOTE(e_i):
41:   Send phase:
42:     if vote_i ≠ nil then
43:       broadcast ⟨VOTE, h, e_i, vote_i⟩
44:   Delivery phase:
45:     while (timerVote not expires) do{}                      /* Collect messages */
46:   Compute phase:
47:     if ∃v' : 2f+1(⟨PROPOSE, h, e_i, v'⟩) ∧ isValid(v') ∧ sentByProposer(h, e_i, v') then
48:       validValue_i ← v'
49:     if ∃v_d, e_d : 2f+1(⟨VOTE, h, e_d, v_d⟩) ∧ isValid(v_d) ∧ decision_i = nil then
50:       decision_i ← v_d
51:     else
52:       e_i ← e_i + 1
53:       v_i ← nil
54:       if validValue_i ≠ nil then
55:         proposal_i ← validValue_i
56:       else
57:         proposal_i ← getValue()
```

value v' pre-proposed by the proposer of the current epoch is valid and has at least $2f+1$ votes, if it is the case, then i decides v' and goes to the next height; otherwise it increases the epoch number and updates the value of $proposal_i$ with respect to $validValue_i$.

4.5.1.2 TendermintBFT for Byzantine Eventual Synchronous System

This section presents the Algorithms 1, 3 & 4 that solve consensus in an eventually syn-

Algorithm 3 (part 1) Eventual Synchronous TendermintBFT for height h executed by i

```
1:  Initialisation:
2:    e_i := 0                                                    /* Current epoch number */
3:    decision_i := nil                                           /* This variable stocks the decision of the committee member i */
4:    lockedValue_i := nil; validValue_i := nil
5:    lockedEpoch_i := −1; validEpoch_i := −1
6:    proposal_i := getValue()                                    /* This variable stocks the value the committee member will (pre-)propose */
7:    v_i := nil                                                  /* Local variable stocking the pre-proposal if delivered */
8:    validEpoch_j := nil                                         /* Local variable stocking the proposer's validEpoch */
9:    vote_i := nil                                               /* This variable stock the value the committee member will vote for */
10:   timeoutPrePropose := Δ_Pre-propose; timeoutPropose := Δ_Propose; timeoutVote := Δ_Vote

11: Round PRE-PROPOSE:
12:   Send phase:
13:     if decision_i ≠ nil then
14:       ∀v, j : (⟨VOTE, h, e_i, v⟩, j) ∈ messagesSet_i, broadcast⟨VOTE, h, e_i, v⟩
15:       return
16:     if proposer(h, e_i) = i then
17:       broadcast ⟨PRE − PROPOSE, h, e_i, proposal_i, validEpoch_i⟩
18:   Delivery phase:
19:     set timerPrePropose to timeoutPrePropose
20:     while (timerPrePropose not expired) ∧ ¬(∃v_j, e_j : sentByProposer(h, e_i, v_j, e_j)) do
21:       if ∃v_j, e_j : sentByProposer(h, e_i, v_j, e_j) then
22:         v_i ← v_j                                             /* v_j is the value sent by the proposer */
23:         validEpoch_j ← e_j                                    /* e_j is the validEpoch sent by the proposer */
24:     if ¬(∃v, epochProp : sentByProposer(h, e_i, v, epochProp)) then
25:       timeoutPrePropose ← timeoutPrePropose + 1
26:   Compute phase:
27:     if 2f+1(⟨PROPOSE, h, validEpoch_j, v_i⟩) ∧ validEpoch_j ≥ lockedEpoch_i ∧ validEpoch_j < e_i ∧ isValid(v_i) then
28:       proposal_i ← v_i
29:     else
30:       if ¬isValid(v_i) ∨ (lockedEpoch_i > validEpoch_j ∧ lockedValue_i ≠ v_i) then
31:         proposal_i ← nil                                      /* Note that isValid(nil) is set to false */
32:       if isValid(v_i) ∧ (lockedEpoch_i = −1 ∨ lockedValue_i = v_i) then
33:         proposal_i ← v_i
```

chronous model in the presence of Byzantine committee members. To achieve consensus in this setting two additional variables need to be used: (i) $lockedEpoch_i$, which is an integer representing the last epoch where committee member i updated $lockedValue_i$, and (ii) $validEpoch_i$, which is an integer representing the last epoch where i updates $validValue_i$. These two new variables are used to not violate the agreement property during the asynchronous period. During such period, different epochs may overlap for different committee members, it is then needed to keep track of the relative epochs when a committee member locks such that it does not accept "outdated" information or information generated during a previous epoch. Moreover, round duration management mechanism needs to be introduced, *i.e.*, increasing timeouts. In the algorithm in a synchronous setting (Algorithm 2), rounds were lasting δ, the known message delay. In an eventually synchronous system, such an approach is not feasible, since, during the asynchronous period, messages may take an unbounded period of time before being delivered. It follows that, since there are at most f Byzantine committee members when a committee member delivers messages from $n - f$ different committee members it can terminate the delivery phase, but such phase may last an unbounded time. On the contrary, in the PRE-PROPOSE round, only the proposer is sending a message, and generally, messages may take a lot of time before being delivered, for such reasons timeouts need to be used to manage the duration of the rounds and adapted to message delays, such that once the system enters in the synchronous period, rounds last enough for messages sent during the current round to be delivered before the end of it.

The algorithm proceeds in 3 rounds for any given epoch e at height h. The description is mainly the same as in Section 4.5.1.1, thus in the following, we just underline the differences:

- Round PRE-PROPOSE (Lines 11 - 33 of Algorithm 3): The description of this round is

Algorithm 4 (part 2) Eventual Synchronous TendermintBFT for height h executed by i

```
1:  Round PROPOSE:
2:    Send phase:
3:      if proposal_i ≠ nil then
4:        broadcast ⟨PROPOSE, h, e_i, proposal_i⟩
5:      broadcast ⟨HeartBeat, PROPOSE, h, e_i⟩
6:    Delivery phase:
7:      set timerPropose to timeoutPropose
8:      while (timerPropose not expires) ∧ ¬2f+1(⟨HeartBeat, PROPOSE, h, e_i⟩) do{}   /* Note that the HeartBeat messages should be
        from different committee members */
9:      if ¬2f+1(⟨HeartBeat, PROPOSE, h, e_i⟩) then
10:       timeoutPropose ← timeoutPropose + 1
11:   Compute phase:
12:     if ∃v' : 2f+1(⟨PROPOSE, h, e_i, v'⟩) ∧ isValid(v') ∧ sentByProposer(h, e_i, v') then
13:       lockedValue_i ← v'
14:       lockedEpoch_i ← e_i
15:       validValue_i ← v'
16:       validEpoch_i ← e_i
17:       vote_i ← v'
18:     else
19:       vote_i ← nil

20: Round VOTE:
21:   Send phase:
22:     if vote_i ≠ nil then
23:       broadcast ⟨VOTE, h, e_i, vote_i⟩
24:     broadcast ⟨HeartBeat, VOTE, h, e_i⟩
25:   Delivery phase:
26:     set timerVote to timeoutVote
27:     while (timerVote not expires) ∧ ¬2f+1(⟨HeartBeat, VOTE, h, e_i⟩) do{}
28:     if ¬2f+1(⟨HeartBeat, VOTE, h, e_i⟩) then
29:       timeoutVote ← timeoutVote + 1
30:   Compute phase:
31:     if ∃v'' : 2f+1(⟨PROPOSE, h, e_i, v''⟩) ∧ isValid(v'') ∧ sentByProposer(h, e_i, v'') then
32:       validValue_i ← v''
33:       validEpoch_i ← e_i
34:     if ∃v_d, e_d : 2f+1(⟨VOTE, h, e_d, v_d⟩) ∧ isValid(v_d) ∧ decision_i = nil then
35:       decision_i ← v_d
36:     else
37:       e_i ← e_i + 1
38:       v_i ← nil
39:       if validValue_i ≠ nil then
40:         proposal_i ← validValue_i
41:       else
42:         proposal_i ← getValue()
```

mainly the same as before. We highlight the fact that an obedient committee member i also takes into account $lockedEpoch_i$ to accept a pre-proposed value.

- Round PROPOSE (Lines 1 - 19 of Algorithm 4): When an obedient committee member i updates $lockedValue_i$ (resp. $validValue_i$), it also updates $lockedEpoch_i$ (resp. $validEpoch_i$) to its current epoch.

- When an obedient committee member i updates $validValue_i$, it also updates $validEpoch_i$ to its current epoch.

We recall that each committee member has a timeout for each round. If during a round committee member i does not deliver at least $2f + 1$ messages sent during that round (or the pre-proposal for the PRE-PROPOSE round), the corresponding timeout is increased. Those messages can be values or heartbeats, in the case in which an obedient committee member has not a value to propose or vote.

4.5.1.3 Correctness of TendermintBFT in Eventual Synchronous Setting

In this section, we prove the correctness of TendermintBFT (Algorithm 3 & 4) in an eventually synchronous system.

Lemma 4.17 (Validity)**.** *In an eventually synchronous system, TendermintBFT verifies the following property: A decided value satisfies the predefined predicate denoted as* `isValid`().

Proof

The proof follows by construction. When an obedient committee member decides a value (Line 35 of Algorithm 4), it checks before if that value is valid (Line 34 of Algorithm 4). Therefore, an obedient committee member only decides a valid value. $\square_{Lemma\ 4.17}$

Lemma 4.18. *In an eventually synchronous system, TendermintBFT verifies the following property: An obedient committee member proposes and votes only once per epoch.*

Proof

We prove this lemma by construction. In Algorithm 4, an obedient committee member proposes (Line 4) and votes only once during the corresponding round (Line 23). At the end of the VOTE round, a committee member changes epoch (Line 37). Therefore, it cannot propose nor vote for that epoch any more. $\square_{Lemma\ 4.18}$

Lemma 4.19. *In an eventually synchronous system, TendermintBFT verifies the following property: At most one value can be proposed by at least $2f + 1$ committee members per epoch, and at most one value can be voted at least $2f + 1$ times by epoch.*

Proof

We prove this lemma by contradiction. Let v, v' such that $v \neq v'$. Since there are $3f + 1$ committee members in the system, if v or v' gets at least $2f + 1$ proposals (resp. votes), it means that at least $f + 1$ committee members propose (resp. vote) for both v and v'. By assumption, there are less than f Byzantine participants by committee, so at least 1 obedient committee member proposes (resp. votes) for both v and v'. That is a contradiction to Lemma 4.18. Therefore, two different values cannot be proposed (resp. voted) at least $2f + 1$ times during the same epoch. $\square_{Lemma\ 4.19}$

Lemma 4.20. *Let v be a value, e an epoch, and the set $L^{v,e} = \{i : i$ obedient $\wedge$ lockedValue$_i = v \wedge$ lockedEpoch$_i = e$ at the end of epoch $e\}$. In an eventually synchronous system, TendermintBFT verifies the following property: If $|L^{v,e}| \geq f + 1$ then no obedient committee member i will have lockedValue$_i \neq v \wedge$ lockedEpoch$_i \geq e$, at the end of each epoch $e' > e$; moreover, a committee member in $L^{v,e}$ only proposes v or nil for each epoch $e' > e$.*

Proof

Let v be a value, e an epoch, and $L^{v,e} = \{i : i$ obedient$\wedge$lockedValue$_i = v \wedge$lockedEpoch$_i = e$ at the end of epoch $e\}$, we assume that $|L^{v,e}| \geq f + 1$. We prove the theorem by induction:

- *Initialisation:* At the end of epoch e, by assumption, we have that $|L^{v,e}| \geq f+1$. There is an obedient committee member in that set, say i ($i \in L^{v,e}$). It means that i updates $lockedValue_i$ to v during epoch e, therefore i delivered $2f+1$ proposals for the value v (Lines 12 - 14 of Algorithm 4). By Lemma 4.19, at most one value can have at least $2f+1$ proposals during epoch e, and since v has at least $2f+1$ proposals, no obedient committee member j can update $lockedValue_j$ to a value $v' \neq v$ during epoch e. At the end of e, $lockedValue_j \neq v \vee lockedEpoch_j < e$.

- Induction: Let $a \geq 1$, we assume that $\forall i \in L^{v,e}$, $lockedValue_i = v$ at the end of each epoch between e and $e+a$, we also assume that if a value was proposed at least $2f+1$ times during these epochs it was either v or *nil*. We prove that at the end of epoch $e+a+1$, no obedient committee member j will have $lockedValue_j = v' \wedge lockedEpoch_j = e+a+1$ with $v' \neq v$.

  Let $i \in L^{v,e}$ such that i delivers a pre-proposal for v, then i will set $proposal_i$ to v; it will propose v since $lockedValue_i = v$ (Lines 27 - 33 of Algorithm 3 & Line 4 of Algorithm 4), in any other case, if i does not deliver a pre-proposal, or delivers a pre-proposal for a value $v' \neq v$, it will set $proposal_i$ to *nil* and will propose *nil* (Lines 27 - 33 of Algorithm 3 & Line 4 of Algorithm 4), since $\texttt{isValid}(nil) = \texttt{false}$ and by assumption, there is no $e' \in \{e, \ldots, e+a\}$ where there were at least $2f+1$ proposals for a value $v' \neq v$, and $lockedEpoch_i \geq e$. All committee members in $L^{v,e}$ will then propose v or *nil* during epoch $e+a+1$. By Lemma 4.18, obedient committee members only propose once per epoch, at least $f+1$ committee members (the ones in $L^{v,e}$) propose v or *nil*; since messages cannot be forged, the only values that can get at least $2f+1$ proposals for the epoch $e+a+1$ are v and *nil*. If an obedient committee member j delivers at least $2f+1$ proposals for v, it sets $lockedValue_j$ to v and $lockedEpoch_j$ to $e+a+1$ (Lines 12 - 14 of Algorithm 4); otherwise, it does not change $lockedValue_j$ nor $lockedEpoch_j$ (Line 19 of Algorithm 4). At the end of epoch $e+a+1$, there is no obedient committee member j such that $lockedValue_j \neq v \wedge lockedEpoch_j = e+a+1$. Moreover, committee members in $L^{v,e}$, only propose v or *nil* during epoch $e+a+1$.

We proved that if $|L^{v,e}| \geq f+1$, no obedient committee member i will have $lockedValue_i \neq v \wedge lockedEpoch_i \geq e$; moreover a committee member in $L^{v,e}$ only proposes v or *nil* for each epoch $e' > e$.

$\square_{Lemma\ 4.20}$

Lemma 4.21 (Agreement). *In an eventually synchronous system, TendermintBFT verifies the following property: If there is an obedient committee member that decides a value v, then eventually all the obedient committee members decide v.*

Proof

Let i be an obedient committee member. Without loss of generality, assume that i is the first obedient committee member that decides, and assume that it decides value v during epoch e. At time t where i decided, no other participant has decided, even those having a bigger epoch number. To decide, i delivered at least $2f+1$ votes for v for epoch e. Since there are less than f Byzantine committee members, and by Lemma 4.18 obedient committee members can only vote once per epoch, so at least $f+1$ obedient committee members voted for v during epoch e, so we have $|L^{v,e}| = |\{j : j \text{ obedient} \wedge lockedValue_j = v \wedge lockedEpoch_j =$

e at the end of epoch $e\}| \geq f + 1$. By Lemma 4.20 committee members in $L^{v,e}$ only propose v or *nil* during each epoch after e, and no obedient committee member j will have $lockedValue_i \neq v \wedge lockedEpoch_i \geq e$. Thanks to the broadcast guarantees, all obedient committee members will eventually deliver the $2f + 1$ votes for v from epoch e; since when an obedient committee member decides, it sends back all votes it delivered than makes it decided (Line 13 of Algorithm 3).

If an obedient committee member j does not decide before delivering these votes, when eventually it delivers them, it will decide v (Lines 34 - 35 of Algorithm 4). Otherwise, it means that j decides before delivering the votes from epoch e.

By contradiction, we assume that j decides a value $v' \neq v$ during an epoch $e' > e$, so j delivered at least $2f + 1$ votes for v' during epoch e' (Lines 34 - 35 of Algorithm 4). Since an obedient committee member only votes once by Lemma 4.18, there are less than f Byzantine committee members and the messages are unforgeable, at least $f + 1$ obedient committee members voted for v' during epoch e'. An obedient committee member votes a non-*nil* value if that value was proposed at least $2f + 1$ times during the current epoch (Lines 12 - 23 of Algorithm 4). By Lemma 4.18 an obedient committee member only proposes once, there are less than f Byzantine committee members and the messages are unforgeable, so at least $f + 1$ obedient committee members proposed v' during e'. Since $e' > e$ and $|L^{v,e}| \geq f + 1$, by Lemma 4.20 there are at least $f + 1$ committee members that proposed v or *nil* during epoch e'. Even if all the $2f$ committee members remaining proposes v', there cannot be $2f + 1$ proposals for v', which is a contradiction. So j cannot decide $v' \neq v$ after epoch e and we assume that e is the first epoch where an obedient committee member decides. $\square_{Lemma\ 4.21}$

Lemma 4.22. *In an eventually synchronous system, if there is an epoch after which when an obedient committee member broadcasts a message during a round, it is delivered by all obedient committee members during the same round, TendermintBFT verifies the following property: If an obedient committee member i updates $lockedValue_i$ to a value v during epoch e, then at the end of the epoch e, all obedient committee members have $validValue = v$ and $validEpoch = e$.*

Proof

We prove this lemma by construction.

Let e be the epoch after which when an obedient committee member broadcasts a message during a round r, it is delivered by all obedient committee members during the same round r. Let i be an obedient committee member, we assume that at the end of epoch $e' \geq e$, i has $lockedValue_i = v$ and $lockedEpoch_i = e'$, it means that i delivered at least $2f + 1$ proposals for v during epoch e' (Lines 12 - 14 of Algorithm 4). Thanks to the reliable broadcast guarantees, and since all messages are propagated, all obedient committee members will deliver these proposals for v, in the worst-case in the VOTE round. Let j be an obedient committee member since j will deliver at least $2f + 1$ proposals for v and epoch e' during the VOTE round, it will set $validValue_j = v$ and $validEpoch_j = e'$ (Lines 31 - 33 of Algorithm 4). $\square_{Lemma\ 4.22}$

Lemma 4.23 (Termination)**.** *In an eventually synchronous system, TendermintBFT verifies the following property: Every obedient committee member eventually decides some value.*

Proof

By construction, if an obedient committee member does not deliver more than $2f + 1$ messages (or 1 from the proposer in the PRE-PROPOSE round) from different committee members during the corresponding round, it increases the duration of its round, so eventually during the synchronous period of the system all the obedient committee members will deliver the pre-proposal, proposals and votes from obedient committee members respectively during the PRE-PROPOSE, PROPOSE and the VOTE round; and messages delivered by obedient an obedient committee member will be delivered by the others at most in the following round. Let e be the first epoch after that time.

If an obedient committee member decides before e, by Lemma 4.21 all obedient committee members eventually decide, which ends the proof. Otherwise, at the beginning of epoch e, no obedient committee member decides yet. Let i be the proposer of epoch e. First, we assume that i is obedient and pre-propose v; v is valid since getValue() always return a valid value (Line 6 of Algorithm 3 & Line 42 of Algorithm 4), and $validValue_i$ is always valid (Lines 12 & 31 of Algorithm 4). We have 2 cases:

- Case 1: At the beginning of epoch e, $|\{j : j \text{ obedient} \wedge (lockedEpoch_j \leq validEpoch_i \vee lockedValue_j = v)\}| \geq 2f + 1$.

  Let j be an obedient committee member where the condition $lockedEpoch_j \leq validEpoch_i \vee lockedValue_j = v$ holds. After the delivery of the pre-proposal v from i, j will update $proposal_j$ to v (Lines 27 - 33 of Algorithm 3). During the PROPOSE round, j proposes v (Line 4 of Algorithm 4), since there are at least $2f + 1$ similar obedient committee members, they will all propose v, and all obedient committee members will deliver at least $2f + 1$ proposals for v (Line 7 of Algorithm 4).

  Obedient committee members will set their variable *vote* to v (Lines 12 - 4 of Algorithm 4), then will vote v, and they will deliver all the votes (at least $2f + 1$) from this epoch (Lines 23 & 25 of Algorithm 4). Since we assume that no obedient committee members decided yet, and since they each deliver at least $2f + 1$ votes for v, they will decide v (Lines 34 - 35 of Algorithm 4).

- Case 2: At the beginning of epoch e, $|\{j : j \text{ obedient} \wedge (lockedEpoch_j \leq validEpoch_i \vee lockedValue_j = v)\}| < 2f + 1$.

  Let j be an obedient committee member where the condition $lockedEpoch_j > validEpoch_i \wedge lockedValue_j \neq v$ holds. When i will make the pre-proposal, j will set $proposal_j$ to *nil* (Line 31 of Algorithm 3) and will propose *nil* (Line 4 of Algorithm 4).

  By counting only the proposed values of the obedient committee members, no value will have at least $2f + 1$ proposals for v. There are two cases:

  - No obedient committee member delivers at least $2f + 1$ proposals for v during the PROPOSE round, so they will all set their variable *vote* to *nil*, then they will vote *nil* and go to the next epoch without changing their state (Lines 19 & 23 - 25 & 36 - 42 of Algorithm 4).
  - If some obedient committee members delivers at least $2f+1$ proposals for v during the PROPOSE round, *i.e.*, some Byzantine committee members send proposals for v to those committee members.

    As in the previous case, they will vote for v, and since there are $2f + 1$ of them, all obedient committee members will decide v. Otherwise, there are less than $2f + 1$ obedient committee members that deliver at least $2f + 1$ proposals for v. Only them will vote for v (Line 23 of Algorithm 4). Without Byzantine committee

members, there will be less than $2f+1$ votes for v, no obedient committee member will decide (Lines 34 - 35 of Algorithm 4) and they will go to the next epoch; if Byzantine committee members send votes for v to an obedient committee member such that it delivers at least $2f + 1$ votes for v during VOTE round, then the obedient committee member will decide (Lines 34 - 35 of Algorithm 4), and by Lemma 4.21 all obedient committee members will eventually decide.

Let k be one of the obedient committee members that delivers at least $2f + 1$ proposals for v during the PROPOSE round, it means that $lockedValue_k = v$ and $lockedEpoch_k = e$. It follows by Lemma 4.22 that at the end of epoch e, all obedient committee members will have $validValue = v$ and $validEpoch = e$.

If there is no decision, either no obedient committee member changes its state, or all obedient committee members change their state and have the same *validValue* and *validEpoch*; therefore, eventually a proposer of an epoch will satisfy Case 1, and that ends the proof.

If i, the proposer of epoch e, is Byzantine and more than $2f + 1$ obedient committee members delivered the same message during PRE-PROPOSE round, and the pre-proposal is valid, the situation is like i was obedient. Otherwise, there are not enough obedient committee members that delivered the pre-proposal, or if the pre-proposal is not valid, then there will be less than $2f + 1$ obedient committee members that will propose that value, which is similar to the case 2.

Since proposers are selected in a round-robin fashion, an obedient committee member will eventually be the proposer, and obedient committee members will decide. $\square_{Lemma\ 4.23}$

Theorem 4.24. *In an eventually synchronous system, TendermintBFT implements the consensus specification.*

Proof

The proof follows directly from Lemmas 4.17, 4.21 and 4.23. By Lemma 4.17, we show that the TendermintBFT satisfies Validity, by Lemma 4.21, we show that the TendermintBFT satisfies Agreement, and by Lemma 4.23, we show that the TendermintBFT satisfies Termination. $\square_{Theorem\ 4.24}$

4.5.1.4 TendermintBFT Message Complexity

Let us consider the following scenario after the asynchronous period (*i.e.*, after τ), in which in the first f epochs, $e_{i+1}, \ldots, e_{i+f}$, there are f Byzantine proposers that cause f obedient committee members (one for each epoch $e_{i+1}, \ldots, e_{i+f}$) to lock on f distinct values. Let j be the last obedient committee member that locked, and let v be its locked value ($lockedValue_j = v$ with $lockedEpoch_j = e_{i+f}$); all the other obedient committee members have their *validValue* set to v and *validEpoch* set to e_{i+f}. This happens thanks to the fact that when an obedient committee member locks on a value, at the end of the epoch every obedient committee member sets its *validValue* to that value. The algorithm terminates when a pre-proposal is proposed and voted by more than $2f + 1$ committee members, *i.e.*, when the pre-proposed value has *validEpoch* greater or equal than the committee member *lockedEpoch*. Thus, during the synchronous period, the first obedient proposer that proposes will lead the algorithm to terminate

in $f + 1$ rounds. Let us consider the case in which there are f obedient committee members locked on f different values with different *lockedEpoch* before τ. Let us assume that j is the last obedient committee member that locked on a value v, thus it has the highest *lockedEpoch* but not all the obedient committee members necessarily have their *validValue* set to v (due to the asynchronous communication). Let us now consider that after τ the first f proposers are Byzantines and stay silent. The following proposers are obedient but their pre-propose value might not be accepted by enough obedient committee members as long as j, with the highest *validEpoch* and *lockedEpoch* proposes. That eventually happens due to the round-robin selection function. Thus, the protocol terminates in a number of epochs proportional to the number of committee members $O(n)$, while the lower bound to solve Byzantine consensus in the worst-case scenario is $f + 1$ [76].

