Master Thesis

Scalability & Trustlines Network Architecture

Côme du Crest
TU Darmstadt

Advisor: Prof. Dr. Sebastian Faust
Date of Submission: 08.10.2018

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Côme du Crest, die vorliegende Master-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden. Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein. Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, October 2018 _____

Thesis Statement pursuant to § 22 paragraph 7 and § 23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Côme du Crest, have written the submitted thesis independently pursuant to § 22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once. In the submitted thesis the written copies and the electronic version for archiving are pursuant to § 23 paragraph 7 of APB identical in content. For a thesis of the Department of Architecture, the submitted electronic version corresponds to the presented model and the submitted architectural plans.

Darmstadt, October 2018 _____

Abstract

The Trustlines Network project intends to create a network of "I Owe You" payments, allowing to replace classical payments systems. However, it is confronted to two scalability problems that this thesis exhibits and seek a solution for. Scalability is deemed problematic when the system is capable of handling a low number of users, but will not work when this number grows. First of all, to each transaction involving two users, a path in the network has to be found connecting the two users; this is called pathfinding and can require a large amount of computing power. An empirical study of the pathfinding algorithm used in the Trustlines Network is conducted and shows how the algorithm can only handle around ten transactions per second on a personal computer. This does not reach the criteria for scalability of the Trustlines Network.

However, it is deemed to be lesser of an issue than the second scalability problem: the problem of the underlying blockchain. In the bigger part of this thesis, different solutions for the scalability of blockchains as well as alternative architectures for the Trustlines Network are presented. At the end of this part, the recommendation is given to the Trustlines Network project to deploy its own provisional blockchain based on currently available solutions, in wait for a more developed project to offer a convincing solution to the scalability problem of blockchains.

Contents

1 Introduction

We live in a world where the economy and industry is more and more driven by technological advancement. Moreover, citizens are becoming increasingly aware of security and privacy in the digital environment. This helps us understand how blockchains became so important today. The domain of application of blockchains is primarily but not limited to finance and transfer of currencies, allowing for pseudonymous and secure transfers with fees independent of the amount of the transaction. However, it is difficult for non technical users to fully understand blockchains and how to interact with them. Additionally, a recurring problem for blockchain related projects is scalability. The biggest historical blockchain: the Bitcoin blockchain, can only process an estimated 7 transactions per second and consumes as much electricity as Switzerland to do so[1].

In this aspect, the Trustlines Network project[2] solves the first problem and is confronted to the latter. Trustlines Network intends to build a level of abstraction for non technical users by providing an intuitive mobile application to make "I Owe You" (IOU) payments, written on the blockchain. The starting idea for Trustlines Network is similar to the original Ripple idea[3], but build on Ethereum[4].

The goal of my thesis is to expose and propose solutions for solving the scalability problems Trustlines Network is confronted to, by discussing and analyzing different approaches to scale use cases for blockchains or blockchains in general. The focus will not be in covering as many approaches as possible but to go deeper in the most notable and promising in the opinion of the blockchain community and my opinion. In a first part I will present the related work for study of pathfinding algorithm and categorization of scalability solutions. In the second part I am going to explain in detail the Trustlines Network's goals, functioning, and architecture. In the third part I will explain how the pathfinding algorithm used by Trustlines Network poses scalability issues. Lastly, in the fourth part I am going to discuss different approaches for scaling consensus mechanisms and how Trustlines Network could apply them.

2 Related Work

In "'Algorithms and Data Structures"[5], the theory of graph traversal, and in particular algorithms for shortest paths are detailed and analyzed theoretically. Simulations of different pathfinding algorithms are presented in [6], the conclusion being that among the tested algorithms Dijkstra's algorithms is the most efficient one for graphs with non negative weights. However the simulation is performed on grid graphs and random graphs with low-weights Hamiltonian cycles, not on graphs mirroring any precise real life scenario. The author also points out that care is needed when applying these algorithms in practice, as changes in graphs can drastically change the performance of the algorithms. "Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport"[7] provides an experimentation of Dijkstra's algorithm conducted on a data set of path queries from the public railroad transport. It shows how, in this case, Dijkstra's algorithm is inefficient but certain optimisations can be adopted to make it run in an acceptable amount of time. However this study[7] was conducted in 1999 and performance of computers have been largely improved since then. From this, I can conclude that the empirical analysis of the pathfinding used for the Trustlines Network is scientifically relevant and necessary to answer the later detailed question of suitability of the algorithm.

Regarding blockchains and scalability, "Blockchain Challenges and Opportunities: A Survey"[8] explains typical consensus algorithms and expose the challenges behind designing a scalable blockchain. In [9], Mingxiao et al. gives an introduction to the different characteristics and principles of consensus algorithms, focusing on proof of work and its security. Scalability limitations of proof of work and byzantine fault tolerance based blockchains and potential solutions are also exposed in [10]. I believe this thesis deepens these works by providing detailed explanations behind the motivation and technical aspects of different scalability solutions. Additionally, in "SoK: Consensus in the Age of Blockchains"[11], the authors categorise first layer solutions to scale blockchains in general by explaining proof of stake, proof of authority, and others, but do not analyze second layer solutions like I do in this thesis.

3 Trustlines Network

3.a Goal of the Project

In this part, I am going to explain the context of Trustlines Network and the technical aspects of the project, before we can see the problems it is confronted to. The main idea of Trustlines Network is to create a mobile payment application based on trust. There are two main different use cases for Trustlines Network that the project intends to test out and validate. The first one is Trustlines Network used by companies, to facilitate payments in business to business operations. The second one being private users in a community using it for daily payments, for example unbanked people of developing countries. As such, Trustlines Network tries to get a broad adoption by creating a user-friendly application, and with abstracting complicated blockchain interactions to users. These two use cases will have different implications for the scalability criteria that we are going to see later. However, as the general explanation of how the Trustlines Network works and the current architecture of the solution is independent of the case of application, and the second use cases might be easier to grasp, I will principally keep my focus on this private user application.

The Trustlines Network intends to leverage the fact that people are willing to lend money to their friends or relatives in their everyday lives to make payments. This can be represented by *credit lines* between two parties: Alice is willing to lend €5 to Bob. If Bob wants to actually use these €5, we have to update the *balance* of this credit line: Bob owes €5 to Alice. A credit line is thus made of two entities, a credit limit, and a balance. To enable the transfer of money in the two ways, we use *trustlines*, that is two credit lines between the same entities but with different directions. In the rest of the thesis, I will refer to the Trustlines Network project as Trustlines Network or simply Trustlines, with a capital "T", and refer to the bidirectional credit lines as trustlines with a lowercase "t".

Furthermore, imagine Bob wants to give €5 to Charles, but does not trust Charles. If Alice has a credit line with Charles, we can represent the €5 transfer from Bob to Charles by: Alice owes €5 to Charles and Bob owes €5 to Alice (see figure 1). This leads us to a network of channels, linking entities that may not know each other, through entities that they trust, and allowing payments in the form of *I Owe You* (IOU) between them.

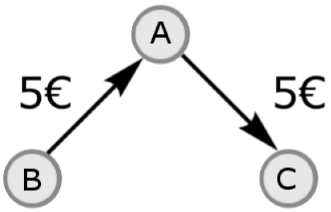

Figure 1: Diagram of a payment from B to C through A

3.b Decentralized Exchange

Additionally, as trustlines networks represent payments of a certain currency, we need a way of making payments between a trustline network in a currency to another network using another currency. This can be done with *gateways*: users accepting trustlines money in exchange for another currency, such as ether (ETH) or accepting ETH for trustlines money. For example, if Alice wants to send €1 to Bob but Bob only accepts Dollars. Alice has to be routed to an ETH gateway in her euro network that sells ETH, she then has to find another ETH gateway in the dollar network to buy the ETH and, lastly, find a path from the gateway to Bob.

The system thus has to maintain a public order book where each user can create offers for buying or selling certain currencies in exchange for ETH or potentially other currencies. This order book will then be used to find the gateways to be used in atomic cross-currency transactions.

3.c Current Architecture

The currently projected architecture to make this system work comprises: a mobile application, relay servers, and a smart contract on the Ethereum blockchain. The mobile application allows users to open trustlines with other users, and to make payments. Relay servers interpret the payment request from users, find a path in the trust network, linking the payer and the payee, and forward the transaction to the on-chain smart contract. Lastly, the smart contract on the blockchain is used to securely update each trustline along the path of the transaction and to store the state of the network.

The main reason for having relay servers is to compute the pathfinding algorithm in the place of the user and its smartphone for efficiency reasons. Indeed, this algorithm can be computationally heavy and requires storing the state of the whole network, necessitating frequent updates and communication with the blockchain.

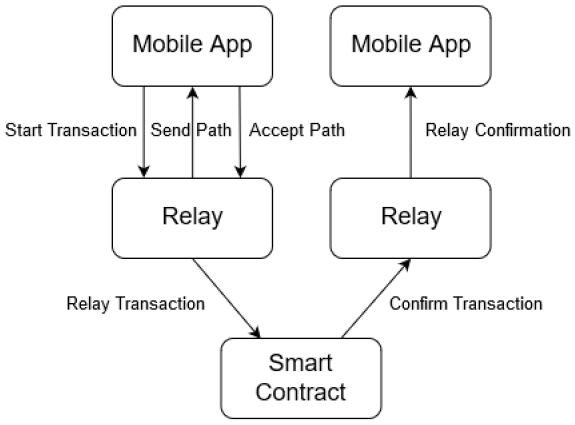

Figure 2: Diagram of the architecture of Trustlines Network

3.d Pathfinding in the Network

I am now going to explain in more details the stakes of pathfinding. There are three principal fees due to transactions currently considered on the trustlines network. The first one is due to *Ethereum transaction fees*; the interaction with the blockchain inevitably costs gas. The second one is an *imbalance fee*. It incentivises to bring balance to the network. It adds costs to transactions that exhaust credit lines proportional to the imbalance they add. The last fee is a *capacity fee* rewarding users that provide transfer capacity to the network by maintaining trustlines.

For the same transfer of money between two users, these fees can vary depending on the path chosen to link the two entities in the network. Usually, the higher the number of *hops* between the payer and the payee (i.e. the number of intermediaries), the higher the fee is. Still, some path are better than other. Relay servers find the cheapest path between two entities, calculating the sum of the three fees.

Moreover, if two similar transactions are submitted at the same time to the blockchain, based on the ordering of the transactions, one transaction could increase the fees of the other by changing the different balances of intermediary trustlines. This will change the imbalance fee the second transaction's issuer has to pay. Worse, it could happen that some trustlines become exhausted and the second transaction fails. It could be hard to explain to non technologist users why this happens and would be a barrier to the wide adoption of the Trustlines Network.

3.e Scalability Issues

A deep explanation of the scalability requirements for the Trustlines Network can be found in part 5.a. However, for the current part and especially for the pathfinding algorithm, it is sufficient to say that the Trustlines Network should be able to handle a thousand transactions per second.

Currently, the method used to find the best path in the network is Dijkstra's algorithm. It is suitable for the current scale of the network of a few users but may be too slow to guarantee a satisfying payment speed for a larger scale network. Based on my knowledge of graph theory and pathfinding algorithms, I make the hypothesis that the employed algorithm will prove inefficient to handle a thousand transactions per second.

Additionally, as already stated, every transactions currently result in an interaction with the Ethereum blockchain. The blockchain only being able to proceed transactions in the order of ten per second, it remains a bottleneck for the scalability of the whole system.

In the next part I am going to explain the work I did to assess the scalability of the pathfinding algorithm. I am then going to explain how I came to change my focus from pathfinding to the other scalability issue, the one of the underlying consensus mechanism.

4 Pathfinding

4.a Design Choices

Throughout this part, I will mathematically refer to the trustlines network as a graph. It is intuitive to represent users as vertices of a graph and credit lines as edges of a directed graph. Currently the pathfinding algorithm used is Dijkstra's Algorithm, considering the shortest path in number of hops. This choice is motivated by the fact that currently the highest contribution to the fees is by far the cost of using the Ethereum blockchain. The cost of using the blockchain is proportional to the number of trustlines to be edited, thus the shortest path in term of hops will be the cheapest path. If two paths have the same number of hops, then the other fees are considered to compare them. However, in the future, or with a different architecture, the different fees could be comparable and will have to be considered differently in the algorithm. Moreover, currently only one path is found, due to the same intent of impacting the least number of trustlines possible, but it could theoretically be cheaper to split the transfer among different paths.

