João Pedro Lobato de Carvalho

Master of Science

A Systems Approach to Minimize Wasted Work in Blockchains

Relatório intermédio para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: João Carlos Antunes Leitão, Assistant Professor, NOVA University of Lisbon

Co-orientador: Bernardo Ferreira, Postdoctoral Researcher, NOVA University of Lisbon

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

February, 2019

Abstract

Blockchain systems and distributed ledgers are getting increasing attention since the release of Bitcoin. Everyday they make headlines in the news involving economists, scientists, and technologists. The technology invented by Satoshi Nakamoto gave to the world a quantum leap in the fields of distributed systems and digital currencies. Even so, there are still some problems regarding the architecture in most existing blockchain systems.

One of the main challenges in these systems is the structure of the network topology and how peers disseminate messages between them, this leads to problems regarding the system scalability and the efficiency of the transaction and blocks propagation, increasing the time required to validate transactions, and, wasting computational power, energy and network resources.

In this work we prupose a novel solution to tackle these limitations. We will design membership and message dissemination protocols, based on the state-of-art, that will boost the efficiency of the overlay network, reducing the number of exchanged messages and the bandwidth used in the network. This solution will also reduce the computational power and energy required amongst the network, since the nodes will not have to proccess redundant network messages, and perform work that will be wasted.

Keywords: Blockchain, Peer-to-Peer, Overlay Networks, Membership, Message Dissemination

Resumo

Os sistemas de cadeia de blocos e livros-razão distribuídos têm ganho cada vez mais atenção desde o lançamento da Bitcoin. Todos os dias aparecem nas manchetes dos jornais económicos, científicos e tecnológicos. A tecnologia inventada por Satoshi Nakamoto deu um salto quantíco nos campos da computação distribuída e das moedas digitais. Mesmo assim, ainda existem alguns problemas relacionados com a arquitetura da maioria dos sistemas de cadeia de blocos.

Uma das principais limitações nestes sistemas é a estrutura da topologia da rede e a forma como os pares disseminam mensagens entre si, levando a problemas relacionados com a escalabilidade do sistema e a eficiência da propagação de transações e blocos, aumentando assim o tempo requerido para validar uma transação e, gastando poder computacional, energia e recursos da rede.

Neste trabalho, propomos uma solução original para atacar estas limitações. Vamos desenvolver protocolos de filiação(redes sobrepostas) e de disseminação de mensagens, baseados no estado da arte, para melhorar a eficiência da rede, reduzindo o número de mensagens trocadas e a largura de banda utilizada na rede. Esta solução irá reduzir também o poder computacional e energia requeridos na rede, uma vez que os nós não terão de processar mensagens redundantes, e então a realização de computações não uteis.

Palavras-chave: Cadeia de Blocos, Entre-Pares, Redes de Sobreposição, Filiação, Disseminação de Mensagens

Contents

Chapter 1

Introduction

Cryptocurrencies and Blockchain systems are very recent fields in Computer Science, that gained a lot of exposure and attention in the recent years [1]. In 2009, Satoshi Nakamoto released the Bitcoin [2] cryptocurrency as a open-source software, ten years after, there are many people who believe that the technology that Satoshi presented with Bitcoin is revolutionary and can solve many real world problems (e.g. economy) [3].

The technology behind the Bitcoin protocol is called Blockchain. Blockchain was the realization that the technology behind the bitcoin could be used not only as a currency but for many different kinds of operations. This led to a huge interest between financial institutions [4] and entrepreneurs, raising investments to discover how blockchain could impact our daily life (e.g. healthcare, insurance, voting).

In 2013, Vitalik Buterin described Ethereum in a white paper [5], and launched it in 2015. Ethereum is currently the second biggest public blockchain released to date. The main difference between Bitcoin and Ethereum are related with the type of operations supported by these systems. In particular, Ethereum supports records for assets such as contracts and not just currency. It also can be used to build smart contracts, which have programming capabilities (discussed in Section 2.1.5).

Blockchain systems are not perfect and there are lots of issues regarding the costs and the energy consumption of the devices that belong to the network[6]. One of the main problems is the protocol used for consensus amongst the network (i.e. proof-of-work), which are currently being replaced in most of these systems[7]. If we talk about numbers, at the current date, each Bitcoin transaction consumes 392KWh of electricity, giving a total of 45.76TWh annually. If we rank all the world countries and Bitcoin, Bitcoin is the 53rd "country"that consumes more electricity (for reference, Portugal is the 51st). If we add Ethereum to the calculations, the sum of both systems would occupy the 45th position[8].

Scalability is also a worrying problem, most of these distributed systems are aimed at a medium-scale network and fail by design to provide a fully functional and reliable operation at large-scale[9].

There are already many new blockchains that aim at solving most of these problems (we will give examples in Section 2.2), but, because of the complexity of these systems, there is still a long way until the appearance of a flawless blockchain. Even though, most of the systems currently available in the industry have active teams that improve those on a daily basis, improving many fundamental constructs of these solutions.

The work to be conducted in this dissertation is motivated by the following observations in current Blockchain systems:

- The resources wasted due to inefficient message dissemination protocols. There are too many redudant messages in the network, spending CPU cycles in the nodes to proccess these messages while also impacting the network bandwidth consumption.
- Computational resources waste due to outdated state on nodes. When new blocks are added to the blockchain, if the dissemination mechanism does not allow all nodes to become aware of the new block, nodes will be performing computations over an outdated state of the blockchain. Those computations will typically not be useful for the progress of the system.

The main goals of this dissertation are: *i)* study in detail how the current blockchain systems enable nodes to join the network and how messages are disseminate through it, *ii)* design an improved dissemination protocol which ensures reliability while reducing the number of messages in the network and lowering overall latency, and *iii)* implement a proof-of-concept to compare with current solutions employed by bitcoin and ethereum systems.

The remaining of this document has the following structure:

Chapter 2 presents a introduction to Blockchain systems and the current systems available, the underlying network protocols in Bitcoin and Ethereum, what is message dissemination and overlay networks, and the solutions found in the literature

Chapter 3 presents the working plan for the elaboration phase of the dissertation

Chapter 2

Related Work

This Chapter overviews the relevant state of art related with the goals of the project (as discussed previously).

In Section 2.1 we start with fundamental concepts related with blockchains and the architecture behind it. Section 2.2 presents the most common blockchain systems in the wild and discusses how they address the challenges related with privacy and scalability. We will also explain in depth the mechanisms used in Bitcoin and Ethereum to create and manage the overlay topology and disseminate messages throughout the network. In Section 2.3 we will explain what is message dissemination and present some relevant solutions found in the literature. In Section 2.4 we will explain what is a membership protocol, and present some fundamental concepts related with them and survey a few concrete solutions found in the literature. Finally, Section 2.5 concludes this chapter with a discussion on the key points of this state of the art survey.

2.1 Blockchain General Concepts

The idea of a distributed virtual currency is older than the Bitcoin system being constantly iconized in the Cyberpunk pop culture and in the mind of every Cypherpunk activist, expressed through their manifesto *"We must come together and create systems which allow anonymous transactions to take place."*[10].

The appearance of Bitcoin gave us a key advancement towards a distributed virtual currency, but also gave the society at large a better understanding on how we can use this technology, also known as Blockchain, to store permanent records in a decentralized and public network. Even though we now have the technology, it is still complex to take full advantage to build relevant applications on top of it. We now introduce the fundamental operaton of the Blockchain technology.

2.1.1 Blockchain Structure

A blockchain system is nothing more than a distributed and public (any node in the system can access it) ledger. This ledger is leveraged to store every transaction (valid) that is executed between any user. If we have Alice, Bob, and Charlie, we can have an example of a ledger for the payments executed between them, as ilustrated in Figure 2.1 where it is recorded that Alice paid Bob and Charlie, Bob paid Charlie and Charlie paid Alice (with Euros as currency).