As for message complexity, since at each epoch, all committee members broadcast messages, it follows that during one epoch the protocol uses $O(n^2)$ messages, thus in the worst-case scenario, the message complexity is $O(n^3)$. This improved the $O(n^4)$ message complexity of PBFT [45]. In PBFT, if the leader is obedient, the complexity boils down to $O(n^2)$. Otherwise, a *view-change* mechanism takes place to change the leader and resume the computation. The view-change is used to avoid that, in case of a faulty leader, if some obedient participant decides on a value v, the other obedient participants cannot decide on a value $v' \neq v$ when the new leader proposes a new value v'. Such a mechanism implies that when a leader is suspected to be faulty, all participants have to collect enough evidence for the view-change. That is, the view-change message contains at least $2f + 1$ signed messages and these messages are sent from at least $2f+1$ participants, which yields a message complexity of $O(n^2)$. These messages are then sent to all participants, therefore, the view-change has $O(n^3)$ message complexity. Since the protocol terminates when there is an obedient leader, which may happen for the first time in epoch $f + 1$, then in the worst-case scenario it has a message complexity of $O(n^4)$. Contrary to the classical view-changed based approaches such as PBFT, instead of exchanging all the messages they already delivered, in TendermintBFT and subsequent proposals like Hot-Stuff, participants in the consensus locally keep track of potentially decided values to preserve the safety, and so reduce the message complexity, hence they avoid the view-change phases, but need some locking mechanisms. That has been called *Linear View Change* by the authors of HotStuff.

In the following, we address the bit complexity of TendermintBFT. As explained in Section 3.1.3, and for simplicity, we can view the bit complexity of a protocol is the number of messages each participant stores while running the protocol. In TendermintBFT, each message is composed as follows:

- PRE-PROPOSE: The marker that the message is from the round PRE-PROPOSE; two integers, one for the current height, and the second for the current epoch; the proposed value, and an integer representing the epoch on which the proposer last updated its *validValue*.
- PROPOSE: The marker that the message is from the round PROPOSE; two integers representing the current height and the current epoch; and a value that represents the proposed block.
- VOTE: The marker that the message is from the round VOTE; two integers representing the current height and the current epoch; and a value that represents the voted block.
- HeartBeat: The marker that the HeartBeat is from the round VOTE or PROPOSE; two integers representing the current height and the current epoch.

Function repeatedConsensus(Π); %Repeated consensus for the set Π of participants%

Init:
(1) $h \leftarrow 1$ %*Height*%; $B \leftarrow \perp$; $C \leftarrow \perp$ %Set of committee members%;
(2) $commitsReceived_i^h \leftarrow \emptyset$; $toReward_i^h \leftarrow \emptyset$; $TimeOutCommit \leftarrow \Delta_{\text{Commit}}$;

while (true) **do**
(3) $B \leftarrow \perp$;
(4) $C \leftarrow$ committeeMembers(h); %Application and blockchain dependant%
(5) **if** ($i \in V$) **then**
(6) $B \leftarrow$ consensus($h, V, toReward_i^{h-1}$); %Consensus function for the height h%
(7) **trigger** broadcast $\langle \text{COMMIT}, (B, h)_i \rangle$;
(8) **else**
(9) **wait until** ($\exists B' : |\text{MoreThanThird}(B', commitsReceived_i^h)|$);
(10) $B \leftarrow B'$;
(11) **endif**
(12) set $timerCommit$ to $TimeOutCommit$;
(13) **wait until**($timerCommit$ expired);
(14) **trigger** decide(B);
(15) $h \leftarrow h + 1$;
endwhile

upon event delivery $\langle \text{COMMIT}, (B', h')_j \rangle$:
(16) **if** $(((B', h')_j \notin commitsReceived_i^{h'}) \wedge (j \in \text{committeeMembers}(h')))$ **then**
(17) $commitsReceived_i^{h'} \leftarrow commitsReceived_i^{h'} \cup (B', h')_j$;
(18) $toReward_i^{h'} \leftarrow toReward_i^{h'} \cup j$;
(19) **trigger** broadcast $\langle \text{COMMIT}, (B', h')_j \rangle$;
(20) **endif**

Figure 4.4: Tendermint Repeated Consensus algorithm for Obedient Participant i.

An obedient committee member keeps in memory, for each epoch for a given height, one message for each type (PROPOSE, VOTE), and at most 2 messages of type HeartBeat from each committee member, and only one PRE-PROPOSE message. An obedient committee member may have at most 1 message from PRE-PROPOSE, n messages from PROPOSE, n messages from VOTE, and $2n$ messages of type HeartBeat. Hence, for each epoch at any given height, a committee member stores at most $4n + 1$ messages of size $O(\log n)$. In the worst-case, for the whole execution, a committee member may store $O(n^2)$ messages. Therefore, the worst-case bit complexity is $O(n^2 \log n)$. A summary of the complexity can be found in Table 3.1.

4.5.2 Tendermint Repeated Consensus Algorithm

We now present the Tendermint algorithm for repeated consensus.

Detailed description of the algorithm. We describe in Figure 4.4 the Tendermint algorithm for the repeated consensus (Definition 4.2). For a participant i, the algorithm proceeds as follows:

- i computes the set of committee members for the current height;
- If i is a committee member, then it calls the consensus function solving the consensus for the current height, then broadcasts the decision, and sets B to that decision;
- Otherwise, if i is not a committee member, it waits for at least $n/3$, or equivalently $f+1$, commits from the same block and sets B to that block;
- In any case, it sets the timer to *TimeOutCommit* to collect more commits and lets it expire. Then i decides B and goes to the next height.

Whenever i delivers a commit, it broadcasts it (Lines 16 - 20 of Figure 4.4).

Data structures. The integer h represents the current height of the participant. C is the current set of committee members. B is the variable that will be set to the block to be appended. $commitsReceived_i^h$ is the set containing all the commits i delivered for the height h. $toReward_i^h$ is the set containing the committee members from which i delivered commits for the height h. *TimeOutCommit* represents the time a committee member has for collecting commits after an instance of consensus. *TimeOutCommit* is set to the default value Δ_{Commit}.

Functions. Let $Height = \mathbb{N}^*$ be the set of all heights, let *Commits* be a set of all possible commits, and let $\mathbb{B}$ be the set containing all possible blocks. We also recall that Π_ρ is the set of participants in the system run.

- committeeMembers : $\Pi \times Height \to 2^{\Pi_\rho}$ is an application dependent and deterministic selection function which gives the set of committee members for a given height with respect to the blockchain history. We have $\forall h \in Height, |\text{committeeMembers}(h)| = n$.
- consensus : $Height \times 2^{\Pi_\rho} \times 2^{Commits} \to \mathbb{B}$ is a consensus algorithm. It outputs for the participant the decision of the consensus (Definition 4.1) among the committee members.
- MoreThanThird : $\mathbb{B} \times 2^{Commits} \to \{\texttt{false}, \texttt{true}\}$ is a predicate which checks if there are commits from at least $f + 1$ different committee members for a given block in the given set.
- $\texttt{isValid} : \mathbb{B} \to \{\texttt{false}, \texttt{true}\}$ is an application-dependent predicate that is satisfied if the given block is valid. If there is a block B such that $\texttt{isValid}(B) = \texttt{true}$, we say that B is valid. We note that for any non-block, we set isValid to false, (*i.e.*, $\texttt{isValid}(\perp) = \texttt{false}$), the validity of the block depends on the blockchain and the application, and isValid is known by all participants.

Note that in Tendermint, the reward for the height h is allocated during the height $h + 1$, and to a subset of committee members who committed the block for h (Line 18 of Figure 4.4).

4.5.2.1 Correctness of Tendermint Repeated Consensus

In this section, we prove the correctness of the Tendermint algorithm for repeated consensus, when assuming that the consensus algorithm used (Line 6 of Figure 4.4) is correct (Definition 4.1). In the proofs here, the lines mentioned refer to the lines in the algorithm presented in Figure 4.4.

Lemma 4.25 (brc-Termination)**.** *In an eventually synchronous system, and assuming that the consensus function is correct, the Tendermint Repeated Consensus algorithm verifies the following property: Every obedient participant has an infinite output.*

Proof

By contradiction, let i be an obedient participant and we assume that i has a finite output. Two scenarios are possible, either i cannot go to a new height, or from a certain height h it outputs only $\perp$.

- If i cannot progress, one of the following cases is satisfied:

  - The function consensus does not terminate (Line 6), which is a contradiction since it violates the Termination property of the consensus (Definition 4.1).
  - i waits an infinite time for receiving enough commits (Line 9), which cannot be the case thanks to the broadcast guarantees and the eventual synchronous assumption, all the obedient committee members terminate the function consensus and broadcast their commit.

- If i does not decide at each height (Line 14), it means that from a given height, i only outputs $\perp$. Let height h be the first such height, and let $h' > h$; it means (i) either i is a committee member for h' and the function consensus returns $\perp$ (Lines 5 & 6), or (ii) i is not a committee member for h' but delivered at least $f + 1$ commits for $\perp$ (Lines 9 & 19).

  (i) If consensus returns the value $\perp$, it means by the Validity property (Definition 4.1) of the consensus that $\texttt{isValid}(\perp) = \texttt{true}$, which is a contradiction with the definition of the function `isValid`.

  (ii) Only committee members commit, and each of them broadcasts its commit (Lines 5 - 7), and $f < n/3$. Since non-committee members collect at least $n/3$ commits, it means that i delivered a commit from at least one obedient committee member. By the Validity property of the consensus (Definition 4.1), obedient participants only decide/commit on valid value, and $\perp$ is not valid, which is a contradiction.

Therefore, if i is an obedient participant, then it has an infinite output. $\square_{Lemma\ 4.25}$

Lemma 4.26 (brc-Agreement)**.** *In an eventually synchronous system, and assuming that the consensus function is correct, Tendermint Repeated Consensus Algorithm verifies the following property: If the h^{th} value of the output of an obedient participant is B, then B is the h^{th} value of the output of any other obedient participant.*

Proof

We prove this lemma by construction. Let i and j be two obedient participants, and let h be a height. Two cases are possible:

- i and j are committee members for the height h, so both calls the function consensus (Lines 5 & 6). By Agreement property of the consensus (Definition 4.1), i and j decide the same value and then output that same value (Line 14).

- At least one of i and j is not a committee member for the height h. Without loss of generality, we assume that i is not a committee member for the height h. Since all the obedient committee members commit the same value, let say B, thanks to the Agreement property of the consensus (Definition 4.1), and since they broadcast their commit (Line 7), eventually there will be more than $2f + 1$ commits for B. So no other value $B' \neq B$ can be present at least $f + 1$ times in the set $commitReceived_i^h$. Therefore, i outputs the same value B as obedient committee members (Line 9). If j is a committee member, that ends the proof. If j is not a committee member, then by the same argument, j outputs the same value B. Hence, both i and j output the same value B for height h.

$\square_{Lemma\ 4.26}$

Lemma 4.27 (brc-Validity). *In an eventually synchronous system, and assuming that the consensus function is correct, Tendermint Repeated Consensus Algorithm verifies the following property: Each value in the output of any obedient participant is valid; it satisfies the predefined predicate denoted* `isValid`.

Proof

We prove this lemma by construction. Let i be an obedient participant, and assume that the h^{th} value of the output of i is B. If i decides a value (Line 14), then that value has been set during the execution and for that height (Line 3).

- If i is a committee member for the height h, then B is the value returned by the function consensus. By the Validity property of the consensus (Definition 4.1), we have $\texttt{isValid}(B) = \texttt{true}$.
- Let h be a height, and B be the h^{th} value of the output of a participant j. If j is not a committee member for the height h, it means that it delivered more than $f + 1$ signed commits from the committee members for the value B (Lines 5 - 7 and 16 - 20), hence at least one obedient committee member committed B, and by the Validity property of the consensus (Definition 4.1), we have $\texttt{isValid}(B) = \texttt{true}$.

Each value in the output of an obedient participant satisfies the predicate `isValid`. $\square_{Lemma\ 4.27}$

Theorem 4.28. *In an eventually synchronous system, and assuming that the consensus function is correct, the Tendermint Repeated Consensus algorithm implements the Repeated Consensus.*

Proof

Assuming that the function consensus is correct, the proof follows directly from Lemmas 4.25, 4.26 and 4.27. By Lemma 4.25, we show that the Tendermint Repeated Consensus algorithm satisfies brc-Termination, by Lemma 4.26, we show that the Tendermint Repeated Consensus algorithm satisfies brc-Agreement, and by Lemma 4.27, we show that the Tendermint Repeated Consensus algorithm satisfies brc-Validity. $\square_{Theorem\ 4.28}$

4.5.3 Fairness of Tendermint

The committee members' selection of Tendermint is part of a separate module. The selection mechanism is today left configurable by the application; therefore, in the following, we do not address this part. The rewarding mechanism, on the other hand, referred as the *Tendermint's reward mechanism* is part of the Tendermint protocol (Lines 13 and 16 - 20 of Figure 4.4).

Recall that in Tendermint, the reward for the height h is allocated during the height $h + 1$, and to a subset of committee members who committed the block for h (Line 18 of Figure 4.4). In more details, Tendermint's reward mechanism works as follows:

- Once a new block is decided for height h, the participants wait for a default duration of *TimeOutCommit* to collect the decision from the other committee members for h, and put them in their set *toReward* (Lines 13 and 16 - 20 of Figure 4.4).

- During the consensus for height h, let us assume that the committee member i proposes the block that will be decided in the consensus. i proposes to reward participants in its set *toReward* (Line 6 of Figure 4.4). Therefore, only participants from which i delivered a commit will get a reward for the block at height $h - 1$.

Note that in Tendermint, the duration of *TimeOutCommit* is fixed, and is not updated at all during the execution.

Lemma 4.29. *In an eventually synchronous system, the reward mechanism of Tendermint is not eventually fair.*

Proof

We assume that the system becomes synchronous and that $TimeOutCommit < \Delta$, where Δ is the maximum message delay in the network. For any height h, let i be a committee member for the height $h - 1$ and j the committee member whose proposal get decided for the height h. It may happen that j did not receive the commit from i before proposing its block. Hence, when the block is decided, i does not get a reward for its effort, which contradicts Condition *f2* (h-completeness) of the reward mechanism fairness. Tendermint's reward mechanism is not eventually fair. $\square_{Lemma\ 4.29}$

Let us observe that to make Tendermint's reward mechanism at least eventually fair (Definition 4.9) it is necessary to increase *TimeOutCommit* for each round until it catches up the message delay. We refer to this variant as the *Tendermint's reward mechanism with modulable timeouts*. Moreover, the commit message should contain enough information to keep track of the participation of the participants in each phase, *i.e.*, to distinguish Byzantine from obedient participants.

Proposition 4.30. *In an eventually synchronous system, when the commit messages are sufficient to detect Byzantine participants, Tendermint's reward mechanism with modulable timeouts is eventually fair.*

Proof

We change the reward mechanism in Tendermint as follows:

- Once a new block is decided, say for height h, participants wait for at most *TimeOutCommit* to collect the decision from the other committee members for that height, and put them in their set *toReward*.
- If a participant did not get the commits from all the committee members for that height before the expiration of the timeout, it increases the timeout for the next height.
- During the consensus at height h, let us assume that i proposes the block that will be decided in the consensus. i gives the reward to the participants in its *toReward*.

In this reward mechanism, *TimeOutCommit* is increased whenever a participant does not have the time to collect all the commits for the previous round. We prove that this reward mechanism is eventually fair.

There is a point in time t from when the system will become synchronous, and all the commits will be delivered by obedient participants before the next height. From the time t, at height h all obedient participants know the exact set of committee members that committed

the block from $h-1$ the previous height, and from those commit messages, they can exclude the set of participants that were not $(h-1)$-obedient. They give to the $(h-1)$-obedient committee members a reward parameter equal to 1, and to the non $(h-1)$-obedient a reward of 0. The committee members in h give the reward to the obedient committee members that committed and which are the only one with a reward parameter greater than 0 for $h-1$, which satisfy the fairness Conditions *f1*, *f2*, and *f3*, so the reward mechanism presented is eventually fair. □*Proposition 4.30*

4.6 Conclusion

We discuss committee-based blockchains. In committee-based blockchains, committees are in charge of deciding the next block to be added to the chain. The participants know who are the committee members, and inside a committee, the committee members run an agreement procedure such that exactly one block is produced. Committee-based blockchain can be used to avoid forks if they implement a deterministic consensus. In this chapter, we have 3 main contributions.

First, we define the problem committee-based blockchains tackle and try to solve: the *Byzantine repeated consensus*. Our definition is a generalisation of the repeated consensus where only crash failures can occur, and we describe how consensus can be used to build a blockchain in an open setting, providing guarantees of the absence of forks.

Given that, in blockchains, participants creating blocks are rewarded for their effort, we discuss the distribution of rewards. In particular, we are interested in whether participants can fairly be rewarded with respect to the effort they are putting in the system. The second main contribution of this chapter is a definition of *fairness* for committee-based blockchains. Once defined, we analyse the importance of the communication model (in particular the time assumption) for fairness.

Lastly, we study *Tendermint*, one of the most used committee-based blockchain systems against the two abstractions, *i.e.*, (i) the repeated consensus for proving the correctness of Tendermint, and (ii) the fairness. In this thesis, we formalise the Tendermint's algorithms for the first time ever. By our formalisation of the protocol, we exhibited bugs in the earlier version of Tendermint leading to its correction. Tendermint shows efficiency compared to classical consensus algorithms such as PBFT. Our analysis of the fairness of Tendermint shows that the current version of Tendermint is not fair; however, a small twist can make it fair. This allows to show that attention should be put on the design of the blockchain protocols.

The question of fairness of blockchain systems highlight another particular question: participants may want to profit from the system and gain more than their effort. In Chapters 5 & 6, we study the behaviours of participants that aim to profit from the system, and we analyse whether the consensus properties are still guaranteed in their presence.

CHAPTER 5

STRATEGIC BEHAVIOURS IN COMMITTEE-BASED BLOCKCHAINS

Contents

Blockchain systems usually have some economical or financial advantages. Hence, participants may try to maximise their profit in the system. Those participants do not necessarily want to harm the system; they often want to stay in the system but gain the most from it. Many articles studied the strategic behaviours of participants in blockchain systems, focusing mainly on proof-of-work Bitcoin-like systems (*e.g.*, [31, 74]). To the best of our knowledge, very few works have been dedicated to analyse or discuss the strategic behaviours in committee-based blockchains, exceptions to be noted are [4] which introduced interesting incentive mechanisms but did not provide a formal framework for their analysis, and [78] that specifically study some behaviours in Algorand. [78] is however too specific, and the participants have few actions. The results presented in this chapter present a more refined and general setting.

In this chapter, we study the strategic behaviours in committee-based blockchains, specifically against the consensus properties. We focus on one committee and study the behaviour inside that committee. We consider a simplified consensus algorithm based on existing or proposed blockchains such as Algorand [85], HotStuff [160], Tendermint [39], *etc.*, which encapsulates the main actions of the participants: sending a message and checking validity. Knowing that those actions have costs, and achieving the consensus gives rewards to the participants, we study using game theory how strategic participants behave while trying to maximise their gains.

5.1 Model

In this section, the reader may refer to Chapter 2 for more in-depth details about the model.

5.1.1 System Model

We consider a system composed of a finite and ordered set Π of n participants, a *committee*, of synchronous sequential participants, namely $\Pi = \{1, \ldots, n\}$, where participant i is said to have index i. Except if stated otherwise, in this chapter, all participants are in the set Π.

Communication. Participants communicate by sending and receiving messages through a *synchronous network*. We assume that each participant proceeds in rounds. A *round* consists of three sequential phases, in order: the send, the delivery, and the compute phases. The delivery phase has a fixed duration that allows collecting all the messages sent by the participants. At the end of a round, a participant exits from the current round and starts the next one. We assume the existence of a *reliable broadcast* primitive. Messages are created with a digital signature, and we assume that digital signatures cannot be forged. When a participant i delivers a message, it knows the participant j that created the message.

Participants Behaviour. In this chapter, we consider that participants are *strategic* (Definition 2.7). Recall that they are a subclass of *rational* participants (Definition 2.6).

Strategic participants are self-interested and their objective is to maximise their expected gain. They will deviate from a prescribed protocol if and only if doing so increases their expected gain. They differ from obedient participants (Definition 2.4) who always follow the prescribed protocol.

We also consider trembling participants. With low probability, an external function can return an unexpected value. They do not want such value, but are not in control of that; therefore, they are not aware when the returning value is "normal" or not. They only know the probability of such an event happening. A trembling participant is also a strategic participant.

We assume that the behaviour of participants is perceived identically by all other participants, that is, a message sent by a participant in a given round is received by all in the delivery phase of that round.

5.1.2 Consensus in Presence of Strategic Participants

As discussed and studied in Chapter 4, committee-based blockchains can be developed using a consensus algorithm. In particular, at each height, the protocol used by the corresponding committee must implement the consensus. In this section, we adapt the definition of the consensus properties in the presence of strategic participants.

We say that a protocol is a consensus algorithm in the presence of strategic participants if the following properties hold:

- *s-Termination:* every strategic participant decides on a value (a block);
- *s-Agreement:* if two strategic participants decide respectively on values B and B', then $B = B'$;
- *s-Validity*: a decided value by any strategic participant is valid; it satisfies the predefined predicate.

If clear from the context, we simply write Termination, Agreement and Validity instead of respectively s-Termination, s-Agreement and s-Validity.

Problem. In this chapter, we study the behaviour of strategic participants in a consensus protocol. The goal is to know whether consensus is possible in committee-based blockchains in the presence of strategic participants.

In the study, we use the notion of Nash equilibrium, which is intuitively a "stable" situation where no participant has an incentive to unilaterally deviate. Nash equilibrium is formally defined in Section 2.4.4 of Chapter 2.

Formally, the question we answer in this chapter is: *What are the different Nash equilibria and do they satisfy the consensus properties?* It is important to note that our goal is not to propose a protocol such that all strategic participants behave as obedient, but rather to study the behaviour of strategic participants in a consensus algorithm under different reward mechanisms.

5.1.3 Protocol Studied

Generally, in committee-based blockchains, the protocol to reach the consensus for one block at a given height is as follows: a proposer proposes a block, and the other members of the committee will check the validity of the block. If the block is valid, the committee members will vote for it and will announce their vote through a message to the other members; if the block is invalid, they will not vote for it. Votes are collected and, if a given threshold is reached, then the block is decided (produced), otherwise, a new proposer will propose another block and the procedure restarts.

As explained above, these two phases encapsulate the main and important ideas of consensus protocol for committee-based blockchains. Moreover, Chan and Shi in [46], extended this two phases approach (Propose and Vote) to present multiple algorithms for different communication and failure models; pointing out the importance and sufficiency of these phases in consensus algorithms for blockchains.

In the following, we first present in details the prescribed protocol. Most use a similar skeleton.

The Prescribed Protocol. The protocol proceeds in rounds. For the sake of simplicity, we consider the height h of the blockchain passed as a parameter to the protocol. Algorithm 5 presents the pseudo-code of the protocol.

For each round t a committee member is designated as the proposer for the round in a round-robin fashion. The `isProposer`(t, h) function returns the identifier of the proposer for the current round (Line 7 of Algorithm 5), the function, by taking as parameter the current height, deterministically selects the proposer based on the information contained in the blockchain up to h (discussions about selection mechanisms can be found in Section 4.3.2 of Chapter 4). Each round is further divided into two sub-rounds: the PROPOSE and the VOTE rounds.

While in PROPOSE, the proposer of the round uses the function `createValidValue`(h) to generate a block. Because a valid block must include the identifier of the h^{th} block in the blockchain, the height h is passed as parameter (Line 8 of Algorithm 5). Once the block is created, a message broadcasting the proposal is sent (Line 9 of Algorithm 5). At Line 11, the proposal is received through a delivery function. Each participant checks if the proposal is valid (Line 13 of Algorithm 5). If so, the participant sets its vote to that value (Line 14 of Algorithm 5).