4.b Explanation of the Algorithm

The algorithm can be found below in listing 1. The algorithm instantiates the distance of every node to infinity. Then, starting from the source, will *visit* each node with the minimal registered distance from source (line 13). Doing so, it will change the distance of all the neighbours v of the currently visited node u, if it is shorter to go through u to visit v (line 21-25). When the visited node is the target, the search is over. The path from source to target can then be found by recursively taking $prev[u]$ from *target* to *source*.

```
1  function Dijkstra(Graph, source, target):
2
3    create vertex set Q
4
5    for each vertex v in Graph:          // Initialization
6      dist[v] := INFINITY                // Unknown distance from source to v
7      prev[v] := UNDEFINED               // Previous node in optimal path from source
8      add v to Q                         // All nodes initially in Q (unvisited nodes)
9
10   dist[source] = 0                     // Distance from source to source
11
12   while Q is not empty:
13     u := vertex in Q with min dist[u]
14                                        // Node with the least distance
15                                        // will be selected first
16
```

```
17      if u == target:                 // we reached the target
18        return dist[], prev[]
19
20      remove u from Q
21
22      for each neighbor v of u        // where v is still in Q
23        alt := dist[u] + length(u, v)
24        if alt < dist[v]:             // A shorter path to v has been found
25          dist[v] := alt
26          prev[v] := u
27
28    return dist[], prev[]
```

Listing 1: Dijkstra's Algorithm

Dijkstra's Algorithm worst case complexity is $O((E+V)*log(V))$ when implemented with a binary heap priority queue[5], like in the current implementation, where E is the number of edges in the graph and V is the number of vertices. However a Fibonacci heap would bring the worst case complexity down to to $O(E+V*log(V))$[5]. For our use case, we would say the algorithm scales sufficiently if it is capable of finding 1.000 paths per second, on a graph with 100.000 nodes with 10 relay servers. The actual average running time of the algorithm depends on the number of nodes the algorithm has to visit. This depends on the average distance from source to target for each path, and the average number of neighbours each node have. The choice of the graph to use and its topology is thus more important than the number of edges and vertices for the accurate simulation of the running time of the algorithm[6][7].

4.c Graph to Use

As a first intuition, the idea behind Trustlines Network being at first similar with that of Ripple[3], the topology of the Trustlines Network could be approximated by that of Ripple. However, the Ripple network is made of a large amount of nodes connected solely to highly connected nodes[12]. This is due to the way users join the Ripple network: users typically pay some fiat or cryptocurrencies to a gateway to open a credit link with them. The resulting graph is *decentralized with centralized hubs* (see figure 3).

On the contrary, I assume that different users would join the Trustlines Network by setting up trustlines with totally different users resulting in a more distributed topology. My hypothesis for the Trustlines Network is thus that it should resemble a social graph, where each users are connected to their friends and family, the available credit in the trustlines being higher for family members than for acquaintances.

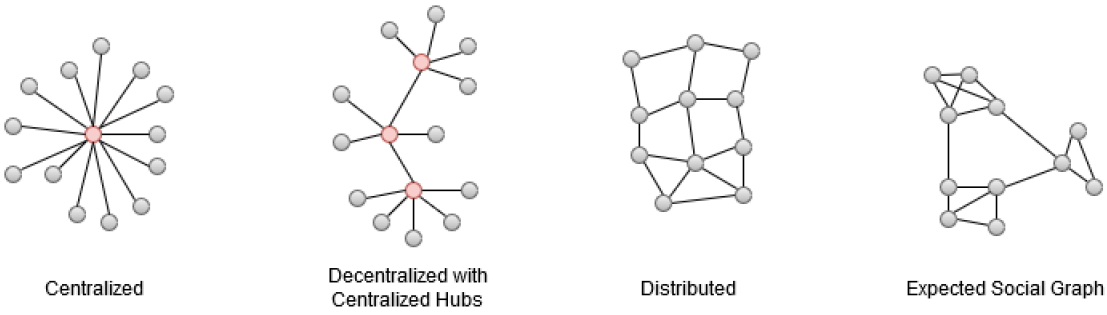

Figure 3: Example of different type of graphs with the corresponding names

As a second thought, I looked into using data from Facebook or Twitter. But this may not prove to be more adequate. Indeed, it is less committing to add someone as a Friend on Facebook than to trust someone with money or anything of value, as it is the case in the Trustlines Network.

The conclusion is that a graph has to be generated for this simulation, and it should not be based on the currently available real world data. As already explained, the important parameters for the computation time of the pathfinding are the average path length and the average number of neighbours of each vertex. For Facebook, the average path length is 3.57[13]. It is expected to be higher for Trustlines Network and an educated guess for an appropriate number would be in between 6 and 8, following the *six degrees of separation* principle[14]. As for the average number of neighbours of each vertex, I estimate generic users will be willing to open trustlines with 1 to 15 other users with an average around 7.

Once these parameters are set, I fixed the number of nodes to 10.000, for a reason of precision on the other parameters and to be able to run the simulation in a reasonable time with my computer.

To generate a graph with the chosen parameters, I started with the *Watts–Strogatz model*, allowing for the random generation of a graph with *small-world properties* with low average path length and high *clustering*[15]. The small-world property signifies that most node will not be directly linked to most other nodes, but will have a low number of hops to reach all the other nodes in the network. Clustering represents the tendency of social graphs to have groups of tightly connected vertices with a low probability to be connected to other clusters. Lastly, for the purpose of the simulation, I verify the generated graph is connected. Running tests on a non connected graph will lead to a lot of impossible path chosen at random. This will result in more computations than for a scenario of usage by regular users, likely asking for possible paths.

I then modified the graph generated via this model to comply with the previously explained requirements for the graph. I added randomness in the generation of the graph by removing or replacing certain edges to represent uncertainty on the average number path length and average number of neighbours. By running simulations with different graphs having varying parameters, I evaluate and account for the lack of precision on the parameters.

4.d Simulation and Results

The graph generation and pathfinding simulation has been written and run with Python 3.6.4, as is the pathfinding in the Trustlines Network. I have run the simulation using a laptop with an Intel Core i7-6700HQ CPU at 2.60GHz (8 CPUs), with 16GB of DDR4 RAM running Windows 10 Education 64-bit. During each simulation round I look for 10,000 paths randomly selected. The time for the selection of the paths to find as well as the generation of the graph is not accounted for in the running time. I then calculate the average number of paths found per second.

Average Path Length	7.074	6.803	7.326	7.851	6.251	8.151
Average Number of Neighbours	7.589	7.605	6.392	5.598	8.781	5.209
Path Per Second	5.579	7.152	9.267	9.922	10.311	12.849

Table 1: Result of the simulation for Dijkstra's Algorithm

As can be seen in the table 1, the number of paths found per second remains in the same order of magnitude as 10. This means that to be able to find 1,000 paths per second as required, an expected number of 100 computers with the same computing power as the one used for the simulation will have to be used. It is reasonable to say that this scalability ratio is not suitable and that employing another algorithm, seems like a better approach than getting the algorithm to run on 100 servers.

4.e Pathfinding Conclusion

With the current architecture of the Trustlines Network, both the pathfinding algorithm, and the blockchain the project relies on, are bottleneck to the number of transactions per second one can envision, and will have to evolve.

There are known algorithm that can perform better than Dijkstra's algorithm. However, they have drawbacks: A* is based on a heuristic to be

determined accordingly with the use case. Other algorithms can do precomputation on the graph prior to the search of the path to be more efficient[7], but will only be efficient if the graph does not vary rapidly.

There is also two families of pathfinding algorithm: centralized and decentralized pathfinding[16][17]. In the former, one entity is aware of the whole state of the graph and can do the computations on its own. As for the latter, entities are only aware of their neighbours, and no global state is available at any time. To find a path, they will recursively query their neighbours, for example, to collectively compute a result.

The type of algorithm and the different variations to use depends greatly on the whole architecture of the system. Thus, finding a scalable architecture should have a higher priority than finding alternatives to scale the pathfinding algorithm. Additionally, at the launch of the Trustlines Network, it could be sufficient to have a capacity of only 10 transactions per second per relay server, whereas having transaction fees of around $0.30, as it is the case on the Ethereum blockchain as of July 2018[18], might prevent widespread adoption.

This is the reason why I came to change my plan of developing and testing a better scaling pathfinding algorithm for trustlines to study different approaches to scale its architecture. The next part of my work constitutes an analysis of different solutions to scale the architecture as a whole, mainly but not solely by scaling the underlying blockchain.

5 Scaling the Architecture

I am going to introduce this part by explaining what I take as a definition for scalability in the blockchain domain. Secondly, I will explain the goal and evaluation criteria for scalability solutions. I will then explain common paradigms as building blocks for scalability solutions before I will detail and analyze different concrete projects for scaling blockchains, relating them when possible to the Trustlines Network architecture.

The current literature does not explicitly agree on what scaling means in the context of blockchain. I will define solving scalability as solving three intertwined problems. The first and most obvious one is *transaction throughput*; the blockchain has to be able to process a high number of transactions per seconds, or one that is growing with the number of users to be deemed scalable. Transaction throughput is the main focus of studies as it is the smallest bottleneck for scalability currently. The second problem is *network bandwidth*, as blockchains work in a peer-to-peer (P2P) network. Traditionally, each node in the network receives and relays every transactions to be written on the blockchain, which can require a high bandwidth as the transaction throughput grows. The last problem is related to how every user needs to process every transaction and store all the data of the blockchain. As such, the processing capability of the blockchain is the one of the least capable node of the network. I will refer to this as the *global processing* problem. Different solutions to scalability intend to tackle one or more of the explained problem.

Additionally, scalability solutions can be categorized in different layers; layer 1 solutions are modifications to the core protocol of an existing blockchain, or even a completely different blockchain (PoS[19], Ouroboros[20], block size extensions, etc ...). Layer 2 solutions are building on top of existing layer 1 solutions or blockchains to add features and can go hand in hand with layer 1 solutions (Plasma[21], Sharding, State Channels, etc ...). Lastly, I will informally define layer 0 solutions for this work as solutions for the architecture of Trustlines Network that do not rely on blockchains.

5.a Evaluation Criteria

The idea behind the current Trustlines Network architecture is to use relay servers to strictly transfer the transactions it receives to the smart contract on the Ethereum blockchain. The smart contract is responsible for verifying transactions and updating every credit lines along the path of each transaction. The scalability of the Trustlines Network is thus inherently lesser than that of the underlying blockchain. The Ethereum blockchain, at the times

of writing, is able to handle 7 transactions a second, whereas the Trustlines Network optimistically projects a thousand or more transactions per second. Additionally, the transactions fees vary between $0.30 and $1 as of July 2018[18], while to achieve widespread adoptions, the fees for a transfer via trustlines should be around $0.05, depending on the use case.

To allow the Trustlines Network to scale, different choices are available: the network can use independently designed scalability solutions of blockchains when they become available, deploy their own smaller-scale adaptation of these solutions, or design alternative architecture from scratch or using the building blocks explained below. This comes to justify the fusion between scaling blockchains and the Trustlines Network.

To evaluate the potential of the different scalability solutions for any application, we have to look at how centralized it is, how secure it promises to be, how many transactions per seconds it can handle, the fees for each transactions, and the speed at which finality is reached. Finality is defined for a transaction when it is not reversible anymore; the history of the blockchain cannot be rewritten so as to revert this transaction.

To evaluate the potential of the solutions for the Trustlines Network project in particular, we have to analyse its objectives. As already mentioned, there are two main use cases for Trustlines that the project intends to test out and validate. The first one is Trustlines used by companies, to facilitate payments in business to business operations, in which case the value of transfers might be such that high fees would pose no problem. The second use case is that where Trustlines is used by private users in a community for everyday life operations, where low fees are important as well as low transaction delay. Another important aspect for this use case is how easy it is for users to join the network, and if we can make all the technical details oblivious to the user. For the sake of simplicity, I will focus on the basic functionality of Trustlines where users make transfers between them, and not take too much consideration on the decentralized exchange part which should result from it.

The project needs to be released as soon as a satisfactory version is developed. We can assume that it will be openly available before the end of 2018. Further, we can imagine that for the first months, the number of users will not be so that the scalability of the network is an issue in terms of transaction throughput, but the high fees remain a problem nonetheless and to ensure smooth user experience, finality has to be reached as fast as possible.

5.b Building Blocks

In this part I am explaining known building blocks that help solving scalability issues in blockchains. These building blocks do not necessarily solve scalability problems on their own but are common to different scaling projects or solutions, which is why I explain them in an independent part of my thesis. It is also important to note that some of the building blocks are not novel but are inspired by established research in distributed systems (sharding[22], election mechanisms[23], etc ...).

5.b.1 Proof of Work, Stake, and Authority

In the current state of Ethereum and Bitcoin for example, the proposer of each block is determined via proof of work (PoW)[24]. PoW implies that every miners attempt to solve a different puzzle for each block; the one solving it is deemed to be the owner (or proposer) for that block and can claim the mining reward. The best strategy to solve the puzzle is to compute random hash values until they are below a certain threshold. The security of PoW relies on the fact that it is computationally hard to solve the puzzle. An attacker that wants to perform a 51% attack, that is, to be able to own proposed blocks 51% of the time in order to exclude other miners' blocks, needs to possess 51% of the hash power of the whole system[25]. Additionally, miners are incentivised to build their blocks on the longest chain to be more likely that other miners build on top of them so that their blocks end up in the final chain. In this way, they are more likely to collect the block reward.