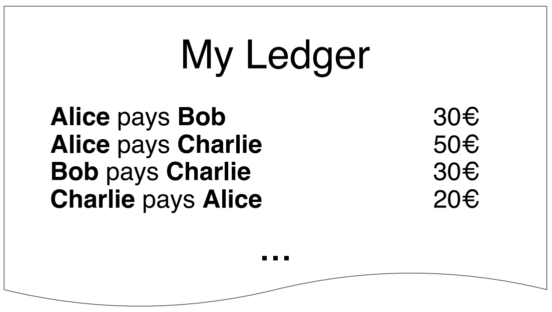

Figure 2.1: Ledger example

Apart of keeping track of all the transactions, there is an additional key functionality in this ledger: any user in the system can add transactions to the ledger. With this functionality arises one big challenge, if we do not trust the remaining users of the system, how can we assure that they will not add transactions without the approval of the other parties involved? To tackle this challenge the blockchain solutions typically rely on digital signatures, being every transaction signed by the sending user. To garantee that the signatures are not forged and that transactions cannot be replayed, every transaction is also numbered and use public-key encryption to sign the transaction, this way, even if we have similar transactions, the signature on the transaction will be different.

In order to distribute the ledger, every node in the system keeps a full copy of it. When some node wants to execute a transaction, it needs to broadcast the same transaction throughout the entire network and every node will add it to their own local ledgers after they verify the sender has enough currency to execute the transaction (as well as validating all relevant cryptographic signatures). This is a naive approach, in the real world we can not expect that the message will be delivered nor garantee the order messages are received by different nodes in the network. Therefore, we can not assume that all the ledgers in the network are equal and can not take different decisions regarding approving or reject particular transactions. To solve this problem, the nodes need a consensus (explained in Section 2.1.3) mechanism to aggree on a single order for processing all submitted transactions. Originally in the Bitcoin system, a consesus mechanism presented in Hashcash[11] was employed to this end, later labeled as proof-of-work (we will detail other consensus mechanisms also in Section 2.1.3). The consensus mechanism has two fundamental functionalities, it provides an extra layer of security and provides a consensus regarding the evolution of the system state. Therefore, instead of having a ledger where nodes insert broadcasted transactions, transactions are grouped in blocks and link those blocks together, effectively building the ledger as a chain of different blocks: the blockchain.

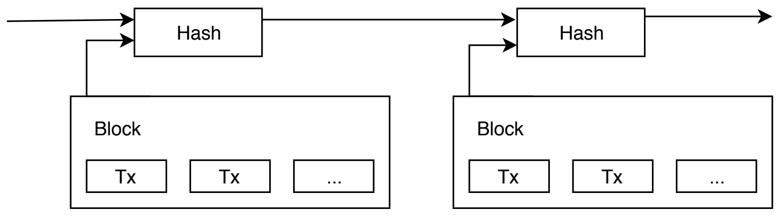

Figure 2.2: Blockchain schematic as in Bitcoin paper

All nodes in the system can create blocks with the transactions that are being broadcasted through the network (and not inserted in the blockchain yet). To create new blocks, the node has to give a proof of his work, which involves a hard cryptographic puzzle that can be easily verifiable. In the case of Bitcoin, the crypto puzzle is based on calculating a number that can be added to the end of the block such that the hash of it will have a pre-configured number of zeros. To garantee the order of the blocks, the block needs to have the hash of the previous block as parts of its contents (see Figure 2.2). In the end, a block becomes composed by the hash of the previous block, the transactions, and the proof of work. After a block is successfully generated it is broadcasted throughout the network, where it will be approved or rejected after other nodes confirm the transactions and the proof of work. If a node receives two different blocks the chain might diverge. Nodes will always select the sub-chain that is larger since that sub-chain has more effective work invested in it; if both sub-chains have a similar length, the node will wait to receive another block broadcast that will make one of these chains longer than the other, therefore, to actually approve a transaction, a node must wait until some blocks are chained on top of the one that contain the transaction.

Every time a node computes a new block, there is a special transaction on top of it, which is, the attribution of a pre-defined ammount of the currency for the block creator (also called miner). This ammount is issued by the system itself (called the block reward) and is used to encourage the nodes to actively participate on the network. This whole system model is fail proof unless most of the nodes in the system are not trustworthy, as we can see in a recent attack against Ethereum classic[12], where the attacker could get more than 51% of the total network processing power, producing a "trustworthy"chain. This attack caused $1.1 million dollar losses and raised the concerns about security on blockchain systems.

In the following sections we will describe how blockchain systems can be classified with regard to some design decisions.

2.1.2 Permissions and Access to Data

Regarding the permissioning, we can divide the blockchain systems in two main categories[13]:

Permissioned blockchains where only a restrict number of nodes can contribute to the consensus of the system and the processing of transactions.

Permissionless blockchains where there are no restrictions regarding the identities of the nodes, thus everyone can mine and create blocks (including submitting and validating transactions).

Blockchain systems can also be public or private, usually permissioned blockchains are also private while permissionless blockchains are public[14].

In a **public blockchain** everyone can download the blockchain ledger and read all the transactions and data.

In a **private blockchain**, only a set of predefined users can read the data and download the ledger.

Permissionless systems let any node that is willing to contribute with processing power to be part of the network, without requiring to prove its identity. This is a perfect scenario for large scale distributed systems removing the centralization of the power to control the network. However, this scenario comes also with a huge cost, regarding the consensus mechanism available, because there are many nodes who are not trustable, which requires longer and more computationally expensive consesus schemes.

2.1.3 Consensus Mechanisms

The consensus mechanism allows nodes to agree on the outcome of all the transactions submitted to the system to ensure all nodes reach the same state. The mechanism has to ensure that the transactions only will be added if they are valid and that the transactions are not recorded more than once.

There are lots of mechanisms used in the community, amongst those, there are three main mechanisms, the proof of work, proof of stake, and the practical use of byzantine fault tolerance distributed protocols.

Proof-of-Work[15], uses a cryptographic puzzle that requires processing power to solve it and determine the winner, usually the complexity of this puzzle increases with the number of times it has been solved.

Proof-of-Stake[16] determine the winner using a combination of randomness and the stake weight (coins that a user held in their wallet), stake based on age can also be used. In non-cryptocurrency blockchains, the Proof-of-Stake is replaced by alternative approaches such as voting, since there is no stake by default.

Practical Byzantine Fault Tolerance mechanisms[17] are often used in permissioned blockchains (e.g. Hyperledger[18]), where new blocks are added if more than $\frac{2}{3}$ of all peers effectively validate them.

Proof-of-Work was the original consensus mechanism used in Bitcoin, but with the growing of the coin value, there was also a significant increase in the number of miners, which led to a huge need of computational power to mine new bitcoins. This need of computational power was proved a waste of resources and energy[19]. Practical Byzantine Fault Tolerance mechanisms are a very good consensus mechanism, but it is not usable in a network where more than one third of the peers are malicious. Proof-of-Stake also have some limitations regarding the centralization of the mining due to the centralization of the stakes, although, Vitalik Buterin (co-founder of Ethereum and co-founder of Bitcoin Magazine) suggested an hybrid approach to this issue where blocks are produced via proof-of-work and signed by a number of stakeholders[20], solving the disadvantage of proof-of-stake (because its high dependence on proof-of-work).

2.1.4 Currency

The blockchain system emerged with the Bitcoin[2], the first modern cryptocurrency designed to prevent the weaknesses in their predecessors, such as double-spending attacks and centralization of the ledgers, preventing scalability[21]. Being a cryptocurrency, it provided a mathematical way to generate currency that is nearly impossible to falsify[22]. The system also specified how the generated currency should be distributed among the users of the network and the operations available to the users (send currency to each other).

