Scalability Solutions

Disclaimer: The content of these research summaries has been written after a year of reading, researching and writing about blockchain technologies and applications. Definitions may vary depending on the paper cited. The summaries provided are subject to further iterations; whereby, the first version relies on my personal understanding of the industry and the technologies. Most of it is based on informal discussions, academic papers, industry whitepapers and primary research. These research summaries may foster from previous research but do not replicate any ideas or content created previously.

For comments, references, contribution proposals etc. please contact Anais Urlichs on Twitter, LinkedIn or email under urlichsanais@gmail.com

This docs has last been updated on: 07.04.2019

Overview

Assuming that a replicated ledger, or blockchain has similar user requirements as a centralised database, it also has to provide linear scalability. Meaning, no matter how many users access the ledger and the data stored within, the ledger has to operate reliable without high latency and risk of failure. Thus, a user should be able to access accurate information in a timely manner. Some databases are accessed by thousand of people, who want to retrieve the latest and most accurate information stored in the database. Accessing data from a database may be referred to as querying. Queries either access the raw information or require the processing of several entries to provide the requested outcome.

Blockchain networks, such as Bitcoin or Ethereum, provide the value transfer between several participants. Information may be submitted, processed and retrieved from the ledger to allow for further operations. Each transaction has to be processed by all full nodes. (For more information, please refer to the summary on Replicated Systems.) If the network is under attack or partitioned, the user/node may not be able to access the requested data. Assuming that the ledger is operating at full capacity, Ethereum is currently able to process around 13 transactions per second, while Bitcoin processes around 8. In comparison, Visa can handle up to 50.000 transactions per second. Given these information, it becomes clear that Visa is able to operate at much higher capacity than the most popular blockchain networks.

Once the user/node has submitted a transaction to the network, she/he has to wait until the transaction is processed by the miners or validators and declared to be final. (For more information, please refer to the summary on Consensus Mechanisms.) The time period required for a user to wait for a transaction to be submitted, processed and finalised may be referred to as the network latency. Note that the latency of a network may be independent of its throughput. The throughput measures how many transactions are processed, or the number of packages sent within a specified time period. While the throughput of a system might be quite high, the latency might be high too and vice versa. A system with limited throughput may have high latency or low latency. The effect of the throughout and latency on a system is design dependent. Systems aim to optimise for a high throughput and low latency. Generally, replicated ledgers have high latency and limited throughput. Scalability solutions aim to optimise and improve latency and throughput to provide a better user

experience and allow for various use cases, which required the timely processing of transactions.

Types of Scalability Solutions

We can differentiate between two categories of scalability solutions. The first one is referred to as Layer 1 and the second one as Layer 2 solution.

Background Information

Overall, blockchains have one chain. New transactions are gathered into blocks of data, which become appended to the existing chain in sequential order. Each additional block references the block prior and so on. Therefore, each additional block is linked to the previous blocks of the blockchains. In case the data of any previous block is modified, the change will tamper with all sequential blocks in the network. Resulting, every change of a block would be visible to all nodes in the network. Consensus Mechanisms, in which the majority of nodes are honest, would prevent any other node in the network from changing the data within previous blocks. If the majority of the network is controlled by one central authority, it will be possible for this entity to make changes to the blocks without anyone being able to prevent it. If that happens, the network is not secure nor tamper-proof. Meaning, information on the network may be modified. The hype about public blockchains is that its mechanism design prevents exactly such situations from happening.

This write-up will refer to the blockchain that deals with the main dependencies between a network of nodes, such as the Ethereum blockchain, as the main chain. All transactions are submitted to the main chain. The nodes on the main chain are responsible to process, verify, and append new blocks to the existing chain. As mentioned, if the full nodes and miners, who do the work on the blockchain, are malicious, then the information on the ledger may be tampered with. Therefore, it is important to have as many honest nodes/miners in the network as possible. The more decentralised the network is, the more nodes may aim to participate in the consensus protocol, and the more likely it may be that the majority of nodes within any given consensus round are honest.