PoW incurs a waste of resources: electricity consumption and hardware components used to produce useless hashes. Proof of stake (PoS)[26] intends to bypass this waste by letting the protocol itself take the role of choosing validators for each blocks. In PoS and later proof of authority (PoA), the term *miner* is replaced with the one of *validator* as to represent that the act of creating a block is not labour intensive. When it is unclear for layer 2 solutions whether they are implemented on top of a PoW, PoS, or PoA system, I will continue to use the term *validator*. To participate in PoS, validators have to prove that they locked valuables, for example cryptocurrencies or tokens in a smart contract for a certain amount of time. Contrary to PoW, the resulting set of eligible validators is known for each block height. The probability for a user to be elected as a validator for a block should be proportional to the amount he locked. To perform a 51% attack on the system, a user then has to hold 51% of all the stakes. The security can also be justified in that a user committed so much to a blockchain probably has no gain from attacking it.

Contrary to PoW, as minting blocks does not require to waste resources, validators can build blocks on top of two different forks. Game-theoretic modeled validators, only interested in getting as much profit as they can from validating, would then build on top of every fork to be more likely to be included in the final chain. This will make it hard to solve forks and decide on a main chain. Additionally, if each fork has an equal pool of validator holding 99% of the stake, a user with 1% of the stakes can switch from one chain to the other and decide on which chain should be final, thus enabling double spending attacks. This problem is known as the *nothing at stake* problem[11].

To mitigate this problem, *slashing* can be introduced[27]: if validators misbehave by building blocks for the same height on two different forks, they can be *slashed* and lose part or all of their stake. However, slashing does not help for long-range attacks. As validators are allowed to withdraw the funds they locked after a certain time (typically 6-12 months), once their funds are withdrawn, they are free to misbehave again. As such, validators can start building blocks on top of past forks where they are still seen as validators, and reconstruct a longer chain.

Weak subjectivity is a theoretical concept introduced in 2014 in [28] that comes to lessen the long-range attack problem. Most consensus algorithm, such as the one of Bitcoin or Ethereum are objective: an outsider receiving a set of all blocks and aware of the rules of the protocol can reconstruct the main chain and the state of the network. Other consensus algorithm, like Ripple, are subjective: different nodes can come to different conclusions based on information outside of the protocol (reputation or the like). Weak subjectivity is the use of subjectivity for long-range decisions and objectivity for short term decisions: an outsider receiving a set of all blocks, aware of the rules of the protocol and of a state from less than X blocks ago thanks to trustful sources or reputation, can reach the same conclusion about the current main chain and state of the network.

Weak subjectivity solves the problem of long-range attacks as users regularly monitoring the blockchain should reject a suddenly appearing new fork with X or more blocks. However, it adds the assumption that new users willing to join the network have a means of accessing a trustworthy state younger than X blocks. This X value can be determined as the time length users have to lock their states in order to be eligible as a validator.

The concept of Proof of Authority (PoA) is similar to the one of PoS, where instead of staking funds validators stake their identities. For example, in the Kovan PoA testnet, validators are companies such as Etherscan or

Parity Technologies. Misbehaving validators are not slashed of any funds but their reputation is harmed. There is no way of withdrawing staked identity, so long-range attacks seen in PoS are not a problem for PoA.

Both PoS and PoA are working on the layer 1 as they are impacting the core protocol of blockchains. PoA can help with increasing the transaction throughput by reducing the number of potential validators in the system. However, PoS is most often used in combination with other building blocks to solve scalability in the long term.

5.b.2 Validator Choice: VRF

The problem remains for PoS and PoA of how validators are chosen by the protocol for each block height. The most intuitive algorithm is the *round robin consensus*[29]: eligible validators are numerated from 0 to $N-1$, the selected validator for block with height h is $i = h \mod N$.

Some protocols also take into account ranking among different validators for the same block height. If the validator with rank i does not propose a block, the block proposed by validator with rank $i+1$ will be considered for example. Round robin allows for easy ordering of validators by taking the validator with rank $i+1$ as the validator following i in the initial numeration of all validators.

Other methods use random but predictable means of electing validators, such as methods based on the hash of the block number. It is important that the randomness used for election is unmanipulable. For example if the randomness used is the hash of the whole block, a validator can generate slightly different blocks and calculate the hash until he finds a value giving him the most chance of being reelected. This method is called a *grinding attack*[30]. Verifiable random functions (VRF) have been introduced as a means to make unmanipulable randomness[31]. VRF are functions that can compute a pseudo-random number based on an input and a secret key. The correctness of the output can then be verified using a public key by anyone, without compromising the secret key.

Additionally, protocols where the validators and their ranks can be determined ahead of time suffer from three fundamental flaws.

Firstly, an evildoer can prepare a denial of service attack on the predetermined first ranked validator, in an attempt to prevent him to propose a block. The second rank validator would attempt this attack to get the reward in its place.

Secondly, due to randomness a validator even with a low proposing power has a chance to be elected for a certain number of blocks in a row. A validator predicting this will have the opportunity to coordinate a double

spend attack.

Lastly, in a similar way, the system becomes vulnerable to adaptive corruption attack. If it is predicted that a low number of validators will be elected for a large set of consecutive blocks, an adversary can attempt to corrupt this small set of validator to perform a double spending attack.

It is thus important to have methods where the elected validators remain unidentifiable until they produce a block or at least until it is their turn to propose a block[32]. We can give as an example Ouroboros[20] as a PoS blockchain protocol using a public verifiable secret sharing scheme, similar to VRFs, where the validators for each blocks can be known long time in advance. Ouroboros Praos[33] is an improvement of Ouroboros where only elected validators can determine that they have been elected, staying private until they publish a block.

5.b.3 Sharding

Sharding refers to the method of partitioning data in a database into *shards* and attributing them to different servers[22]. This will reduce the load on each server as queries about one shard will result in computations from only one server. Applied to blockchains, sharding can be seen both as a layer 1 or a layer 2 solution attempting to solve the global processing problem, depending on how it is applied. One way to shard a blockchain is to split users by group of addresses and put all transactions involving these users in a shard. Each shards are then monitored and handled by a subset of all the validators. Each group of validators is then ignoring transactions of other shards and transactions are no longer processed globally[34].

The main issue with sharding is cross-shard communications: how to handle transactions involving users from different shards. Validators in the shard of the receiver may have no idea whether the sender possesses the necessary funds for the transaction. Each transaction impacting more than one shard will add overload to the system. The way shards are defined has to be thought through for efficiency depending on the use case.

Another issue is how the size of each shard is decided upon. The smaller the shard is, the more computation validators can handle but the higher the overload of the cross-shard communications. Indeed, validators will have fewer data to monitor but it will be more likely that transactions impact more than one shard. Additionally, the higher the number of shards, the lower the security of each shards as the subset of validators responsible for a particular shard will be smaller and will be more likely to be corrupt.

5.b.4 Off-chain transactions

An other important second layer approach to solving the global processing problem is to put transactions off the blockchain while attempting to have a similar level of security as if they were done on chain. This can be done for example via state channels[35] or side-chains[21] and usually calls for staking on the main chain. The main idea is to lock some funds on the main chain to be allowed to trade them extensively off-chain, while later being able to unlock the funds you are now entitled to on-chain. We thus always have a *lock* or *entry* mechanism to move funds from on-chain to off-chain and an *unlock* or *exit* mechanism to move funds the other way around.

In the case of state channels for example, two users can *open* a channel together by *locking* a certain amount on-chain. These users can then exchange proofs for transactions off-chain effectively exchanging currencies up to the locked amount. A high number of transactions can be done off-chain between these two users and only the last valid transaction exchanged is of importance. Indeed, users can *close* the channel by transmitting the last transaction to the main chain and claiming their remaining funds or part of their counterpart funds.

The concepts of off-chain computations and sharding are theoretical and can be implemented in very different manners by different projects. In the next part we are going to take a look at specific projects to solve scalability issues of blockchains that I chose to detail for the way they take different building blocks and combine them together. Additionally, I chose projects because they propose novel approaches to the best of my knowledge, on variations of the building blocks or completely different from these building blocks.

5.c Plasma

For the analysis of different projects, I am first going to study Plasma, a work in progress initiated by Joseph Poon and Vitalik Buterin[21]. I will explain the *Minimal Viable Plasma*[36] (MVP), as a first step to understanding the paradigm. Then I will provide an analysis of this MVP, before explaining a more developed solution: Plasma Cash. Finally we will see how this principle could be applied to scale the Trustlines Network.

5.c.1 Minimal Viable Plasma

Plasma is a layer 2 solution relying on the building block of *Off-chain computations*. To shift transactions off-chain, Plasma uses a tree of child chains each reporting to their parents. The root of the tree being the Ethereum

blockchain. The security guarantee of the child chain is performed by deploying a *Plasma smart contract* on the parent chain, capable of enforcing state transitions of their child chains. Thus, only block headers are sent to the parent or root blockchain periodically, which allows to reduce the amount of writing on the parent chain. This grants a higher transaction throughput on the child chain than that of the parent chain.

The main security assumption for Plasma is that users should monitor their blockchain and interact with the parent chain whenever they detect malicious transactions or block being withheld. Users should be able to provide a proof for the misbehaving validator of the child chain to the Plasma contract of the parent chain. This interaction should allow for the system to revert to a previous state before the invalid block and punish the validator of this faulty block.

In the following I will describe a minimalist implementation of Plasma, based on the current details of [36], as a first step to understanding Plasma. In this MVP, the assumption is that in case of a Byzantine behaviour, users should exit the Plasma child chain instead of trying to recover it to a safe state. In the minimal Plasma chain, blocks consist simply of a timestamp and of the Merkle roots of a depth-16 Merkle tree with each leaves corresponding to a transaction. Transactions are inspired by the bitcoin transactions and are made of two inputs, two outputs, and a signature by the owner of the inputs. The inputs of one transaction are simply the outputs of a previously included transaction, we call them *unspent transaction output* (UTXO).

The Plasma contract has four functions: deposit, submitBlock, startExit, and challengeExit. *deposit* is called by users that want to get credit in a child chain, through this function, they lock funds in the Plasma contract as a guarantee. A new block in the Plasma chain is then created granting credit to the user equivalent to the locked funds. *submitBlock* is called by the validators of the child chain to submit a Merkle tree root for transactions of the new block, creating a new block in the Plasma chain. *startExit* is used by any member of the child chain providing a UTXO of the child chain to initiate an exit query and withdraw their funds from the smart contract. The UTXO proves that the user is entitled to a certain amount of Ether for example. *challengeExit* can be called by anybody that wants to challenge an exit, providing a proof that the UTXO of the exit was in fact spent by the user, and that this spending transaction was included in a block.

Honest users are incentivised to challenge malicious exits to protect their funds locked in the Plasma contract. Indeed, if fraudulent users steal the

funds from the Plasma contract, honest users will not be able to withdraw their legitimate funds when exiting. Challenger could be further incentivised by punishing the fraudulent exit attempt and giving the exiter's locked funds to the challenger.

The purpose of the validators is thus to receive transactions from users, validate them, and periodically compute a Merkle tree with leaves corresponding to transactions. The root of the Merkle tree is then sent to the on-chain Plasma contract to confirm the history of the Plasma chain. In the MVP, it is suggested that the validator be a single operator for simplifying the explanation, however the protocol would work in the same way with a PoS or PoA algorithm choosing block proposers among a set of operators.

When an exit query is created providing a UTXO, the age of the block containing the UTXO is stored, if the transaction dates from more than 7 days, the stored age is 7 days instead. Queries that have an age of more than 14 days old are finalized in order of age from the oldest to the youngest by the smart contract, giving their funds back to the issuer. This leaves other users at least 7 days to challenge the exit queries providing a proof that this exit is fraudulent. Moreover, since exits are ordered based on the time of their corresponding UTXO, if a validator creates an invalid block "spending" transactions he does not own, cheated users will be able to create exit queries for these stolen UTXOs and will see their exit queries processed before the malicious validator resulting in no economic harm.

5.c.2 Analysis of the MVP

The working draft presenting Plasma[21] was published on August 11, 2017. The explanation of the Minimal Viable Plasma[36] was released on January 3, 2018. The project is progressing, and like most scalability related project, to the best of my knowledge, there is no communication on an estimated date of deployment.

This solution, in its present form, is as decentralized as the Ethereum blockchain due to the fact that its security relies on it. In spite of a centralized validator for Plasma chains, the validator is constrained by the decentralized blockchain. Moreover, users are guaranteed not to lose anything even if every other users and validator of the Plasma chain collude together, as long as the root chain is secure.

Plasma was not designed to reach finality quickly, as it requires transactions to be written first on the child chain and then on the root blockchain to be finalized. The finality is thus at most as fast as the one of the Ethereum blockchain.

With regards to transaction throughput, Karl Floersch, researcher working on Plasma, announced during an impromptu talk at the Ethereum Community Conference of 2018, that theoretically the MVP can scale to more than 1000 transaction per second. However, the problem arises that with the current description, in case of a misbehaving Plasma chain validator, all users may try exiting at the same time. A spike of transactions will appear on the parent chain increasing the cost of gas to the point where they will have to pay ten to a hundred times the projected cost of transaction. Vitalik Buterin commented that "this is indeed the fundamental flaw in all channel systems, Raiden and Lightning Network included, and is the reason why the scalability of this system can't go too far above the scalability of the main chain. 2-3 orders of magnitude probably but not that much more."[36] I am unsure about the importance of the problem, as users have 7-14 days to exit the Plasma chain and can dilute their exit queries throughout these days when the fees are low enough for them.