Algorithm 5 Prescribed protocol for a participant i at a given height h

```
1:  Initialisation:
2:    vote := nil
3:    t := 0                                                    /* Current round number */
4:    decidedValue := nil

5:  Round PROPOSE(t):
6:    Send phase:
7:      if i == isProposer(t, h) then
8:        proposal ← createValidValue(h)      /* The proposer of the round generates a block, i.e. the value to be proposed */
9:        broadcast ⟨PROPOSE, h, t, proposal⟩
10:   Delivery phase:
11:     delivery ⟨PROPOSE, h, t, v⟩ from proposer(h, t)          /* The participant collects the proposal */
12:   Compute phase:
13:     if isValid(v) then
14:       vote ← v                  /* If the delivered proposal is valid, then the participant sets a vote for it */

15: Round VOTE(t):
16:   Send phase:
17:     if vote ≠ nil then
18:       broadcast ⟨VOTE, h, t, vote⟩      /* If the proposal is valid, the participant sends the vote to the others */
19:   Delivery phase:
20:     delivery ⟨VOTE, h, t, v⟩            /* The participant collects all the votes for the current height and round */
21:   Compute phase:
22:     if |⟨VOTE, h, t, v⟩| ≥ ν ∧ decidedValue = nil ∧ vote ≠ nil ∧ vote = v then
23:       decidedValue ← v; exit            /* The valid value is decided if the threshold is reached */
24:     else
25:       vote ← nil
26:       t ← t + 1
```

While in VOTE, any participant that sets its vote to the current valid proposal sends a message (of type vote) to the other members of the committee (Line 18 of Algorithm 5). During the delivery phase, every participant collects sent messages. During the compute phase, each participant verifies if a quorum of ν votes for the current proposal has been reached. Let us note that ν, the majority threshold, is a parameter here because it is the object of our study to establish the quorum ν in presence of the participants. If the quorum is reached, the participant voted for the value and did not already decide for the current height, then it decides for the current proposal (Line 23 of Algorithm 5) and the protocol ends; in that case, we say that the block (or the proposal) is produced. If the quorum is not reached, then a new round starts (Line 26 of Algorithm 5).

Remark 5.1. *Let us note that the protocol in an environment assuming only obedient and Byzantine participants trivially implements consensus if f, the number of Byzantine participants, is such that $f < \nu$, and $n - f \geq \nu$. If $f \geq \nu$, on the other hand, the consensus cannot be guaranteed.*

In the following, we describe the actions strategic participants have. We present it as a protocol shown in Algorithm 6. The definition of the game and of the actions is done in the next section. We consider the choice of (i) checking or not the validity of a block and (ii) sending or not the vote for a proposed block. We consider that the actions of checking the validity of blocks and of sending the message (of type vote) are costly.

Protocol of Strategic Participants. Strategic participants choices are explicitly represented in Algorithm 6 by dedicated variables, namely, $action^{\text{check}}$ and $action^{\text{send}}$. Each action, initialised at a default value of *nil*, can take values from the set $\{\texttt{false}, \texttt{true}\}$. For participant i, the values of $action^{\text{check}}$ and $action^{\text{send}}$ are set by calling respectively the functions σ_i^{check}, and σ_i^{send}, returning its strategy.

Note that an obedient participant (who always follow the prescribed protocol) takes its actions such that Algorithm 6 corresponds to Algorithm 5, *i.e.*, $action^{\text{check}} = action^{\text{send}} = \texttt{true}$.

Algorithm 6 Pseudo-code for a given height h modelling the strategic participant i's behaviour

```
1:  Initialisation:
2:    vote := nil
3:    t := 0                                                    /* Current round number */
4:    decidedValue := nil
5:    action^check := nil
6:    action^send := nil
7:    validValue[] := {⊥, ⊥, ..., ⊥}                            /* ∀t, validValue[t] ∈ {⊥, false, true} */

8:  Round PROPOSE(t):
9:    Send phase:
10:     if i == isProposer(t, h) then
11:       proposal ← createValidValue(h)
12:       broadcast ⟨PROPOSE, h, t, proposal⟩
13:   Delivery phase:
14:     delivery ⟨PROPOSE, h, t, v⟩ from proposer(h, t)
15:   Compute phase:
16:     action^check ← σ_i^check()   /* σ_i^check() ∈ {false, true} sets the action of checking or not the validity of the proposal */
17:     if action^check == true then
18:       validValue[t] ← isValid(v)                  /* The execution of isValid(v) has a cost c_check */
19:     action^send ← σ_i^send(validValue[t])   /* σ_i^send : {⊥, false, true} → {false, true} sets the action of sending the vote or not */
20:     if action^send == true then
21:       vote ← v                  /* The participant decides to send the vote, the proposal might be invalid */

22: Round VOTE(t):
23:   Send phase:
24:     if vote ≠ nil then
25:       broadcast ⟨VOTE, h, t, vote⟩                  /* The execution of the broadcast has a cost c_send */
26:   Delivery phase:
27:     delivery ⟨VOTE, h, t, v⟩          /* The participant collects all the votes for the current height and round */
28:   Compute phase:
29:     if |⟨VOTE, h, t, v⟩| ≥ ν ∧ decidedValue = nil ∧ vote ≠ nil ∧ vote = v then
30:       decidedValue = v; exit
31:     else
32:       vote ← nil
33:       t ← t + 1
```

The strategy σ_i^{check} determines if i, the receiving participant chooses to check the validity of the proposal or not, which is a costly action. If the participant chooses to check the validity (Line 17 of Algorithm 6), it will also update the knowledge it has about the validity of the proposal and it will pay a cost c_{check}. Otherwise, the participant keeps not knowing if the proposal is valid or not ($validValue[t]$ remains at $\perp$). Note that this value remains at $\perp$ even if the participant is the proposer. This is because we assumed, without loss of generality, that checking validity has a cost and that the only way of checking validity is by executing the $\texttt{isValid}(v)$ function.

Note that, as it will be defined in Section 5.1.4, the strategy σ_i^{send} depends on the knowledge the participant has about the validity of the proposal. The strategy determines if the participant chooses to send its vote for the proposal or not (Line 19 - 25 of Algorithm 6). If the participant chooses to send a message for the proposal, it will pay a cost c_{send}.

Let us note that a strategic participant that did not check the validity of the block could consider as the decision of the committee an invalid value if it collects more than ν votes for an invalid proposal. We also note that in our model, the Agreement property always holds, since, at the end of each round, all participants have the same set of messages delivered.

We now define the game that represents the protocol.

5.1.4 Game

Action space. After receiving the proposal block, each participant first decides whether to check the block's validity or not (at cost c_{check}), and second decides whether to send a message

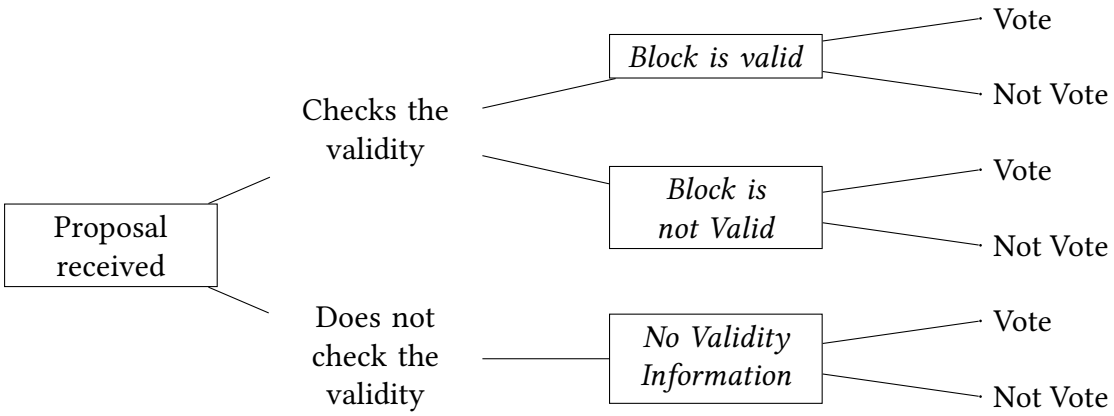


Figure 5.1: Decision Tree of One Participant after Reception of the Proposal.

(at cost c_{send}) or not.

Information sets. At the beginning of each round $t > 1$, the information set of the participant, η_i^t, includes the observation of the round number t, the participant's own type θ_i, as well as the observation of what happened in previous rounds, namely (i) whether the participant decided to check validity, and in that case, it knows the validity of the block, (ii) how many messages were sent, and (iii) whether a block was produced or not.

Then, in each round $t > 1$, the participant decides whether to check the validity of the current block. At this point, denoting by b_t the block proposed at round t, when the participant does not decide to check validity, the $\texttt{isValid}(b_t)$ function is set to the null information set, while if the participant decides to check, $\texttt{isValid}(b_t)$ is equal to `true` if the block is valid and `false` otherwise. Therefore, at this stage, the participant information set becomes $H_i^t = \eta_i^t \cup \texttt{isValid}(b_t)$, which is η_i^t augmented with the validity information participant i has about b_t, the proposed block.

Strategies. At each round $t \geq 1$, the strategy of participant i is a mapping from its information set into its actions. At the point at which the participant can decide to check block validity, its strategy is given by $\sigma_i^{\text{check}}(\eta_i^t)$. Finally, after making that decision, the participant must decide whether to send a message or not, and that decision is given by $\sigma_i^{\text{send}}(H_i^t)$. The decision tree of a participant is depicted in Figure 5.1. We note that when the participant does not check the validity of the proposal, it does not know if the block is valid or not.

We denote by $\sigma = (\sigma_1, \ldots, \sigma_n)$, the *strategy profile* where $\forall i \in \{1, \ldots, n\}$, participant i uses strategy σ_i, where $\sigma_i(H_i^t)$ is the pair $(\sigma_i^{\text{check}}(\eta_i^t), \sigma_i^{\text{send}}(H_i^t))$.

Rewards and Costs for Strategic Participants. In this chapter, we study the cases in which:

1. when a block is produced, only the participants which sent a message are rewarded (and receive R), as it is done in some blockchain systems (*e.g.*, [15]);

2. whenever a block is produced, all committee members are rewarded (and receive R).

In our analysis, we will explicitly state the case we are studying.

In addition, we assume that when an invalid block is produced, all strategic participants incur a cost κ. *The reward R, given to the participants when a block is produced, is larger than*

the cost c_{check} of checking validity, which in turn is larger than the cost c_{send} of sending a message. Additionally, we assume that the reward obtained when a block is produced is smaller than the cost κ of producing an invalid block. That is,

$$\kappa > R > c_{\text{check}} > c_{\text{send}} > 0.$$

Objective of Strategic Participants. Let T be the endogenous round at which the game stops. If a block is produced at round $t \leq n$, then $T = t$. Otherwise, if no block is produced, $T = n + 1$. In the latter case, the *termination* property is not satisfied.

As explained above, we study two type of rewards. The analyses are done independently. In each setting, all strategic participants have the same gain function.

1. **Reward Only Sender**: When the reward is given only to participants that vote for the produced block, at the beginning of the first round, the expected gain of strategic participant i is:

$$U_i = E\left[\begin{array}{c}\left(R \times \mathbb{1}_{(\sigma_i^{\text{send}}(H_i^T)=\texttt{true})} \times \mathbb{1}_{(\text{block produced at } T)} - \kappa \times \mathbb{1}_{(\text{invalid block produced})}\right) \\ -\sum_{t=1}^{T}\left(c_{\text{check}} \times \mathbb{1}_{\sigma_i^{\text{check}}(\eta_i^t)=\texttt{true})} + c_{\text{send}} \times \mathbb{1}_{(\sigma_i^{\text{send}}(H_i^t)=\texttt{true})}\right)\end{array} \middle| \eta_i^1\right], \tag{5.1}$$

   where $\mathbb{1}_{(.)}$ denotes the indicator function, taking the value 1 if its argument is `true`, and 0 if it is `false`.

2. **Reward All**: When the reward is given to the whole committee once a block is produced, at the beginning of the first round, the expected gain of strategic participant i is:

$$U_i^{\text{all}} = E\left[\begin{array}{c}\left(R \times \mathbb{1}_{(\text{block produced at } T)} - \kappa \times \mathbb{1}_{(\text{invalid block produced})}\right) \\ -\sum_{t=1}^{T}\left(c_{\text{check}} \times \mathbb{1}_{\sigma_i^{\text{check}}(\eta_i^t)=\texttt{true})} + c_{\text{send}} \times \mathbb{1}_{(\sigma_i^{\text{send}}(H_i^t)=\texttt{true})}\right)\end{array} \middle| \eta_i^1\right]. \tag{5.2}$$

Equilibrium concept. In this chapter, we consider that the participants are playing Nash equilibria, and we focus only on their behaviour during the first round. The following sections present our results.

In Sections 5.2 & 5.3, we do not have trembling hand effects; therefore, we cannot have invalid blocks since the proposal should be valid (Line 11 of Algorithm 6). Focusing on liveness issues, we study whether participants vote or not in equilibrium.

In Section 5.4, trembling effects are considered, and the proposal may be invalid. Therefore, for safety reasons, participants may check the proposal's validity before voting or not.

5.2 Reward Only Committee Members that Vote

In this section, we consider that only committee members that voted for a produced block are rewarded. Equation (5.1) describes the gain of each strategic participant. We note that, since we focus only on the first round, Equation (5.1) can be simplified as:

$$E\left[(R\mathbb{1}_{(\text{block produced at round 1})} - c_{\text{send}})\mathbb{1}_{\sigma_i^{\text{send}}(H_i^1)} - c_{\text{check}}\mathbb{1}_{\sigma_i^{\text{check}}(h_i^1)} - \kappa\mathbb{1}_{(\text{invalid block produced at round 1})}\right].$$

We study the different equilibria with respect to the value of ν, the minimum number of votes required to consider a block as produced.

First, we analyse the case where 1 vote for a proposed block is sufficient to consider it as produced, *i.e.*, $\nu = 1$.

Proposition 5.2. *In one round, with only strategic participants in the committee, if $\nu = 1$, and when only participants that vote for the produced block are rewarded, there is only one Nash equilibrium. In the unique equilibrium, all participants vote for the proposed block.*

In this equilibrium, all participants vote, and the block is produced. No participant has an incentive to deviate and not send, such deviation will mean for the participant that it will not be rewarded while the block is produced.

Proof

We first prove that the strategy profile where all participants vote for the proposed block is a Nash equilibrium. In that strategy profile, the gain of a participant is $R - c_{\text{send}}$. If a participant deviates and does not send a vote, the block is produced in any case ($\nu = 1$), therefore the gain of the deviating participant is 0, which is lower than the gain at equilibrium. The strategy profile is indeed a Nash equilibrium.

We now prove that there is no more Nash equilibrium. Let i, j be two participants such that $\sigma_i^{\text{send}} \neq \sigma_j^{\text{send}}$. Without loss of generality, we assume that $\sigma_i^{\text{send}} = \texttt{false}$ and $\sigma_j^{\text{send}} = \texttt{true}$. In this case, the block is produced since $\nu = 1$. The gain of participant i is 0, while the gain of j is $R - c_{\text{send}}$. If instead i votes, it will have a gain of $R - c_{\text{send}} > 0$.

If no participant votes, the block is not produced, and all participants have a reward of 0. If instead a participant deviates and votes, the block will be produced, and the deviating participant will have a reward of $R - c_{\text{send}} > 0$. The only Nash equilibrium is the strategy profile where all participants vote. $\square_{Proposition\ 5.2}$

Remark 5.3. *Note that in the Nash equilibrium of Proposition 5.2, the consensus properties are satisfied, in particular, there is always a block produced at the end of the first round.*

We now consider the situation where strictly more than one vote is needed to consider a block as produced, *i.e.*, $\nu \in \{2, \ldots, n\}$.

Proposition 5.4. *In one round, with only strategic participants in the committee, if $\nu > 1$, and when only participants that vote for the produced block are rewarded, there are two Nash equilibria; either (i) all participants vote, or (ii) no participant votes.*

In the first equilibrium, if a strategic participant anticipates that no participants will vote, its only vote will not make the proposal produced, since $\nu > 1$, therefore, the participant is better off not voting. In the second type of equilibrium, if a participant anticipates that all other participants are voting, it is better off voting as well; otherwise, if the participant does not send, it will not have a reward.

Proof

We prove that the strategy profiles described in the proposition are Nash equilibria.

- First, we prove that the strategy profile where no participant votes is a Nash equilibrium. The gain at equilibrium of any participant is 0. If one participant deviates and does vote, only it sends a vote, and the block is not produced since $\nu > 1$, its gain at deviation is $-c_{\text{send}}$, which is lower than the gain at equilibrium. The strategy profile is indeed a Nash equilibrium.

- We now prove that the strategy profile where all participants vote is a Nash equilibrium. The gain at equilibrium of any participant is $R - c_{\text{send}}$. If one participant deviates and does not send a vote, its gain at deviation is $0 < R - c_{\text{send}}$. The strategy profile is indeed a Nash equilibrium.

Moreover, considering one round at a time, there is no more equilibrium. To prove that, let $X^{\sigma^{\text{send}}} = \{i : \sigma_i^{\text{send}} = \texttt{true}\}$ be the set of all participants that decided to vote.

- When $|X^{\sigma^{\text{send}}}| < \nu - 1$: Assume by contradiction that there exists a Nash equilibrium such that participant i votes, *i.e.*, $\sigma_i^{\text{send}} = \texttt{true}$. Since $|X^{\sigma^{\text{send}}}| < \nu - 1$, the block is not produced, so the gain of i is $-c_{\text{send}}$; if instead i decides not to vote, its gain would be $0 > -c_{\text{send}}$. Contradiction, the strategy profile is not a Nash equilibrium.
- When $|X^{\sigma^{\text{send}}}| \geq \nu - 1$: Assume by contradiction that there exists a Nash equilibrium such that participant i does not vote, *i.e.*, $\sigma_i^{\text{send}} = \texttt{false}$. The gain at equilibrium of i is 0. If instead, i deviates and votes, its gain will be $R - c_{\text{send}} > 0$, since the block will be produced in any case. Contradiction, the strategy profile is not a Nash equilibrium.

□*Proposition 5.4*

Remark 5.5. *There are two Nash equilibria in Proposition 5.4. In the equilibrium where no participant votes, Termination is not guaranteed at round 1. In the second equilibrium where there are n votes, the consensus properties are satisfied in the first round.*

5.3 Reward All Committee Members

In this section, we consider that all committee members are rewarded once a block is produced. Equation (5.2) describes the gain of each strategic participant. We note that, when focusing only the first round, Equation (5.2) can be simplified as:

$$E\left[R\mathbb{1}_{(\text{block produced at round 1})} - c_{\text{send}}\mathbb{1}_{\sigma_i^{\text{send}}(H_i^1)} - c_{\text{check}}\mathbb{1}_{\sigma_i^{\text{check}}(h_i^1)} - \kappa\mathbb{1}_{(\text{invalid block produced at round 1})}\right].$$

We study the different equilibria with respect to the value of ν, the minimum number of votes required to consider a block as produced.

First, we analyse the case where 1 vote for a proposed block is sufficient to consider it as produced, *i.e.*, $\nu = 1$.

Proposition 5.6. *In one round, with only strategic participants in the committee, if $\nu = 1$, and when all participants are rewarded once a block is produced, in the Nash equilibria, exactly one participant votes, and the others do nothing.*

If the participant supposed to vote does not vote, no block is produced, and hence it does not have any reward, therefore, it prefers voting. Since a block is always produced in equilibrium, if a participant not supposed to send deviates and votes, it will pay the cost of sending for nothing since it will be rewarded even without voting.

Proof

We first prove that the strategy profile where exactly one participant votes for the proposed block is a Nash equilibrium. Without lack of generality, assume that the participant with index 1 votes, so $\sigma_1^{\text{send}} = \texttt{true}$. In that strategy profile, the gain of participant 1 is $R - c_{\text{send}}$, and the gain of the other participants is R. If participant 1 deviates and does not send a vote, the block is not produced and its gain at deviation is $0 < R - c_{\text{send}}$. If another participant not supposed to send deviates and votes, its gain at deviation will be $R - c_{\text{send}} < R$. The strategy is indeed a Nash equilibrium.

Moreover, we show that in all equilibria in this setting, there is only one participant that necessarily votes.

- The strategy profile where no participant votes is not a Nash equilibrium. In fact, the gain at equilibrium of a participant is 0, while if it deviates, its gain will be $R - c_{\text{send}} > 0$.
- By contradiction, assume that there is a Nash equilibrium such that two participants i and j vote. The gain of i and j at equilibrium is $R - c_{\text{send}}$; if (let us say) i deviates and does not send, its gain at deviation will be $R > R - c_{\text{send}}$. Contradiction, the strategy profile where strictly more than one participant votes is not a Nash equilibrium.

$\square_{Proposition\ 5.6}$

Remark 5.7. *Note that there exists at most n equilibria corresponding to Proposition 5.6. In all the equilibria corresponding to Proposition 5.6, the consensus properties are satisfied.*

We now consider the situation where strictly more than one vote is needed to consider a block as produced, *i.e.*, $\nu \in \{2, \dots, n\}$.

Proposition 5.8. *In one round, with only strategic participants in the committee, if $\nu > 1$, and when all participants are rewarded once a block is produced, in the Nash equilibria, either (i) exactly ν participants vote, or (ii) no participant votes.*

If a strategic participant anticipates that no participants will vote, since $\nu > 1$, its only vote will not make the proposal produced, therefore, it is better off not voting. In the other type of equilibrium, exactly ν participants vote; if a participant supposed to send does not vote, the block is not produced and the deviating participant is not rewarded any more; if a participant not supposed to send deviates (by voting) it will incur a cost of sending, when it will be rewarded in any case, so it prefers not to send.

Proof

We prove that the strategy profiles described in Proposition 5.8 are Nash equilibria.

- First, we prove that the strategy profile where no participant votes is a Nash equilibrium. The gain at equilibrium of any participant is 0. If one participant deviates and does send a vote, only it sends a vote, and the block is not produced since $\nu > 1$, its gain at deviation is $-c_{\text{send}}$, which is lower than the gain at equilibrium. The strategy profile where no one sends is indeed a Nash equilibrium.
- We now prove that the strategy profile where exactly ν participants vote is a Nash equilibrium. Without loss of generality, assume that the ν first participants are supposed to send, and participants with index bigger than ν do not send. The gain at equilibrium of a participant supposed to send is $R - c_{\text{send}}$. If it deviates and does not send a vote, its gain at deviation is $0 < R - c_{\text{send}}$. The gain of a participant not supposed to send is R. If it deviates and sends a vote, its gain at deviation is $R - c_{\text{send}} < R$. The strategy is indeed a Nash equilibrium.

Moreover, considering one round at a time, there is no more equilibrium. To prove that, let $X^{\sigma^{\text{send}}} = \{i : \sigma_i^{\text{send}} = \texttt{true}\}$ be the set of all participants that decided to vote.

- When $1 \leq |X^{\sigma^{\text{send}}}| < \nu$: Assume by contradiction that there exists a Nash equilibrium such that participant i votes, *i.e.*, $\sigma_i^{\text{send}} = \texttt{true}$. Since $|X^{\sigma^{\text{send}}}| < \nu$, the block is not produced, so the gain of i is $-c_{\text{send}}$; if instead i deviates and does not vote, its gain would be $0 > -c_{\text{send}}$. Contradiction, the strategy profile is not a Nash equilibrium.

	Reward All	Reward Only Senders
$\nu = 1$	Proposition 5.6 In equilibrium, exactly one message is sent: **Consensus**	Proposition 5.2 In equilibrium, All participants send a message: **Consensus; but inefficient: too costly**
$\nu > 1$	Proposition 5.8 In equilibrium, either: - No message is sent: **No Termination: Coordination failure**, or - Exactly ν messages are sent. **Consensus**	Proposition 5.4 In equilibrium, either: - No message is sent: **No Termination: Coordination failure**, or - All participants sent a message: **Consensus; but inefficient: too costly**

Table 5.1: Summary of the Equilbria with Strategic Participants

- When $|X^{\sigma^{\text{send}}}| > \nu$: Assume by contradiction that there exists a Nash equilibrium such that participant i does vote, *i.e.*, $\sigma_i^{\text{send}} = \texttt{true}$. The gain at equilibrium of i is $R - c_{\text{send}}$. If instead, i deviates and does not vote, its gain will be R, which is bigger than $R - c_{\text{send}}$. Contradiction, the strategy profile is not a Nash equilibrium.

□*Proposition 5.8*

Remark 5.9. *There are two types of Nash equilibria in Proposition 5.8.*

- *The equilibrium where no participant votes do not guarantee* Termination *at round 1.*
- *In the second type of equilibrium in this setting, there are exactly ν messages sent. There can be at most $\binom{n}{\nu}$[1] $+ 1$ equilibria corresponding to that setting. In each of them, the consensus properties are satisfied.*

A summary of the different equilibria in Sections 5.2 & 5.3 can be found in Table 5.1. We note that when only 1 vote is required to consider a proposal as produced, in all equilibria, blocks are always produced. When we require strictly more than 1 vote to consider a block as produced, although there are equilibria where the consensus is guaranteed, there is also an equilibrium where no participant votes, anticipating that the others will not vote as well: that is a coordination failure, leading to a violation of the Termination property. This is true in the two reward mechanisms: reward all committee members, or reward only the members that voted. However, in the equilibria where all committee members are rewarded, fewer messages are sent, making it a more efficient mechanism with respect to the number of messages sent.

5.4 Trembling Strategic Participants

Now, we assume that there is some negligible probability p for the `createValidValue` function (Line 11 of Algorithm 6) to return an invalid proposal, and all participants are aware of the trembling effect.