Many blockchains came after Bitcoin (most of them not designed for trading, but for applications) and most of them keep the idea of providing a currency available for the users of the system, mainly for the maintenance, giving it as a reward to the users (mainly the ones who process the transactions).

2.1.5 Smart Contracts

Smart contracts[23] are similar to contracts in the real world, except they are digital. They are computer protocols intended to facilitate, verify, and enforce a contract. To achieve these goals, smart contracts have two properties, they are immutable and distributed.

With smart contracts, blockchains gain a programming capability where they can build and deploy decentralized applications. This comes from the development of the blockchain system, where the developers provide a Turing-complete scripting language (e.g.: Ethereum[24]). Some blockchains have limited programming capability, making it harder to develop applications on top of them (e.g.: Bitcoin).

2.2 Blockchain Systems

After Bitcoin there was a sudden emergence of alternative systems (also known as altcoins) that in one way or another improved the original feautures of Bitcoin or just tried to solve the some of the limitations of blockchain systems.

2.2.1 Systems Overview

Litecoin[25] and Dash[26] appeared as alternatives to Bitcoin in order to improve the high transaction latency, reducing the block mining time and allowing faster transactions.

Peercoin[16] presented proof-of-stake as an alternative to the proof-of-work approach, changing how the consensus operates in the network. Ripple[27] proved the application of the blockchain technology in financial industry with its real-time gross settlement system. NEM[28] demonstrates that it is possible to have multiple ledgers in the same blockchain and provided a new version of proof-of-stake: the proof-of-importance.

Ethereum gave us a Turing-complete scripting language proving that it is possible to perform full computations accross the blockchain, although it was possible to perform computations in prior blockchains, it was unnoticed untill the appearance of Ethereum.

IOTA[29] appeared to try to mitigate the scalability problems with the Tangle concept. Instead of having the transactions grouped into blocks, IOTA allows transactions to be entagled together. This project was directed to Internet of Things applications, but it can also be applied to other application domains.

Regarding privacy, Monero[30] worked with privacy in mind. The main feature of this system is the usage of an obfuscated public ledger where anybody can broadcast or send transactions to, but no one (other than the sender and the receiver) can tell the source, amount, or destination. Enigma[31] provides privacy through secret contracts. Secret contracts are smart contracts that enforce correctness of the system and privacy, since they can hide the data completely from other nodes. Enigma uses secure multi-party computation[32], which let the different peers perform computations over their data while maintaining their inputs secret.

In the end of 2015, the Linux Foundation announced the Hyperledger Project[18] which consists in a group of tools and frameworks for the development of blockchain-based systems. This project is also supported by the research community and industry (e.g. IBM and Intel), giving it high credibility with the big names involved on the project. The most famous framework of the project is Hyperledger Fabric[33], that provides a blockchain infrastructure for developing applications on demand.

2.2.2 Bitcoin

We already discussed the design and core features of Bitcoin[2], this is because of the importance of this system as a pioneer in the domain of blockchain. It was the first blockchain system applied in the real world and it is still reliable after more than 10 years.

We will now present and explain the underlying mechanisms of the Bitcoin network.

Join Mechanism

Nodes join the network[2][34][35] by discovering a bootstrap node and sending to it a version message (with the node version number, block, and current time). The bootstrap node will respond with his version message and then both nodes send an acknowledgment message indicating that the connection has been established.

After the initial connection is established, the new node sends a *getaddr* message in order to gather additional peers. When a node receive a *getaddr* request it sends the addresses that have a timestamp in the last 3 hours, to a maximum of 2500 addresses (selected at random). Nodes also attempt to keep a minimum number os connections p (this is a protocol parameter, in the case of Bictoin is 8), if the number of open connections is below p, the node will try to connect with known addresses, these peers do not close open connections even if they more than p open connections. Therefore, the total number of open connections is likely to be higher for nodes that accept incoming connections (nodes behind firewalls will have p open connections).

With the current join mechanism, the overlay network will be unstructured, forming a random graph, with potentially a very high degree (i.e. each node will have a large number of neighbors).

Node Discovery

In order to discover other nodes[2][34][35], there are many possibilities regarding which methodology to employ. When a client joins the Bitcoin network for the first time, it can try to discover neighbors through:

- **Hard coded DNS seeds** The client has a group of hard coded DNS hostnames. After querying one or more of those DNS services, is expected a response with multiple nodes address that may accept new incoming connections. These addresses will initially have a timestamp equal to zero, therefore they will not be announced in response to *getaddr* messages.
- **Hard coded seed addresses** The client has a group of hard coded seed addresses that may be used as a last resort. There are mechanisms to cancel the connections to these nodes as soon as possible in order to avoid overloading them. These addresses will initially have a timestamp equal to zero.
- **Command line addresses** The user can specify multiple IP addresses through the command line. These addresses will initially have a timestamp equal to zero.
- **Text file addresses** The client can have multiple IP addresses in the file "addr.txt"under the bitcoin data directory. These addresses will initially have a timestamp equal to zero.

If the client is currently in the network, there are additional mechanisms to discover additional overlay neighbors:

- **Local client external address** The client discover is external address (using a third-party service) and announce its public address to the network.
- **Connect callback address** After making the initial connection, the client issues a *getaddr* message to gather more peers (explained previously).
- **Ongoing addr advertisements** The client can listen ongoing *addr* advertisments since nodes advertise addresses in different situations (when they relay them, when they advertise their own, and when a new connection is made).

- **Addresses stored in a database** The client also adds peer addresses to a local database, if it needs additional peers, it can try to connect to the peers in its local database.

Nodes Leave and Failure

Bitcoin does not have a way for nodes to leave the network[2][34][35]. Nodes leaving the network and failing will have their addresses lingering in the network for some time. Nodes by default send a heartbeat message to peers from time to time (before 30 minutes of inactivity). If a node does not receive a message from a peer for more than 90 minutes, it will assume the connection with the peer has failed and it will purge it from their known peer addresses set.

Message Dissemination

To disseminate transactions and blocks in the network, the bitcoin node announces their availability with an *inv* message to its neighbors (once verified)[2][34][35][36]. If a neighbor does not have the transaction or the block, it will send a *getdata* message to the original node with the hashes of the information being requested. After that, the information is sent to the neighbor.

The message dissemination efficiency in the current model will depend heavily on the topology of the network, that in this case its fundamentally random.

There was also recent scientific research to improve information dissemination in the Bitcoin network[37] based in neighbour rankings. It showd an improvement in the performance when compared with the techniques employed today for information dissemination, but the neighbour ranking did not provide a relevant improvement in some of the metrics (e.g. bandwidth consumption).

2.2.3 Ethereum

Ethereum is an open-source and permissionless blockchain system that enable users to build and run smart contracts and distributed applications (ĐApps). The Ethereum blockchain implements a Turing-complete language providing a good framework for users to build smart contracts and ĐApps. These applications, are offered the possibility of creating tokens, therefore, most of the startups that launch tokens, are actually issuing a distributed application over the Ethereum network.

The principal characteristic of Ethereum is its active community and the continuous evolution of the Ethereum protocol that strives to optimize many of the different subprotocols in use on their blockchain.

Node Discovery and Join Mechanism

Ethereum implements its own peer discovery protocol[38], based on the Kademlia protocol[39]. Originally, Kademlia was designed for storing and locating resources in a peer-to-peer network and also as a mechanism to locate particular nodes. In the Ethereum network, it is only used for node discovery and topology maintenance. The only RPC calls enabled in the discovery protocol are *Ping* and *FindNode*, where *FindNode* accepts the node identifier (not Kademlia ID) as an argument. The Kademlia ID is calculated with the hash function applied over that node identifier (generating a 32-byte value) which is used to calculate the "distance"between nodes (leveraging a XOR based computation).