Another scalability bottleneck is due to the idea that each users need to monitor the whole Plasma chain for misbehaviours, meaning Plasma does not truly solve the *global processing* problem and will put a limit to the amount of transaction this chain can handle in any case. To solve this Problem, a variant of the Minimal Viable Plasma was proposed: Plasma Cash.

5.c.3 Plasma Cash

In this variant, deposits on the Plasma contract of the parent chain would create a unique ERC721 token that represent the exact amount of deposited Ether. ERC721 tokens are simply non-fungible tokens in the way not all of them are similar. Each ERC721 token has a unique identifier (ID) and contrary to ERC20 tokens, their value cannot be split or merged.

Thanks to this unique identifier, instead of having a non meaningful position in the Merkle tree of the transactions, transfers spending these tokens would be included in the Merkle tree in the position corresponding to their ID. For example, if the ID of the token is 0, the transactions regarding this particular token have to be on the leftmost position of the Merkle tree. As a result, users only have to monitor the Plasma chain at the specific index of the tree corresponding to their tokens ID, verifying that no transactions try to spend their belongings.

We can see this is sufficient since only tokens with an ID corresponding to a deposited token can be withdrawn in an exit. As a result, attackers trying to withdraw a token they do not own will harm a specific victim instead of harming every Plasma users. Thus, contrary to the MVP, it is not necessary for users to watch out for attacks on tokens they do not own.

The exit mechanism in Plasma Cash is similar to the one of the MVP. To exit, users need to provide a proof of a UTXO in the correct position of a Merkle tree in a block. Others can challenge this exit by providing an older UTXO for that token, or a proof of a transaction spending this UTXO. The difference with the MVP is that if the operator goes rogue by creating transactions spending tokens he does not own and tries to exit using these transactions, he would need to make an exit for each and every one of the tokens. The users will always be able to exit before the rogue validator with their most recent transaction of their tokens. This makes the attack more expensive for the validator to attempt and more rewarding for the honest users.

We can highlight another difference with the MVP, in that, sending a token to someone require sending the whole history of the token to this user for him to verify its correctness. Otherwise, the receiver has no idea whether the token is valid or not since he did not monitor the whole Plasma chain. This would require a large exchange of data in the long term. A potential solution to this problem is producing checkpoints for tokens once a year by withdrawing them and redepositing them, allegedly deleting the history. Otherwise privacy preserving cryptography like ZK-SNARK can be used to prove the coin is valid while keeping the history private[37]. Additionally, privacy of transactions could be envisioned as users only need to know about the part of the Merkle tree corresponding to their tokens. The validator, providing the Merkle branches, could hide information about other tokens to the user.

One of the initial drawback of Plasma Cash is that coins have a fixed denomination. A solution to allow users to break their tokens into smaller pieces is to allow token IDs to have decimals. Theoretically, users could transfer split tokens on the Plasma Chain and later combine fragments of tokens into a large one before exiting. Otherwise, probabilistic payments could be employed to represent payments smaller than a token: instead of transferring half a token, users could transfer a whole token with a 50% chance.

To summarize the paradigm of Plasma, a side chain is built by a constrained validator, receiving off-chain transactions from different users. Users join the Plasma chain by committing Ether or creating tokens on the main

chain. These users verify the behavior of the validator for part or all of the transactions included in the blocks. The security is guaranteed by the fact that they can exit the side chain whenever malicious behavior are suspected.

5.c.4 Application to the Trustlines Network

In the current Trustlines Network architecture, everything is handled by a smart contract deployed on the blockchain. Since it is unclear in the current state of Plasma how smart contracts would be deployed and work with Plasma chains, the use of this smart contract has to be rethought.

As a first step to building a Trustlines Network in a Plasma-like architecture, we need a smart contract on the root chain. This smart contract has to allow for the creation of trustlines for users. It also has to set the rules for modifying trustlines and closing them. It is responsible for storing the history of the side chain as list of Merkle roots. Finally it must state how users can exit the side chain and how exit queries can be challenged.

We then need a central validator for the side chain, whose role is to process the queries of users to send credit via trustlines or open new trustlines. It needs to compute the pathfinding, and include the transactions in an indexed Merkle tree where each leaf corresponds to one user (see figure 4). Transactions will be included at every leaves corresponding to an involved user. As a result, users will need to monitor their leaf and every leaves impacted by their transactions to verify its validity.

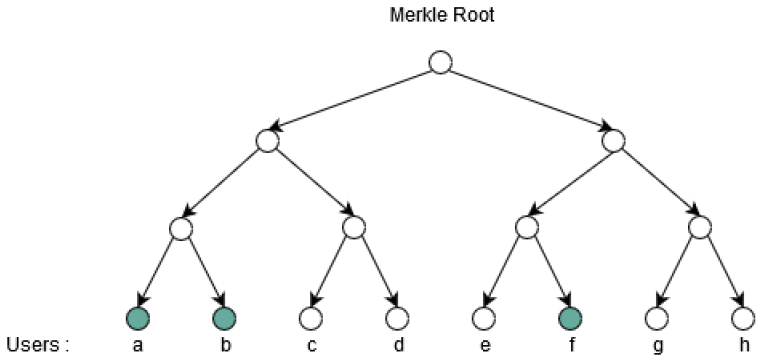

Figure 4: Merkle Tree for transaction from a to f through b, only the leaves of a, b, and f contains a transaction. Each of these users have to verify all the leaves to ensure the transaction is correct

When users are offline, the state of their trustlines can still be modified by the validator as the way they can be modified is restrained by the on-chain smart contract. As users turn back online, they have to gather the blocks they missed and verify the status of their accounts. If we use the

same time window as Plasma of 7 days for exit procedures, users have to go online at least every weeks to ensure security of their accounts.

To guarantee the exit procedure, users would need to store the history of modification of their trustlines. Moreover, we need to take into account a new function on the on-chain Trustlines contract: the *Lock* function. This lock signals that a user wants to exit the chain and does not want to see its trustlines updated through other user's transactions by the validator anymore. The lock is required as users are not the only one responsible for the modifications of their accounts, unlike in Plasma where users can guarantee that the UTXO they provide for exiting will not be spent by others. Once a lock transaction for Alice is send and included in the root chain, Alice knows what state of her account she should exit with and the validator knows not to use Alice as an intermediary for transfers anymore.

In scenarios were the validator is honest, it will be sufficient for Alice to exit to provide the last state of her account right before her accounts got locked. In case of a challenge from other users complaining that they have a different balance for their trustlines with her, Alice will need to provide the full history of modification of her trustlines that she kept throughout her time monitoring the blockchain. Users can then challenge this proof by providing a transaction omitted by Alice that modifies her account before the locking of her account.

In the case were Alice wants to exit the side chain because the validator is misbehaving, for example including faulty transactions resulting in conflicting view of trustlines between users, she will have to give a proof of the invalid block and the state of her account as an argument to the exit. It can then be challenged by someone stating to have a different view on Alice's account, which will be solved in the same way as the regular honest scenario explained above.

However, if someone exits due to a malicious transaction from the validator (or one of the validators), it is likely that everybody will want to exit or at least roll back to the state before the malicious transaction was included in the chain. This is explained by the fact that the final account of the cheated user will be set to the status it had before the invalid transaction, so its neighbors will want to go back to that status to have the same view on their trustlines with the cheated user. As a result, this will create a propagation of users wanting to go back to the valid state for their personal safety. It is thus necessary that every user monitor exit attempts for a valid request due to a faulty validator.

In the case where there is more than one validator and other validators are still honest, users do not need to exit in mass. Since they keep the history of the modification of their trustlines, the remaining honest validators can agree on the number of the last valid block. Users can roll back their status to the one corresponding to that block and continue using the network. This means that if the user exits with the invalid transaction after 7 days it will roll back to 7 days prior. This period of 7 days for exit and challenge could be shortened but it remains a problem in any case to have to rollback transactions.

The procedure of sending your whole history is very costly, but unlike in Plasma Cash, it happens only during exits and not during transfers. Moreover, it happens only if the exiter was malicious, the validator misbehaved, or other malicious users challenge the exit without basis. These malicious behaviours could be punished in some way to be detailed at a later point. Alternatively, like in Plasma Cash, ZK-SNARK could be envisioned to make the proof more succinct, or checkpoints could be published by the side chain validator so that users would only need to store and prove their history since the last checkpoint. For Trustlines, checkpoints could be efficient as the impact of a big set of transaction on the network can be efficiently reduced to a smaller set of transactions, simply by summing up their impacts.

To summarize, a Plasma-like architecture could be used for the Trustlines Network with some challenges: writing the on-chain smart contract to allow for detection of invalid blocks provided by users. The problem of having to rollback the whole network in case validators are dishonest has to be solved and the way of communication between users and validator has to be defined. Additionally, the status of the validator has to be specified: it could be the one who created a particular network, so each communities with different networks would manage their own side chain. A different approach could envision multiple validators either staking in the on-chain contract or in a proof of authority manner.

The speed at which finality is reached depends on this validator. In a network where the validator is highly trusted, we might not need for the inclusion of the Merkle root in the root chain to consider a transaction finalized, since the probability of the validator cheating is low and so is the risk.

It seems this solution is most suitable for the use case where businesses are the users of the Trustlines Network rather than private users.

Lastly, throughout this description I omitted the use of relay servers in the Trustlines Network, principally in order to simplify the explanation,

but the actors of the Plasma-like sidechain could be relay servers instead of users of the Trustlines Network. As such, users would not have to be able to contact the blockchain to lock their accounts, could stay offline more than 7 days, and could leave the task of verifying the behavior of the validator to relay servers.

5.d DFINITY

DFINITY is an ongoing project, intending to create the "Internet Computer with unlimited capacity". It promises to deliver a lot: a self-governing, unbounded, blockchain system based on randomness, supposedly decentralized and capable of growing in terms of storage and computation capacity. What is of most interest for us is the white paper on the consensus system, which was published on the 23rd of January 2018[38].

I decided to write a detailed explanation of the protocol for DFINITY and its challenges because it gathers different building blocks from layer 1 and layer 2 solutions and arrange them together to build a completely new blockchain. The explanation of the protocol gives a detailed example of how verifiable random functions, sharding, proof of stake, and off-chain transactions building blocks could be used in practice. In subsection 5.b, I mostly referred to projects other than DFINITY, also exemplifying different building blocks, in order to refer to the source of the idea to the best of my knowledge. However, the goal of this part is not to establish a comparison between different protocols but rather go in detail into one example.

To create its blockchain, the DFINITY protocol works in rounds, each round creating a new block. At the beginning of each round, a group of users called the *committee* jointly creates a random number. This random number is used to establish a priority ranking of all users in the network. Each users can then propose a block, but the higher their priority the more likely it will be included in the final blockchain. Proposed blocks are received by *notaries* that wait for a certain time before notarizing the block with the highest priority and sending it to all users. The random number is then used to select the new committee and a new round is started.

In the next section, I am going to explain the functioning of the *Verifiable Random Beacon*, before we will see how users propose and *notarize* blocks. Then, I am going to explain the principle of validation tree introduced by DFINITY.

5.d.1 Random Beacon

The goal of the random beacon is to create a stream of deterministic, verifiable random numbers. It is required to be deterministic so that users cannot influence the choice of this number, and verifiable so that users not participating in the protocol can guarantee it was properly generated (see part 5.b.2). To do so, the previous random number is used to select the next committee among groups. This committee will then use (t,n)-threshold BLS signature scheme[39], signing the old random number to create a new one. An initial random number is thus needed for this beacon, which can be taken as a nothing up my sleeve number such as the hash of the string "DFINITY". For the rest of this explanation, we will assume a random number ξ_r has already been produced from the previous round, and we will see how a new number ξ_{r+1} is produced.

To begin every epoch (consisting of a duration of a fixed amount of blocks), m groups of n users are created based on the first generated random number of this epoch. The group creation follows the formula:

$$Group(\xi_r, j) = Perm(PRG(\xi_r, j))(\{1, ..., n\})$$

where ξ_r is the first random number of the epoch, $j \in [1; m]$ is the group number, PRG is a pseudo-random number generator using ξ_r as a seed, and getting the j-th generated number. $Perm$ is the Fisher-Yates shuffle[40] that uses a random number to produce a permutation of all the candidates. We finally take the n first members of the permutation to form the group.

Once groups are formed, members have to perform a distributed key generation to create a group public key and private key shares for each members for the threshold BLS signature scheme. Once the keys are created, a verification vector is published on the blockchain, containing the group public key used to verify the correct generation of the random number by the group if he is picked as a committee. The verification vector also contains information to verify the honest behaviour of individual users participating in the generation.