When proposing a value, there is a probability that the hand of the participant trembles and proposes an invalid block instead of a valid block; *i.e.*, in some sense, we take into account the possibility of making a mistake for the proposal. The idea of trembling hand and acknowledging errors has been studied in economics (*e.g.*, [57]).

[1] $\binom{n}{\nu} = C_n^\nu$ is the number of combinations for choosing ν out of n elements.

Note that now, checking the validity of a block may be important, there is a risk of producing an invalid block, breaking the validity property of the consensus. To ensure that the reward covers the costs of checking and sending in this setting we assume that $(1-p)(R-c_{\text{send}})-c_{\text{check}}>0$, that is, the reward covers the costs. We also note that it is better for the participants to send (resp. not to send) without checking than checking and sending (resp. not sending) irrespective to the block validity; that would mean incurring a cost $-c_{\text{check}}$ for nothing. It is also not in their best interest to check the validity of the proposal and vote if the proposal is invalid, that would mean increasing the chances of producing an invalid block and incurring a cost $-\kappa$. In the analyses, we then consider only the three relevant strategies: a strategic participant can (i) vote without checking proposal validity, (ii) not vote nor check proposal validity, and (iii) check the proposal validity and vote only if the proposal is valid.

In the following, we make the same analyses as in Sections 5.2 & 5.3, *i.e.*, we analyse the behaviours of strategic participants when only voters are rewarded, and their behaviours when all committee members are rewarded.

5.4.1 Reward Only Committee Members that Vote

In this subsection, we consider that only committee members that voted for a produced block are rewarded. Equation (5.1) describes the gain of each strategic participant.

We study the different equilibria with respect to the value of ν, the minimum number of votes required to consider a block as produced.

First, we analyse the case where 1 vote for a proposed block is sufficient to consider it as produced, *i.e.*, $\nu=1$.

Proposition 5.10. *In one round, with only strategic participants in the committee, if $\nu=1$, when only participants that vote for the produced block are rewarded, and if there is a probability p that the proposer proposes an invalid block, there are two Nash equilibria. In equilibrium, either (i) if $\kappa \geq R-c_{send}+c_{check}/p$, all participants check the validity of the proposal and vote only if it is valid; or (ii) all participants vote for the proposal without checking the validity of the proposal.*

As in Proposition 5.2, one can note that in equilibrium, all participants do (try to) vote.

Proof

We prove that the strategy profiles described in the proposition are Nash equilibria.

- First, we prove that the strategy profile where all participants check the proposal validity and vote only if the proposal is valid is a Nash equilibrium.

  The expected gain at equilibrium of any participant is $(1-p)(R-c_{\text{send}})-c_{\text{check}}>0$. If one participant deviates and does not send a vote nor checks, its gain at deviation is 0; if it deviates and votes without checking, its expected gain at deviation is $R-c_{\text{send}}-p\kappa$, which is lower than the gain at equilibrium if and only if $\kappa \geq R-c_{\text{send}}+c_{\text{check}}/p$, which is our assumption, therefore, the gain at equilibrium is better than the gain if the participant deviates. The strategy profile where all participants check the proposal validity and vote only if the proposal is valid is a Nash equilibrium.

- We now prove that the strategy profile where all participants vote without checking the proposal validity is a Nash equilibrium.

  The gain at equilibrium of any participant is $R-c_{\text{send}}-p\kappa$. Even if one participant deviates, the block will be produced in any case, no matter its validity. If a participant

deviates by checking validity and votes only if the proposal is valid, its gain at deviation is $(1-p)(R-c_{\text{send}})-c_{\text{check}}-p\kappa$; if the participant deviates and does not check proposal's validity nor votes, its expected gain at deviation is $-p\kappa$. In any case, the gain at deviation is lower than the gain at equilibrium. The strategy profile is indeed a Nash equilibrium.

Moreover, considering one round at a time, there is no more equilibrium.

First, we prove that in equilibrium, at least one participant should choose to vote. If no participant votes, no block is produced and the gain of each participant is 0; if a participant deviates and checks the proposal validity and votes only if it is valid, its gain at deviation is $(1-p)(R-c_{\text{send}})-c_{\text{check}}>0$. Let i,j be two participants such that $\sigma_i^{\text{send}}\neq\sigma_j^{\text{send}}$. Without loss of generality, we assume that $\sigma_i^{\text{send}}=\texttt{false}$ and $\sigma_j^{\text{send}}=\texttt{true}$. In this case, the block is produced since $\nu=1$. The gain of i is 0 if the proposal is valid (resp. $-\kappa$ if the proposal is invalid); if instead i votes, it will have a gain of $R-c_{\text{send}}$ (or resp. $R-c_{\text{send}}-\kappa$), i is better off by deviating.

Now we can prove that in equilibrium, participants have the same strategy for checking validity. Let i,j be two participants such that $\sigma_i^{\text{check}}\neq\sigma_j^{\text{check}}$, without loss of generality, we assume that $\sigma_i^{\text{check}}=\texttt{true}$. The expected gain of i is $(1-p)(R-c_{\text{send}})-c_{\text{check}}-p\kappa$ if instead it deviates and does not check, but its expected gain is $R-c_{\text{send}}-p\kappa$, which is greater than the gain before deviation. □*Proposition 5.10*

Remark 5.11. *There are two Nash equilibria in Proposition 5.10. In the equilibrium where all participants check, if the proposal is invalid, there is no Termination at the first round, however, Validity is always ensured. While in the second equilibrium where no participant checks, Termination is always guaranteed at the end of the first round, even if the proposal is invalid, which violates the Validity.*

We now consider the situation where strictly more than one vote is needed to consider a block as produced, *i.e.*, $\nu\in\{2,\dots,n\}$.

Proposition 5.12. *In one round, with only strategic participants in the committee, if $\nu>1$, when only participants that vote for the produced block are rewarded, and if there is a probability p that the proposer proposes an invalid block, there are three Nash equilibria. Either (i) no participant votes nor checks the proposal validity; or (ii) if $\nu<n$, all participants vote for the proposal without checking the validity of the proposal; or (iii) if $\kappa\geq R-c_{send}+c_{check}/p$, $n-\nu+1$ participants check the validity of the proposal and vote only if it is valid, and the $\nu-1$ remaining participants only vote without checking the validity of the proposal.*

Proof

We prove that the strategy profiles described in the proposition are Nash equilibria.

- First, we prove that the strategy profile where no participant votes is a Nash equilibrium. The gain at equilibrium of any participant is 0. If one participant deviates and does send a vote, there is only 1 vote and the block is not produced since $\nu>1$, the gain at deviation is $-c_{\text{send}}<0$. If the participant deviates by checking block validity, it will pay the cost $-c_{\text{check}}-(1-p)c_{\text{send}}<0$. The strategy profile is indeed a Nash equilibrium.

- We now prove that the strategy profile where all participants vote without checking the proposal validity is a Nash equilibrium. Let $\nu<n$, the gain at equilibrium of any

participant is $R - c_{\text{send}} - p\kappa$. Even if one participant deviates, the block will be produced in any case (since $\nu < n$) no matter its validity. If a participant deviates by checking validity and sending if the proposal is valid, its gain will be $(1-p)(R-c_{\text{send}})-c_{\text{check}}-p\kappa$; if the participant deviates and does not check proposal's validity nor votes, its expected gain at deviation is $-p\kappa$, the gain at deviation is lower than the gain at equilibrium. The strategy profile is indeed a Nash equilibrium.

- It remains to prove that the strategy profile where some participants are supposed to check the proposal validity and check only if the block is valid and the remaining participants vote without checking block validity is also a Nash equilibrium.

  We can first note that only valid blocks can be produced following the equilibrium, and invalid blocks do not have the necessary ν votes, since only $\nu - 1$ participants vote without checking, and so for invalid proposal.

  - The expected gain of a participant not supposed to check is $(1 - p)(R - c_{\text{send}})$. If it deviates and does not send, its gain at deviation is 0; if it deviates by checking and sending a message only if the proposal is valid, its expected gain at deviation is $(1 - p)(R - c_{\text{send}}) - c_{\text{check}}$, which is lower than the gain at equilibrium.
  - The expected gain of a participant supposed to check is $(1-p)(R-c_{\text{send}})-c_{\text{check}}$. If it deviates and does not send, its gain at deviation is 0. If it deviates by voting without checking the proposal's validity, any block proposed will be produced, no matter its validity since ν votes are sent in any case, so the expected gain of the deviating participant is $R - c_{\text{send}} - p\kappa$, which is lower than the gain at equilibrium if and only if $\kappa \geq R - c_{\text{send}} + c_{\text{check}}/p$.

  The strategy profile is indeed a Nash equilibrium.

Moreover, considering one round at a time, there is no more equilibrium. We sketch the proof by exhibiting the main other equilibrium candidates.

- Let $x \geq 0$. Assume by contradiction that there exists an equilibrium where $n - \nu - x$ participants check the block validity and vote only if the proposal is valid, and the remaining $\nu + x$ participants vote without checking the block validity.

  That means any block proposed will be produced since $\nu + x \geq \nu$ participants vote without checking validity. Let i be a participant supposed to check. It expected gain is $R - c_{\text{send}} - c_{\text{check}} - p\kappa$, while if i deviates and send without checking proposal validity, its expected gain will be $R - c_{\text{send}} - p\kappa$. Contradiction, the strategy profile is not an equilibrium.

- Let $x > 1$. Assume by contradiction that there exists an equilibrium where $n - \nu + x$ participants check the block validity and vote only if the proposal is valid, and the remaining $\nu - x$ participants vote without checking the block validity.

  Let i be a participant supposed to check. It expected gain is $(1-p)(R-c_{\text{send}})-c_{\text{check}}$. If i deviates and send without checking proposal validity, there will be $\nu - x + 1 < \nu$ votes for an invalid block proposed, and so it will not be produced, where there will be n votes for a valid block proposed; the expected gain at deviation for i is $(1 - p)(R - c_{\text{send}})$. Contradiction, the strategy profile proposed is not an equilibrium.

□*Proposition 5.12*

Remark 5.13. *There are three types of Nash equilibria in Proposition 5.12.*

- *The equilibrium where no participant votes do not guarantee* Termination *at round 1.*
- *In the equilibrium where no participant checks, Termination is always guaranteed at the end of the first round, even if the proposal is invalid, which violates the Validity property.*
- *In the last equilibrium, valid blocks are produced and invalid blocks are not. Termination is not guaranteed at round* 1 *but Validity is always ensured. There can be at most* $\binom{n}{n-\nu+1}$ *equilibria corresponding to that setting.*

5.4.2 Rewarding All Committee Members

In this section, we consider that all committee members are rewarded once a block is produced. Equation 5.2 describes the gain of each strategic participant.

We study the different equilibria with respect to the value of ν, the minimum number of votes required to consider a block as produced.

First, we analyse the case where 1 vote for a proposed block is sufficient to consider it as produced, *i.e.*, $\nu = 1$.

Proposition 5.14. *In one round, with only strategic participants in the committee, if* $\nu = 1$*, when all participants are rewarded once a block is produced, if there is a probability* p *that the proposer proposes an invalid block, and if* $\kappa \geq R - c_{send} + c_{check}/p$*, in all Nash equilibria, exactly one participant checks the validity of the proposal and votes only if it is valid, while the other participants do nothing.*

As in Proposition 5.6, one can note that in equilibrium, the task of validating (checking) and producing a block is delegated to one participant.

Proof

We prove that the strategy profile described in the proposition is a Nash equilibrium. We prove that the strategy profile where exactly one participant votes for the proposed block is a Nash equilibrium. Without lack of generality, assume that the participant with index 1 is the one supposed to check the proposal validity and to vote only if it is valid, and all other participants do not send nor check. In that strategy profile, the gain of participant 1 is $(1-p)(R - c_{\text{send}}) - c_{\text{check}}$, and the gain of the other participants is R.

If participant 1 deviates and sends a vote without checking, the block is always produced, even if it is invalid, so its gain at deviation is $R - c_{\text{send}} - p\kappa$ which is lower than the gain at equilibrium if $\kappa \geq R - c_{\text{send}} + c_{\text{check}}/p$. If participant 1 deviates and does not send a vote nor checks, the block is never produced, therefore, the gain of i at deviation is 0, which is lower than the gain at equilibrium. The other participants having a reward of R cannot do better, since R is the maximum reward a participant can get.

The strategy profile is indeed a Nash equilibrium.

Moreover, we now prove that in all equilibria in this setting, there is exactly one participant that necessarily checks and votes.

- The strategy profile where no participant checks nor votes is not a Nash equilibrium. In fact, the gain at equilibrium of a participant is 0, while if it deviates by checking the block validity and voting only if the proposal is valid its gain will be $(1-p)(R-c_{\text{send}}) - c_{\text{check}} > 0$.

- By contradiction, assume that there is a Nash equilibrium such that two participants i and j check the proposal validity and vote only if the block is valid. The gain of i and j at equilibrium is $(1-p)(R-c_{\text{send}})-c_{\text{check}}$; if i deviates and does not send nor check, its gain at deviation will be $(1-p)R > (1-p)(R-c_{\text{send}})-c_{\text{check}}$. Contradiction, the strategy profile is not a Nash equilibrium.
- By contradiction, assume that there is a Nash equilibrium where there are two participants i that checks the proposal validity and vote only if the block is valid, and j that votes without checking validity. The gain of i at equilibrium is $R-(1-p)c_{\text{send}}-c_{\text{check}}-p\kappa$; since the block will be produced anyway, if i deviates and does not send nor check, its gain at deviation will be $R-p\kappa > R-(1-p)c_{\text{send}}-c_{\text{check}}-p\kappa$. Contradiction, the strategy profile is not a Nash equilibrium.
- By contradiction, assume that there is a Nash equilibrium such that two participants i and j vote without checking validity. The gain of i at equilibrium is $R-c_{\text{send}}-p\kappa$; if i deviates and does not send nor checks, its gain at deviation will be $R-p\kappa > R-c_{\text{send}}-p\kappa$. Contradiction, the strategy profile is not a Nash equilibrium.

□*Proposition 5.14*

Remark 5.15. *Note that there exists at most n equilibria corresponding to Proposition 5.14. In all the equilibria corresponding to Proposition 5.14, if the proposal is invalid, there is no Termination at the first round, however, Validity is always ensured.*

We now consider the situation where strictly more than one vote is needed to consider a block as produced, *i.e.*, $\nu \in \{2,\dots,n\}$.

Proposition 5.16. *In one round, with only strategic participants in the committee, if $\nu > 1$, when all participants are rewarded once a block is produced, if there is a probability p that the proposer proposes an invalid block, and if $\kappa \geq R - c_{send} + c_{check}/p$, in all Nash equilibria, either (i) no participant votes, or (ii) 1 participant checks the proposal validity, and votes only if it is valid, exactly $\nu - 1$ other participants vote without checking validity, and the others do nothing.*

Proof

We prove that the strategy profiles described in the proposition are Nash equilibria.

- First, we prove that the strategy profile where no participant votes is a Nash equilibrium. The gain at equilibrium of any participant is 0. If one participant deviates and does send a vote, there is only 1 vote, and the block is not produced since $\nu > 1$, the gain at deviation is $-c_{\text{send}} < 0$. If the participant deviates by checking block validity, it will pay the cost of checking for nothing and will have the gain $-c_{\text{check}} - (1-p)c_{\text{send}} < 0$. The strategy profile is indeed a Nash equilibrium.
- It remains to prove that the strategy profile where some participants are supposed to check the proposal validity, and send only if the block is valid; some participants vote without checking block validity; and the others do nothing is a Nash equilibrium.

  We can first note that only valid blocks can be produced following the equilibrium, and invalid blocks do not have the necessary ν votes, since only $\nu - 1$ participants vote without checking.

– First, the participants that do not send nor check validity have an expected gain of $(1-p)R$. Let i be such a participant. If i deviates and votes without checking, any proposal will be produced, no matter its validity, therefore, the gain of the participant at deviation is $R - c_{\text{send}} - p\kappa$, which is lower than the gain at equilibrium. If instead, i deviates and checks the validity of the proposal and votes only if it is valid, only valid blocks will be produced, so the gain at deviation will be $(1-p)(R - c_{\text{send}}) - c_{\text{check}}$,
which is lower than the gain at equilibrium.

– Now, turns to the participants not supposed to check but supposed to send. Their expected gain at equilibrium is $(1-p)R - c_{\text{send}}$. Let i be such a participant. If i deviates and does not send nor checks, no block will be produced and its gain at deviation is $0 < (1-p)R - c_{\text{send}}$. If it deviates by checking and sending vote only if the proposal is valid, its expected gain at deviation is $(1-p)(R - c_{\text{send}}) - c_{\text{check}}$, which is lower than the gain at equilibrium since $c_{\text{send}} < c_{\text{check}}$.

– Finally, we can analyse the one participant supposed to check. Without loss of generality, assume that it is the participant with index 1. The expected gain of participant 1 is $(1-p)(R - c_{\text{send}}) - c_{\text{check}}$. If it deviates and does not send, no block will be produced, so its gain at deviation is $0 < (1-p)(R - c_{\text{send}}) - c_{\text{check}}$; if it deviates by voting without checking the proposal's validity, any block proposed will be produced, no matter its validity since ν votes are sent in any case; therefore, the expected gain of participant 1 at deviation is $R - c_{\text{send}} - p\kappa$, which is lower than the gain at equilibrium if and only if $\kappa \geq R - c_{\text{send}} + c_{\text{check}}/p$.

The strategy profile is indeed a Nash equilibrium. No participant can increase its gain by deviating.

Moreover, considering one round at a time, there is no more equilibrium in this setting.

First, let us note that in any case, exactly ν participants should vote (counting also those supposed to send after checking). If there are less than ν participants supposed to send (but at least one), no block is produced so one such participant can deviate and not send, economising its cost. If there are more than ν participants supposed to send, one can deviate by not voting and economising that cost.

We can show that the other main equilibrium candidates are not equilibria.

- By contradiction, assume that there exists an equilibrium where ν participants vote without checking the proposal's validity and the others do not send nor check.

  Let i be a participant supposed to send. Its expected gain at equilibrium is $R - c_{\text{send}} - p\kappa$, while if i deviates by checking the proposal validity and sending only if valid, only valid proposals will be produced, so its expected gain will be: $(1-p)(R - c_{\text{send}}) - c_{\text{check}}$ which is greater than the equilibrium. Contradiction, the strategy profile is not an equilibrium.

- By contradiction, assume that there exists an equilibrium where ν participants vote (counting also those supposed to send after checking) and the others do not send nor check. Suppose that in the set of participants supposed to send, at least two i and j check the validity of the proposal and vote only if it is valid. In this strategy profile, only valid proposals will be produced. The expected gain at equilibrium of i is $(1-p)(R - c_{\text{send}}) - c_{\text{check}}$. If instead, i deviates and always send without checking validity, its expected gain at deviation is $(1-p)R - c_{\text{send}}$, which is greater than the gain at equilibrium. Contradiction, the strategy profile is not an equilibrium.

□*Proposition* 5.16

	Reward All	Reward Only Senders
$\nu = 1$	Proposition 5.14 In the equilibrium, one message sent if valid: **Validity**	Proposition 5.10 In the equilibrium, either - n messages sent only if valid: **Validity** - n messages always sent: **Validity not guaranteed**
$\nu > 1$	Proposition 5.16 In equilibrium, either: - No message is sent: **No Termination** - $\nu - 1$ messages always sent + 1 if valid: **Validity**	Proposition 5.12 In equilibrium, either: - No message is sent: **No Termination** - $\nu - 1$ messages always sent + $(n - \nu + 1)$ if valid: **Validity** - (if $\nu < n$) n messages always sent: **Validity not guaranteed**

Table 5.2: Summary of the Equilbria with "Trembling" Strategic Participants

Remark 5.17. *There are two types of Nash equilibria in Proposition 5.16.*

- *Termination is not guaranteed at round 1 in the equilibrium where no participant votes.*
- *In the second type of equilibrium in this setting, there are exactly ν votes when the proposal is valid, but $\nu - 1$ votes when the proposal is invalid. Termination is not guaranteed at round 1 but Validity is ensured. There can be at most $n \times \binom{n-1}{\nu-1}$ equilibria corresponding to that setting.*

A summary of the different equilibria of this Section 5.4 can be found in Table 5.2.

We note that when all participants are rewarded once a block is produced, there is no "bad" equilibrium, *i.e.*, an equilibrium where Validity is violated, while when only participants that send a vote for a produced block are rewarded when $\nu < n$, there exists a "bad" equilibrium (Propositions 5.10 & 5.12).

5.5 Discussions

Before concluding, we discuss some interesting points that are not directly addressed in the core of this chapter.

Fixed amount of Reward for the Committee

First, we quickly highlight what happens if there is a fixed reward for the committee members that is shared by them. Let μ be the number of participants that are rewarded in the committee, and let R/μ be the fraction of the reward each rewarded participant gets. Our equilibrium analyses still hold, but attention should be given to the bounds. For example, in Proposition 5.10, instead of $\kappa \geq R - c_{\text{send}} + c_{\text{check}}/p$, we should have $\kappa \geq R/n - c_{\text{send}} + c_{\text{check}}/p$, since all participants vote in case of a valid proposal (here, $\mu = n$).

Obedient Participants

Recall that obedient participants always follow the prescribed protocol; they take actions such that Algorithm 6 corresponds to Algorithm 5, *i.e.*, they always check the validity of the proposal, and vote only if the proposal is valid.

We did not include obedient participants in our analyses for the following reason: their presence does not change the different equilibria we have; they may, however, change the bounds under which some equilibria exist.

Denote by $\mathfrak{o}$ the number of obedient participants in the committee. Generally, if there are $\mathfrak{o}$ obedient participants and ν messages are required for the production of a block, $\mathfrak{o}$ votes are guaranteed for valid blocks only; then for the strategic participants, the goal is to give the $\nu - \mathfrak{o}$ remaining votes.

5.6 Conclusion

In this chapter, we analyse the behaviours of strategic participants in committee-based blockchains. Strategic participants are selfish, they are self-interested, and the question of whether the consensus properties are guaranteed in their presence was of interest. We study the behaviours of strategic participants under the assumption of synchrony of messages and under different mechanisms of rewards: (i) reward only the senders of votes when a block is produced; and (ii) reward all committee members once a block is produced. Our study took into account the number of votes required to consider a block produced. Once a block is proposed, a participant has the choice of checking the validity of the proposal and then can vote or not for the proposal, knowing that sending a message and checking validity are costly.

When 1 message is required for producing a block, in equilibrium the consensus is always guaranteed at the first round. When strictly more than 1 message is required for the production of a block, there exist equilibria where the consensus is guaranteed at the first round, but there can be coordination failures, no one sends a vote and no block is produced. These equilibria hold with the two mechanisms of rewards, with a difference that when all participants are rewarded, fewer messages are sent.

We moreover investigate the case where an invalid block may be proposed, we found that although the equilibria seem to resemble those in the case where all blocks are valid, there is a slight but important difference. In the case of rewarding only the voters, there are equilibria where invalid blocks could be produced. These latter equilibria exist whenever the number of messages required to consider a block produced is strictly lower than the number of participants; it resembles the *verifier's dilemma* [146] in proof-of-work blockchains, but not exactly, since there is no tension with other sources of allocation (*e.g.*, mining in Bitcoin).

While the assumption of trembling strategic is not that generic and can raise questions about its usage, it leads us to consider participants that purposely proposes invalid blocks, or more generally participants that aim to prevent the consensus properties. We call such participants *malicious*, and in the next chapter, we study whether consensus is possible in the presence of both malicious and strategic participants.

CHAPTER 6

WHEN MALICIOUS COME INTO THE GAME

Contents

Chapter 5 started an analysis of the behaviours of strategic participants in committee-based blockchains. That important analysis does not take into account the presence of adversarial behaviours in the committees. As in Chapter 4 and more generally in distributed systems, some participants may have adversarial or arbitrary behaviours: the Byzantine participants. To take part of those behaviours into account, in this chapter we consider in our analyses the presence of malicious and strategic participants, where malicious participant's objective is to prevent consensus in the blockchain system, and the strategic participants want to maximise their expected gain. We then analyse the behaviour of the participants in such a system. Note that a malicious participant is not fully a Byzantine. Malicious participants have an objective and do their best to achieve it at each moment, while Byzantine participants are more general and can behave arbitrarily. A small comparison is proposed in Section 2.4.2 of Chapter 2.

In this chapter, we do focus on the behaviour inside one committee at a time, and we do not consider what happened before that committee, what will happen after that committee, and how the committee members are selected.

6.1 Model

In this section, the reader may refer to Chapter 2 for more in-depth details about the model.

6.1.1 System Model

We consider a system composed of a finite and ordered set Π, called *committee*, of n synchronous sequential participants, namely $\Pi = \{1, \ldots, n\}$, where participant i is said to have index

i. Except if stated otherwise, in this chapter, all participants are in the set Π.