The Kademlia's routing table consists of a set of lists (or *k-buckets*) where each *k-buckets* holds a maximum of *k* entries. The number of *k-buckets* is the same as the number of bits of the Kademlia ID. Each *k-bucket* corresponds to a distance to the other nodes, this method guarantees that the node will have different peers at different distances, improving the reliability of their neighboring connections and the routing on the Kademlia network.

For a node to discover a bootstrap node (when he wants to join the network) he has two alternatives. He can get one bootstrap node from a hardcoded list in the implementation or he can ask to a well known discovery endpoint setted for that particular purpose (some of these endpoints are maintained by the Ethereum Foundation).

When the joining node finds a bootstrap node, it starts a join process where it will initialize its own buckets and querying the bootstrap node for neighbors (eventualy removing the bootstrap node to avoid overloading it).

With the current join mechanisms, the overlay of the Ethereum network is fundamentally structured. Its a structure is similar to all Kademlia DHT-based networks.

Nodes Leave and Failure

When a node wish to leave the network[40], it will send a *Disconnect* message to its peers. This message receives a *reason* parameter where nodes can tell their peers if they are leaving the network or if there is another reason for the disconnecting (e.g. incompatible P2P protocol version). After sending the message, hosts wait a short time before disconnecting themselves, for peers to disconnect first.

In Ethereum there is no mechanism to handle node failures, but a node can easily detect if one of its peers has failed the next time it sends him a message (because of the use of TCP connections). Even though, this mechanism can lead failed nodes to remain in the buckets of correct nodes for long periods of time.

Message Dissemination

There are many different types of messages in the Ethereum protocol[40] but the three key messages are the *NewBlock*, the *NewBlockHashes*, and the *Transactions*.

If a node has a block that the peers should know about, it will send a *NewBlock* message with the block itself.

When a new block (it can be more than one) appears on the network, the node sends a *NewBlockHashes* message to his peers. This message can have a maximum of 256 block hashes. If the peer does not have at least one of the blocks corresponding to those hashes, it sends a *GetBlocks* message requesting the blocks unknown to him, then it will receive a *Blocks* message with the requested blocks contents.

When a node has a transaction (it can be more than one) that the peers should include in their transaction queue, the node will send a *Transactions* message.

The efficiency of the message dissemination protocol, likewise to the bitcoin system, is highly dependent of the network topology. However the Ethereum network, due to its structured nature, appears to be more efficient than the Bitcoin network for message dissemination.

2.3 Message Dissemination

2.3.1 Dissemination Protocols

Dissemination protocols define how messages are disseminated throughout the network. One of the typical example of a dissemination protocol is the flood mechanism, which will flood all the network links with the message being broadcasted. Of course this is not efficient and there are many protocols that achieve better network usage compared with flooding. We will now present key data dissemination schemes and how they define the way messages travel over the overlay network topology.

Publish-Subscribe

Publish-Subscribe[41] is a architectural messaging design pattern. With this pattern, the peers who send messages to the network (*publishers*) and the peers who receive the messages in the network (*subscribers*) do not have to know each other. Publishers categorize the messages with two different subscription models: *topic-based* and *content-based*. Subscribers express their interest in a category of messages (based in the subscription models being employed) and only receive messages related with that (or those) category (or categories). The process of selecting the subset of messages for each subscriber is called filtering.

In *topic-based* solutions, messages are classified into different topics. Subscriptions include all the topics of interest to the subscriber. When a message has a topic that matches with a peer subscription, the message is delivered to the subscriber.

In *content-based* solutions, messages receive multiple attributes where subscriptions are allowed to define ranges over these attributes. When the attributes of a message matches the specification of the subscription, the message is delivered to the subscriber.

The advantages of the publish-subscribe model are:

- There is no need for peers to know the existence of each other or the rest of the system topology. If the messages are broadcasted through the network correctly, it is guaranteed that the peers will receive the messages they subscribed.
- It provides better scalability than centralized approaches (e.g. client-server model). This advantage comes from the distributed system point of view, since there is no overload in a central peer and there is the possibility of doing parallel operations.

There is also a big weakness in this model related with the first advantage. Like we said before, the messages need to be broadcasted through the network, correctly. Such system needs to provide strong guarantees of message delivery, otherwise peers may lose most of the messages disseminated in the system.

Gossip Protocols

Gossip protocols[42] (also known as anti-entropy protocols or epidemic protocols), are an efficient way to spread messages over the network. They were proposed to solve many problems in distributed systems (e.g. publish-subscribe problems[43]).

In a gossip protocol all the peers collaborate in order to disseminate information through the network. When a node wants to broadcast a message, it sends the message to a number of random nodes (the number of nodes is parameterizable, and usually called *fanout*). When a message is received, there are two scenarios, the node receives the message for the first time or has already received it. In the first scenario, the node will forward the message to a number of random nodes (typically using the same *fanout*). In the second scenario, the node simply discards the message.

Likewise to Publish-Subscribe, gossip protocols are foundations for highly scalable systems. It also provides robustness to systems because of the intrinsic redudancy in gossip-based dissemination, being able to mask network omissions and node failures.

It is also important to note that there are different strategies to implement gossip protocols:

- **Pull:** Nodes query random peers for information, if the peers have new informantion (e.g. messages), they forward it to the node.
- **Eager push:** Nodes send information to random peers as soon as they receive new information.
- **Lazy push:** Nodes send a unique identifier to random peers when they receive new information. If the receiving peer does not have the corresponding information, it will make an explicit pull request.

These strategies are good for specific scenarios. There also exist the possibility of combining different strategies to build hybrid approaches.

Epidemic Broadcast Trees

The Epidemic Broadcast Trees[44] paper presented a gossip protocol, named push-lazy-push multicast tree (or *Plumtree*), that shows how to mitigate the message redundancy in gossip-based dissemination protocols without compromising their robustness. In order to do that, the authors rely on a multicast tree that covers all nodes in the network, effectively achieving a structured overlay on top of a random overlay (we discuss overlay networks further ahead).

The *Plumtree* protocol has two main components:

- **Tree consctruction:** responsible for choosing the links in the random overlay network that will forward the messages using an eager push strategy.
- **Tree repair:** responsible of repairing the tree when node failures happen. The main objective is to guarantee that all nodes are covered by the spanning tree.

After constructing the spanning tree, each node maintains two set of peers, one destined to use eager push gossip and the other to use lazy push gossip. The whole proccess of creating both of these sets garantees that: *i)* when the first broadcast is terminated, a tree is created, *ii)* assuming a stable network, the generated spanning tree minimizes the message latency (to the sender of the original message), and *iii)* there is a decentralized and timely mechanism to detect and recover from failures (thanks to lazy push messages).

This protocol offers high reliability even in presence of a large number of faults. It also showed lower latencies and less message redundancy with a low cost scheme for build and maintain the topology.

2.4 Overlay Networks

In the context of peer-to-peer systems we can define an overlay network[45] as a virtual (or logic) network deployed on top of another (typically the physical) network in order to improve the utilization of the information and the resources in the system. This virtual network is composed by many logical links between the different peers that compose the system, independent from the physical network. Based on the organization scheme and the graph formed considering the nodes and the links between them, we can classify an overlay network as unstructured or structured[46].

Structured Overlays are created with a well defined organization scheme as basis, typically, based on unique identifiers for each node, having very predictable topologies. Because of the well defined structure of these networks, we can easily find the location of a node even without a whole image of the system. These networks usually have a lower network diameter, enable more reliable links between nodes which can be organized considering a specific performance metric (e.g. latency or hop count) and also enable easier content finding through the whole network since such contents can be indexed based on node identifiers.

Unstructured Overlays usually have a random topology and are unpredictable, even with complete information about the system filiation, in the sense that even knowing all the details about one joining node, one can not predict where the node will stay in the network and what will be the nodes that will be connected to it. Because of this random nature (that imposes very few restrictions), these networks are often more reliable in scenarios where a large number of nodes may fail or where a large number of changes in the overlay topology can happen (e.g. in churn scenarios)[47].