The group G_j to be the committee for round $r + 1$ is selected as the group number $j = \xi_r \mod m$. To generate the random number for round $r + 1$, each user i of the committee will produce a signature:

$$\sigma_{r+1,i} = Sign(r + 1||\xi_r, sk_i)$$

Where sk_i is the secret key of i produced during the distributed key generation phase. The final signature σ_{r+1} is then recomposed from the individual shares, following the threshold-BLS protocol. The final random number ξ_{r+1} is the hash of σ_{r+1}.

$$\xi_{r+1} = H(\sigma_{r+1})$$

Note that this protocol allows users to reach consensus on a random value, and that the consensus on the final blockchain results from this. This random beacon is an election mechanism based on VRFs (see 5.b.2), as it will decide on the proposer for the blocks of each round as we will see later. We can also point out that a mechanism to protect from sybil attacks has to be put in place to prevent users to register a large number of users to be placed in different committees. A proposed mechanism by DFINITY is PoS where users lock a number of *dfinities* (the DFINITY project cryptocurrency) for at least three months in order to participate.

5.d.2 Block ranking

We saw how the random number is generated for each round and how it selects the committee for that round; we are now going to see how it influences the decision on block proposer, and later another role for committees: *notarization* For each round, a ranking of the N total users is computed as a permutation of the list of users$(1, 2, ..., N)$: $Perm(\xi_r)(\{1, 2, ..., N\})$. Each user can then propose blocks including the round number, the data payload, the number corresponding to the block creator, the hash of the previous block, and the notarization of the previous block. Additionally, each blocks have a different weight depending on the ranking of its owner: $weight = 2^{-ownerRank}$ where the first owner in the permutation has a rank of 0.

Block weights are used to decide on the main fork to build on top of. The weight of a chain is defined as the sum of the weight of all its blocks. Block proposers are incentivised to build on top of the heaviest chain to see their blocks in the final chain, just as in Bitcoin or Ethereum. One block per user can be proposed by each users for each round, thus creating an infinitely growing blockchain. In the next part I explain how the decision on the final chain is made.

5.d.3 Notarization and Finality

The purpose of notarization is to reach finality and to prevent committees from mining blocks and withholding them to later publish them to revert transactions. To perform notarization, for each round, committee members pool blocks from every users for a certain globally set time: $BlockTime$.

Committee members will then produce signature shares on the heaviest block they received and broadcast them along with the block, until a full group signature is produced on a single block. This signature is called the notarization and acts a proof that the block was broadcast timely and not withheld. Due to latency in the network or misbehaving users, more than one block may be notarized each round, which causes no problem for the consensus, since users will build block on top of the highest ranked one.

Once a committee notarizes a block, users can move to the next round $r+1$, where a new committee is selected from the random number ξ_r, a new random number $\xi_r + 1$ is generated as explained above, and new blocks are being proposed and notarized.

For users to have a view on the state of the final chain, they locally run the *finalization* procedure at the end of each round. We know that blocks must include a notarization of the previous block to be valid. Moreover, to be notarized, blocks have to be broadcast during their round. Thus, when round r is over, we know the set of every block that have been notarized for rounds $<= r - 1$ is final and no more blocks will be added. At the end of round r we can not decide on the set of notarized blocks for this round r, since notarization of blocks are included in their children. Users can then compute the final chains of notarized blocks from round $r - 2$ and take the blocks common to every non orphaned chain as final.

5.d.4 Validation Tree

Very few information is available concerning the *payload* part of the blocks to be validated in the blockchain. DFINITY intends to implement a mechanism similar to the Ethereum Virtual Machine, and plans to scale to billions of transactions per second with exabytes of data. It is obvious that not every users in the system will be able to validate every transactions, and that not every transaction can be written on the blockchain. In this part, I will explain the approach of DFINITY on solving this *global processing* problem via sharding (see 5.b.3). Their approach is based on a novel concept to me, extending on the idea of sharding: the *Validation Tree*, which I will explain based on my interpretation of the currently available information[41][42]. I would define this mechanism as a layer 2 solution, as the DFINITY protocol constitutes a complete functioning blockchain without it and it is not currently part of the core DFINITY protocol description.

The Validation Tree is analogous to a Merkle Tree where each node is a so-called *Validation Tower*. At the leaves of the tree, instead of having data to be hashed, there is *state leaves* which are shards of the state managed by a subset of the users of the system (see figure 5 below). Similarly to Merkle

trees, the goal of the validation tree is to create at its root a hash of the state of the system to be included in the blockchain. Validation tree is to validation what Merkle trees are to data.

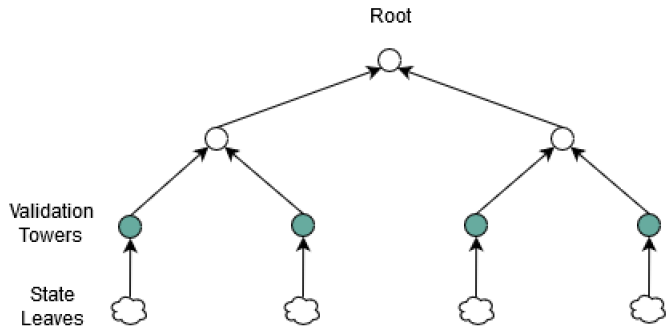

Figure 5: Validation Tree with Validation towers in white and leaves Validation Towers in blue, each mapping to a single state leaf

Leaves validation towers are assigned a shard of state data and receive transactions and proposed consequential transformations impacting this data. They output a hash of the validated new state. Other intermediary towers take as input hash values from two lower rank towers and output a hash of these two values. Finally, the root tower will send its output root hash to the blockchain, as a proof of the state of the system. The goal of validation towers is then to validate its assigned data with a small subset of validators while keeping the same amount of security as what is achieved on the blockchain.

To do so, each tower proceed in an infinitely growing number of *levels*. Each level corresponds to a small group of users (typically 10) picked at random using the number ξ provided by the random beacon previously described. For each level, selected users will validate received inputs for this round and for all the previous d rounds ($d \simeq 10$). A level is considered validated if half of the selected validators attests for the input of the tower for this round. An input of a tower is considered validated by a tower if it has been validated by d consecutive levels of that tower.

As security measure, once a user participated in a level, he cannot participate in the next $d - 1$ levels of that tower. This ensures validation of inputs are done by different users. As a result, an attacker needs to control more than half of the randomly chosen validators of d consecutive levels of a tower in order to validate a malicious transaction.

A long time can be observed between a state being updated and the confirmation of this update being included in the blockchain. However, clients may accept a transaction as finalized once the shard announces it as decided, ignoring this long processing time.

5.d.5 Summary

To summarise, the architecture of the DFINITY protocol can be seen as made of three layers (see figure 6). The first layer is the consensus layer, where the random beacon operates, and blocks are created and finalised. The second layer is the validation tree and serves the purpose of validating the transactions and state transitions that are passed as a data to the blockchain. The third and last layer is the storage layer where shards are handled by different groups of users to store parts of the global state of the system.

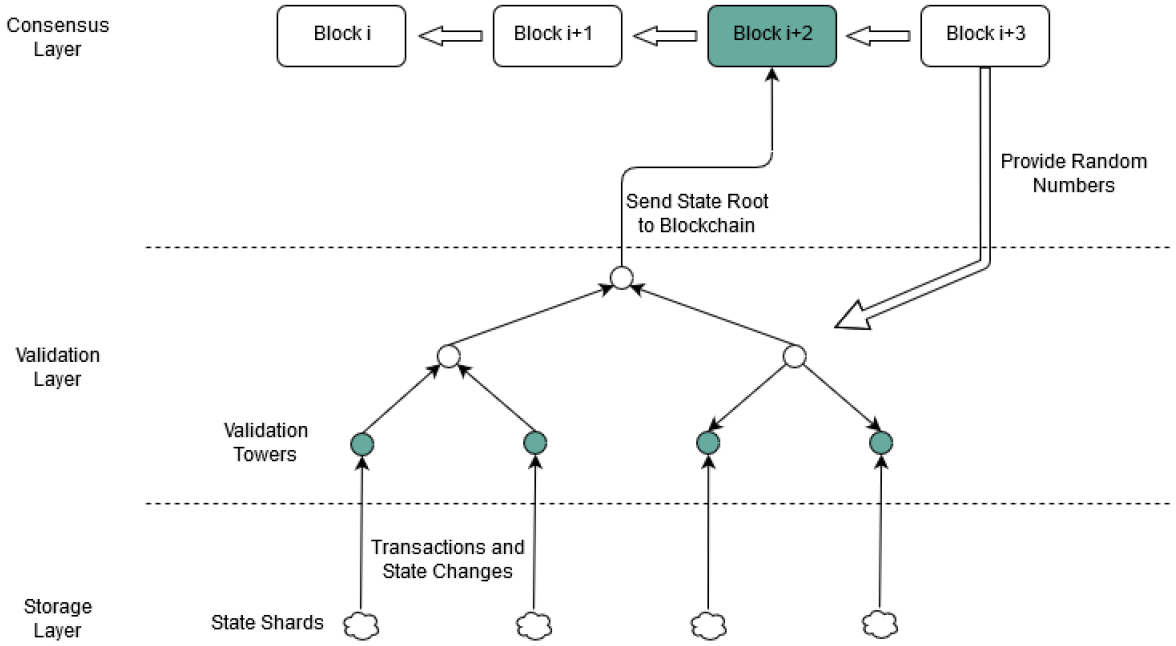

Figure 6: The three layers of the DFINITY protocol

5.d.6 Security Assumptions

Predominantly Honest Committees

The security of the DFINITY consensus protocol is based on various assumptions, which I will attempt to explain in detail. The DFINITY white paper[38] uses a byzantine adversary model, where a certain number of users are deemed byzantine and behave how they like; they can collude together

to attempt any coordinated attacks. The following two assumptions restrict the number of possible byzantine users in the system.

The first assumption is that there exists $\beta > 2$ such that strictly less than $1/\beta$ of users are byzantine. This is equivalent to saying that at least half of the participants in the protocol are honest. The number of participants is defined as by users identified by the sybil resistance layer (see 5.d.1). This means that in the case where users have to stake a specific amount of dfinities to participate, half of the stake is considered as belonging to honest users. Note also that, in this case, to each real life person multiple entities could be associated depending on how many times they are able to lock the expected amount of dfinities. The second assumption is that half of the participants of every committees elected during the protocol are honest. This second assumption only make sense under the mildly adaptive adversary assumption: the adversary is able to corrupt users but its corruption is slower than the time during which the committee will be active.

To verify the second assumption, for a committee size of n, one has to calculate the probability of picking more than $n/2$ dishonest users from a pool of N users, $1/\beta$ of which are dishonest. This is given by the *cumulative hypergeometric distribution function* (CHDF). This function represents the probability that a number of successful draws from a specified set with size N and probability of success $1/\beta$ with n draws without replacement is lower than a given number (here $n/2$). As the number of total users N increases towards infinity, this function grows towards the *cumulative binomial distribution function* (CBDF), and is lower bounded by it, since the probability of successful draw is lower than 0.5. The CBDF represents the same event of successfully picking an adversary from a set, but includes the replacement of the picked user afterward, intuitively similar to having an infinite set.

The total number of users of the system being unknown, it can be more judicious to look at the CBDF than the CHDF. You can find in the table below (table 2) the minimal committee size for a given probability of failure ρ depending on the value of β and whether the CHDF or the CBDF is used for calculation. We can see that the difference in size for the committees depending on the function used are not significant for a number of users of 10,000.

The assumption stated in the white paper[38] that $\beta > 2$ is too weak. For $\beta \simeq 2$, the probability of failure will always be $\simeq 0.5$ for acceptable committee size. DFINITY plans to have a committee size of 400. For a probability of failure $\rho < 2^{-40}$, this means we need $\beta \geqslant 3.050$ and for a

$-log_2\rho$	$\beta = 3$	$\beta = 4$	$\beta = 5$
40	$405 - 423$	$169 - 173$	$111 - 111$
64	$651 - 701$	$277 - 287$	$181 - 185$
80	$811 - 887$	$349 - 363$	$227 - 235$
128	$1255 - 1447$	$555 - 593$	$365 - 383$

Table 2: Minimal committee sizes to reach probability ρ of failing to have a honest majority in the committee, depending on the share of dishonest users $1/\beta$. Lower number is computed via CHDF for a number of users of 10,000 and higher number was computed via CBDF. Source: [38], values verified personally.

probability of failure $\rho < 2^{-128}$, this means $\beta \geqslant 4.926$. The assumption should thus be that less than 33% of the users are dishonest.

The fact that half of the committee is honest comes to justify a set of security assumptions of the security analysis. Indeed, it is implicit in the white paper that the threshold of the signature scheme is $\lfloor n/2 \rfloor + 1$ so that if committees are predominantly honest, every random number produced in the beacon uses a signature share of a honest user, thus cannot be predicted by dishonest users. This also implies that every notarization include a signature from a honest user, and that the block was validated at least by one such user.