Communication. Participants communicate by sending and receiving messages through a *synchronous network*. We assume that each participant proceeds in rounds. A *round* consists of three sequential phases, in order: the send, the delivery and the compute phases. The delivery phase has a fixed duration that allows collecting all the messages sent by the participants. At the end of a round, a participant exits from the current round and starts the next one. We assume the existence of a *reliable broadcast* primitive. Messages are created with a digital signature, and we assume that digital signatures cannot be forged. When a participant i delivers a message, it knows the participant j that created the message.

Participants Behaviour. In this chapter, we consider two types of participants. A participant is either *malicious* (Definition 2.8) or *strategic* (Definition 2.7). Note that both types of participants can be seen as *rational* (Definition 2.6).

Strategic participants are self-interested and their objective is to maximise their expected gain. They will deviate from a prescribed protocol if and only if doing so increases their expected gain. We denote by $\mathfrak{s}$ the number of strategic participants in the committee. The malicious participants' objective is to prevent the protocol from achieving its goal, no matter the cost. We denote by $\mathfrak{m}$ the number of malicious participants in the committee. We have that $\mathfrak{m}, \mathfrak{s} \in \mathbb{N}$ and $\mathfrak{m} + \mathfrak{s} = n$.

We assume that the behaviour of participants is perceived identically by all other participants, that is, a message sent by a participant in a given round is received by all in the delivery phase of that round.

6.1.2 Consensus in Presence of Rational Participants

As discussed and studied in Chapter 4, committee-based blockchains can be developed using a consensus algorithm. In particular, at each height, the protocol used by the corresponding committee must implement the consensus. In this section, we adapt the definition of the consensus properties in the presence of rational participants.

We say that a protocol is a consensus algorithm in the presence of rational participants if the following properties hold:

- *r-Termination:* every rational participant decides on a value (a block);
- *r-Agreement:* if two rational participants decide respectively on values B and B', then $B = B'$;
- *r-Validity*: a decided value by any rational participant is valid; it satisfies the predefined predicate.

If clear from the context, we simply write Termination, Agreement and Validity instead of respectively r-Termination, r-Agreement and r-Validity.

Problem. In this chapter, we study the behaviours of rational participants in a consensus protocol. The goal is to know whether consensus is possible in committee-based blockchains in the presence of rational participants. Formally, the question this chapter tackles is: *Are there equilibria where the consensus properties are satisfied?* It is important to note that our goal is not to propose a protocol such that all rational participants behave as obedient, but rather studying the behaviour of rational participants in a consensus algorithm.

Recall that, from Section 2.4.4, an equilibrium is a "stable" situation where no rational participant has an incentive to unilaterally deviate.

Protocol of Rational Participants. The protocol studied in this chapter is slightly different from the protocol in Chapter 5. In this chapter, as it can be noted in Algorithm 7, the participants also have the choice of sending a valid or an invalid proposal (Lines 12 - 16). Both malicious and strategic participants have the same action space, *i.e.*, both follow Algorithm 7.

Rational participants' choices are explicitly represented in Algorithm 7 by dedicated variables, namely, $action^{\text{propose}}$, $action^{\text{check}}$, and $action^{\text{send}}$. Each action, initialised at a default value of *nil*, can take values from the set $\{\texttt{false}, \texttt{true}\}$. For participant i, the values of $action^{\text{propose}}$, $action^{\text{check}}$, and $action^{\text{send}}$ are set by calling respectively the functions $\sigma_i^{\text{propose}}$, σ_i^{check}, and σ_i^{send}, returning the strategy of the participant i.

Algorithm 7 Pseudo-code for a given height h modelling the rational participant i's behaviour

```
1:  Initialisation:
2:    vote := nil
3:    t := 0                                                    /* Current round number */
4:    decidedValue := nil
5:    action^propose := nil
6:    action^check := nil
7:    action^send := nil
8:    validValue[] := {⊥, ⊥, ..., ⊥}                            /* validValue[r] ∈ {⊥, false, true} */

9:  Round PROPOSE(t):
10:   Send phase:
11:     if i == isProposer(t, h) then
12:       action^propose ← σ_i^propose()   /* σ_i^propose() ∈ {false, true} sets the action of proposing a valid block or an invalid one */
13:       if action^propose == true then
14:         proposal ← createValidValue(h)
15:       else if action^propose == false then
16:         proposal ← createInvalidValue()
17:       broadcast ⟨PROPOSE, h, t, proposal⟩
18:   Delivery phase:
19:     delivery ⟨PROPOSE, h, t, v⟩ from proposer(h, t)
20:   Compute phase:
21:     action^check ← σ_i^check()   /* σ_i^check() ∈ {false, true} sets the action of checking or not the validity of the proposal */
22:     if action^check == true then
23:       validValue[t] ← isValid(v)                  /* The execution of isValid(v) has a cost c_check */
24:     action^send ← σ_i^send(validValue[t])   /* σ_i^send : {⊥, false, true} → {false, true} sets the action of sending the vote or not */
25:     if action^send == true then
26:       vote ← v                  /* The participant decides to send the vote, the proposal might be invalid */

27: Round VOTE(t):
28:   Send phase:
29:     if vote ≠ nil then
30:       broadcast ⟨VOTE_i, h, t, vote⟩                  /* The execution of the broadcast has a cost c_send */
31:   Delivery phase:
32:     delivery ⟨VOTE, h, t, v⟩          /* The participant collects all the votes for the current height and round */
33:   Compute phase:
34:     if |⟨VOTE, h, t, v⟩| ≥ ν ∧ decidedValue = nil ∧ vote ≠ nil ∧ vote = v then
35:       decidedValue = v; exit
36:     else
37:       vote ← nil
38:       t ← t + 1
```

Strategy $\sigma_i^{\text{propose}}$ determines if the proposer i chooses to produce a valid proposal or an invalid one (Lines 12 - 16 of Algorithm 7). In both cases, the proposal is sent in broadcast (Line 17 of Algorithm 7).

6.1.3 Game

Action Space. As proposer, the participant decides which block to send to each of the other participants. This means to choose whether to send valid or invalid blocks. Then, at each round t, after receiving a block, each participant first decides whether to check the block's validity or not (at cost c_{check}), and second decides whether to send a message (at cost c_{send}) or not.

Information Sets. At the beginning of each round $t > 1$, the information set of the participant, η_i^t, includes the observation of the round number t, the participant's own type θ_i, as well as the observation of what happened in previous rounds, namely (i) whether the participant decided to check validity, and if it did, the knowledge of whether the block was valid or not, (ii) how many messages were sent, and (iii) whether a block was produced or not. At round 1, if i is strategic, η_i^1 only includes the participant's private information about its own type, θ_i, if i is a malicious participant, it knows which participants are malicious, η_i^1 also includes the private information of all participant's type.

Then, during each round $t > 1$, the participant decides whether to check the validity of the current block. At this point, denoting by b_t the block proposed at round t, when the participant does not decide to check validity, the $\texttt{isValid}(b_t)$ function is set to the null information set, while if the participant decides to check, $\texttt{isValid}(b_t)$ is equal to `true` if the block is valid and `false` otherwise. Therefore, at this stage, the participant information set becomes $H_i^t = \eta_i^t \cup \texttt{isValid}(b_t)$, which is η_i^t augmented with the validity information participant i has about b_t, the proposed block.

Strategies. At each round $t \geq 1$, the strategy of participant i is a mapping from its information set into its actions. If the participant is selected to propose the block, its choice is given by $\sigma_i^{\text{propose}}(\eta_i^t)$. Then, at the point at which the participant can decide to check block validity, its strategy is given by $\sigma_i^{\text{check}}(\eta_i^t)$. Finally, after making that decision, the participant must decide whether to send a message or not, and that decision is given by $\sigma_i^{\text{send}}(H_i^t)$.

Rewards and Costs for Strategic Participants. In this chapter, we study the case where when a block is produced, only the participants which sent a vote message are rewarded (and receive R).

In addition, we assume that when an invalid block is produced, all strategic participants incur a cost κ.

The reward R, given to the participants when a block is produced, is larger than the cost c_{check} of checking validity, which in turn is larger than the cost c_{send} of sending a vote message. Additionally, we assume that the reward obtained when a block is produced is smaller than the cost κ of producing an invalid block. That is, $\kappa > R > c_{check} > c_{send} \geq 0$.

Payoff of Malicious Participants. Let ω be an outcome of the game. If ω does not satisfy Termination, then the malicious participants have a payoff of $\tilde{\kappa}_{\text{Termination}}$. If ω does not satisfy Validity, then the malicious participants have a payoff of $\tilde{\kappa}_{\text{Validity}}$.

$$\tilde{\kappa}(\omega) = \begin{cases} \tilde{\kappa}_{\text{Validity}}, & \text{if } \omega \text{ does not satisfy Validity} \\ \tilde{\kappa}_{\text{Termination}}, & \text{if } \omega \text{ does not satisfy Termination} \\ 0, & \text{if } \omega \text{ satisfies Agreement, Validity and Termination} \end{cases},$$

The malicious participants have lexicographic preferences on the outcome of the game, in order, they prefer:

1. Outcomes that do satisfy Termination, but not Validity;
2. Outcomes that do satisfy Validity, but not Termination;
3. Outcomes that do satisfy Termination and Validity;

which is summarised by the following assumption, $\tilde{\kappa}_{\text{Validity}} > \tilde{\kappa}_{\text{Termination}} > 0$. We also assume that malicious participants do not incur any costs, no matter their actions.

Objective of Strategic Participants. Let T be the endogenous round at which the game stops. If a block is produced at round $t \leq n$, then $T = t$. Otherwise, if no block is produced, $T = n+1$. In the latter case, the *Termination* property is not satisfied. At the beginning of the first round, the expected gain of strategic participant i is:

$$U_i = E\left[\begin{array}{c}\left(R \times \mathbb{1}_{(\sigma_i^{\text{send}}(H_i^T)=\texttt{true})} \times \mathbb{1}_{(\text{block produced at } T)} - \kappa \times \mathbb{1}_{(\text{invalid block produced})}\right) \\ -\sum_{t=1}^{T}\left(c_{\text{check}} \times \mathbb{1}_{\sigma_i^{\text{check}}(\eta_i^t)=\texttt{true})} + c_{\text{send}} \times \mathbb{1}_{(\sigma_i^{\text{send}}(H_i^t)=\texttt{true})}\right)\end{array} \middle| \eta_i^1\right],$$

where $\mathbb{1}_{(.)}$ denotes the indicator function, taking the value 1 if its argument is `true`, and 0 if it is `false`.

Then, at the beginning of round $t > 1$, if $T \geq t$, the continuation payoff of the strategic participant with the information set η_i^t is:

$$W_{i,t}(\eta_i^t) = E\left[\begin{array}{c}\left(R \times \mathbb{1}_{(\sigma_i^{\text{send}}(H_i^T)=\texttt{true})} \times \mathbb{1}_{(\text{block produced at } T)} - \kappa \times \mathbb{1}_{(\text{invalid block produced})}\right) \\ -\sum_{s=t}^{T}\left(c_{\text{check}} \times \mathbb{1}_{(\sigma_i^{\text{check}}(h_i^s)=\texttt{true})} + c_{\text{send}} \times \mathbb{1}_{(\sigma_i^{\text{send}}(H_i^s)=\texttt{true})}\right)\end{array} \middle| \eta_i^t\right]. \tag{6.1}$$

Objective of Malicious Participants. Let T be the endogenous round at which the game stops. If a block is produced at round $t \leq n$, then $T = t$. Otherwise, if no block is produced, $T = n+1$. In the latter case, the *Termination* property is not satisfied. At the beginning of the first round, the expected payoff of malicious participant i is:

$$\tilde{U}_i = E\left[\ \tilde{\kappa}_{\text{Validity}} \times \mathbb{1}_{(\text{invalid block produced at } T)} \ \middle| \eta_i^1\right],$$

and if no block is produced at round n, the expected payoff of malicious participant i is:

$$\tilde{U}_i = \tilde{\kappa}_{\text{Termination}},$$

Then, at the beginning of round $t > 1$, if $T \geq t$, the continuation payoff of the malicious participant with the information set η_i^t is:

$$\tilde{W}_{i,t}(\eta_i^t) = E\left[\ \tilde{\kappa}_{\text{Validity}} \times \mathbb{1}_{(\text{invalid block produced at } T)} + \tilde{\kappa}_{\text{Termination}} \times \mathbb{1}_{(T=n+1)} \ \middle| \eta_i^t\right].$$

Equilibrium concept. We assume in this Chapter that there is at least one malicious in the system ($\mathfrak{m} \geq 1$). If there are no malicious, the results from Chapter 5.2 apply, if not (*i.e.*, $\mathfrak{m} \geq 1$) let $\mathfrak{m}$ be a random variable in $\{1, \ldots, n\}$, it means that the number of malicious participants in the committee may not be known in advance.

In this section, as opposed to Chapter 5 we study the case of multiple rounds. Meaning, if there is no decision at round 1, the participants go to round 2 and so on. Moreover, strategic participants do not know the indices of the malicious participants. The game is then dynamic

and with asymmetric information. The relevant equilibrium concept to consider is *Perfect Bayesian Equilibrium* (Definition 2.13 of Chapter 2) intuitively defined as follows:

A Perfect Bayesian equilibrium is such that all participants 1) choose actions maximising their objective function, 2) rationally anticipate the strategies of the others, and 3) draw rational inferences from what they observe, using their expectations about the strategies of the others and Bayes law, whenever it applies.

In a perfect Bayesian equilibrium, participants best-respond to one another. It imposes additional restrictions to take into account the fact that the game is dynamic and that participants can have private information, and therefore, must draw rational inferences from their observation of actions and outcomes. The rationality of inferences in perfect Bayesian equilibrium implies that (i) each participant has rational expectations about the strategies of the others, and (ii) each participant's beliefs are consistent with Bayes law when computing probabilities conditional on events that have strictly positive probability on the equilibrium path. Perfection in perfect Bayesian equilibrium implies that at each node starting a subgame the participants' strategies form a Nash equilibrium of that subgame. In this context, to show that a strategy is optimal it is sufficient to show that it dominates any one-shot deviation [33].

6.2 Equilibrium where Validity is not Guaranteed

First, we show that finding an equilibrium satisfying the consensus properties is not easy. We show in the next proposition that there exists an equilibrium where any proposed block is produced at the end of the first round. This equilibrium in the proposition violates the Validity property.

Proposition 6.1. *Let* $\mathfrak{m}$ *be a random variable such that* $\mathfrak{m} \leq n - \nu - 1$*, then there exists a Perfect Bayesian equilibrium described as follows:*

- *As proposer:*
  - *a strategic participant proposes a valid block;*
  - *a malicious participant proposes an invalid block.*
- *When receiving a proposed block,*
  - *strategic participants do not check the block validity and always send a message;*
  - *malicious participants can either do not check and always send, or do check and send a message if and only if the block is invalid.*

Since $\mathfrak{m} + \mathfrak{s} = n$, the condition of the proposition which can be rewritten as $\mathfrak{s} \geq \nu + 1$ implies that, when all strategic participants but one send a message, they meet the majority threshold ν, such that the block is produced. The condition also means that $\mathfrak{m} \in \{1, \ldots, n - \nu - 1\}$, which implies that, if all participants send a message, no strategic participant is pivotal, so the block is produced with or without its vote. Under these conditions, each strategic participant understands it is not pivotal: if the block is invalid, strategic participants will send messages, so that the block will be produced irrespective of the strategic participant's own action. Moreover, if the block is valid, malicious participants have no preference about checking or not a block validity, since the blocks proposed will be produced at the end of the first round.

Thus, strategic participants understand that they are not pivotal and that whatever they do, given the equilibrium behaviour of the other strategic participants and of the malicious

participants, all blocks will be produced. Consequently, they have no interest in checking the validity of the block. In fact, their expected gain when they send a message is:

$$R - c_{\text{send}} - \Pr(\theta_1 = \theta^m)\kappa.$$

Since no strategic participant by itself is pivotal, if a strategic participant decides to check the block's validity and check if and only if the block is valid, then its expected gain is

$$-c_{\text{check}} + (1 - \Pr(\theta_1 = \theta^m))(R - c_{\text{send}}) - \Pr(\theta_1 = \theta^m)\kappa,$$

which is lower than the gain at equilibrium.

Finally, note that, in the equilibrium of Proposition 6.1, a block is produced at round 1, so the *Termination* property is satisfied, but, when the proposer is a malicious participant, invalid blocks are produced, so the *Validity* property is not satisfied.

Proof

- First, we study the strategic participants' behaviours. If a strategic participant is selected to be the proposer, it prefers to propose a valid block than to propose an invalid block. Indeed, if it proposes an invalid block, that block will be produced. In that case, the gain of the proposer is $R - c_{\text{check}} - c_{\text{send}} - \kappa$. If instead, the strategic participant proposes a valid block, this block will be produced and his gain will be $R - c_{\text{check}} - c_{\text{send}}$.

  Now, turn to the actions of strategic participants who are not proposers. The equilibrium gain of these participants is

  $$R - c_{\text{send}} - \Pr(\theta_1 = \theta^m)\kappa.$$

  If instead of playing the equilibrium strategy, a strategic participant does not send a message, its expected gain is $-\Pr(\theta_1 = \theta^m)\kappa$, which by assumption ($R > c_{\text{send}}$) is lower than the equilibrium expected gain.

  Another deviation is to check the block's validity and send a message only if the block is valid, which brings expected gain equal to

  $$-c_{\text{check}} + (1 - \Pr(\theta_1 = \theta^m))\,(R - c_{\text{send}}) - \Pr(\theta_1 = \theta^m)\kappa.$$

  This is lower than the equilibrium expected gain if

  $$-c_{\text{send}} + R - \Pr(\theta_1 = \theta^m)\kappa > -c_{\text{check}} + (1 - \Pr(\theta_1 = \theta^m))\,(R - c_{\text{send}}) - \Pr(\theta_1 = \theta^m)\kappa,$$

  which holds since it is equivalent to

  $$0 > -c_{\text{check}} - \Pr(\theta_1 = \theta^m)(R - c_{\text{send}}).$$

  The other possible deviations are also dominated: Checking the block's validity and sending a message only when the block is invalid, yields expected gain

  $$-c_{\text{check}} + \Pr(\theta_1 = \theta^m)(R - c_{\text{send}} - \kappa),$$

  which is lower than the equilibrium expected gain. Checks the validity of the block and always sending a message yields

  $$R - c_{\text{send}} - \Pr(\theta_1 = \theta^m)\kappa - c_{\text{check}},$$

  which is again dominated, as is also checking and not sending, which yields $-c_{\text{check}} - \Pr(\theta_1 = \theta^m)\kappa$.

- Now, we focus on malicious participants. If a malicious participant is selected to be the proposer, it prefers to propose an invalid block than to propose a valid block. Indeed, if it proposes a valid block, that block will be produced. In that case, the gain of the proposer is 0. If instead, the malicious proposer proposes an invalid block, this block will be produced and his gain will be $\tilde{\kappa}_{\text{Validity}}$.

  If a malicious participant is not a proposer, its expected gain at equilibrium is:

$$\Pr(\theta_1 = \theta^m)\tilde{\kappa}_{\text{Validity}}.$$

  All its deviations yield the same expected gain since $\nu < n$.

$\square_{Proposition\ 6.1}$

In some sense, the situation where too many participants are supposed to check the proposal validity than needed is not stable. In fact, a single strategic participant will prefer not to check since the others are supposed to do so, leading to the disastrous situation of producing invalid blocks.

6.3 Equilibrium where Termination is not Guaranteed

Even if the previous equilibrium (Proposition 6.1) does violate the Validity property, it does ensure the Termination property. In contrast, we may as well have an equilibrium where the Termination property is violated. We present the latter in the following proposition.

Proposition 6.2. *Let $\mathfrak{m}$ be a random variable such that $\mathfrak{m} < \nu$ and $n - \mathfrak{m} \geq \nu$, then there exists a Nash equilibrium in which strategic participants never check blocks' validity nor send messages, and malicious participants check validity but send a message only if the block is invalid, so that no block is ever produced.*

Condition $\mathfrak{m} < \nu$ in Proposition 6.2 implies that malicious participants cannot reach the threshold on their own. This precludes producing invalid blocks. Therefore, the *Validity* property is satisfied. Unfortunately, the condition also implies that there exists an equilibrium in which the *Termination* property also fails to hold. The intuition is the following:

In Proposition 6.2, each strategic participant anticipates that no other participant will send a message when the block is valid[1]. In this context, each strategic participant knows that, if it were to send a message in favour of a valid block, it would be the only one to do so. Because the threshold ν is strictly larger than 1, the block would not be produced. Therefore, sending a message is a dominated action for the strategic participant. The equilibrium in Proposition 6.2 reflects that strategic participants' actions are strategic complements and they must coordinate on sending messages to have valid blocks produced. Proposition 6.2 shows that, in equilibrium, there can be a coordination failure, such that no block is ever produced.[2]

Note that even if malicious participants also do not check validity nor check, the resulting strategy profile is still an equilibrium.

[1] Malicious participants send messages but only when the block is invalid.

[2] If $\mathfrak{m} = 0$, and with $\nu = 1$, there exists a unique equilibrium (Proposition 5.2), in which all participants check validity and send a message if and only if the block is valid. In that equilibrium Validity and Termination are satisfied. However, this obtains only if there are no malicious participants. As soon as $\mathfrak{m} \geq 1$, if $\nu = 1$, Proposition 6.1 applies and validity is not satisfied.

Proof

Consider a strategic participant who anticipates that other strategic participants will not send any message at any round. If it follows the equilibrium strategy and does not send a message, its gain is 0. This must be compared to the gain of the participant if its deviates:

- If it sends a message without checking its expected gain is
$$-c_{\text{send}} + \Pr(\text{invalid}) \Pr(\mathfrak{m} = \nu - 1)(R - \kappa).$$
- If it checks the block's validity and sends a message only when the block is valid, its expected gain is
$$-c_{\text{check}} - \Pr(\text{valid})c_{\text{send}}.$$
- If it checks the block's validity and sends a message only when the block is invalid, its expected gain is
$$-c_{\text{check}} + \Pr(\text{invalid})(\Pr(\mathfrak{m} = \nu - 1)(R - \kappa) - c_{\text{send}}).$$
- If it checks the validity of the block and always sends a message, its expected gain is
$$-c_{\text{send}} - c_{\text{check}} + \Pr(\text{invalid}) \Pr(\mathfrak{m} = \nu - 1)(R - \kappa).$$
- If it checks and does not send a message, its gain is $-c_{\text{check}}$.

Clearly, the participant is better off following the equilibrium strategy.

Consider now a malicious participant. It payoff at equilibrium is $\tilde{\kappa}_{\text{Termination}}$. No matter its action, no block can be produced and the payoff of the malicious participant will remain $\tilde{\kappa}_{\text{Termination}}$. So it weakly prefers following the equilibrium strategy.

□*Proposition 6.2*

6.4 Equilibrium where both Termination and Validity are Guaranteed

While there exists an equilibrium in which either termination (Proposition 6.2) is not verified or validity (Proposition 6.1) is not guaranteed, this does not necessarily imply there is no equilibrium that satisfies both termination and validity. To have both properties, it must be that, in equilibrium, while malicious participants propose and send messages for invalid blocks, sufficiently many strategic participants find it in their own interest to check the validity of the block and to send vote messages in support of valid blocks. The problem is that some strategic participants might be tempted to free ride and let the others bear the cost of checking. To avoid this situation, it must be that (at least some) strategic participants anticipate they are pivotal, *i.e.*, if they fail to check block validity and send messages in support of valid blocks, this may derail the participant at their own expense.

To make this point, we look for an equilibrium in which some strategic participants check the validity of the block and send a message if and only the block is valid, and this results in valid blocks being immediately produced and invalid blocks being rejected. Before proving that such an equilibrium exists, we characterise the expected continuation payoffs to which it would give rise.

Lemma 6.3. *Consider a candidate equilibrium in which some strategic participants check the validity of the block and send a message if and only if the block is valid, while the other strategic participants send messages without checking validity, and this results in valid blocks being immediately produced and invalid blocks being rejected. In such an equilibrium, if it exists, the expected payoff of malicious participants is*

$$\pi_{malicious} = 0,$$

while the continuation payoff, at round t, of the strategic participants who are supposed to check block validity is

$$\pi_{check}(t) = R - c_{send} - \phi(t)c_{check},$$

while the expected continuation payoff, at round t, of the strategic participants who are not supposed to check block validity is

$$\pi_{send}(t) = R - \psi(t)c_{send},$$

where $\phi(\mathfrak{m}) = 1$, $\psi(\mathfrak{m}+1) = 1$ and both ϕ and ψ satisfy property P defined below.

Definition 6.4. *A function g satisfies property P, if $g(t) = 1 + \frac{\mathfrak{m}-t+1}{n-t+1}g(t+1), \forall t < \mathfrak{m}$.*

In the candidate equilibrium, participants will reach a point at which the block is valid and all strategic participants send a message so that the block is produced. This gives rise to a payoff $R - c_{\text{send}}$, the first part of $\pi_{check}(t)$. The second part of $\pi_{check}(t)$, $\phi(t)c_{\text{check}}$, is the expected cost of checking the block validity, where $\phi(t)$ is the expected number of times the participant expects to check validity before a block is produced. Similarly, in $\pi_{send}(t)$, $\psi(t)c_{\text{check}}$ is the expected cost of sending messages, where $\psi(t)$ is the expected number of times the participant expects to send messages before a block is produced.