2.4.1 Distributed Hash Tables (Structured Overlay Networks)

One of the best examples of structured overlays are Distributed Hash Tables (DHTs). They are a decentralized system that enable the same lookup features as a Hash Table, mapping keys to values.

In the end of the nineties and the begin of the new millenium, many peer-to-peer systems appeared in order to take the full capabilities and resources of the Internet to build file-sharing services, some examples of these systems are Gnutella[48], Napster[49] and BitTorrent[50]. Some of these systems were not fully decentralized, relying on a centralized server to index the files in the network, this model would mantain the issues related with single points of failure[51]. After that, there was a motivation to research DHTs in order to provide the decentralized operation and efficiency features.

The appearance of protocols such as Chord[52] and Pastry[53] ignited the DHTs as a hot research topic, enabling the appearance of advanced protocols like Kademlia[39], the overlay protocol used by the Kad network[54].

The main reason for the boost of popularity of this type of overlay network is their high flexibility and scalability properties, providing an application-level routing infrastructure with minimal information about the membership of the network.

The Hash Table has two functions **get(key)** and **put(key, value)**. When we call the get function, the Hash Table will hash the key in order to find the bucket that has it, in a DHT a key is hashed in order to find the node responsible for that key (e.g. the name of the file we want lookup in file sharing services). To ensure that every time that the number of nodes in the system changes, we do not have to redistribute all of the items again over objects, one usually resorts to consistent hashing[55].

2.4.2 Membership Protocols (Unstructured Overlay Networks)

Some membership protocols are applied to unstructured overlay networks in order to define a management strategy. We will now give examples of membership protocols and how they manage networks.

One key algorithm used by these protocols are random walks[56]. A random walk is a type of random process (or stochastic process), which envolves taking a series of steps where each step is determined with some random probability. Random walks are used in many different fields apart of epidemiology (e.g. physics).

Cyclon

Cyclon[57] is a gossip-based membership management protocol designed to manage unstructured overlays, building graphs with low diameter, low clustering, similar node degrees and being resilient to node failures (in terms of overall connectivity). Its operation is based on a partial view of the whole network with a parameterizable fixed size maintained individually by each node in the system, nodes in the partial view of a node are considered their neighbors.

The protocol uses a shuffle algorithm to continuously change the neighbors of a peer from time to time. The algorithm works as follows, pediodically, a node increments the neighbors age and then choose the oldest node in its neighbors and exchange a subset of its neighbors with a subset of the other node neighbors. If the other node does not reply to the shuffle message, the node will assume that the node failed and it will remove it from the neighbors when a node executes one of there steps it also creates a new identifier with age sera that is sent alongside the scope of its partial view. There is no distinction between nodes leaving and nodes failing.

To join the network, a node has to know another node that belong to the overlay. The join operation is done with random walks on the overlay. This proccess has an important observation: *"... due to the randomness of the connectivity graph, a random walk of length at least equal to the average path length is guaranteed to end at a random node of the overlay, irrespectively of the starting node."*, with this property, there is also a guarantee that the in-degree of all nodes will remain unchanged.

Scamp

Scamp[58] is a gossip-based membership protocol characterized for removing the need of a complete knowledge of the global membership and having two partial views, one where the nodes select the target nodes for sending gossip messages (*outView* for reference) and other with the nodes from which they receive gossip messages (*inView*). Scamp has a scheme where the size of the views converge to an optimal value for gossip to succeed. This value is a function of the system size (we have to keep in mind that these nodes do not know the system size).

When a node wants to join the network, it sends a subscription request to a member of the overlay network and puts that node in its *outView*. When a node receives a subscription request, it forwards the node identifier to all the nodes in their *outView* and creates a fixed number (design parameter to determine the proportion of failures tolerated) of additional copies that are sent to randomly chosed nodes also in the *outView*. When a node receives one of the forwarded subscription requests, it adds the new subscriber in its views with a probability depending on the size of its local partial view, otherwise, it forwards the subscription to a random node in its *outView*. To prevent node isolation, there is a periodic check mechanism that nodes can use.

When a node wants to leave the network, it sends an unsubscription message, which is disseminated through the whole network as a gossip message. When a node receives an unsubscription message, it removes the node from its views (only if it is present in its views, of course). If the node fails ungracefully, the protocol guarantees that eventually all other nodes in the system will remove the failed node from their partial views.

HyParView

HyParView[59] is also a gossip-based membership protocol that addresses the challenge of minimizing the restauration time in the presence of large numbers of (concurrent) node failures, ensuring high levels of reliability in such scenarios. HyParView uses two partial views, one with the peers with whom that node comunicates (*active view*) and another with different peers from the whole overlay (*passive view*), used for fault recovery.

When a node wants to join the network, it sends a join request to a node already present in the system. A node that receives a join request adds the new node to its active view (if it has to drop a node from the view, it sends a disconnect message to the node being removed from its views) and it will forward the join request to all its active view nodes. This request will be propagated through the overlay using a random walk. The protocol has two configuration parameters which define when a node will be added to the active and passive views, these are the *Active Random Walk Length* and the *Passive Random Walk Length*, respectively.

There is no distinction between leaving or failed nodes. Nodes have a reactive mechanism to maintain their active view always updated. When a node suspects that one node in their active view has failed, it will remove it and substitute for one in the passive view, removing every nodes that it fails to connect to from both views. When sending a neighbor request, the node can set the priority as high (if the node does not have any elements in its active view) or as low otherwise. When receiving such requests, if the priority is set to high, the node will be added in the receiver active view (if it has to drop a node from the view, it sends a disconnect message), otherwise it will accept the request only if it has a free slot in its active view.

To maintain the passive view, nodes perform a shuffle operation with random peers. The node will create a set with its own identifier, a subset of the active view and a subset of the passive view (the numbers of elements for each subset are protocol parameters). After the creation of this set, the node sends a shuffle request to a random peer which will be propagated using a random walk, likewise to peer identifiers in the join request. The node at the end of this random walk, will send a shuffle reply with a subset of its passive view, with the same number of peer identifiers as in the original shuffle request. In the end, both nodes will integrate the received shuffle messages in their own passive views, ensuring that all the nodes in the overlay have a diversified passive view.

2.4.3 Topology Mismatch Problem

The topology mismatch between the overlay network and the underlying physical IP topology can impose a huge stress on the physical network, leading to issues related with the efficiency of applications and network overhead. This problem is known as *Topology*

Mismatch Problem and can be found on structured and unstructured overlay networks alike.

T-Man[60] and X-BOT[61] appeared to solve this problem. These protocols allow to bias the network overlay in order to optimize some criteria. X-BOT appeared after T-Man and proved itself to preserve some key properties (e.g. the overlay connectivity) in scenarios that even T-Man was not able to. For instance, when T-Man uses latency as a criteria to bias the topology of an unstructured overlay network, it might compromise the overlay connectivity and partition the network. Both of these works were further studied to create optimized DHT protocols[62][63].

2.5 Summary and Discussion

Blockchain systems are continuously evolving. In this chapter we presented how general blockchain systems operate, some of those and how they are solving the blockchain challenges. For entrepreneurs, it is a good idea on having a different blockchain system solving each problem, because that way they keep diferentiate from each other and continue the innovation, giving space for new businesses each with an advantage. But on a systems view, it is better to have a flawless blockchain. The blockchain system that is working best from a systems prespective is Ethereum[5], they keep working on the protocols under the hood, improving these core protocols in order to get better system performance, instead of being afraid of the change, as the Bitcoin community.

This is one more reason to study the Ethereum system in depth. We saw how Bitcoin and Ethereum networks are defined and disseminate information between peers.

After some understanding on both of these systems, we can relate the challenges existing in blockchain systems to the overlay topology of the network, that is not maintained correctly, and to the message dissemination protocols used, that do not deliver messages about transactions and blocks effectively.