Semi-Synchronous Network

The security analysis part of the white paper gives timing assumptions showing why the protocol is efficient in the normal scenario where users are honest. For this analysis, DFINITY assumes a semi-synchronous network where the network traversal time can be modeled by a random variable whose probability distribution is known. The actual required assumption is that the network traversal time is bounded by δ: if a honest user emits a message at time t, every users in the network will have received this message before $t + \delta$.

This is notably, necessary to estimate the value *BlockTime* committee members have to wait to receive blocks before notarizing them. Intuitively, if a user starts waiting for blocks at time t, it means he received a notarisation for the previous round and entered the new round. Other users will begin the new round at most at $t + \delta$ when they received this same notarisation. Committee members produce their signatures on the previous random number and propagate them in the network, producing the random number for the round before $t + 2\delta$. At that time, block proposers can produce and send their blocks to every users before $t + 3\delta$. As a result, if the highest ranked proposer is honest, committees waiting for $BlockTime = 3\delta$

will have received its block and only this block will be notarized, effectively accelerating finalisation. For this analysis we ignored the time users take to compute signatures and blocks, or include this time in the estimation of δ.

If dishonest highly ranked block proposers are present in the system, they can withhold their blocks for a little less than $BlockTime$ to attempt to make the committee create two notarisations, allegedly creating a fork that future block proposers have to decide upon, slowing down finality but not preventing persistence or liveness. Similarly, the incorrect estimation of δ will make it possible for two or more blocks to be notarized in some honest cases. It is hard to tell theoretically the impact of the imprecision on the choice of δ and whether adversaries will ever take advantage of it. The scheme is thus showed to be optimised for semi-synchronous networks, however it is still persistent and live under asynchronous networks and gracefully handles network splits.

If the network is split into two similarly sized parts, the random selection of committees will reach a point where less than half of the committee is in a part, this part will stop advancing in the protocol rounds. The other separated part will stop, eventually at a different instant, for the same reason. The choice of committees being deterministic, and knowing the threshold for creating a new random number or creating a notarisation is $\lfloor n/2 \rfloor + 1$, it is impossible that both separated parts of the network continue the protocol on their side. This ensures there will not be conflicting view on the state of the blockchain when both parts are reunited. However, if only a small number of users are eclipsed from the network, the remaining users will be able to proceed with the protocol.

Cryptography Assumptions

Lastly, for the scheme to remain secure, classical cryptography assumptions are made throughout its description[38]: access to a collision-resistant hash function, cryptographically secure seeded pseudo-random number generator, adversaries are computationally bounded, and the computational Diffie-Hellman problem is hard for elliptic curves with pairings. A specific additional assumption is that the BLS-threshold signature scheme[39] produces an unpredictable deterministic random number. Any of these assumptions failing will result in a potential attack of the protocol.

5.d.7 Analysis

The system currently described does not punish dishonest users proposing a malformed block, failing notarization, or misbehaving in the validation tree. Moreover, there is no apparent incentives to participate in the consensus

protocol or the validation tree protocol, since every user can propose a block and rewards are only provided for block creators. I believe mechanism design and cryptoeconomic incentives can be applied to solve these problems.

As currently described, to "join the system", one has to stake dfinities and publish a public key. As such the system is said to be pseudonymous. In practice the system will only be pseudonymous if one can acquire dfinities in a pseudonymous way. Moreover, it might be hard or impossible for some users to get dfinities at the launch of the system, which can hinder decentralization.

The system is said to be designed via notarization and constant block time to reach finality rapidly, however the process of the validation tree make this finality longer to reach. Though, the assumption that users will take for final a transaction that was validated by a shard is sound and initial confirmation could be reached quickly.

The system does not suffer from selfish mining thanks to the timing aspect of the notarization protocol. Adversaries cannot withhold blocks during a round since the block needs to be notarized by the committee of this round to be validated and included in the blockchain. As a result, unless the adversary controls the majority of this round's committee, at least one honest user will have knowledge about this block and propagate it through the network.

Additionally, the nothing at stake problem can be mitigated by weak subjectivity (see 5.b.1, [28]). If the participants of the consensus protocol have to stake dfinities for 6 months, and honest users reject forks dating from more than 6 months, then regular participants cannot be tricked to accept a long fork forged by malicious validators that have already withdrawn their stakes. However, new participants that may not know which fork appeared when can be tricked and need trust in one of the two fork creators.

As currently described, state and transactions emitted by users are not stored in the blockchain; there is no public history of what happens in the system. Only proofs of inclusion analogue to Merkle proofs can be provided by users that monitored a shard. To validate transactions and state transitions, validators of leaves validation towers have to store the state of their corresponding shard. This means that every time users maintaining/-validating a shard change, the new uses has to find out about the state of their shard, which can lead to a high overhead. If the pools of validators of validation towers do not change rapidly enough over time, an adaptive adversary will potentially be able to corrupt a sufficient number of users and

break the liveness of at least one shard. Moreover, there is no description of cross shards communications or transactions impacting data across multiple shards.

Lastly, the word *random* is used throughout every description of the DFINITY consensus but is not properly defined. It is apparent that the random number produced by the beacon should be uniformly distributed in the range of the application using the produced number: at least the number of users. Otherwise, certain registered users will be more often highly ranked than other for block proposals and attackers would attempt to corrupt these users to damage the system.

The argument from DFINITY for justifying the randomness of the number produced by the beacon is as follows: "Take a unique and deterministic signature scheme, then the signature is a random number, as if it were predictable, the signature scheme would not be secure." I believe this argument is faulty from a simple thought experiment. One can take any deterministic signature scheme, change the signing algorithm to append 8 zeros at the end of the signature, and change the verification algorithm so that it verifies the 8 zeros, remove them, and then run the original verification algorithm to produce its result. This scheme still produces a unique and deterministic signature. However if the application takes the signature modulus 256, the result will always be 0.

This degenerate case shows that the security of signature schemes comes from a certain unpredictability in the resulting signature, but that signatures also have a predictable part, otherwise they would not be verifiable. Finally, as the random number produced is a hash of the signature, the properties of the particular hash function used seem more relevant to me and have to be considered in order to determine if the number produced is random and uniformly distributed. The signature would only assure the deterministic property of the random number.

To conclude, DFINITY is a project that promises a lot: near-instant finality, high scalability and security comparable to current blockchains. Moreover, it plans to work hand in hand with Ethereum and should provide an efficient way for programmers to develop their solution on top of it. However, we can see that there are still unsolved theoretical challenges, and that it is unclear when it will be available. The progress of DFINITY has to be watched closely as to when a beta version will be available and what is its actual scalability potential.

Lastly, contrary to the Plasma paradigm, from which Trustlines could take the idea and build its own implementation of the consensus, it seems infeasible to build a small scale working implementation of DFINITY for Trustlines. The documentation for DFINITY is not furnished enough, and the project has too many components. It is worthy enough to note that it could still be a viable solution for the scalability problem of Trustlines in the long run.

5.e Peer-to-peer Architecture

5.e.1 General Description

A radically different architecture is possible for Trustlines: a peer-to-peer (P2P) architecture. In this paradigm, there would be no need for a global consensus; individual users would only be interested in the status of their own trustlines and agreement would only be reached in between two neighbours. This results in very low transaction costs, near-instant transaction time, and high decentralization. I am going to describe this protocol, omitting a few easily solvable imprecisions.

The motivation behind a P2P architecture is that in the Trustlines Network, two adjacent users trust each others up to a certain amount of credit. Therefore, we can have the assumption that two users can agree on the balance of their trustlines entirely without the help of a blockchain. I want to highlight that the assumption is that an identified cheating user will be liable in real life and settlement can happen outside of the system, however it is important to users that they know they have been abused. This allows us to avoid all the mechanisms put in place by hashed-timelock-based payment channels network[43] for single hops transactions. However, we can not assume this trust to be transitive, so we need to design a different off-chain mechanism to treat multi-hop transactions.

For Alice to make a transaction to Charles through Bob, she would send a message to Bob for him to relay to Charles, saying she wants to make the transaction using this path and a signature. The participants will then lock some of their trustlines credit for this transaction, preventing other transactions from interfering. Secondly, they update their trustlines along the lines from Alice to Charles. Lastly, Charles has to confirm receiving the transaction to Alice, that signs the confirmation and transmit it along the path used for the transaction (see table 3).

Number	Users	Action
1	A → B	Message: "I want to send x to C", with path and signature
2	B → C	Transmission of message
3	C & B	Lock the funds
4	B & A	Lock the funds
5	A & B	Update the trustline
6	B & C	Update the trustline
7	C → A	Send confirmation of receipt
8	A → B	Send signed confirmation of receipt

Table 3: Abstract protocol for a transaction from A to B through C

The first four steps of the protocol are required to prevent parts of the path becoming unusable during the transaction because of another transaction exhausting a creditline. It is also necessary to prevent B from spending the credit he acquired from A before the transaction is over and validated.

For steps number 5 and 6, it is necessary to update the trustlines starting from A, to make sure intermediaries increase their balance with the previous user in the chain before decreasing their balance with the next one. This ensures honest intermediaries do not lose anything in the process.

Similarly, the confirmation of receipt is needed so that intermediaries along the path know the transaction was successful. If no confirmation is received in a certain time period, they can safely revert their trustlines balances to the one before the transaction, assuming that some users in the path cheated. To exemplify this security, imagine the protocol stops before step 6: B does not update its trustlines with C in an attempt to receive more credit than he gives. A will not receive a confirmation from C and will revert its balance with B who will not have gained anything in the process.

5.e.2 Offline Intermediaries

The main problem with such an architecture is that, contrary to an on-chain architecture, users need to be actively participating in the protocol to act as intermediaries for transactions. As Trustlines is implemented as a mobile application, users will regularly turn offline abruptly and for an unpredictable time.

The current description of the protocol, does not pose a problem for users who suddenly disconnect. It is impossible to distinguish a honest user having a bad connection from an evil user attempting a denial of service attack by turning offline halfway through the transaction. However, it is possible via the confirmation of receipt to detect this scenario, cancel the transaction, and attempt the transaction via another intermediary node.

As a result, some users might be unable to reach the destination of their payments if too many intermediaries are offline.

The severity of this problem depends on the use case for Trustlines: communities of private users will turn offline more often than businesses. Moreover, depending on the connectivity of the graph and the percentage of time users are unreachable, this could happen not to be a problem at all. However, this will definitely be a problem for early adopters. The P2P architecture could thus be envisioned after reaching moderate adoption, when scalability issues become concrete. It is difficult to evaluate this criteria theoretically and an empirical study would have to be conducted.

To alleviate the offline problem, one can imagine a fallback system where the function of intermediaries of unreachable nodes can be assured by a third party. As such, relay servers could be used to "host" the status of different users, and could be used when they turn offline. Relay servers would have the authority to modify the trustlines of the hosted user based on a transaction, but not create transactions. Once a user turn back online, he would ask the relay server for its new status and get synchronized back into the network.

This creates a new problem as relay servers would have to be trusted, otherwise they can disrupt the system by accepting trustlines updates and not informing the hosted user about these updates later on. We can detect a misbehaving relay server by making users and relay servers store proofs of transactions modifying their trustlines, as in the Plasma approach. This history has to be kept until two adjacent neighbours of a trustline are online and can agree on their status. Note that this is a more reasonable requirement than the one about the history of transactions in Plasma. Once a node goes online, it can then ask its neighbours if they agree on the status provided by the relay server. In case some of its neighbours are offline, it has to wait until a later point to do this verification. History of transactions have to be kept both by the relay servers and users to be able to detect whether the relay server or the user lies about the new status of the trustline.

Since we are able to detect a misbehaving relay server, a punishment system can be established, such as in a proof of stake or proof of authority consensus. However, the loss due to a misbehaving relay server could be unbounded whereas the punishment would always be bounded.

Since the beginning of this section, I have assumed the path for the transaction between users was already known, or that it was not a problem to find it. I believe this is not a strong assumption and is backed by research on decentralised routing algorithms, for example in [16] and [17], even though

describing their method fell out of the scope of this thesis as explained earlier in 4.e. Moreover, without relay servers, depending on the pathfinding algorithm used, the system can achieve a good privacy guarantee. Indeed, as no single entity need not be aware of the state of the whole system, the global status of a user's trustlines is only known to him. Whereas relay servers would lower the privacy back to its state in the current architecture.

To conclude this part, the idea of P2P architecture could be envisioned for the Trustlines Network if it decides to focus its use case on businesses, or could be taken over by other projects intending to build a trust network not relying on mobile users.

5.f Permissioned Chains

The goal of this part is to present a practical solution deployable in the next few months to allow especially for low transaction costs for the Trustlines Network. A provisional solution can be to use a permissioned chain with a selected set of validators (see 5.b.1). This permissioned blockchain allowing only a limited number of validators make it easier to scale. Additionally, it can be effortless to run an application designed for the traditional PoW Ethereum blockchain on a PoA or PoS chain designed to run the Ethereum Virtual Machine (EVM), and the other way around.