Proof

We prove this Lemma in 3 parts:

1. Proof of the first part of the lemma, concerning the malicious participants: Since in the candidate equilibrium there is always a valid block produced, then the malicious participants have a gain of 0.

2. Proof of the second part of the lemma, concerning the strategic participants who are expected to check validity:

   At round $t = \mathfrak{m}$, participants know that all $\mathfrak{m} - 1$ previous proposers were malicious and that there are now $n-\mathfrak{m}+1$ potential proposers, out of which only one is malicious and $n - \mathfrak{m} > \nu$ are strategic. The expected gain of the strategic participants who are supposed to check is

   $$-c_{\text{check}} + \frac{n-\mathfrak{m}}{n-\mathfrak{m}+1}(R - c_{\text{send}}) + \frac{1}{n-\mathfrak{m}+1}(R - c_{\text{send}}),$$

   where the first term is the cost of checking validity, the second term corresponds to the case in which the current proposer is strategic and proposes a valid block that is immediately produced, and the third term corresponds to the case in which the proposer is malicious, the block is rejected, and we move to the next round, at which a valid block is finally produced (without needing any further validity check). This equilibrium payoff simplifies to

   $$R - c_{\text{send}} - c_{\text{check}},$$

reflecting that eventually a valid block will be produced, and that from round $\mathfrak{m}$ on the participant will need to check validity only once. This equilibrium payoff implies that

$$\phi(\mathfrak{m}) = 1.$$

Now turn to round $t < \mathfrak{m}$. If round $t \leq \mathfrak{m}$ is reached, the previous $t-1$ proposers were malicious. There remains $n-(t-1)$ potential proposers. Out of them a fraction

$$\frac{\mathfrak{m}-(t-1)}{n-t+1}$$

is malicious, while the complementary fraction

$$\frac{n-\mathfrak{m}}{n-t+1}$$

is strategic. This fraction being the probability that the next proposer is strategic.

To prove the property stated in the Proposition by backward induction, we now prove that if this property is satisfied at round $t+1$, that is if

$$\pi_{check}(t+1) = R - c_{\text{send}} - \phi(t+1)c_{\text{check}},$$

then it is satisfied at round t.

Suppose the strategic participant follows the equilibrium strategy of checking and sending if and only if the block is valid. Its expected gain from round t on is

$$-c_{\text{check}} + \frac{n-\mathfrak{m}}{n-t+1}(R - c_{\text{send}}) + \frac{\mathfrak{m}-(t-1)}{n-t+1}\pi(t+1),$$

where the first term is the cost of checking the block at round t, the second term is the probability that the block is valid and produced multiplied by the payoff in that case, and the third term is the probability that the block is invalid and rejected multiplied by the payoff in that case. Substituting the value of $\pi_{check}(t+1)$, using that the property is verified at round $t+1$, the expected gain writes as

$$-c_{\text{check}} + \frac{n-\mathfrak{m}}{n-t+1}(R - c_{\text{send}}) + \frac{\mathfrak{m}-(t-1)}{n-t+1}(R - c_{\text{send}} - \phi(t+1)c_{\text{check}}).$$

That is

$$R - c_{\text{send}} - \left(1 + \frac{\mathfrak{m}-(t-1)}{n-t+1}\phi(t+1)\right) c_{\text{check}},$$

which, using the definition of $\phi(t)$, is $R - c_{\text{send}} - \phi(t)c_{\text{check}}$.

3. Proof of the third part of the lemma, concerning the strategic participants who are just expected to send messages:

   Again, we prove that if the property is satisfied at round $t+1$, i.e., $\pi_{send}(t+1) = R - \psi(t+1)c_{\text{send}}$, then it is satisfied at round t. Suppose the strategic participant follows the equilibrium strategy of not checking blocks' validity and always sending a message. Its expected gain from round t on is

$$c_{\text{send}} + \frac{n-\mathfrak{m}}{n-t+1}R + \frac{\mathfrak{m}-t+1}{n-t+1}\pi_{send}(t+1),$$

where the first term is the cost of sending a message at round t, the second term is the probability that the block is valid and produced multiplied by the payoff in that case, and the third term is the probability that the block is invalid and rejected multiplied by the continuation payoff in that case. Substituting the value of $\pi_{send}(t+1)$, the expected gain writes as

$$-c_{\text{send}} + \frac{n-\mathfrak{m}}{n-t+1}R + \frac{\mathfrak{m}-t+1}{n-t+1}(R - \psi(t+1)c_{\text{send}}).$$

That is

$$R - \left(1 + \frac{\mathfrak{m}-t+1}{n-t+1}\psi(t+1)\right) c_{\text{send}},$$

which, using the definition of $\psi(t)$, is $R - \psi(t)c_{\text{send}}$.

$\square_{Lemma\ 6.3}$

Relying on Lemma 6.3, we now establish that our candidate equilibrium is indeed an equilibrium. To do so, denote the highest index of all malicious participants by i_M; formally, $i_M = \max\{i : \theta_i = \theta^m\}$. We recall that each participant knows its own index in the committee.

Proposition 6.5. *Let $\mathfrak{m}$ the number of malicious participants be a constant such that $\mathfrak{m} < \nu$ and $n - \mathfrak{m} > \nu$; let also $\mathfrak{m}$ be known by all participants. If the cost κ of producing an invalid block is large enough, in the sense that*

$$\kappa > \alpha(t)c_{check} - \beta(t)c_{send}, \forall t < \mathfrak{m}, \tag{6.2}$$

where

$$\alpha(t) = \frac{(n-t+1)\phi(t) - (\mathfrak{m}-t+1)\Pr(i_M \geq n-\nu+\mathfrak{m}+2 | T \geq t)\phi(t+1)}{(\mathfrak{m}-t+1)\Pr(i_M < n-\nu+\mathfrak{m}+2 | T \geq t)}$$

and

$$\beta(t) = \frac{\Pr(i_M \geq n-\nu+\mathfrak{m}+2 | T \geq t)}{\Pr(i_M < n-\nu+\mathfrak{m}+2 | T \geq t)},$$

in addition, if the reward is large enough relative to the costs in the sense that

$$R \geq \max\left[\frac{n}{n-\mathfrak{m}}c_{send}, c_{send} + \frac{n}{n-\mathfrak{m}}c_{check}\right],$$

there exists a Perfect Bayesian equilibrium in which the strategy of participants is the following:

- *As proposer, a strategic participant proposes a valid block, while a malicious participant proposes an invalid block.*
- *At any round $t \leq \mathfrak{m}$, when receiving a proposed block, (i) the strategic participants with index $i \in \{t, \ldots, n-\nu+\mathfrak{m}+1\}$ check the block validity and send a message only if the block is valid, while (ii) the strategic participants with index $i \in \{n-\nu+\mathfrak{m}+2, \ldots, n\}$ do not check the validity of the block but send a message, and (iii) malicious participants check the blocks' validity and send a message if and only if the block is invalid.*
- *If round $t = \mathfrak{m}+1$ is reached, strategic participants send a message without checking if the block is valid. At this point, the block is valid and produced.*

Hence, in equilibrium, termination occurs no later than at round $\mathfrak{m}+1$.

On the equilibrium path, invalid blocks (proposed by malicious participants) are rejected, while valid blocks (proposed by strategic participants) are produced. This implies that, if round $t=\mathfrak{m}+1$ is reached, the participants know that during all the previous ($\mathfrak{m}$) rounds the proposers were malicious (to draw this inference, the strategic participants use their anticipation that all participants play equilibrium strategies; hence the perfect Bayesian nature of the equilibrium). Consequently, at round $\mathfrak{m}+1$, the proposer must be strategic, and all participants anticipate the proposed block is valid. Therefore, no strategic participant needs to check the validity of the block but all send a message, which brings them expected gain equal to $R-c_{\text{send}}$. This is larger than their gain from deviating (e.g., by not sending a message or by checking the blocks).

At previous rounds $t\leq\mathfrak{m}$, participants know that all $t-1$ previous proposers were malicious and that there remains $\mathfrak{m}-t+1$ malicious participants with index strictly larger than $t-1$ (as above, this rational inference is a feature of the perfect Bayesian equilibrium we characterise). Do the equilibrium strategies of the strategic participants preclude the production of an invalid block by malicious participants? To examine this point, consider the maximum possible number of messages that can be sent if the proposer is malicious. In equilibrium the $\nu-\mathfrak{m}-1$ participants with indexes strictly larger than $n-\nu+\mathfrak{m}+1$ are to send a message without checking it. The worst-case scenario (maximising the number of messages sent when the block is invalid) is that none of these participants is malicious. In that case, in equilibrium, the number of messages sent when the block is invalid is $\mathfrak{m}+(\nu-\mathfrak{m}-1)=\nu-1$, so that we narrowly escape production of the invalid block. In contrast, if one of the strategic participants deviated from equilibrium and sent a message without checking the block, in the worst-case scenario, this would lead to producing an invalid block. Thus, in that sense, the strategic participants with index strictly lower than $n-\nu+\mathfrak{m}+1$ are pivotal. Hence, they check block validity, because, under the condition stated in the proposition, the cost of producing an invalid block is so large that strategic participants do not want to run that risk.

Proof

For clarity, we decompose the proof in 6 steps.

1. The first step concerns the possible deviations of malicious participants. At equilibrium, the gain of a malicious participant is 0. The malicious participants weakly prefer the equilibrium strategy, since all deviations of a malicious participant yield to a payoff of 0. In fact, in equilibrium, if a malicious participant deviates by sending a valid block, the block will be immediately produced, hence a gain of 0. Since checking has no cost for the malicious participants, they prefer to check, since it gives them information about the block's validity. For a malicious participant, in equilibrium, sending a message does not have an impact on which block is decided or not, valid blocks are immediately produced, and invalid blocks are never produced, but since sending a message does not cost anything for the malicious participant, it can send messages for invalid blocks with no impact.

2. This step is to note that strategic proposers strictly prefer to propose a valid block than an invalid one. This is because, when they follow their equilibrium strategy of proposing a valid block, it is produced and the proposer gets $R-c_{\text{check}}-c_{\text{send}}$, while if they propose an invalid block, it is rejected, and we move to the next round, in which, in equilibrium, the participant gets at most $R-c_{\text{check}}-c_{\text{send}}$ (and possibly less). Indeed, this participant incurs the cost of checking validity at the next round, because the strategic

participants who are not expected to check validity have indexes above $n-\nu+\mathfrak{m}+1$, which are above $\mathfrak{m}+1$, so that they do not get to propose blocks.

3. The next step concerns the actions of the strategic participants when round $t=\mathfrak{m}+1$ is reached. At that round, all participants know the proposer must be strategic and the proposed block valid. In equilibrium, no strategic checks validity but all send a message. Any other action would be dominated.

4. The fourth step concerns the most relevant deviation, in which a strategic participant expected to check block validity fails to do so. If at round t a strategic participant supposed to check, deviates and sends a message without checking block validity, its expected continuation payoff is

$$\left(1-\frac{\mathfrak{m}-(t-1)}{n-t+1}\right)(R-c_{\text{send}})+\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M<n-\nu+\mathfrak{m}+1|T\geq t)\,(R-c_{\text{send}}-\kappa)$$
$$+\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M\geq n-\nu+\mathfrak{m}+1|T\geq t)\,(\pi(t+1)-c_{\text{send}})\,.$$

The first term is the payoff obtained by the deviating strategic participant if the current block is valid, and therefore immediately produced. The second term is the payoff obtained by the deviating participant when he was pivotal and triggered production of an invalid block. To see this, consider the number of messages when the block is invalid, the strategic participant is deviating and the indexes of all the malicious participants are strictly lower than $n-\nu+\mathfrak{m}+2$: $\mathfrak{m}$ messages are sent by the malicious participants, 1 message is sent by the deviating strategic participant, $\nu-\mathfrak{m}-1$ messages are sent by the strategic participants with an index above than or equal to $n-\nu+\mathfrak{m}+2$. The resulting total number of messages is ν and the block is produced. The last term corresponds to the case in which the deviating strategic participant is not pivotal, and the invalid block is not produced, so that we move to the next round.

Substituting the value of $\pi_{check}(t+1)=R-c_{\text{send}}-\phi(t+1)c_{\text{check}}$ from Lemma 6.3, the expected continuation value of the deviating participant is

$$\left(1-\frac{\mathfrak{m}-(t-1)}{n-t+1}\right)(R-c_{\text{send}})+\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M<n-\nu+\mathfrak{m}+2|T\geq t)\,(R-c_{\text{send}}-\kappa)$$
$$+\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M\geq n-\nu+\mathfrak{m}+2|T\geq t)\,(R-c_{\text{send}}-\phi(t+1)c_{\text{check}}-c_{\text{send}})\,.$$

Or

$$(R-c_{\text{send}})-\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M<n-\nu+\mathfrak{m}+2|T\geq t)\kappa$$
$$-\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M\geq n-\nu+\mathfrak{m}+2|T\geq t)\,(\phi(t+1)c_{\text{check}}+c_{\text{send}})\,.$$

The equilibrium condition is that this deviation payoff must be lower than the equilibrium continuation payoff of the participant

$$R-c_{\text{send}}-\phi(t)c_{\text{check}}.$$

That is

$$\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M < n-\nu+\mathfrak{m}+2|T\geq t)\kappa > \phi(t)c_{\text{check}}$$
$$-\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M \geq n-\nu+\mathfrak{m}+2|T\geq t)\left(\phi(t+1)c_{\text{check}}+c_{\text{send}}\right).$$

Note that

$$\phi(t)\geq\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M \geq n-\nu+\mathfrak{m}+2|T\geq t)\phi(t+1),$$

since by the definition of $\phi(t)$ this inequality is equivalent to

$$1+\frac{\mathfrak{m}-(t-1)}{n-t+1}\phi(t+1)\geq\frac{\mathfrak{m}-(t-1)}{n-t+1}\Pr(i_M \geq n-\nu+\mathfrak{m}+2|T\geq t)\phi(t+1),$$

which indeed holds. Thus, we can write the equilibrium condition as

$$\kappa > \alpha(t)c_{\text{check}}-\beta(t)c_{\text{send}}, \forall t<\mathfrak{m},$$

as stated in the proposition.

5. Other possible deviations for strategic participant supposed to check block's validity are easier to rule out:

   First, the participant could do nothing (neither check nor send). Relative to the equilibrium payoff, this deviation economises the cost of checking (c_{check}). If the current proposer is malicious, the participant then obtains the same payoff after a one-shot deviation as on the equilibrium path ($\pi_{check}(t+1)$). If the current proposer is strategic, the block is produced, but the participant does not earn any reward. Therefore, the deviation is dominated if

   $$\frac{n-\mathfrak{m}}{n-t+1}(R-c_{\text{send}})\geq c_{\text{check}},$$

   which holds under the condition, stated in the proposition, that $R \geq \max\left[\frac{n}{n-\mathfrak{m}}c_{\text{send}}, c_{\text{send}}+\frac{n}{n-\mathfrak{m}}c_{\text{check}}\right]$.

   Second, the participant could check the block validity, and then send a message irrespective of whether the block is valid or not. This would generate a lower payoff than the main deviation, shown above (in 4.) to be dominated.

   Third, the participant could check validity but then send no message. When the current proposer is a malicious participant, this one-shot deviation yields the same payoff as the equilibrium strategy. When the current proposer is strategic, this deviation yields a payoff of $-c_{\text{check}}$, which is lower than the equilibrium payoff $R-c_{\text{send}}-c_{\text{check}}$.

   Fourth, the participant could check the block's validity and send a message only if the block is invalid, which is trivially dominated.

6. Finally, turn to deviations of strategic participants supposed to send messages without checking blocks' validity.

   First, consider the possibility to abstain from sending a message. This economises the costs c_{send}, but, in case the block is valid and produced, this implies the participant loses the reward R. The deviation is then dominated if

   $$\frac{n-\mathfrak{m}}{n-t+1}R\geq c_{\text{send}},$$

which holds under the condition, stated in the proposition, that $R \geq \max\left[\frac{n}{n-\mathfrak{m}}c_{\text{send}}, c_{\text{send}} + \frac{n}{n-\mathfrak{m}}c_{\text{check}}\right]$.

Second, consider the possibility of checking validity and sending a message only for valid blocks. This deviation would imply the participant would have to incur the cost of checking (c_{check}), but it would economise the cost of sending a message when the block is invalid. The deviation is dominated if

$$c_{\text{check}} \geq \frac{\mathfrak{m} - t + 1}{n - t + 1} c_{\text{send}},$$

which holds, since by assumption $c_{\text{check}} \geq c_{\text{send}}$.

Other deviations, such as checking validity but never sending messages, or checking validity and always sending messages, or checking validity and sending only if the block is invalid, are trivially dominated.

□*Proposition 6.5*

6.4.1 Estimation of κ in Proposition 6.5

Recall that from Lemma 6.3, $\forall t < \mathfrak{m}, \phi(t) = 1 + \frac{\mathfrak{m}-t+1}{n-t+1}\phi(t+1)$ and $\phi(\mathfrak{m}) = 1$. The distribution of malicious participants in the committee is given by a Bernoulli distribution. The repetition of binomial here is not a Bernoulli since the draws are dependent. If all malicious participants are already placed in the committee, say with indices lower than t, then at index $t+1$ the participant is not a malicious any more. We need to compute $\forall t \leq n, \Pr(i_M \geq n - \nu + \mathfrak{m} + 2)$.

Note that $\forall t < \mathfrak{m}, \Pr(i_M = t) = 0$.

Let $a \geq 0$ such that $\mathfrak{m} + a \leq n$, the probability $\Pr(i_M = \mathfrak{m} + a)$ is equal to:

$$\sum_{i_0=0<i_1<i_2<\cdots<i_a\leq \mathfrak{m}+a-1} \left(\prod_{\lambda=0}^{a-1} \prod_{k=i_\lambda+1}^{i_{\lambda+1}-1} \frac{\mathfrak{m} - k + (\lambda+1)}{n-k+1} \times \frac{n - \mathfrak{m} - \lambda}{n - i_{\lambda+1} + 1} \right) \times \prod_{k=i_a+1}^{\mathfrak{m}+a-1} \frac{\mathfrak{m} - k + (a+1)}{n-k+1} \times \frac{1}{n - (\mathfrak{m}+a) + 1} \tag{6.3}$$

If the last malicious is at position $\mathfrak{m} + a$, it means that all $\mathfrak{m} - 1$ first malicious have an index strictly lower than $\mathfrak{m} - a$ as well as a strategic participants.

The last term "$1/(n - (\mathfrak{m} + a) + 1)$" represents that the last (ultimate) malicious participant is placed at position $\mathfrak{m} + a$. $\forall \lambda > 0$, i_λ and $i_{\lambda+1}$ corresponds to the positions of two consecutive strategic participants, *i.e.*, the participants with index i_λ and $i_{\lambda+1}$ are strategic ($\theta_{i_\lambda} = \theta_{i_{\lambda+1}} = \theta^s$) and all participants with index $k \in \{i_\lambda + 1, \ldots, i_{\lambda+1} - 1\}$ are malicious, $\theta_k = \theta^m$. The penultimate product corresponds to the sequence of malicious participants immediately followed by the last one (at position $\mathfrak{m} + a$). Note that if $i_a = \mathfrak{m} + a - 1$, then $\theta_{\mathfrak{m}+a-1} = \theta^m$, the penultimate product collapses to 1.

The formula represents the sum over all configurations where the malicious participant with the highest index is placed at position $\mathfrak{m} + a$.

Using Equation (6.3), we compute the probabilities needed, and then α and β from Equation (6.2). Note that the conditions of Proposition 6.5 imply that $\mathfrak{m} < n/2$.

Table summarising the value of α and β for different values of $n, \mathfrak{m}$ and ν can be found in Appendix A.2. It has to be noted that values of α and β have been computed only at round 1, giving us not a tight bound for κ, but only a lower bound.

For example, in a committee of 40 participants, if there are 10 malicious participants in the committee, for different values of ν, we have:

- If $\nu = 11$, *i.e.*, the smallest possible value for ν, a lower bound for κ is $5.29 \times c_{\text{check}}$;
- If $\nu = 20 = n/2$, then a lower bound for κ is $77.74 \times c_{\text{check}} - 18.11 \times c_{\text{send}}$;
- If $\nu = 30$, *i.e.*, the highest possible value for ν, then a lower bound for κ is $9,614.24 \times c_{\text{check}} - 2,402.24 \times c_{\text{send}}$.

The minimal value of κ is increasing with the number of required messages needed for block's production. One can interpret that as follows: for a good equilibrium to exist, the more messages are required to consider a block as valid, the higher the cost of producing invalid blocks should be.

That can be explained since the more messages are required, the fewer participants supposed to check are pivotal, hence not checking validity may be an interesting deviation. To avoid that, the corresponding cost of invalid should be high to avoid such behaviours, *e.g.*, free riding. The same reasoning applies when the number of malicious participants is fixed and the size of the committee is increasing.

6.5 Conclusion

In this chapter, we introduced malicious participants. Malicious participants want to prevent consensus, and they are rational in the sense that they do take action maximising the chance of breaking the consensus. In particular, they prefer making an invalid block produced, or making no block produced. To study malicious behaviours in a system prone to malicious and strategic behaviours, we extend the model that allows studying strategic participants to malicious participants. We then study the behaviours of both malicious and strategic participants in committee-based blockchains when rewards are given only to participants that vote for the produced block. Strategic participants incur the costs of their actions, while malicious participants do not care about such costs. At any round, a participant in the committee is selected to be the proposer in a round-robin fashion. The proposer has the choice of proposing a valid or an invalid block, then upon receiving the proposal, each participant can check or not the validity of the proposal and then send a vote message for the block or not. If the number of votes for the proposal is higher than a given threshold, the block is considered produced, otherwise, the next round starts with the next proposer.

In these settings, focusing on one height at the time, and under the assumption of synchrony of communication between participants, we found multiple equilibria. We show that while there exists a good equilibrium where the consensus properties are always satisfied, there also exists an equilibrium where the Termination property is not satisfied or an equilibrium where the Validity property is not guaranteed.

CHAPTER 7

CONCLUSIONS

Contents

7.1 General Conclusions of the Thesis

Blockchain systems promise many advantages related to classical centralised systems. Many problems or rather questions are however raised by these systems. The scientific community is lately highly interested in the topic. The main questions we tackle in this thesis can be summarised in the following: "*Do blockchains even work? Are they usable in real-life settings?*".

In this thesis, we mainly focus on committee-based blockchains: blockchains in which committees are used to update the blockchains, and they rely on decades of research from the distributed systems community to ensure the absence of forks. The interest in these blockchains is still increasing.

To answer the question, "*Do blockchains even work?*", we first need to know what they are, and what they are trying to achieve. To do so, we first describe the abstraction that committee-based blockchains are solving, namely, the *Byzantine repeated consensus*, by extending the definition and incorporating Byzantine participants. Once the problem formally defined, we can prove whether committee-based blockchains are correct against the abstraction or repeated consensus. That is what we did, taking as use case the Tendermint blockchain. About Tendermint, our first contribution is the formalisation of its algorithms. Although already in use, before this thesis, Tendermint was not formalised nor proved. Our first work shows that the Tendermint consensus algorithm suffers from many bugs, and one of the bug was inherent to a bad design of the algorithm. Our work led to the correction and the proposal of a new algorithm by Tendermint, which solved the bugs and that we present in this manuscript (acknowledged in [39]). These contributions are detailed in Chapter 4.

The second question we tackle "*Are blockchains usable in real-life settings?*", do not have a simple response. In fact, giving a meaning to that question is the first step. Knowing that participants that produce blocks in the blockchain are rewarded, we first try to respond to the question by understanding if these blockchains are *fair*, *i.e.*, if participants are rewarded proportionally to their effort. If a participant produces less effort that another one, the latter should not be rewarded less than the former. The definition of fairness in blockchains such as Bitcoin cannot be applied to committee-based blockchain. In this thesis, we define the notion

of fairness for committee-based blocks and split the two key components, the *selection mechanism*, and the *reward mechanism*. Discussing the fairness in Chapter 4, we show that ensuring fairness, at least in the sense of reward mechanism, needs some (eventual) synchrony of the communication. We moreover show that the (eventual) synchrony of the communication is not a sufficient condition for fairness as we presented a mechanism that evolves in (eventual) synchronous communication, but which is not fair. A sort of summary of this study can be, fairness is hard to ensure. Furthermore, we can think of participants that want to gain the most reward possible, for as little effort as possible. That is the second meaning we may give to the question "*Are blockchains usable in real-life settings?*". If participants are rational, in the sense that they want to maximise their expected gain, is the system still correct? To respond to that latter question, we model the blockchain consensus algorithms inside committees as a game between the participants in committees. Our model combines approaches from distributed systems by considering malicious and obedient behaviours and approaches from economics by considering strategic participants. We show in Chapters 5 & 6 that although there exists at least an equilibrium satisfying all consensus properties, there exist many other equilibria in which either the Termination or the Validity properties cannot be guaranteed.