We also presented the state of the art protocols related with membership and message dissemination. Those protocols present the base idea to solve the blockchain challenges.

In the next chapter we will provide some insight on how to solve one of the main challenges regarding blockchain leveraging on the technologies presented so far.

Chapter 3

Work Plan

In this Chapter we present the work plan for the course of the thesis.

In Section 3.1 we will revisit the problem of Blockchain systems (stated earlier) that we aim at addressing. In Section 3.2 we will explain how we plan to tackle the problem at hand. Section 3.3 discusses our preliminary plans for evaluating our solution comparing with the state of the art, and further details the performance metrics that will be considered. In Section 3.4 we will present a detailed schedule for the remainder work to be conducted in the context of the thesis.

3.1 Revisiting the Thesis Problem

The message dissemination protocols used in blockchain systems are based on flooding all neighbors with messages concerning both transactions and blocks. The effectiveness of this message flooding mechanisms is highly dependent on the overlay network topology. Considering a topology supporting the operation of a blockchain system, we can improve the efficiency of the flooding mechanism, but it is not enough, the system network will still have many redudant messages and significant network bandwith will be used in a non-necessary way, while also having significant overhead in terms of proccessing power (CPU cycles), and (ultimatelly), energy consumption.

There is also a waste of computational resources due the unawareness of the new blocks in the blockchain (due poor dissemination mechanisms). Nodes may perform computations over an outdated state of the blockchain that will not be useful for the progression of the system.

These are some of the main problems with blockchain systems in the moment. Apart of the waste of resources mentioned earlier, it will also translate to scalability problems too. Blockchain systems depend on the capacity to proccess blocks and transactions accross the network, if we do not have mechanisms that allow to do that efficiently, as the network grows, the time to proccess blocks and transactions will also grow (unfortunately above linearly).

This thesis will only tackle these problems in the view of the efficiency of the system network usage.

3.2 Envisioned Solution

To tackle the existing limitations of blockchain systems, we have to change not only the structure of the overlay network, but also the way nodes disseminate messages over that network. We will develop a protocol that create a biased overlay network with a gossip message dissemination approach. The goal is to significantly reduce the network cost, processing cost, and dissemination latency as to benefit the overall operation of the blockchain system.

3.2.1 Overlay Network and Membership

Our solution consists in creating an overlay network that combines two different forms os topology bias, leveraging on X-BOT[61] as a departure point to build such a solution. We have to keep in mind that this approach needs to be adapted to a blockchain system and it does not consists only in applying the X-BOT protocol to create an overlay network for blockchains. Applying X-BOT over the blockchain system might create new attack vectors where an attacker can exploit the neighbors of a node, blocking all the transactions or blocks mined by that element. The topology bias criteria that we plan to initially study are:

- **Geolocation:** The physical location of nodes and the proximity between them and their peers.
- **Secondary Bias:** A secondary characteristic that combined with the first topology bias, will garantee the overlay network connectivity (subject of study).

The geolocation bias will guarantee that the nodes will be physically close to each other so we can expect lower latency between them. The problem associated with the geolocation bias is the possibility of formation of clusters and network partitions, a phenomenon studied in previous works[59][61].

To address the challenges inherent to handle the geolocation bias, we have to select a secondary bias in order to maintain the network connectivity. For the secondary bias, we will test different approaches and compare results, for reference, some secondary bias that will be studied are the node uptime and other stability characteristics, we will also study the usage of an improved secondary bias adapted for other blockchain characteristics, namely the consensus mechanism (e.g. figuring out the probability of the next mining nodes and connect with them), like previous works tried to achieve[37].

The way bias are applied over the overlay network topology will also be further studied and tested. Currently, we consider two different approaches. The first approach nodes will give a score (based on the two topology bias) to every peers, and the node will connect with peers with higher scores, the second approach will separate the neighbors in two sets where each set will be influenced by each bias. The results of both of these approaches will be highly dependent on the secondary bias.

Blockchain networks usually do not have the need for lookup node information in the network, since every node is a replica of the full system state (i.e. the blockchain) including both blocks and transactions. Because of this feature we do not have to guarantee the communication between nodes that are not in the neighbour list of a specific node.

Join and leave (either leaving or failing) mechanisms will be addressed in the membership protocol. The way nodes discover a bootstrap node to join the network will not be addressed in this work.

3.2.2 Message Dissemination

We want to minimize message redundancy, lower latency, and improve robustness, therefore we will use an approach similar to the one originally proposed in epidemic broadcast trees[44] but it will also be adapted for our use case, particularly in what respects to the use of a biasing mechanism in the underlying unstructured overlay network.

3.3 Evaluation

Our solutions will be evaluated against the two most used Blockchain systems (Bitcoin and Ethereum) and any scientific work that might be relevant or interesting to use as a baseline.

We plan to measure latency, throughput, bandwith, and number of messages effectively traversing the network. We think our solution will greatly outperform any other approach in latency, bandwidth, and number of messages in the network, and might outperform in throughput as well, although somewhat less when compared with other metrics.

We also plan to study the overlay topological properties that result for applying our topology biasing mechanisms. To this end we will follow the methodology originally proposed in X-BOT[61].

For the evaluation we will rely initially on simulations and will implement our own simulator because the few simulators that exist are oudated or are not adequated to blockchain systems. We also have in mind running tests in servers provided by Amazon or Google, but we are still analyzing this options because of the limited resources these services provide for free. Such experiments would show the feasability of our proposed solutions in real execution environments.

3.4 Scheduling

To achieve our objectives we consider the following tasks and subtasks as defining the key actions to be tackled during the course of the thesis.

3.4.1 Tasks

Protocol Development

This task comprehends the design and specification of protocols (both overlay management and data dissemination), the implementation of the protocols themselves and the implementation of the simulator that will be used for the first phase of evaluation. The implementation subtask also comprehends the implementation of the state-of-art protocols, with possible adjustments.

Evaluation

This task consists on the evaluation of the different existing approaches and our own proposed solutions. This task has two different substasks the testing and the comparison and it is divided in two phases. The first phase will have the preliminary testings for writing articles and the second phase will have a more detailed evaluation for the thesis.

Writing

This task involves all the writing to be performed during the course of the thesis. It has two different subtasks, the *Paper writing* and *Thesis writing*. The first is destined for possible papers reporting the current work in different conferences, the second is for preparing the final thesis document.

The selected conferences were selected based on the topic and the timing for submission period. The conferences selected are: SRDS 2019, Middleware 2019 and Inforum 2019. The conferences can change depending on the status of the work in the submission period. The number of papers to submit will depend on the effective results obtained.

Tasks Duration

Table 3.1 presents how much time is predicted for executing each task and subtask. These durations might differ from the reality and were defined mostly based on the total amount of available time and the relative complexity of tasks and subtasks.