Assuming that we have a network of 50.000 nodes, of which 5000 (10%) are malicious. Each consensus round will draw 1021 nodes at random. In this case scenario, it is unlikely that all nodes, who have been picked are malicious. In contrast, if the network would have fewer nodes and pick fewer verifiers but the same number of malicious nodes, it would be more likely that the majority of nodes are malicious. Summarising, the security of the network depends on its design and the number of nodes securing it. If we split up the number of nodes within one network across multiple networks, then the security on the individual networks will be less than on the original network. This is one of the issues that scaling solutions are concerned with.

Layer 1

Layer 1 scaling solutions refer to any solution that may make the main chain more scalable without separating individual nodes from the main chain. Note that the chain may still be split up into multiple sections but nodes can change between the different sections and are not dependent on the security within an individual section. This should not be confused with a

network partition. These summaries refer to network partitions as involuntary segmentation of the main chain into smaller sections. Each section may believe that it is operating with the same level of security and consistency as before the network split, without realising that a partition has happened. In contrast, the purposeful separation of the main chain into individual segments may still allow nodes to switch between and communicate with other sections. The benefit of Layer 1 solutions is that the security of the main chain is still preserved. Either the main chain remains as a whole and benefits from technical solutions that provide higher scalability, or the main chain is segmented according to predefined mechanisms, which generally allow nodes to communicate with other segments. No individual section of the chain should operate in complete isolation from the other chains. The goal of Layer 1 scaling solutions is to allow the network to distribute the processing of transactions between several groups of nodes. Resulting, only a subsection of the network has to process a subset of the transactions. In comparison to the entire network verifying all transactions.

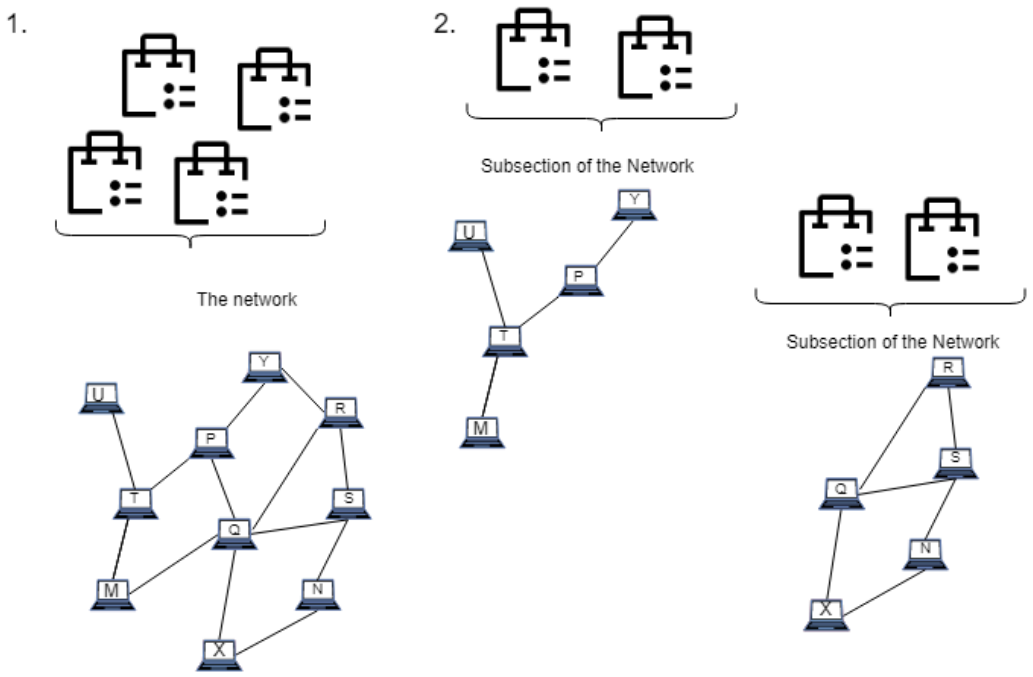

Figure 1 Transactions have to be processed by all nodes in the network (1) vs. a subsection of nodes processing a subsection of transactions in the network (2)

Layer 2

In contrast to Layer 1 scaling solutions, Layer 2 solutions allow individual nodes to become separated from the main chain. This usually happens for a predefined duration of time or until a specific condition is met. Highly abstracted, a group of nodes decide to transfer value between each other for a certain time or until predefined conditions are met, leave the main chain into a separate chain or transaction channel, transfer value (e.g. tokens) between each other, and enter back into the main chain. Once the group of nodes reenters the main chain, a smart contract will verify that all conditions are met and that the information provided is accurate.