These works particularly show what remains to be done. Quoting Marie Skłodowska-Curie, "*One never notices what has been done; one can only see what remains to be done*".

7.2 Perspectives / Future Works

In this section, we present research questions opened up from the works presented in this manuscript.

The Problem Solved by Blockchains. In this thesis, we propose an abstraction for the problem solved by committee-based blockchains, it is interesting to know how other blockchains cope with such abstraction. Generalising blockchains needs to be done, and the problem solved by blockchains such as Bitcoin, or Ethereum needs to be defined. It has to be noted that works are being done on this line of research. Some effort to specify and formalise generic blockchain systems are being done [19, 22]. When committee-based blockchains need and should solve consensus to guarantee the absence of forks, blockchains like Bitcoin do not have that requirement. The problem solved may probably need to take into account probability and exponential convergence.

Correctness of Blockchain Protocols. Once the problem solved by blockchains (specifically forkable blockchains) is clearly and formally defined, one can think of their correctness against such definition.

More generally, managing the full expressiveness of distributed systems is still an ongoing work and formal techniques have been used to study some general problems (*e.g.*, [50, 66]). [9] is a step in that direction, proving a simplified variant of Algorand using formal tools. Hopefully, the advances in formal tools, coupled with interests from distributed systems researchers will allow to easily and quickly formally verify and prove complex distributed system applications such as blockchains, as claimed in [152]. Formally verified algorithms will be free of bugs. However, formally verifying already implemented codes is even harder. Usually, the key components are re-encoded, and the proof is done on the encoding. One can note that between the (proved) encoding and the code really implemented, there can be slight differences, due to the programmer or the programming environment. An even better idea will be to generate

correct-by-construction algorithms ready to be executed. The path for that is quite exciting, and efforts are already being done [10].

Complexity of Consensus Algorithms. In this manuscript, we gave a small glance at the complexity of consensus algorithms (Table 3.1), and some improvements that have been done lately, thanks to the interest in the scalability of blockchain systems. In synchronised setting, it was proved [63] that the lower bound on bit complexity for Byzantine consensus is $\Omega(n^2)$. There exist trade-offs between round complexity and bit complexity, and there may be trade-offs between the many concepts of complexity. It can be interesting to found the theoretical lower bound of rounds needed to achieve consensus using the linear view change paradigm of Tendermint and HotStuff, without using additional rounds nor threshold signatures as done in HotStuff. Such additional mechanisms yield to an increase of other complexity concepts such as the bit complexity.

Fairness. An important next step in the sense of fairness will be to integrate the variation of the merit of participants during the execution of the blockchain. Merit parameters varying over time should be considered. In fact, our proposal copes with systems where merits do not vary, or where the merit is fixed only after an infinite time. This is a big limitation of the definition for real-life applications where infinity is more of a mathematical concept. When considering that blockchains may evolve for only a finite time, or that participants take part in the system for a finite time, it is more interesting and realistic to extend the fairness definition to sufficiently long period of time, and not only over an infinite duration. Research from *returns to scale* [84] in economics may help. Basically, our notion of fairness implies constant returns to scale. However, in real life, when a participant invests more, it should gain more than another one investing less. There are needs to investigate such concepts and study it against decentralisation specifically. The collaboration between computer science and economics must be greater, more developed, and strengthen.

The selection of committee members is also a hard question. To guarantee equal chances or even equity among the different participants, the selection mechanism is really important. It will be interesting to propose and analyse different mechanisms and show what can be done, and what cannot, in a deterministic manner, in a probabilistic one, *etc.*

Censorship is another big question that needs more research. How to make sure that the transactions of all users are eventually in the blockchains. This can be viewed as a fairness question, but from the point of view of the users, the transaction issuers, and not the block creators. All participants, and specifically block creators do not necessarily have the exact same view and order of the transactions issued, the problem of ensuring user fairness is not trivial. It may even be impossible in a general setting. Proving if such property can be guaranteed or not is an interesting research topic; and in case of impossibility, finding the minimal conditions needed to guarantee fairness of users is a challenging question.

Rational Behaviours. Our work raises an immediate research question that needs to be answered. As we have seen, using different reward schemes lead to different equilibria. In committee-based blockchains specifically, and in presence of malicious participants, it is interesting to study the different equilibria when all participants are rewarded when a block is produced, and not only the participants that voted. That mechanism is interesting since, with no malicious, we show that there is no equilibrium violating the Validity property of the consensus.

Another direct question that may follow from our work considering rational behaviour is

the impact of the time assumptions. The time assumption guaranteeing agreement, it may be interesting to consider analyses with softer assumptions on communication delay between participants. Such analyses may highlight whether the Agreement property can always be guaranteed, which we think not.

More puzzling questions are whether it is possible to find blockchain systems where there is unicity of equilibrium, or rather if there are mechanisms that can prevent the existence of bad equilibria.

More generally, a complex yet interesting path is to model blockchain systems with fewer assumptions, meaning model real-word consensus protocols. These analyses are much more complex to handle and quite delicate, but they will probably have huge impacts on the field, and on the understanding of the systems. It is also important to analyse the whole blockchain system and not just one committee at the time. If the selection mechanism is known, participants can try to strategically be part of committees. There are many interesting paths around this topic, and all of them need to be investigated.

Another quite exciting question is how our results and theoretical analysis compare to real behaviours of participants. To do such comparison, behavioural sciences need to be integrated, which again will request more collaboration among different research fields.

Solution Concept. In this manuscript, when dealing with rational participants, we use the notion of pure Nash equilibrium, and when we consider dynamic Bayesian games, we consider refinement of Nash equilibrium. It can be interesting to analyse our system with other solution concepts. We can foresee many paths of research. We mention some of them.

First, by considering mixed strategies, *i.e.*, each participant instead of choosing deterministically one strategy to play for the game, chooses a distribution of the different strategies. The different equilibria that may arise can teach many lessons. Note that this is different from the concept of approximate equilibria.

Another path can be to consider cooperation between strategic participants. In this manuscript, we consider that only malicious do cooperate. Allowing cooperation may help participants to achieve higher gains; that needs investigation.

Other Interesting Problems. In this manuscript, we do not really discuss some topics. The next problems discussed are still related to blockchains; they are interesting and need to be investigated as well. Importantly, most of the problems existed before blockchains, but the blockchain technology increases the research on the topics. We just give a brief description of some of them.

Communication, Dynamicity and Network. We assume throughout this thesis that the communication network is reliable, no loss of messages, and the existence of a reliable broadcast. We should point that such assumptions require a lot of communication and redundancy. Research is still being done about the requirement for this reliability (*e.g.*, [35]). Extensions of results presented in this manuscript can be to consider less demanding assumptions about the reliability of the underlying network and communication. It would be interesting to provide protocols that work with fewer assumptions. Such protocols will offer more resilience to real-life issues. We should note that recent works (*e.g.*, [23]) consider relaxed assumptions on the network, where messages can be lost, with the guarantee that messages sent infinitely often, are eventually delivered by obedient participants. However, the techniques used require a lot of messages to be exchanged. It will be interesting to analyse the

intrinsic impact of the network assumptions on the complexity of the agreement algorithm, as well as for the behaviour of rational participants in such systems.

In the same topic of the underlying network assumption, we recall that we consider a dynamic setting in this manuscript. However, we impose a quite strong definition of correctness (obedient) for the participants, which is: a participant is obedient if it is part of the system since the beginning, do not leave it at any point in time, and always follows the prescribed protocol. That definition can, for good reason, be seen as strong. In particular, for blockchains, participants can enter at any point in time, since the system is dynamic. We need such participants to eventually have the blockchain from its beginning as well. Mechanisms ensuring that need to be analysed to guarantee good information in the system.

Last but not the least; we would like to mention the following two topics. Notions of *smart contracts* (and hence scalability, cross chains interactions and interoperability, decentralised applications, formal language and verification, *etc.*) were not discussed in this manuscript, but we cannot point out enough how important and interesting these topics are, in particular since they define what can be done with blockchain systems. The same goes to *cryptography*, which is at the core of blockchain systems, without which there cannot be such systems. These are currently hot topics of research, and they do provide interesting and promising results for the usability, and the common good that blockchains are. For example, analyses done in this manuscript need to be extended with less constrains on the cryptographic hypotheses.

"*We can only see a short distance ahead, but we can see plenty there that needs to be done.*"
– Alan Turing

PUBLICATIONS

The works presented in this manuscript gave rise to multiple publications.

The contributions of Chapter 4 have been published in the proceedings of the following international conferences:

- Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. On Fairness in Committee-based Blockchains. In *2nd International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2020)*, October 26-27, 2020, Toulouse, France, [18].

- Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Dissecting Tendermint. In *Networked Systems - 7th International Conference (NETYS 2019)*, Marrakech, Morocco, June 19-21, 2019, [17].

- Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Correctness of Tendermint-Core Blockchains. In *22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, December 17-19, 2018, Hong Kong, China, 2018, [15].

The contributions of Chapter 5 have been published in the proceedings of following international conference:

- Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Rational Behaviors in Committee-Based Blockchains. In *24th International Conference on Principles of Distributed Systems (OPODIS 2020)*, December 14-16, 2020, Strasbourg, France and Online, [13].

Part of the contributions of Chapter 6 has been published in the proceedings of the following international conference:

- Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Rational vs Byzantine Players in Consensus-based Blockchains. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, Auckland, New Zealand, May 9–13, 2020, 2020, [14].

The contributions of Chapters 4 and 6 have also been presented in the proceedings of the following French national congresses, respectively:

- Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Blockchains basées sur du Consensus Répété. In *ALGOTEL 2019 – 21èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Saint Laurent de la Cabrerisse, France, June 2019, [16].

- Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Consensus en Présence de Participants Rationnels et Byzantins. In *ALGOTEL 2020 – 22èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Lyon, France, September 2020, [12], and got the *best student paper award*.

Outside the scope of this thesis, the author also published the following paper resulting from his internship.

- Yackolley Amoussou-Guenou, Souheib Baarir, Maria Potop-Butucaru, Nathalie Sznajder, Leo Tible, and Sébastien Tixeuil. On the encoding and solving partial information games. In *Networked Systems - 8th International Conference, NETYS 2020*, Marrakech, Morocco, June 03-05, 2020, [11].

BIBLIOGRAPHY

[1] Ittai Abraham, Lorenzo Alvisi, and Joseph Y. Halpern. Distributed computing meets game theory: combining insights from two fields. *SIGACT News*, 42(2):69–76, 2011. doi:10.1145/1998037.1998055. 38

[2] Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 53–62. ACM, 2006. doi:10.1145/1146381.1146393. 38

[3] Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. Distributed protocols for leader election: A game-theoretic perspective. *ACM Trans. Economics and Comput.*, 7(1):4:1–4:26, 2019. doi:10.1145/3303712. 38

[4] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. *CoRR*, abs/1612.02916, 2016. arXiv:1612.02916. 31, 32, 38, 77

[5] Yehuda Afek, Yehonatan Ginzberg, Shir Landau Feibish, and Moshe Sulamy. Distributed computing building blocks for rational agents. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 406–415. ACM, 2014. doi:10.1145/2611462.2611481. 38

[6] Marcos Kawazoe Aguilera. A pleasant stroll through the land of infinitely many creatures. *SIGACT News*, 35(2):36–59, 2004. doi:10.1145/992287.992298. 14

[7] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Michael Dahlin, Jean-Philippe Martin, and Carl Porth. BAR fault tolerance for cooperative services. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles, SOSP 2005, Brighton, UK, October 23-26, 2005*, pages 45–58. ACM, 2005. doi:10.1145/1095810.1095816. 21, 38

[8] Victor Allombert, Mathias Bourgoin, and Julien Tesson. Introduction to the tezos blockchain. In *17th International Conference on High Performance Computing & Simulation, HPCS 2019, Dublin, Ireland, July 15-19, 2019*, pages 1–10, 2019. doi:10.1109/HPCS48598.2019.9188227. 26, 30

[9] Musab A. Alturki, Jing Chen, Victor Luchangco, Brandon M. Moore, Karl Palmskog, Lucas Peña, and Grigore Rosu. Towards a verified model of the algorand consensus

protocol in coq. In *Formal Methods. FM 2019 International Workshops - Porto, Portugal, October 7-11, 2019, Revised Selected Papers, Part I*, pages 362–367, 2019. doi:10.1007/978-3-030-54994-7_27. 26, 116

[10] Rajeev Alur and Stavros Tripakis. Automatic synthesis of distributed protocols. *SIGACT News*, 48(1):55–90, 2017. doi:10.1145/3061640.3061652. 117

[11] Yackolley Amoussou-Guenou, Souheib Baarir, Maria Potop-Butucaru, Nathalie Sznajder, Leo Tible, and Sébastien Tixeuil. On the encoding and solving partial information games. In *Networked Systems - 8th International Conference, NETYS 2020, Marrakech, Morocco, June 03-05*, 2020. 122

[12] Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Consensus en Présence de Participants Rationnels et Byzantins. In *ALGOTEL 2020 – 22èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Lyon, France, September 2020. URL: https://hal.archives-ouvertes.fr/hal-02874641. v, 8, 122

[13] Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Rational Behaviors in Committee-Based Blockchains. In *24th International Conference on Principles of Distributed Systems, OPODIS 2020, December 14-16, Strasbourg, Online, France*, 2020. doi:10.4230/LIPIcs.OPODIS.2020.12. v, 7, 121

[14] Yackolley Amoussou-Guenou, Bruno Biais, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Rational vs byzantine players in consensus-based blockchains. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pages 43–51. International Foundation for Autonomous Agents and Multiagent Systems, 2020. URL: dl.acm.org/doi/10.5555/3398761.3398772. v, 8, 121

[15] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Correctness of Tendermint-Core Blockchains. In *22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 16:1–16:16, 2018. doi:10.4230/LIPIcs.OPODIS.2018.16. v, 7, 57, 82, 121, 139

[16] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Blockchains basées sur du Consensus Répété. In *ALGOTEL 2019 - 21èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Saint Laurent de la Cabrerisse, France, June 2019. URL: https://hal.archives-ouvertes.fr/hal-02119956. v, 7, 122

[17] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Dissecting tendermint. In *Networked Systems - 7th International Conference, NETYS 2019, Marrakech, Morocco, June 19-21, 2019, Revised Selected Papers*, volume 11704 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2019. doi:10.1007/978-3-030-31277-0_11. v, 7, 121

[18] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. On Fairness in Committee-based Blockchains. In *2nd International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2020), Toulouse, France, October 26-27*, 2020. v, 7, 121

[19] Emmanuelle Anceaume, Antonella Del Pozzo, Romaric Ludinard, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Blockchain abstract data type. In *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019*, pages 349–358. ACM, 2019. doi:10.1145/3323165.3323183. 116

[20] Emmanuelle Anceaume, Antoine Guellier, Romaric Ludinard, and Bruno Sericola. Sycomore: A permissionless distributed ledger that self-adapts to transactions demand. In *17th IEEE International Symposium on Network Computing and Applications, NCA 2018, Cambridge, MA, USA, November 1-3, 2018*, pages 1–8. IEEE, 2018. doi:10.1109/NCA.2018.8548053. 39

[21] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 30:1–30:15. ACM, 2018. doi:10.1145/3190508.3190538. 31

[22] Antonio Fernández Anta, Kishori M. Konwar, Chryssis Georgiou, and Nicolas C. Nicolaou. Formalizing and implementing distributed ledger objects. *SIGACT News*, 49(2):58–76, 2018. doi:10.1145/3232679.3232691. 116

[23] Lăcrămioara Aştefanoaei, Pierre Chambart, Antonella Del Pozzo, Thibault Rieutord, Edward Tate, Sara Tucci-Piergiovanni, and Eugen Zălinescu. Tenderbake - classical BFT style consensus for public blockchains. *CoRR*, abs/2001.11965, 2020. arXiv:2001.11965. 118

[24] Christian Badertscher, Juan A. Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. But why does it work? A rational protocol design treatment of bitcoin. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 34–65. Springer, 2018. doi:10.1007/978-3-319-78375-8_2. 37

[25] Roberto Baldoni, Marin Bertier, Michel Raynal, and Sara Tucci-Piergiovanni. Looking for a definition of dynamic distributed systems. In *Parallel Computing Technologies, 9th International Conference, PaCT 2007, Pereslavl-Zalessky, Russia, September 3-7, 2007, Proceedings*, volume 4671 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2007. doi:10.1007/978-3-540-73940-1_1. 13, 18

[26] Amotz Bar-Noy, Xiaotie Deng, Juan A. Garay, and Tiko Kameda. Optimal amortized distributed consensus. *Inf. Comput.*, 120(1):93–100, 1995. doi:10.1006/inco.1995.1101. 45

[27] Thomas Bayes. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London*, 53:370–418, 1763. doi:10.1098/rstl.1763.0053. 22

[28] Marianna Belotti, Nikola Bozic, Guy Pujolle, and Stefano Secci. A vademecum on blockchain technologies: When, which, and how. *IEEE Commun. Surv. Tutorials*, 21(4):3796–3838, 2019. doi:10.1109/COMST.2019.2928178. 39

[29] Marianna Belotti, Sofiane Kirati, and Stefano Secci. Bitcoin pool-hopping detection. In *4th IEEE International Forum on Research and Technology for Society and Industry, RTSI 2018, Palermo, Italy, September 10-13, 2018*, pages 1–6. IEEE, 2018. doi:10.1109/RTSI.2018.8548376. 37

[30] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 17-19, 1983*, pages 27–30. ACM, 1983. doi:10.1145/800221.806707. 32

[31] Bruno Biais, Christophe Bisière, Matthieu Bouvard, and Catherine Casamatta. The Blockchain Folk Theorem. *The Review of Financial Studies*, 32(5):1662–1715, 04 2019. doi:10.1093/rfs/hhy095. 4, 37, 77

[32] Binance. Binance chain, 2020. [Online; accessed 2020 July 10]. URL: https://docs.binance.org/guides/intro.html. 57

[33] David Blackwell. Discounted dynamic programming. *The Annals of Mathematical Statistics*, 36(1):226–235, 1965. doi:10.1214/aoms/1177700285. 102

[34] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 104–121. IEEE Computer Society, 2015. doi:10.1109/SP.2015.14. 39

[35] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast with honest dealer made practical. *J. Braz. Comput. Soc.*, 25(1):9:1–9:23, 2019. doi:10.1186/s13173-019-0090-x. 118

[36] Fatemeh Borran and André Schiper. A leader-free byzantine consensus algorithm. In *Distributed Computing and Networking, 11th International Conference, ICDCN 2010, Kolkata, India, January 3-6, 2010. Proceedings*, volume 5935 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2010. doi:10.1007/978-3-642-11322-2_11. 33

[37] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, 1987. doi:10.1016/0890-5401(87)90054-X. 17

[38] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, 1985. doi:10.1145/4221.214134. 16

[39] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus. Technical report, Tendermint, jul 2018. arXiv:1807.04938. 32, 34, 42, 57, 77, 115

[40] Ethan Buchman. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains. Master thesis, Guelph University, june 2016. URL: hdl.handle.net/10214/9769. 57

[41] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 524–541. Springer, 2001. doi:10.1007/3-540-44647-8_31. 31, 43

[42] Christian Cachin and Marko Vukolic. Blockchain consensus protocols in the wild (keynote talk). In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 1:1–1:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.1. 57

[43] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multiparty computation. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 639–648. ACM, 1996. doi:10.1145/237814.238015. 37

[44] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distributed Syst.*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546. 13

[45] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, pages 173–186. USENIX Association, 1999. URL: https://dl.acm.org/citation.cfm?id=296824. 29, 31, 32, 34, 42, 57, 69

[46] Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. *IACR Cryptol. ePrint Arch.*, 2020:88, 2020. URL: https://eprint.iacr.org/2020/088. 79

[47] Tushar Deepak Chandra, Vassos Hadzilacos, Sam Toueg, and Bernadette Charron-Bost. On the impossibility of group membership. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, Philadelphia, Pennsylvania, USA, May 23-26, 1996*, pages 322–330. ACM, 1996. doi:10.1145/248052.248120. 47

[48] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996. doi:10.1145/226643.226647. 31, 49

[49] Sylvain Charlebois. How blockchain technology could transform the food industry. *The conversation*, 20, 2017. URL: https://theconversation.com/how-blockchain-technology-could-transform-the-food-industry-89348. 4

[50] Bernadette Charron-Bost, Henri Debrat, and Stephan Merz. Formal verification of consensus algorithms tolerating malicious faults. In *Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings*, volume 6976 of *Lecture Notes in Computer Science*, pages 120–134. Springer, 2011. doi:10.1007/978-3-642-24550-3_11. 116

[51] Kaylash Chaudhary, Ansgar Fehnker, Jaco van de Pol, and Mariëlle Stoelinga. Modeling and verification of the bitcoin protocol. In *Proceedings Workshop on Models for Formal Analysis of Real Systems, MARS 2015, Suva, Fiji, November 23, 2015*, volume 196 of *EPTCS*, pages 46–60, 2015. doi:10.4204/EPTCS.196.5. 26

[52] Allen Clement, Edmund L. Wong, Lorenzo Alvisi, Michael Dahlin, and Mirco Marchetti. Making byzantine fault tolerant systems tolerate byzantine faults. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2009, April 22-24, 2009, Boston, MA, USA*, pages 153–168. USENIX Association, 2009. URL: http://www.usenix.org/events/nsdi09/tech/full_papers/clement/clement.pdf. 32

[53] Cosmos. List of project using tendermint and the cosmos ecosystem. https://cosmonauts.world, 2020. [Online; accessed 2020 July 10]. 57

[54] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. (leader/randomization/signature)-free byzantine consensus for consortium blockchains. *CoRR*, abs/1702.03068v2, 2017. arXiv:1702.03068v2. 32, 33, 42, 43

[55] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. DBFT: efficient leaderless byzantine consensus and its application to blockchains. In *17th IEEE International Symposium on Network Computing and Applications, NCA 2018, Cambridge, MA, USA, November 1-3, 2018*, pages 1–8. IEEE, 2018. doi:10.1109/NCA.2018.8548057. 33, 34