Tasks	Weeks
Protocol Development	**17**
Design	8
Implementation	10
Simulator implementation	7
Evaluation	**11**
Approaches Testing	4
Approach Comparison	2
Protocol performance test	5
State-of-art performance test	5
Results Comparison study	4
Writing	**18**
Paper writing	10
Thesis writing	8

Table 3.1: Schedule table

3.4.2 Gantt Chart

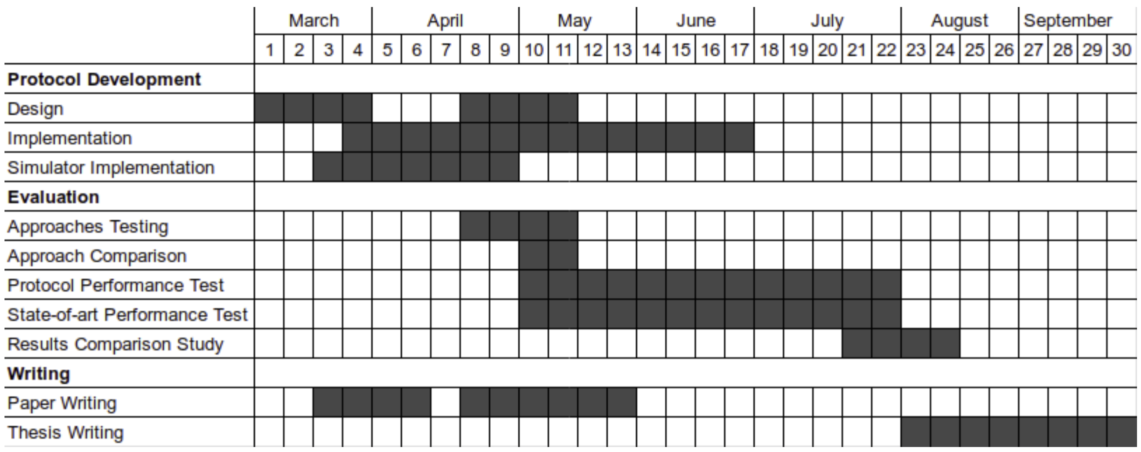

Figure 3.1: Gantt chart with planned scheduling

Figure 3.1 presents the gantt chart with the planned scheduling for the work to be conducted in the context of this project.

3.5 Final Remarks

Blockchain systems promise to have a significant impact on the technological landscape, but they still have to tackle some limitations until they can be efficiently used in our daily lives.

In this document we presented what a blockchain system is, how they internally work, identified some key limitations in their designs and discussed how the new systems that are appearing, are solving some of these issues. There are still many topics that were not presented or not deepened in this document.

We believe that currently, blockchain systems might be an overhyped hot topic. The technology has the potential to be applied to many society problems, but the topic it is not yet mature to efficiently address those problems and, currently, it is not showing any significant improvements (e.g. cryptocurrencies vs. VISA number of transactions) with the exception of anonymity. With our work, we want to solve one of the main problems regarding the scalability of the system and the efficiency of the communication between the network nodes.

To achieve our objective, we will work on a novel solution that will create an overlay network with a topology biased towards supporting highly efficient and fast data dissemination, providing better node neighbors without sacrificing the overlay connectivity nor any key blockchain characteristic. We will also reduce the number of messages that are disseminated in the network using an adaption of epidemic broadcast trees as a gossip protocol in order to effectively broadcast transactions and blocks over the network without redundancy.

We also discussed our initial plan to evaluate our solution and compare it with the current state of the art.

Bibliography

[1] T. Abate. *Stanford computer scientists launch the Center for Blockchain Research*. 2018. URL: https://engineering.stanford.edu/news/stanford-computer-scientists-launch-center-blockchain-research.

[2] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf*.

[3] D. Pollock. *The Fourth Industrial Revolution Built On Blockchain And Advanced With AI*. 2018. URL: https://www.forbes.com/sites/darrynpollock/2018/11/30/the-fourth-industrial-revolution-built-on-blockchain-and-advanced-with-ai/.

[4] L. Shen. *Blockchain Will Be Used By 15% of Big Banks By 2017*. 2016. URL: http://fortune.com/2016/09/28/blockchain-banks-2017/.

[5] V. Buterin. *Ethereum White Paper: A Next Generation Smart Contract And Decentralized Application Platform*. Tech. rep. URL: http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.

[6] M. Orcutt. *Blockchains Use Massive Amounts of Energy - But There's a Plan to Fix That*. 2017. URL: https://www.technologyreview.com/s/609480/bitcoin-uses-massive-amounts-of-energybut-theres-a-plan-to-fix-it/.

[7] A. Hertig. *Ethereum's Big Switch: The New Roadmap to Proof-of-Stake*. 2017. URL: https://www.coindesk.com/ethereums-big-switch-the-new-roadmap-to-proof-of-stake.

[8] *Bitcoin Energy Consumption Index*. 2019. URL: https://digiconomist.net/bitcoin-energy-consumption.

[9] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer. "On Scaling Decentralized Blockchains - (A Position Paper)." In: *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*. 2016, pp. 106–125. DOI: 10.1007/978-3-662-53357-4_8. URL: https://doi.org/10.1007/978-3-662-53357-4_8.

[10] E. Hughes. *A Cypherpunk's Manifesto*. 1993. URL: https://www.activism.net/cypherpunk/manifesto.html.

[11] A. Back. *Hashcash - A Denial of Service Counter-Measure*. Tech. rep. 2002. URL: http://www.hashcash.org/papers/hashcash.pdf.

[12] J. A. Lanz. *Hacker Responsible for 51Returns Part of the Stolen Funds*. 2019. URL: https://ethereumworldnews.com/hacker-51-percent-attack-ethereum-classic-returns-funds/.

[13] A. Kadiyala. *Nuances Between Permissionless and Permissioned Blockchains*. 2018. URL: https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483.

[14] P. Jayachandran. *The difference between public and private blockchain*. 2017. URL: https://www.ibm.com/blogs/blockchain/2017/05/the-difference-between-public-and-private-blockchain/.

[15] M. Jakobsson and A. Juels. "Proofs of Work and Bread Pudding Protocols." In: *Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99), September 20-21, 1999, Leuven, Belgium*. 1999, pp. 258–272.

[16] S. N. Sunny King. "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake." In: (2012). URL: https://peercoin.net/whitepapers/peercoin-paper.pdf.

[17] J. Connell. *On Byzantine Fault Tolerance in Blockchain Systems*. 2017. URL: https://cryptoinsider.com/byzantine-fault-tolerance-blockchain-systems/.

[18] *An Introduction to Hyperledger*. Tech. rep. The Linux Foundation, 2018. URL: https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf.

[19] A. Tayo. *Proof of work, or proof of waste?* 2017. URL: https://hackernoon.com/proof-of-work-or-proof-of-waste-9c1710b7f025.

[20] V. Buterin. *On Stake*. 2014. URL: https://blog.ethereum.org/2014/07/05/stake/.

[21] D. O. Michael Peirce. *Scaleable, Secure Cash Payment for WWW Resources with the PayMe Protocol Set*. 1997. URL: https://www.w3.org/Conferences/WWW4/Papers/228/.

[22] E. Rykwalder. *The Math Behind Bitcoin*. 2014. URL: https://www.coindesk.com/math-behind-bitcoin.

[23] *Smart Contracts: The Blockchain Technology That Will Replace Lawyers*. URL: https://blockgeeks.com/guides/smart-contracts/.

[24] B. Allen. *Turing-completeness: How Ethereum Does what it Does*. 2017. URL: https://thebitcoinmag.com/turing-completeness-ethereum/1712/.

[25] *Litecoin Homepage*. URL: https://litecoin.org/.

[26] D. D. Evan Duffield. *Dash: A Privacy-Centric Crypto-Currency*. Tech. rep. URL: https://whitepaperdatabase.com/wp-content/uploads/2017/09/Dash-Whitepaper.pdf.

[27] *Product Overview: A technical overview of xCurrent*. Tech. rep. Ripple, 2017. URL: https://ripple.com/files/ripple_product_overview.pdf.

[28] *NEM: Technical Reference*. Tech. rep. NEM Foundation, 2018. URL: https://www.nem.io/wp-content/themes/nem/files/NEM_techRef.pdf.

[29] S. Popov. *The Tangle*. Tech. rep. 2017. URL: http://untangled.world/wp-content/uploads/2017/09/iota1_3.pdf.

[30] S. N. Shen Noether. *Monero is Not That Mysterious*. Tech. rep. 2014.

[31] G. Zyskind, O. Nathan, and A. Pentland. "Enigma: Decentralized Computation Platform with Guaranteed Privacy." In: *CoRR* abs/1506.03471 (2015). arXiv: 1506.03471. URL: http://arxiv.org/abs/1506.03471.