Within the time period, during which nodes are not on the main chain, they do not have access to the security of the main chain. Therefore, the security has to be provided through the mechanism design of the Layer 2 solution. The purpose of Layer 2 scaling solutions is to take transactions away from the main chain and only append a final state/balance of the individual nodes to the main chain. Resulting, the main chain does not have to verify individual transactions that lead to the desired end-state between a group of nodes but solely the end-state.

The sections below will discuss specific designs of Layer 1 and Layer 2 scaling solutions.

Sharding (Layer 1)

Background

The term sharding is taken from computer science and database design. Generally, databases can be categorised into relational and non-relational databases. While non-relational databases do not follow clear patterns that allow for the automatic processing of data, relational databases do. Each entry into the database will have a clear identifier that allows for the retrieval of the data. Imagine a relational database with hundreds of rows and columns. Each row will have a different set of data linked to a unique identifier. However, a machine will not be able to filter out the desired row(s) without iterating through the entire database. This iteration may be highly time consuming, depending on the size of the database and the complexity of the data that is stored and processed within.

To make the filtering and processing of data more efficient, the data may be split up into several chunks. For example, if a database stores all first names in alphabetical order, then it may be beneficial to split it up into several groups. The first one may have all names that start with the letter A to I, the second one from J to P and the last one from Q to Z. Depending on the letter of the first name, the machine will have to look up the information within a different section of the database, which reduces the number of rows it will have to filter through.

Sharding

A similar design is used in the sharding of replicated ledgers. However, in that case, sharding is much more difficult to implement since the data is not stored on one server but replicated across multiple machines. Resulting, it is more difficult to keep the ledger in synchrony. Another major difference is that each shard usually does not store different information. For example, one shard will not only process utility tokens, while another shard is specialised on processing security tokens. Instead, each shard represents a separate blockchain that will process whatever transactions are received and aligned with its requirements. Usually, all shards will also depend on the same design properties and run the same consensus protocol. This allows for higher consistency across shards.

Summarising, a blockchain is one main chain that processes all transactions. Thus, the main chain is a central point of failure and throughput blockage. To allow for more efficient transaction processing, the main chain is either replaced or complemented by multiple separate chains. Each chain has to process only a subsection of all incoming transactions. All nodes on the main chain will be shuffled and randomly allocated across shards. This way,

a malicious node has it more difficult to coordinate an attack with other malicious nodes across the network since it does not know which chain it will be assigned to. Note that the design of the shards varies between blockchains. Some blockchains may not choose to shuffle nodes and randomly assign new nodes to separate shards. Those nodes will then remain on their assigned shard until there are reasons to reallocate nodes to another shard.

All chains may communicate with each other if needed. The latter property is required to process cross-shard transactions. Assuming that Alice is on chain 1 and Bob is on chain 4. Alice wants to transfer 0.2 eth to Bob and submits a transaction to her chain. The miner/validator nodes on Alice's chain have to first verify that Alice has enough funds to pay transaction fees and send 0.2 eth to Bob. Along with a proof that Alice has all funds available, they will send the transaction to the validators on Bob's chain. The validators on Bob's chain trust the proof provided from the validators on Alice's chain and submit the 0.2 eth into Bob's address. This is a highly simplistic description of the processes that actually happen and the mechanisms cross-shard communication may depend on.