[56] Flaviu Cristian. Reaching agreement on processor-group membrship in synchronous distributed systems. *Distributed Computing*, 4(4):175–187, 1991. doi:10.1007/BF01784719. 32, 47

[57] Fiery Cushman, Anna Dreber, Ying Wang, and Jay Costa. Accidental Outcomes Guide Punishment in a "Trembling Hand" Game. *PloS one*, 4(8), 2009. doi:10.1371/journal.pone.0006699. 87

[58] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11598 of *Lecture Notes in Computer Science*, pages 23–41. Springer, 2019. doi:10.1007/978-3-030-32101-7_2. 30

[59] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore, January 4-7, 2016*, pages 13:1–13:10. ACM, 2016. doi:10.1145/2833312.2833321. 29

[60] Carole Delporte-Gallet, Stéphane Devismes, Hugues Fauconnier, Franck Petit, and Sam Toueg. With finite memory consensus is easier than reliable broadcast. In *Principles of Distributed Systems, 12th International Conference, OPODIS 2008, Luxor, Egypt, December 15-18, 2008. Proceedings*, volume 5401 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2008. doi:10.1007/978-3-540-92221-6_5. 46

[61] Danny Dolev. Unanimity in an unknown and unreliable environment. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 159–168. IEEE Computer Society, 1981. doi:10.1109/SFCS.1981.53. 17

[62] Danny Dolev, Cynthia Dwork, and Larry J. Stockmeyer. On the minimal synchronism needed for distributed consensus. *J. ACM*, 34(1):77–97, 1987. doi:10.1145/7531.7533. 12

[63] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985. doi:10.1145/2455.214112. 117

[64] Shlomi Dolev and Sergio Rajsbaum. Stability of long-lived consensus. *J. Comput. Syst. Sci.*, 67(1):26–45, 2003. doi:10.1016/S0022-0000(03)00063-1. 46

[65] Assia Doudou and André Schiper. Muteness detectors for consensus with byzantine processes. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998*, page 315. ACM, 1998. doi:10.1145/277697.277772. 31

[66] Cezara Dragoi, Thomas A. Henzinger, and Damien Zufferey. Psync: a partially synchronous language for fault-tolerant distributed algorithms. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 400–415. ACM, 2016. doi:10.1145/2837614.2837650. 116

[67] Ashutosh Dhar Dwivedi, Gautam Srivastava, Shalini Dhar, and Rajani Singh. A decentralized privacy-preserving healthcare blockchain for iot. *Sensors*, 19(2):326, 2019. doi:10.3390/s19020326. 3

[68] Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988. doi:10.1145/42282.42283. 18, 32, 42, 57

[69] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992. doi:10.1007/3-540-48071-4_10. 26, 41

[70] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 585–605. Springer, 2015. doi:10.1007/978-3-662-48000-7_29. 31

[71] European Union. General data protection regulation – gdpr. https://gdpr-info.eu/, 2018. [Online; accessed 2020 July 10]. 4

[72] Ittay Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 89–103. IEEE Computer Society, 2015. doi:10.1109/SP.2015.13. 37

[73] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, pages 45–59. USENIX Association, 2016. URL: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal. 29, 30

[74] Ittay Eyal and Emin Gün Sirer. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, 2018. doi:10.1145/3212998. 35, 36, 37, 49, 77

[75] Giulia C. Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, and Gerui Wang. Compounding of wealth in proof-of-stake cryptocurrencies. In *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11598 of *Lecture Notes in Computer Science*, pages 42–61. Springer, 2019. doi:10.1007/978-3-030-32101-7_3. 35

[76] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982. doi:10.1016/0020-0190(82)90033-3. 32, 69

[77] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985. doi:10.1145/3149.214121. 28, 32, 43, 55

[78] Mehdi Fooladgar, Mohammad Hossein Manshaei, Murtuza Jadliwala, and Mohammad Ashiqur Rahman. On incentive compatible role-based reward distribution in algorand. In *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020*, pages 452–463. IEEE, 2020. doi:10.1109/DSN48063.2020.00059. 38, 77

[79] Drew Fudenberg and Jean Tirole. Perfect bayesian equilibrium and sequential equilibrium. *Journal of Economic Theory*, 53(2):236 – 260, 1991. doi:10.1016/0022-0531(91)90155-W. 24

[80] Fangyu Gai, Baosheng Wang, Wenping Deng, and Wei Peng. Proof of reputation: A reputation-based consensus protocol for peer-to-peer network. In *Database Systems for Advanced Applications - 23rd International Conference, DASFAA 2018, Gold Coast, QLD, Australia, May 21-24, 2018, Proceedings, Part II*, volume 10828 of *Lecture Notes in Computer Science*, pages 666–681. Springer, 2018. doi:10.1007/978-3-319-91458-9_41. 31

[81] Juan A. Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 648–657. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.75. 37, 38

[82] Juan A. Garay and Aggelos Kiayias. Sok: A consensus taxonomy in the blockchain era. In *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, volume 12006 of *Lecture Notes in Computer Science*, pages 284–318. Springer, 2020. doi:10.1007/978-3-030-40186-3_13. 39

[83] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015. doi:10.1007/978-3-662-46803-6_10. 29, 35, 47, 49

[84] Gregory M. Gelles and Douglas W. Mitchell. Returns to scale and economies of scale: Further observations. *The Journal of Economic Education*, 27(3):259–261, 1996. doi:10.1080/00220485.1996.10844915. 117

[85] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017. doi:10.1145/3132747.3132757. 30, 32, 33, 38, 42, 77

[86] Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. SBFT: A scalable and decentralized trust infrastructure. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*, pages 568–580, 2019. doi:10.1109/DSN.2019.00063. 33

[87] Fabíola Greve, Murilo Santos de Lima, Luciana Arantes, and Pierre Sens. A time-free byzantine failure detector for dynamic networks. In *2012 Ninth European Dependable Computing Conference, Sibiu, Romania, May 8-11, 2012*, pages 191–202. IEEE Computer Society, 2012. doi:10.1109/EDCC.2012.28. 53

[88] Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. Byzantine agreement with a rational adversary. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 561–572. Springer, 2012. doi:10.1007/978-3-642-31585-5_50. 38

[89] Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. The consensus number of a cryptocurrency. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 307–316. ACM, 2019. doi:10.1145/3293611.3331589. 33

[90] Rachid Guerraoui and Jingjing Wang. On the unfairness of blockchain. In *Networked Systems - 6th International Conference, NETYS 2018, Essaouira, Morocco, May 9-11, 2018, Revised Selected Papers*, volume 11028 of *Lecture Notes in Computer Science*, pages 36–50. Springer, 2018. doi:10.1007/978-3-030-05529-5_3. 35

[91] Saurabh Gupta. A non-consensus based decentralized financial transaction processing model with support for efficient auditing. Master thesis, Arizona State University, june 2016. URL: http://hdl.handle.net/2286/R.I.39437. 33

[92] Önder Gürcan, Antonella Del Pozzo, and Sara Tucci-Piergiovanni. On the bitcoin limitations to deliver fairness to users. In *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I*, volume 10573 of *Lecture Notes in Computer Science*, pages 589–606. Springer, 2017. doi:10.1007/978-3-319-69462-7_37. 35

[93] Önder Gürcan, Alejandro Ranchal Pedrosa, and Sara Tucci-Piergiovanni. On cancellation of transactions in bitcoin-like blockchains. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part*

I, volume 11229 of *Lecture Notes in Computer Science*, pages 516–533. Springer, 2018. doi:10.1007/978-3-030-02610-3_29. 35

[94] Andreas Haeberlen and Petr Kuznetsov. The fault detection problem. In *Principles of Distributed Systems, 13th International Conference, OPODIS 2009, Nîmes, France, December 15-18, 2009*, pages 99–114. Springer, 2009. doi:10.1007/978-3-642-10877-8_10. 53

[95] Joseph Y. Halpern and Xavier Vilaça. Rational consensus: Extended abstract. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 137–146. ACM, 2016. doi:10.1145/2933057.2933088. 38, 39

[96] Maurice Herlihy and Mark Moir. Enhancing accountability and trust in distributed ledgers. *CoRR*, abs/1606.07490, 2016. arXiv:1606.07490. 35, 57

[97] S. M. Riazul Islam, Daehan Kwak, Humaun Kabir, Md. Mahmud Hossain, and Kyung Sup Kwak. The internet of things for health care: A comprehensive survey. *IEEE Access*, 3:678–708, 2015. doi:10.1109/ACCESS.2015.2437951. 3

[98] Andreas Kamilaris, Agusti Fonts, and Francesc X. Prenafeta-Boldύ. The rise of blockchain technology in agriculture and food supply chains. *Trends in Food Science & Technology*, 91:640 – 652, 2019. doi:10.1016/j.tifs.2019.07.034. 4

[99] Dimitris Karakostas, Aggelos Kiayias, Christos Nasikas, and Dionysis Zindros. Cryptocurrency Egalitarianism: A Quantitative Approach. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2020), pages = 7:1–7:21, series = Open Access Series in Informatics (OASIcs), ISBN = 978-3-95977-108-5, ISSN = 2190-6807, publisher = Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, address = Dagstuhl, Germany, volume = 71, Paris, France, May 06-07*, 2019. doi:10.4230/OASIcs.Tokenomics.2019.7. 35

[100] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2nd edition, 2014. URL: https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269. 4, 14

[101] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 365–382. ACM, 2016. doi:10.1145/2940716.2940773. 36, 37

[102] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017. doi:10.1007/978-3-319-63688-7_12. 30, 35

[103] Aggelos Kiayias and Aikaterini-Panagiota Stouka. Coalition-safe equilibria with virtual payoffs. *CoRR*, abs/2001.00047, 2020. arXiv:2001.00047. 37

[104] Kim Potter Kihlstrom, Louise E. Moser, and P. M. Melliar-Smith. Byzantine fault detectors for solving consensus. *Comput. J.*, 46(1):16–35, 2003. doi:10.1093/comjnl/46.1.16. 53

[105] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. https://www.peercoin.net/whitepapers/peercoin-paper.pdf, 2012. [Online; accessed 2020 July 10]. 30

[106] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 279–296. USENIX Association, 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kogias. 29, 32

[107] Jovan Komatovic, Rachid Guerraoui, Petr Kuznetsov, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, and Andrei Tonkikh. Dynamic Byzantine Reliable Broadcast. In *24th International Conference on Principles of Distributed Systems, OPODIS 2020, December 14-16, Strasbourg, Online, France*, 2020. 17

[108] Dariusz R. Kowalski and Achour Mostéfaoui. Synchronous byzantine agreement with nearly a cubic number of communication bits. In *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 84–91. ACM, 2013. doi:10.1145/2484239.2484271. 34

[109] Raymond Shih Ray Ku. The creative destruction of copyright: Napster and the new economics of digital technology. *The University of Chicago Law Review*, 69(1):263–324, 2002. doi:10.2307/1600355. 1

[110] Jae Kwon. Tendermint: Consensus without mining. https://tendermint.com/static/docs/tendermint.pdf, 2014. [Online; accessed 2020 July 10]. 56

[111] Jae Kwon and Ethan Buchman. Tendermint. https://tendermint.readthedocs.io/en/master/specification.html (visited on 2018-05-22). 58

[112] Nicolas Lagaillardie, Mohamed Aimen Djari, and Önder Gürcan. A computational study on fairness of the tendermint blockchain protocol. *Information*, 10(12):378, 2019. doi:10.3390/info10120378. 35

[113] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi:10.1145/357172.357176. 32, 42, 43

[114] Lawrence Lessig. Code is law. *Harvard magazine*, 1:2000, 2000. URL: https://harvardmagazine.com/2000/01/code-is-law-html. 4

[115] Kfir Lev-Ari, Alexander Spiegelman, Idit Keidar, and Dahlia Malkhi. Fairledger: A fair blockchain protocol for financial institutions. In *23rd International Conference on Principles of Distributed Systems, OPODIS 2019, December 17-19, 2019, Neuchâtel, Switzerland*, volume 153 of *LIPIcs*, pages 4:1–4:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.OPODIS.2019.4. 35, 53, 54

[116] Ziyao Liu, Nguyen Cong Luong, Wenbo Wang, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. A survey on blockchain: A game theoretical perspective. *IEEE Access*, 7:47615–47643, 2019. doi:10.1109/ACCESS.2019.2909924. 39

[117] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996. URL: https://dl.acm.org/doi/book/10.5555/2821576. 4, 5, 11

[118] Anna Lysyanskaya and Nikos Triandopoulos. Rationality and adversarial behavior in multi-party computation. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 180–197. Springer, 2006. doi:10.1007/11818175_11. 38

[119] Katya Malinova and Andreas Park. Tokenomics: When Tokens Beat Equity. SSRN Scholarly Paper ID 3286825, Social Science Research Network, Rochester, NY, December 2018. doi:10.2139/ssrn.3286825. 4

[120] Dahlia Malkhi. The BFT lens: Tendermint, 2018. [Online; accessed 2020 July 10]. URL: https://dahliamalkhi.wordpress.com/2018/04/03/tendermint-in-the-lens-of-bft. 57

[121] Mohammad Hossein Manshaei, Murtuza Jadliwala, Anindya Maiti, and Mahdi Fooladgar. A game-theoretic analysis of shard-based permissionless blockchains. *IEEE Access*, 6:78100–78112, 2018. doi:10.1109/ACCESS.2018.2884764. 38

[122] Jean-Philippe Martin and Lorenzo Alvisi. Fast byzantine consensus. *IEEE Trans. Dependable Secur. Comput.*, 3(3):202–215, 2006. doi:10.1109/TDSC.2006.35. 42

[123] MATLAB. *version 9.6 (R2019a)*. The MathWorks Inc., Natick, Massachusetts, 2019. URL: https://www.mathworks.com. 55

[124] Alexandre Maurer, Sébastien Tixeuil, and Xavier Défago. Communicating reliably in multihop dynamic networks despite byzantine failures. In *34th IEEE Symposium on Reliable Distributed Systems, SRDS 2015, Montreal, QC, Canada, September 28 - October 1, 2015*, pages 238–245. IEEE Computer Society, 2015. doi:10.1109/SRDS.2015.10. 17

[125] Tian Min, Hanyi Wang, Yaoze Guo, and Wei Cai. Blockchain games: A survey. In *IEEE Conference on Games, CoG 2019, London, United Kingdom, August 20-23, 2019*, pages 1–8. IEEE, 2019. doi:10.1109/CIG.2019.8848111. 4

[126] Ahmed Afif Monrat, Olov Schelén, and Karl Andersson. A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7:117134–117151, 2019. doi:10.1109/ACCESS.2019.2936094. 39

[127] Francesc D. Muñoz-Escoí and Rubén de Juan-Marín. On synchrony in dynamic distributed systems. *Open Comput. Sci.*, 8(1):154–164, 2018. doi:10.1515/comp-2018-0014. 18

[128] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf, 2008. [Online; accessed 2020 July 10]. 1, 26, 28, 43

[129] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. URL: http://www.jstor.org/stable/1969529. 23

[130] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007. doi:10.1017/CBO9780511800481. 6

[131] Dev Ojha and Christopher Goes. F1 Fee Distribution. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*, volume 71 of *OpenAccess Series in Informatics (OASIcs)*, pages 10:1–10:6, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.Tokenomics.2019.10. 49

[132] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, and Cian Montgomery. Sawtooth: An introduction. https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_Sawtooth_WhitePaper.pdf, 2018. [Online; accessed 2020 July 10]. 30

[133] Martin J Osborne. *An Introduction to Game Theory*, volume 3. Oxford university press New York, 2004. doi:10.1145/1095810.1095816. 24

[134] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673, 2017. doi:10.1007/978-3-319-56614-6_22. 29

[135] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 315–324. ACM, 2017. doi:10.1145/3087801.3087809. 27, 35, 36

[136] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. doi:10.1145/322186.322188. 14

[137] Serguei Popov. The tangle. https://www.iota.org/foundation/research-papers, 2016. [Online; accessed 2020 July 10]. 3, 39

[138] Julien Prat, Vincent Danos, and Stefania Marcassa. Fundamental pricing of utility tokens. Technical report, THEMA (THéorie Economique, Modélisation et Applications), Université de Cergy-Pontoise, France, 2019. URL: https://thema.u-cergy.fr/IMG/pdf/2019-11.pdf. 4

[139] Julien Prat and Benjamin Walter. An Equilibrium Model of the Market for Bitcoin Mining. SSRN Scholarly Paper ID 3143410, Social Science Research Network, Rochester, NY, March 2018. URL: https://ssrn.com/abstract=3143410. 4

[140] Michel Raynal. *Fault-Tolerant Message-Passing Distributed Systems - An Algorithmic Approach*. Springer, 2018. doi:10.1007/978-3-319-94141-7. 14, 17

[141] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004. doi:10.1007/978-3-540-25937-4_24. 16, 26, 41, 44

[142] Juan M. Roman-Belmonte, Hortensia De la Corte-Rodriguez, and E. Carlos Rodriguez-Merchan. How blockchain technology can change medicine. *Postgraduate Medicine*, 130(4):420–427, 2018. PMID: 29727247. doi:10.1080/00325481.2018.1472996. 3

[143] Fahad Saleh. Blockchain Without Waste: Proof-of-Stake. SSRN Scholarly Paper ID 3183935, Social Science Research Network, Rochester, NY, January 2019. doi:10.2139/ssrn.3183935. 37

[144] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, volume 9603 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2016. doi:10.1007/978-3-662-54970-4_30. 36, 37

[145] Nicolas Six, Nicolas Herbaut, and Camille Salinesi. Which blockchain to choose? A decision support tool to guide the choice of a blockchain technology. *CoRR*, abs/2004.06080, 2020. arXiv:2004.06080. 39

[146] Smith + Crown Intelligence (SCI). Verifier's Dilemma. https://sci.smithandcrown.com/glossary/verifiers-dilemma. [Online; accessed 2020 July 10]. 95

[147] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), Sep. 1997. doi:10.5210/fm.v2i9.548. 2, 4

[148] Michael Bedford Taylor. The evolution of bitcoin hardware. *IEEE Computer*, 50(9):58–66, 2017. doi:10.1109/MC.2017.3571056. 27

[149] The LibraBFT Team, Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. State machine replication in the libra blockchain. Technical report, The Libra Assn., 2019. version 2020-05-26, [Online; accessed 2020 July 10]. URL: https://developers.libra.org/docs/state-machine-replication-paper. 57

[150] Tendermint. Tendermint: Tendermint Core (BFT Consensus) in Go. https://github.com/tendermint/tendermint. [Online; accessed 2020 July 10]. 57

[151] Tendermint Inc. Tendermint core & sdk, 2020. [Online; accessed 2020 July 10]. URL: https://tendermint.com/about/. 57

[152] Pierre Tholoniat and Vincent Gramoli. Formal verification of blockchain byzantine fault tolerance. *CoRR*, abs/1909.07453, 2019. arXiv:1909.07453. 116

[153] Sam Toueg. Randomized byzantine agreements. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing, Vancouver, B. C., Canada, August 27-29, 1984*, pages 163–178. ACM, 1984. doi:10.1145/800222.806744. 42

[154] Takashi Ui. Bayesian nash equilibrium and variational inequalities. *Journal of Mathematical Economics*, 63:139 – 146, 2016. doi:10.1016/j.jmateco.2016.02.004. 24

[155] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. Spin one's wheels? byzantine fault tolerance with a spinning primary. In *28th IEEE Symposium on Reliable Distributed Systems (SRDS 2009), Niagara Falls, New York, USA,*

September 27-30, 2009, pages 135–144. IEEE Computer Society, 2009. doi:10.1109/SRDS.2009.36. 33

[156] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior (60th-Anniversary Edition)*. Princeton University Press, 2007. URL: http://press.princeton.edu/titles/7802.html. 5

[157] Wikipedia contributors. Rational choice theory — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Rational_choice_theory&oldid=962917833, 2020. [Online; accessed 2020 July 10]. 21

[158] Jeannette M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8–24, 1990. doi:10.1109/2.58215. 4

[159] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. https://ethereum.github.io/yellowpaper/paper.pdf, 2014. [Online; accessed 2020 July 10]. 2, 30

[160] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 347–356. ACM, 2019. doi:10.1145/3293611.3331591. 32, 33, 34, 42, 57, 77

[161] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: a survey. *IJWGS*, 14(4):352–375, 2018. doi:10.1504/IJWGS.2018.10016848. 39

APPENDIX A

APPENDIX

A.1 Some TendermintBFT Earlier Bugs

The bugs reported here were present in the earlier version of TendermintBFT formalised and analysed in [15].

Timeout Management. In an earlier version of TendermintBFT, the timeout management was sometimes sloppy. In more details, at certain points, the timeouts were not increasing to allow participants to collect sufficiently many messages, even when the synchronous period is reached, and therefore violating the Termination property. This bug seemed to be an implementation omission and was easily fixed.

Unlock Management. In an earlier version of TendermintBFT, a participant could unlock when it receives enough proof messages from a more recent epoch than the one it is locked on. Intuitively, when a participant locked on an epoch e, for a block B, it means that it had delivered proposals for B from at least $2f + 1$ participants during epoch e. When a new block is proposed for a new epoch, with $2f + 1$ proposals from an epoch $e' > e$, the participant should unlock, since there is a much recent lock value. However, the algorithm allowed to unlock even on the same block the participant is locked on. The problem is that the participants may lock and vote for B at epoch e such that B got $2f + 1$ proposals/votes, and some participant decides. At epoch e' the block B is again proposed in addition to the $2f + 1$ proposals from epoch e. The participants locked on epoch e can unlock, but because of asynchronous communication, they may not lock again, nor decide. When a new epoch starts, assume in the synchronous period, say e'', a completely different value B' can be pre-proposed, such that all remaining participants will propose, eventually vote and decide on B'. Therefore, some obedient participants decide on B, and some others decide on B', the Agreement property could be violated. For that bug, we proposed a simple fix as well; the problem as before seems to be an omission.

Live-lock Issue. Another bug was present in the earlier version of TendermintBFT preventing the Termination property. This bug lied in the design of the locking mechanism. Basically, whenever a new value is proposed, the participants increase the value of the epoch they locked on. Even during the synchronous periods, Byzantine participants may collude to split the obedient participants into two groups locked on two different values, and force

them to alternatively increase the value of their locked epoch. When it is the time for one group to propose a new block, the second group rejects it since they have higher lock value; and the same for the second group, because Byzantine make each group increase their locked value at the end of a round without impacting the messages sent during the corresponding phases. Therefore, each group refuse the proposal of the other, and no obedient participant ever decides, which violates the Termination property.

Exploiting this bug requires a strong collaboration between Byzantine participants, and needs Byzantine participants to send messages at the right time such that it does not diffuse too fast to other participants. The problem was that obedient participants are not aware even during synchronous rounds when other obedient participants lock. That was solved in the most recent version of TendermintBFT (presented in Chapter 4), where two new variables are used to keep track of the latest locked value and epoch a participant heard of from other obedient participants.

A.2 Table Containing Values of α and β from Proposition 6.5

f	ν	$\alpha(1)$	$\beta(1)$
1	2	3	0
$n = 3$ where $\nu = 2$			

f	ν	$\alpha(1)$	$\beta(1)$
1	2	4	0
$n = 4$			

f	ν	$\alpha(1)$	$\beta(1)$
1	3	5.33	0.33
$n = 4$			

f	ν	$\alpha(1)$	$\beta(1)$
1	2	5	0
2	3	3.5	0
$n = 5$ and ν is minimal			

f	ν	$\alpha(1)$	$\beta(1)$
1	3	6.25	0.25
2	3	3.5	0
$n = 5$ and $\nu = 3$			

f	ν	$\alpha(1)$	$\beta(1)$
1	4	8.33	0.67
2	3	3.5	0
$n = 5$ and ν is maximal			

f	ν	$\alpha(1)$	$\beta(1)$
1	2	10	0
2	3	6	0
3	4	4.56	0
4	5	3.92	0
$n = 10$ where ν is minimal			

f	ν	$\alpha(1)$	$\beta(1)$
1	5	14.29	0.43
2	5	9.04	0.61
3	5	5.98	0.43
4	5	3.92	0
$n = 10$ where $\nu = 5$			

f	ν	$\alpha(1)$	$\beta(1)$
1	9	33.33	2.33
2	8	23.5	3.5
3	7	12.65	2.43
4	6	5.83	0.67
$n = 10$ where ν is maximal			

f	ν	$\alpha(1)$	$\beta(1)$
1	2	20	0
2	3	11	0
3	4	7.77	0
4	5	6.18	0
5	6	5.25	0
6	7	4.67	0
7	8	4.29	0
8	9	4.04	0
9	10	3.89	0
$n = 20$ where ν is minimal			

f	ν	$\alpha(1)$	$\beta(1)$
1	10	33.33	0.67
2	10	25.36	1.44
3	10	21.98	2.132
4	10	18.92	2.55
5	10	15.45	2.55
6	10	11.77	2.13
7	10	8.39	1.44
8	10	5.71	0.67
9	10	3.89	0
$n = 20$ where $\nu = 10$			

f	ν	$\alpha(1)$	$\beta(1)$
1	19	133.3	5.67
2	18	191	18
3	17	218.2	31.57
4	16	194.4	37.45
5	15	135.5	32.56
6	14	76.62	21.59
7	13	35.85	11.05
8	12	14.49	4.18
9	11	5.71	0.82
$n = 20$ where ν is maximal			

		ν	$\alpha(1)$	$\beta(1)$
f	1	2	40	0
	2	3	21	0
	3	4	14.38	0
	4	5	11.08	0
	5	6	9.11	0
	6	7	7.81	0
	7	8	6.89	0
	8	9	6.21	0
	9	10	5.69	0
	10	11	5.29	0
	11	12	4.97	0
	12	13	4.71	0
	13	14	4.51	0
	14	15	4.34	0
	15	16	4.21	0
	16	17	4.1	0
	17	18	4.02	0
	18	19	3.96	0
	19	20	3.92	0
$n = 40$ where ν is minimal				

		ν	$\alpha(1)$	$\beta(1)$
f	1	20	72.73	0.82
	2	20	62.66	2.08
	3	20	66.14	3.88
	4	20	73.33	6.22
	5	20	81.14	9
	6	20	87.59	11.97
	7	20	91.15	14.75
	8	20	90.8	16.92
	9	20	86.19	18.11
	10	20	77.74	18.11
	11	20	66.49	16.92
	12	20	53.87	14.75
	13	20	41.33	11.97
	14	20	30.06	9
	15	20	20.80	6.22
	16	20	13.8	3.88
	17	20	8.92	2.08
	18	20	5.78	0.82
	19	20	3.92	0
$n = 40$ where $\nu = 20$				

		ν	$\alpha(1)$	$\beta(1)$
f	1	39	533.33	12.33
	2	38	1,561	77
	3	37	3,764.86	281.29
	4	36	7,254.26	724.32
	5	35	11,395.19	1,423.26
	6	34	14,913.26	2,235.82
	7	33	16,556.67	2,402.21
	8	32	15,818.71	3,162.5
	9	31	13,156.81	2,959
	10	30	9,614.24	2,402.24
	11	29	6,218.84	1,708.81
	12	28	3,582.49	1,073.33
	13	27	1,847.31	598.91
	14	26	1,847.38	298.22
	15	25	856.39	298.22
	16	24	358.45	132.84
	17	23	47.68	18.56
	18	22	16	5.42
	19	21	5.83	0.9
$n = 40$ where ν is maximal				