[32] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. "Secure Multiparty Computations on Bitcoin." In: *Commun. ACM* 59.4 (Mar. 2016), pp. 76–84. ISSN: 0001-0782. DOI: 10.1145/2896386. URL: http://doi.acm.org/10.1145/2896386.

[33] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukoli, S. W. Cocco, and J. Yellick. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains." In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. Porto, Portugal: ACM, 2018, 30:1–30:15. ISBN: 978-1-4503-5584-1. DOI: 10.1145/3190508.3190538. URL: http://doi.acm.org/10.1145/3190508.3190538.

[34] Bitcoin.org. *Developer Guide - Bitcoin*. URL: https://bitcoin.org/en/developer-guide.

[35] *Bitcoin Wiki*. URL: https://en.bitcoin.it/wiki/.

[36] C. Decker and R. Wattenhofer. "Information propagation in the Bitcoin network." In: *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*. 2013, pp. 1–10. DOI: 10.1109/P2P.2013.6688704. URL: https://doi.org/10.1109/P2P.2013.6688704.

[37] M. M. João Marçal Luís Rodrigues. "Adaptive Information Dissemination in the Bitcoin Network." In: *34th ACM Symposium on Applied Computing (SAC 2019), Limassol, Cyprus, April 8-12, 2019*. 2019.

[38] *Ethereum devp2p Wiki*. 2015. URL: https://github.com/ethereum/devp2p/wiki.

[39] P. Maymounkov and D. Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." In: *Peer-to-Peer Systems* (2002), pp. 53–65.

[40] *Ethereum Wiki*. 2019. URL: https://github.com/ethereum/wiki/wiki.

[41] S. Pallickara, H. Bulut, and G. C. Fox. "Fault-Tolerant Reliable Delivery of Messages in Distributed Publish/Subscribe Systems." In: *Fourth International Conference on Autonomic Computing (ICAC'07), Jacksonville, Florida, USA, June 11-15, 2007*. 2007, p. 19. DOI: 10.1109/ICAC.2007.18. URL: https://doi.org/10.1109/ICAC.2007.18.

[42] R. van Renesse, D. Dumitriu, V. Gough, and C. Thomas. "Efficient Reconciliation and Flow Control for Anti-entropy Protocols." In: *Proceedings of the 2Nd Workshop on Large-Scale Distributed Systems and Middleware*. LADIS '08. Yorktown Heights, New York, USA: ACM, 2008, 6:1–6:7. ISBN: 978-1-60558-296-2. DOI: 10.1145/1529974.1529983. URL: http://doi.acm.org/10.1145/1529974.1529983.

[43] P. Th, E. R. Guerraoui, S. B. Handurukande, A. m. Kermarrec, and P. Kouznetsov. "Lightweight probabilistic broadcast." In: *ACM Trans. Comput. Syst* 21 (2003).

[44] J. Leitão, J. Pereira, and L. E. T. Rodrigues. "Epidemic Broadcast Trees." In: *26th IEEE Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, China, October 10-12, 2007*. 2007, pp. 301–310. DOI: 10.1109/SRDS.2007.27. URL: https://doi.org/10.1109/SRDS.2007.27.

[45] D. DD and D. O'Mahony. "Overlay networks: A scalable alternative for P2P." In: *Internet Computing, IEEE* 7 (Aug. 2003), pp. 79 –82. DOI: 10.1109/MIC.2003.1215663.

[46] M. Castro, M. Costa, and A. Rowstron. *Peer-to-peer overlays: structured, unstructured, or both*. Tech. rep. Microsoft, 2004. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2004/07/structella-tr.pdf.

[47] R. Baldoni, S. Bonomi, A. Rippa, L. Querzoni, S. Tucci Piergiovanni, and A. Virgillito. "Evaluation of Unstructured Overlay Maintenance Protocols under Churn." In: *26th International Conference on Distributed Computing Systems Workshops (ICDCS 2006 Workshops), 4-7 July 2006, Lisboa, Portugal*. 2006, p. 13. DOI: 10.1109/ICDCSW.2006.49. URL: https://doi.org/10.1109/ICDCSW.2006.49.

[48] M. Ripeanu. "Peer-to-Peer Architecture Case Study: Gnutella Network." In: *1st International Conference on Peer-to-Peer Computing (P2P 2001), 27-29 August 2001, Linköping, Sweden*. 2001, pp. 99–100. DOI: 10.1109/P2P.2001.990433. URL: https://doi.org/10.1109/P2P.2001.990433.

[49] *Napster Homepage*. URL: https://us.napster.com/.

[50] *BitTorrent Homepage*. URL: http://www.bittorrent.org/.

[51] K. Dooley. *Designing Large-Scale LANs*. O'Reilly, 2002. URL: https://userpages.umbc.edu/~dgorin1/451/lan_design/LANs_Design_Types.htm.

[52] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications." In: *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '01. San Diego, California, USA: ACM, 2001, pp. 149–160. ISBN: 1-58113-411-8. DOI: 10.1145/383059.383071. URL: http://doi.acm.org/10.1145/383059.383071.

[53] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems." In: *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*. Heidelberg, Germany, Nov. 2001, pp. 329–350.

[54] D. Wang, H. He, and J. Shi. "The Measurement and Analysis of KAD Network." In: *Trustworthy Computing and Services - International Conference, ISCTCS 2012, Beijing, China, May 28 - June 2, 2012, Revised Selected Papers*. 2012, pp. 117–123. DOI: 10.1007/978-3-642-35795-4_15. URL: https://doi.org/10.1007/978-3-642-35795-4_15.

[55] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web." In: *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*. STOC '97. El Paso, Texas, USA: ACM, 1997, pp. 654–663. ISBN: 0-89791-888-6. DOI: 10.1145/258533.258660. URL: http://doi.acm.org/10.1145/258533.258660.

[56] C. Gkantsidis, M. Mihail, and A. Saberi. "Random walks in peer-to-peer networks: Algorithms and evaluation." In: *Perform. Eval.* 63.3 (2006), pp. 241–263. DOI: 10.1016/j.peva.2005.01.002. URL: https://doi.org/10.1016/j.peva.2005.01.002.

[57] S. Voulgaris, D. Gavidia, and M. V. Steen. "CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays." In: *Journal of Network and Systems Management* 13 (2005), p. 2005.

[58] A. J. Ganesh, A. Kermarrec, and L. Massoulié. "SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication." In: *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings*. 2001, pp. 44–55. DOI: 10.1007/3-540-45546-9_4. URL: https://doi.org/10.1007/3-540-45546-9_4.

[59] J. Leitão, J. Pereira, and L. Rodrigues. "HyParView: A membership protocol for reliable gossip-based broadcast." In: *In IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE Computer Society, 2007, pp. 419–428.

[60] M. Jelasity and O. Babaoglu. "T-Man: Gossip-based overlay topology management." In: *In 3rd Int. Workshop on Engineering Self-Organising Applications (ESOA'05)*. Springer-Verlag, 2005, pp. 1–15.

[61] J. Leitão, J. P. da Silva Ferreira Moura Marques, J. Pereira, and L. E. T. Rodrigues. "X-BOT: A Protocol for Resilient Optimization of Unstructured Overlay Networks." In: *IEEE Trans. Parallel Distrib. Syst.* 23.11 (2012), pp. 2175–2188. DOI: 10.1109/TPDS.2012.29. URL: https://doi.org/10.1109/TPDS.2012.29.

[62] A. Montresor, M. Jelasity, and O. Babaoglu. "Chord on demand." In: *In Proceedings of the 5th International Conference on Peer-to-Peer Computing (P2P 2005).* IEEE, 2005, pp. 87–94.

[63] J. L. João Carvalho Nuno Preguiça. "Ouroboros: Uma DHT Auto-organizável Tolerante a Churn." In: *Sétimo Simpósio de Informática* (2015).