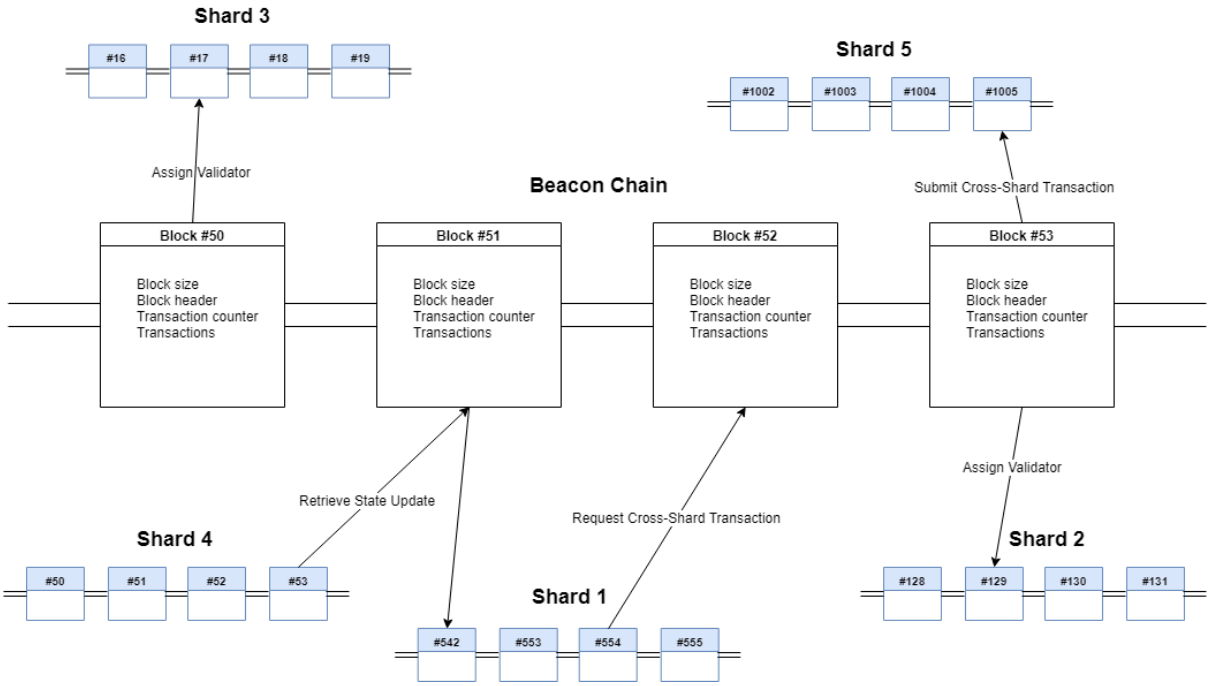

Figure 2 Simplified sharded ledger with beacon chain

Furthermore, the following paper outlines how cross-shard validation may happen. Special considerations also have to be taken for light nodes using the ledger. Generally, if the ledger runs on one main chain it can provide additional security assurance to light nodes. In the case of sharded blockchains, the security given by the network size is missing and light nodes may even end up on malicious shards. This paper outlines how fraud proofs can help to provide additional security assurance.

Several sharding designs implement a beacon chain to do certain tasks in the network; for instance, to provide additional security, retrieve and save snapshots and foster cross-shard communication. Snapshots provide the hash of the latest state of e.g. the shard. A beacon chain is a separate blockchain that connects to all shards. Nodes may enter the network

through the beacon chain and then either stay as validators on the beacon chain or become assigned to particular shards (either as users and/or validators). The beacon chain may be responsible to assign and reshuffle nodes across shards, whereby it may create additional shards or reduce the number of shards available, depending on the number of nodes on the network. Thus, the beacon chain is connected to all shards, which allows shards to communicate with each other. In this case, the beacon chain is responsible to route messages between shards. The downside of such dependency between shards and the beacon chain is that the beacon chain might turn into the bottleneck of the system. In case the beacon chain fails, all shards may also be compromised.

The implications of an individual chain in a sharded system may have on the rest of the system is highly design dependent. There are several projects that work on a sharding-based scalability solution, including:

- Near Protocol
- Ethereum 2.0
- Zilliqa

Side Chain/Subchain (Layer 2)

After understanding the principles of sharded ledgers, side chains, also called subchains, should be quite easy to grasp. However, side chains should not be confused with sharded ledgers. While a sharded ledger may not have a main chain or a beacon chain that shards depend on, side chains are blockchains that are still dependent on the main chain. Overall, the ledger may have a main chain, which processes all transactions like a normal chain but then has several chains running in parallel. The main chain would not necessarily be responsible for processing any transactions from the side chains. However, the side chain may wish to submit ‘snapshots’ to the main chain.