5.f.1 PoA Test Networks

Since no computational power is wasted in a PoW, PoA allows for lower transaction fees, stable block time, and effectively higher transaction throughput. However, PoA chains are inherently centralized and the validators identities have to be public, which falls out of the ideology of the project. Additionally, members of the blockchain community could be reluctant to embrace an application deployed on such a chain. This solution must only be seen as temporary and perhaps will not even be suitable to reach a medium level of adoption.

The Trustlines Network is currently deployed on the PoA Kovan testnet[44], for its intended purpose: closed beta testing. However, no applications on Kovan should have any economic value. Indeed, testnets may not mind crashes, restarts, or rollbacks. An attack on the Ropsten testnet has been observed in the past, abusing the low hashing power of the PoW chain and multiplying the gas limit of blocks by a factor of a thousand to freeze the whole network. Even though Kovan's PoA protects it from such an attack, it should only be used to test applications before deploying them and do not

provide any guarantees of stability or security. Moreover, testnets provide limited amount of test ether to users, in a controlled fashioned. To maintain the Trustlines Network in the long run, participants have to be able to pay for transactions easily, without a central third party controlling distribution of coins.

5.f.2 Trustlines Network Chain

To avoid the drawbacks of test networks, the idea emerged for the project to deploy its own blockchain for the sole purpose of running the Trustlines Network. Since Trustlines is already deployed and running on Kovan, it could be possible to use the same blockchain protocol with a different validator set. Kovan is using the Aura engine developed by Parity Technologies[45] to run its blockchain. We thus planned to adapt the same engine to make it work with our requirements. The goal of this approach is to make as few adjustments as possible in order to be able to have a working version of a blockchain satisfying the requirements of the Trustlines Network as soon as possible. Most of the knowledge about Kovan and Aura comes from their GitHub or Wiki page[44][45] as well as from analyzing the blocks for the Kovan blockchain on Etherscan[46]. In-depth understanding of Aura also came through discussions with developers at Parity Technologies.

Validators Choice

To prevent having public entities validating the chain while still allowing a higher scalability and lower transaction fees, we envisioned auctioning validation rights to pseudonymous users on the Ethereum main chain. This makes the chain a permissioned PoS chain. To avoid having a single person acquiring validator rights more than once, we will have to privately identify bidders through a "Know Your Customer" process, or only allow addresses from a known list of addresses already identified by another party. However, we have to make sure the identity of said parties cannot be revealed at a later point. This requirement falls from the ideology of the project and is not purely linked to a technical requirement. The auction of validation rights will occur before the Trustlines blockchain is started, and winners will see their addresses included in the genesis block of the blockchain to be able to act as validators, just as is the case for Kovan.

To determine the number of validators suitable for the Trustlines blockchain, I looked at the number of validators or miners for other blockchains. In Aura, a block and all of its transactions are considered final when at least 50% of the validators have built another block on top of it; that is

to say the security of the chain is compromised if 50% of the validators are misbehaving. It thus makes sense to compare the number of validator needed in Aura with other chains requiring 50% of honest participant, in strict term of number or in terms hashing or staking power. The number of validators running the Kovan test network is only 10[44], as the identity of validators is well-known and this is only a test network, this low number of validators is deemed sufficient. I believe a blockchain with pseudonymous validators should have more validators as a misbehaving entity will suffer less harm. Regarding the Ethereum blockchain, as of August 31st 2018, three mining pools gather over 50% of the block production of the blockchain[46]. However, one can argue that a collusion of these three mining pools would not result in a full control of the network, as miners noticing misbehaviours would leave the pools and join another smaller pool, or mine by themselves. I believe the number of 90 different miners for the last 7 days in Ethereum is more relevant to evaluate the required number of validators for a chain. I thus have reasons to think that 100 active validators will be sufficient to run the Trustlines blockchain.

As we plan to auction validation rights on the Ethereum chain and bidders might not follow with their intent of being a validator, it is necessary to provision for more than 100 validators; an educated guess of 150 validator rights auctioned is sufficient if two thirds of the buyers actually participate in the protocol. Note that fulfilling these requirements about the validator set does not require a large amount of development work: one smart contract has to be deployed on the Ethereum chain to allow for a one time auction of validation rights, and a modification of the list of validators in the genesis block of the Kovan chain has to be made.

Offline Validators Problem

As our validator set is quite different from the one of Kovan, we imagine that a significant part of the initially registered validators might not participate in the protocol; I will explain why offline validators are a problem and how we can solve this problem. Aura works in slots of fixed time; to every slots correspond a single block proposer selected in a predefined order among the set of validators[45]. This poses two problems for Trustlines. First of all, if the selected validator is offline, no block will be proposed for this slot, effectively reducing transaction throughput. Secondly, for the protocol to be secure, 50% of the validators have to behave honestly, and to reach finality 50% of the validators have to be online.

The first intuitive solution to deal with offline validators is to have a ranking of different proposers for each time slots, this will increase the like-

liness that at least one block is proposed in every slot. However, this does not help regarding finality if more than 50% of the validators are offline or misbehaving. This also transforms the previously rather linear blockchain into one with potential forks at every block height and adds the problem of resolving conflicting forks. Additionally, different slashing conditions for validators will potentially have to be employed. Currently the only way validators can misbehave is by proposing two different blocks for the same block height when they are selected as block proposer, the slashing condition is thus fairly simple but will potentially prove insufficient for a blockchain with more complicated forks.

In order to keep a simple linear blockchain with the same simple slashing as the original Aura protocol, another solution has been thought out. This solution is to remove inactive validators directly from the set of eligible validators. To do so, a first removal of inactive validators takes place at the launch of the blockchain. During the first day of the launch, blocks will only be created by a single "trusted entity". Every validators will have to send a special transaction proving they are online and willing to participate during this period. Every validator that did not send a transaction during this period are considered offline and are removed from the list of validators. This process allows to effectively remove users that changed their mind about participating in the protocol after acquiring rights or that fails to participate. Note that validators will be able to receive transactions from every other honest validators wishing to prove online and will be able to verify the correct behaviour of the trusted entity. No other transactions will be included in the blockchain during this period and the trusted entity's block proposition powers will be removed afterward. To remove further inactive validators, forks can take place at regular time periods. The community of validators and other persons involved in the Trustlines Network can discuss around which validator to remove in order to keep the lowest number of inactive validators.

In addition to the offline validator problem posed by the simple round robin validator selection, a situation could arise where a list of consecutive validators would want to censor a certain validator by never building on top of its block but always on top of the preceding block. This would result in the censored validator not receiving any rewards. It is unclear how Aura handles this situation and how many colluding consecutive validators are necessary to censor one block proposer. However, a simple fix to avoid this situation from repeating at every loop of the round robin election mechanism used in Aura is to shuffle the selection of the validator by taking the hash of the round number to select the validator instead of the actual round number (see part 5.b.2).

Token Bridge

Validators will be rewarded for proposing blocks and including transactions in blocks by minting and receiving Trustlines tokens. This token will thus be the native coin of the Trustlines chain. To allow users to interact with the blockchain and allow transaction fees to be paid, tokens have to be air-dropped. Additionally, for validators to be correctly incentivised to participate in the protocol, the tokens have to be transferable from the Trustlines chain to the Ethereum chain to be potentially listed on exchanges and be valuable. The tokens would only be bought on the Ethereum chain if they can be transferred back to the Trustlines chain and used to pay transaction fees. We thus need a bidirectional bridge between the Ethereum chain and the Trustlines chain allowing for the transfer of Trustlines tokens between the two chains.

The parity-bridge[47] offers an implementation of such a bridge. In this implementation, smart contracts are deployed both on the *Home* (Ethereum) chain and the *Foreign* chain (for us, Trustlines chain). These smart contracts are managed by the same set of chain validators. Validators verify and sign request for cross-chain transfers. A user that wants to transfer tokens from the Home chain to the Foreign chain will deposit their tokens on the bridge contract in Home. Validators will get notified of this event and will send a transaction on Foreign stating they accept the transfer. Once a sufficient number of validators (for example 50% of the total number of validators) have accepted the transfer, the balance of the user is increased on Foreign (see figure 7).

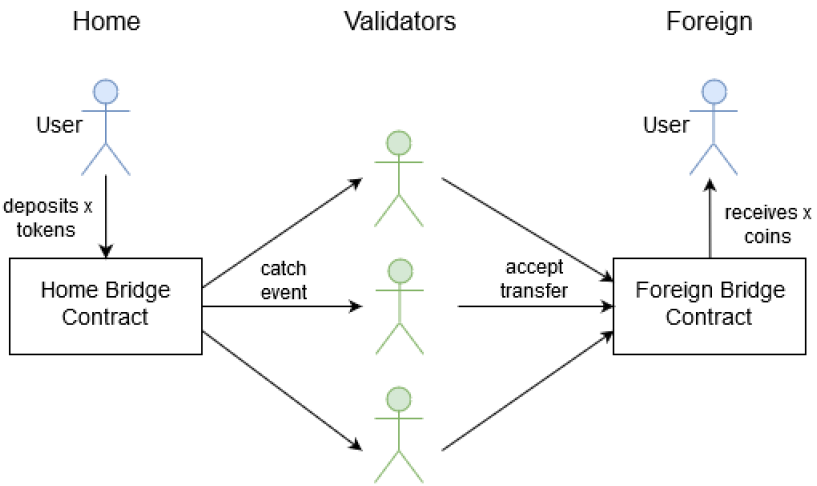

Figure 7: A transfer of tokens by the user in blue from the Home chain to the Foreign chain using the bridge

When a user wants to transfer his tokens from Foreign towards Home, he can deposit his tokens on the smart contract of Foreign. Validators will catch this event and sign a message containing the address of the user, the value to be transferred, and the hash of the original transaction. Validators then submit this signature to the contract on Foreign. Once enough signatures have been submitted, a validator can send a collection of all the signatures on the Home contract to enable the payment to the user (see figure 8). Signatures are collected on Foreign as validators can include the transaction to the bridge contract in the block they produce and do not need to pay for transaction fees like they would normally do on Home. For this same reason, the validator sending the collection of signatures on Home has to be rewarded by the originator of the transfer to compensate for the transaction fees.

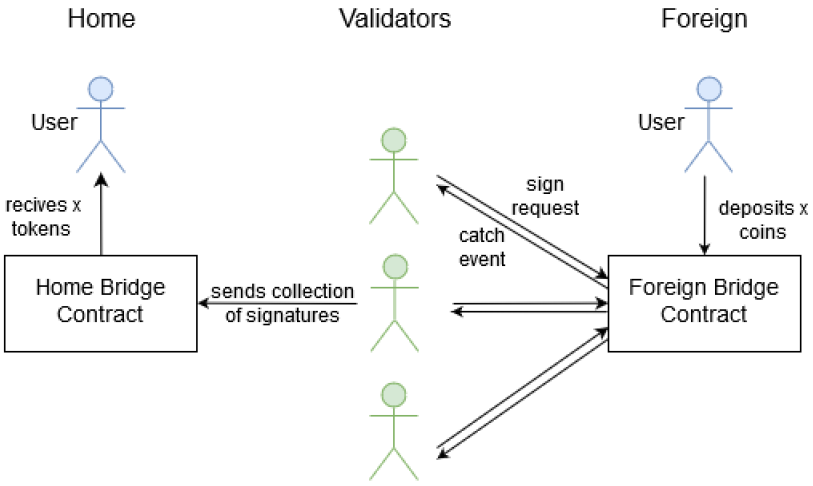

Figure 8: A transfer of tokens by the user in blue from the Foreign chain to the Home chain using the bridge

Implementing the above described bridge brings some challenges. The implementation from Parity bridges the native coin ether in Home to an ERC20 token on a Foreign chain. Our requirements are actually inverse, we want to bridge an ERC20 token on Home to a native coin on Foreign. This notably implies that only ERC20 tokens will be available on Home to reward validators proceeding a transfer from Foreign to Home, unless the user provides ether in another way. However, the hardest problem to resolve is trying to keep the validator set consistent between the bridge validator set on both chains and the chain validator set. As we decided on planning for regular forks, mostly to kick out inactive validators of the chain validator set, the bridge and chain set will eventually differ. Disgruntled former chain validator could later collude to steal the ERC20 tokens from the Home bridge

contract. It is thus important to maintain a certain cohesion between the two validator sets.

A way to achieve this is to make the validators fork the bridge on both chains every time they want to modify one of the validator set. With the current implementation of parity-bridge, this results in the forking of the ERC20 token, as the bridge and token are handled in the same contract. Forking a token is a sensitive action as users will have to be aware of this fork and stop using the old token if it proves that the fork is desirable and the old token has no value anymore. A way around this would be to separate the bridge and the token, but for the same reasons, users will have to be aware of legit forks happening on the bridge so they do not use the old bridge anymore.

To conclude, most of the requirement for the deployment of a Trustlines blockchain can be met with the Aura engine. However, if a bridge is to be set, some technical aspects have to be rethought and a certain amount of work has to be performed. As an additional benefit of the Trustlines blockchain, if coins are airdropped on the Trustlines chain to potentially interested users, these users will not have to worry about getting cryptocurrencies to pay out for transactions and are abstracted complicated interactions with the blockchain, as the project intended initially.