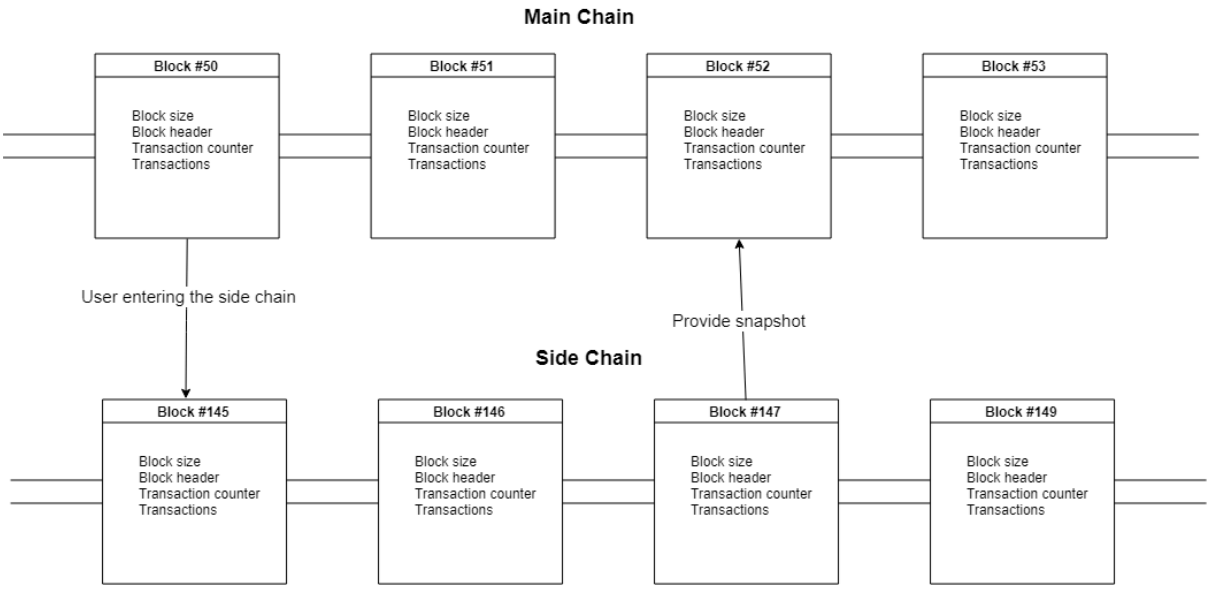

Figure 3 Simple representation of a side chain

The snapshot will be submitted by the side chain to the main chain. In case the side chain becomes compromised, it will be easier to recover from an attack. Additionally, other

applications on the main chain may be able to access information on the side chain through the snapshot. Note that the main chain should remain completely unaffected by any applications that run on the side chain. However, if the main chain becomes compromised, so might the side chain.

Applications that run on the main chain are dependent on the main chain's transaction throughput and latency. Thus, if an application requires its users to submit multiple transactions to the main chain, the main chain might not be capable to process those within a sufficient time frame. In the case of decentralised gaming, the user cannot wait several minutes until the ledger registered the newly acquired weapon. Thus, sidechains may be designed according to the needs and requirements of the application running on top. Depending on the design of the side chain, its dependency on the main chain, the type of application utilizing the main chain and the number of nodes verifying transactions, its level of security will vary.

Plasma (Layer 2)

Scaling solutions foster from quite similar ideas. One of those is the assumption that if the main chain cannot be made more scalable without compromising on decentralisation and security, it might be possible to take several chains, which each process only a limited number of transactions, and operate them in parallel. Plasma is based on similar ideas as it can be viewed as a system of ordered side chains. Generally, there are two different kind of Plasma designs, Plasma and Plasma Cash. Both are quite similar with some twerks. Therefore, Plasma will be described first.

Plasma is built on a Plasma smart contract that runs on the main chain. A Plasma smart contract provides the access to a side chain, called the child chain. Each child chain may reference additional child chains within. The plasma child chains are dependent recursively on the child chain that is closer to the main chain. The information on the child chains are periodically summarised into a Merkle root and posted on the main chain by the Plasma operator. The Plasma operator may be an elected node, or the node that first initiated the Plasma chain through the smart contract. The Merkle root that is published onto the main chain is the hash of the current state of the Plasma chain and all transactions within. This can be imagined similar to a court system. Each country might have a supreme court that is responsible for highly important matters. For all smaller disputes and less impactful decisions, each state, city and town may have their own legal institutions. Those will only be able to make decisions that affect the area that is under the subject of the regional legislations. However, the supreme court can handle decisions that affect all other institution. In case of a dispute between participants on the Plasma chain, they may rely on the security of the main chain and the latest state update of the Plasma chain to resolve the dispute. Each child chain is responsible for the state and the child chains recorded within.

When nodes want to enter into the Plasma smart contract, they have to deposit the amount of funds they wish to use on the Plasma child chain. Nodes may not transfer more funds on the child chain than they have locked up in escrow on the main chain. Those funds are then locked up in the smart contract on the main chain and identical copies of the tokens are created on the Plasma child chain. In the case of Ethereum, the main chain transacts Ether

(eth), those are deposited by a user into the Plasma smart contract, and the same amount of eth are created as eth' on the Plasma child chain. While the Ethereum main chain runs on the balance/account based model, Plasma chains run on the UTXO based model (similar to Bitcoin). This allows the Plasma chain to keep track of several separate outputs per user. The user would be able to transact different UTXOs with various users, while the system (the child chain) does not have to worry about the overall balance of the user.

A user/node may stay on the child chain until a specific condition is met or the user initiates a Plasma exit to the main chain. By existing the Plasma chain, the nodes will have to submit the Merkle proof of the current state of its tokens. This is followed by a challenging period, within which other users may challenge the state of the node that wishes to exit the Plasma chain. If no other user challenges the state of the exiting node, it will be able to enter back onto the main chain and withdraw funds from the Plasma smart contract in accordance to the funds recorded in its last state. Thus, if Bob exits the child chain with 1 eth' he received from Alice but did not make any transactions himself, then he will be able to withdraw the funds that he locked up in the plasma smart contract and 1 eth from the funds that Alice locked up prior to her entering the Plasma chain.