5.g Architecture Scaling Conclusion

In this part, we have understood different mechanisms that can be put in place to create a scalable consensus mechanism. We have also understood how these mechanisms can be put together to make a whole protocol via the examples of Plasma and DFINITY. The value of these two protocols to solve the scalability problem of the Trustlines Network has been analysed and discussed. Later, I presented two alternative solutions for Trustlines: a peer-to-peer architecture and a permissioned blockchain.

Over the time of my thesis, I have seen different project for building a scalable blockchain progress. It appears to me that by the time the Trustlines team would develop a usable smaller sized version of Plasma or other solutions, a better alternative would be offered to them. As a result, I would address the recommendation to the Trustlines project to follow on the perspective of deploying their own blockchain with as little modifications from Aura or the parity-bridge as possible while keeping an eye on the evolution of other projects such as Plasma, DFINITY, Ethereum 2.0, Cardano or other.

6 Conclusion

We have seen in the first part the goals of the Trustlines Network project and how it is confronted to scalability issues. In the second part I measured experimentally the capacity of the pathfinding algorithm used in Trustlines. We saw that the algorithm was not sufficient but understood that the underlying consensus mechanism was a more important problem to solve and that the way the pathfinding algorithm should be modified depends on the architecture of the Trustlines system. I thus decided to focus my research on the last part of this thesis on scalability of blockchains and other potential solutions for the Trustlines Network's architecture. I concluded in this part that the Trustlines team should deploy their own permissioned chain requiring the least amount of work, in wait for bigger projects to provide better fitting solutions.

References

[1] A. de Vries, "Bitcoin's Growing Energy Problem", *Joule*, vol. 2, no. 5, pp. 801–805, 2018, ISSN: 25424351. DOI: 10.1016/j.joule.2018.04.016. [Online]. Available: https://doi.org/10.1016/j.joule.2018.04.016.

[2] H. Hees, G. Friis, and K. Naerland, "Trustlines Network whitepaper (Draft version 0.3)", no. February, pp. 0–19, 2017. [Online]. Available: https://trustlines.network/whitepaper{_}v03.pdf.

[3] *Ripplepay.com*. [Online]. Available: https://classic.ripplepay.com/ (visited on 08/19/2018).

[4] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger", *Ethereum Project Yellow Paper*, pp. 1–32, 2014, ISSN: 1098-6596. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[5] K. Mehlhorn and P. Sanders, "Algorithms and data structures: The basic toolbox", *Algorithms and Data Structures: The Basic Toolbox*, pp. 1–300, 2008, ISSN: 0302-9743. DOI: 10.1007/978-3-540-77978-0. arXiv: 9780201398298.

[6] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest paths algorithms: Theory and experimental evaluation", *Mathematical Programming*, vol. 73, no. 2, pp. 129–174, 1996, ISSN: 0025-5610. DOI: 10.1007/BF02592101. [Online]. Available: http://link.springer.com/10.1007/BF02592101.

[7] F. Schulz, D. Wagner, and K. Weihe, "Dijkstra ' s Algorithm On – Line : An Empirical Case Study from Public Railroad Transport", pp. 110–123, 1999. DOI: 10.1007/3-540-48318-7_11.

[8] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang, "Blockchain Challenges and Opportunities : A Survey Shaoan Xie Hong-Ning Dai Huaimin Wang", *International Journal of Web and Grid Services*, pp. 1–24, 2016. DOI: 10125/41338. [Online]. Available: http://inpluslab.sysu.edu.cn/files/blockchain/blockchain.pdf.

[9] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain", *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, vol. 2017-Janua, pp. 2567–2572, 2017. DOI: 10.1109/SMC.2017.8123011.

[10] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9591, pp. 112–125, 2016, ISSN: 16113349. DOI: 10.1007/978-3-319-39028-4_9.

[11] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the Age of Blockchains", 2017. arXiv: 1711.03936. [Online]. Available: http://arxiv.org/abs/1711.03936.

[12] P. Moreno-Sanchez, M. B. Zafar, and A. Kate, "Listening to Whispers of Ripple: Linking Wallets and Deanonymizing Transactions in the Ripple Network", *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 436–453, 2016, ISSN: 2299-0984. DOI: 10.1515/popets-2016-0049. [Online]. Available: https://www.degruyter.com/view/j/popets.2016.2016.issue-4/popets-2016-0049/popets-2016-0049.xml.

[13] S. Edunov, C. Diuk, I. O. Filiz, S. Bhagat, and M. Burke, *Three and a half degrees of separation*, 2016. [Online]. Available: https://research.fb.com/three-and-a-half-degrees-of-separation/.

[14] J. Travers and S. Milgram, *An Experimental Study of the Small World Problem.pdf*, 1969.

[15] D. J. J. Watts and S. H. H. Strogatz, "Collective dynamics of 'small-world' networks.", *Nature*, vol. 393, no. 6684, pp. 440–442, 1998, ISSN: 0028-0836. DOI: 10.1038/30918. arXiv: 0803.0939v1. [Online]. Available: http://dx.doi.org/10.1038/30918{\%}5Cnhttp://www.nature.com/nature/journal/v393/n6684/abs/393440a0.html.

[16] P. Prihodko, S. Zhigulin, M. Sahno, and A. Ostrovskiy, "Flare: An Approach to Routing in Lightning Network", p. 40, 2016. [Online]. Available: http://bitfury.com/content/5-white-papers-research/whitepaper{_}flare{_}an{_}approach{_}to{_}routing{_}in{_}lightning{_}network{_}7{_}7{_}2016.pdf.

[17] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions", 2017. DOI: 10.14722/ndss.2018.23252. arXiv: 1709.05748. [Online]. Available: http://arxiv.org/abs/1709.05748.

[18] *Ethereum Avg. Transaction Fee chart*. [Online]. Available: https://bitinfocharts.com/comparison/ethereum-transactionfees.html (visited on 09/01/2018).

[19] V. Buterin, *What Proof of Stake Is And Why It Matters*, 2013. [Online]. Available: https://bitcoinmagazine.com/articles/what-proof-of-stake-is-and-why-it-matters-1377531463/ (visited on 06/20/2018).

[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10401 LNCS, pp. 357–388, 2017, ISSN: 16113349. DOI: 10.1007/978-3-319-63688-7_12.

[21] J. Poon and V. Buterin, "Plasma : Scalable Autonomous Smart Contracts Scalable Multi-Party Computation", pp. 1–47, 2017. [Online]. Available: https://plasma.io/plasma.pdf.

[22] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globally distributed database", *Google's Globally Distributed Database*, vol. 31, no. 3, pp. 1–22, 2013, ISSN: 0734-2071. DOI: 10.1145/2491245. arXiv: arXiv:1011.1669v3. [Online]. Available: http://dl.acm.org/citation.cfm?id=2491245{\%}5Cnpapers3://publication/doi/10.1145/2491245.

[23] R. S. Prasad, K. V. B. Rao, and G Murali, "Secure Leader Election for Intrusion Detection in MANET", vol. 5, no. 1, pp. 9–16, 2015.

[24] A. Back, "Hashcash - A Denial of Service Counter-Measure", no. August, pp. 1–10, 2002. [Online]. Available: http://www.hashcash.org/Papers/Hashcash.Pdf.

[25] J. a. Kroll, I. C. Davey, and E. W. Felten, "The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries", *The Twelfth Workshop on the Economics of Information Security (WEIS 2013)*, no. Weis, pp. 1–21, 2013, ISSN: 15437221. DOI: June11-12,2013. arXiv: arXiv:1311.0243.

[26] K. Sunny and S. Nadal, "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake", 2012, ISSN: 1098-6596. DOI: 10.1017/CBO9781107415324.004. arXiv: 1703.04057. [Online]. Available: http://peerco.in/assets/paper/peercoin-paper.pdf{\%}0Ahttp://fc17.ifca.ai/preproceedings/paper{_}73.pdf{\%}0Ahttp://arxiv.org/abs/1606.06530{\%}0Ahttps://papers.ssrn.com/sol3/papers.cfm?abstract{_}id=2977811{\%}0Ahttp://dl.acm.org/citation.cfm?doid=2976749.2978389{\%}0Ahttp.

[27] V. Buterin and V. Griffith, "Casper the Friendly Finality Gadget", pp. 1–10, 2017. arXiv: 1710.09437. [Online]. Available: http://arxiv.org/abs/1710.09437.

[28] V. Buterin, *Proof of Stake: How I Learned to Love Weak Subjectivity*, 2014. [Online]. Available: https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/ (visited on 06/18/2018).

[29] A.-D. A. Arpaci-Dusseau Remzi, "Operating Systems: Three Easy Pieces", *Arpaci-Dusseau*, vol. Electronic, no. 0.91, p. 665, 2015.

[30] Vitalik Buterin, *Validator Ordering and Randomness in PoS*. [Online]. Available: https://vitalik.ca/files/randomness.html (visited on 06/18/2018).

[31] S. Micali, M. Rabin, and S. Vadhan, "Verifiable Random Functions", *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 120–130, 1999, ISSN: 0272-5428. DOI: 10.1109/SFFCS.1999.814584. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=814584.

[32] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies", *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, 2017. DOI: 10.1145/3132747.3132757. [Online]. Available: https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf.

[33] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10821 LNCS, pp. 66–98, 2018, ISSN: 16113349. DOI: 10.1007/978-3-319-78375-8_3.

[34] Raul Jordan, *How to Scale Ethereum: Sharding Explained*. [Online]. Available: https://medium.com/prysmatic-labs/how-to-scale-ethereum-sharding-explained-ba2e283b7fce (visited on 09/01/2018).

[35] S. Dziembowski, S. Faust, and K. Hostáková, "Foundations of State Channel Networks", pp. 1–56, 2018. [Online]. Available: https://eprint.iacr.org/2018/320.pdf.

[36] Vitalik Buterin, *Minimal Viable Plasma*, 2018. [Online]. Available: https://ethresear.ch/t/minimal-viable-plasma/426 (visited on 06/18/2018).

[37] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation", *Proceedings - IEEE Symposium on Security and Privacy*, pp. 238–252, 2013, ISSN: 10816011. DOI: 10.1109/SP.2013.47.

[38] T. Hanke, M. Movahedi, and D. Williams, "DFINITY Technology Overview Series Consensus System",

[39] L. Ben, B. Dan, and S. Hanav, "Short Signature from the Weil Pairing", *J. Cryptology*, vol. Vol.17,No. Pp. 1–24, 2004, ISSN: 0933-2790. DOI: 10.1007/3-540-45682-1_30. [Online]. Available: http://crypto.stanford.edu/{~}dabo/abstracts/weilsigs.html.

[40] D. E. Knuth, *The Art of Computer Programming (vol. 2_ Seminumerical Algorithms) (3rd ed.)* 1997.

[41] DFINITY, *DFINITY website's FAQ*, 2018. [Online]. Available: https://dfinity.org/faq (visited on 07/02/2018).

[42] D. Williams and T. Hanke, *A Brief Tour of Dfinity*, 2016. [Online]. Available: https://www.youtube.com/watch?v=DP8X-NuG4z0{\&}feature=youtu.be (visited on 07/02/2018).

[43] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments", *Technical Report (draft)*, p. 59, 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf.

[44] Kovan Team, *Kovan Testnet Proposal*, 2017. [Online]. Available: https://github.com/kovan-testnet/proposal (visited on 09/04/2018).

[45] Parity Technologies, *Aura - Authority Round*. [Online]. Available: https://wiki.parity.io/Aura (visited on 09/04/2018).

[46] Etherscan Team, *Etherscan*. [Online]. Available: https://etherscan.io/ (visited on 09/04/2018).

[47] Parity Technologies, *Parity Bridge github*. [Online]. Available: https://github.com/paritytech/parity-bridge (visited on 09/04/2018).

[48] J. Stark, *Making Sense of Ethereum's Layer 2 Scaling Solutions: State Channels, Plasma, and Truebit*, 2018. [Online]. Available: https://medium.com/l4-media/making-sense-of-ethereums-layer-2-scaling-solutions-state-channels-plasma-and-truebit-22cb40dcc2f4 (visited on 06/18/2018).

[49] D. Mazières, "The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus", pp. 1–45, 2015.

[50] M. Castro, M. Castro, B. Liskov, and B. Liskov, "Practical Byzantine fault tolerance", *OSDI {'}99: Proceedings of the third symposium on Operating systems design and implementation*, no. February, pp. 173–186, 1999, ISSN: 07342071. DOI: 10.1.1.17.7523. arXiv: arXiv:1203.6049v1.

[51] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains (A position paper)", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9604 LNCS, pp. 106–125, 2016, ISSN: 16113349. DOI: 10.1007/978-3-662-53357-4_8. arXiv: arXiv:1311.0243.

[52] Karl Floersch, *Plasma Cash Simple Spec*, 2018. [Online]. Available: https://karl.tech/plasma-cash-simple-spec/ (visited on 06/18/2018).