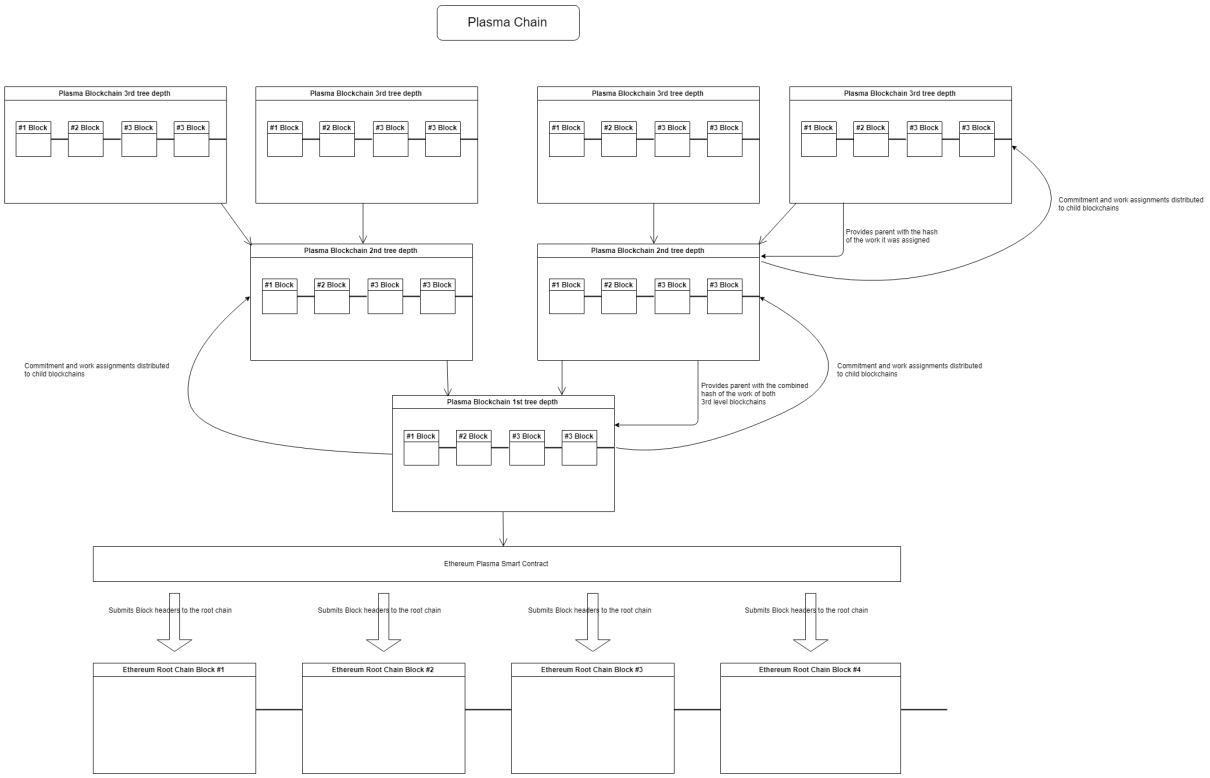

Figure 4 An overview of a main chain and several Plasma child chains.

Plasma cash is a bit different to Plasma in that it utilises non-fungible tokens. Each token has a unique token ID. Depending on the position of the token in the network, the token ID will vary. This has been done to limit the size of the Plasma chain that the user has to download. With Plasma cash, users may only access parts of the Plasma chain, which they will use and interact with. A Sparse-Merkle tree is an ordered Merkle tree that checks the inclusion of transactions through Merkle proofs. You can read further into Plasma Cash here and Plasma. The main drawback of Plasma Cash is that if Alice transfers a token to Bob, she will not only have to transfer the ID of the token but also its entire history. If the token

was first registered by Charlie 15 blocks ago, then Alice will have to provide the entire history of the token along with it. Otherwise, the token may not be valid. While this data would still be less than storing an entire blockchain, it may grow over time, depending on the lifespan of the tokens on the child chain.

Payment Channel (Layer 2)

State Channels

State Channels are off chain Layer 2 scaling solutions for public ledgers such as Ethereum. While other scaling solutions such as Sharding and Plasma allow multiple parties to enter onto a different chain, State Channels are designed to accommodate an exchange of value between a few participants. State Channels are not separate blockchains such as described earlier. Instead, a predefined number of participants enter into the payment channel through a smart contract on the main chain. Similar to the Plasma design, each participant has to lock up their funds in accordance to the amount they would like to transact on the State Channel with other participants. Once this is done, they are directly connected to each other and can transfer funds.

Raiden Network

A Raiden Network is similar to a state channel. The difference is that a state channel does not only allow participants to transfer funds between each other but also to run decentralised Applications (dApps). In contrast, Raiden Networks are payment channels that connect multiple nodes with each other. Such as in the graphics provided below.

An issue that both Raiden Networks and State Channels face is that if Bob goes offline, he cannot check whether or not Alice and Charlie keep behaving honestly. In case either Alice or Charlie want to leave the payment channel, while Bob is online, but submit the wrong state to the smart contract on the main chain, Bob will have a chance to dispute the information provided either by Alice or Charlie by submitting his latest state. Resulting, a malicious node will not be able to steal funds from the payment channel as long as all nodes are online and check each other. However, if Alice transfers 5 vETH to Bob, Bob goes offline, and Alice initiates a channel exit with the wrong state, she might be able to get through with it. Instead of submitting her current state of 3 vETH, which Alice received from Charlie, Alice claims to still have 5vETH. Even though Charlie is online and sees what Alice is doing, he might not even care about Alice submitting the wrong state since that means that he gets to keep his 3vETH that he transferred to Alice beforehand. Bob is offline, and the smart contract on the main chain does not know the latest state of the network.

One of the proposed solutions to this problem are watchtowers. Watchtowers are smart contracts that run on the payment channel and keep track of the latest state of the transactions. Each transaction will receive an ID to allow for the sequential ordering of transactions. With each new transaction the ID number increases. Once a new transaction is submitted, the watchtower will replace its latest state with the new state. Resulting, when a node submits its current state to the watchtower, the watchtower will compare the provided state with the latest registered state. If the state IDs match, the node is allowed to exit the payment channel. However, if the IDs differ, the state provided by the node is wrong and the node is not allowed to withdraw its funds from the smart contract on the main chain.

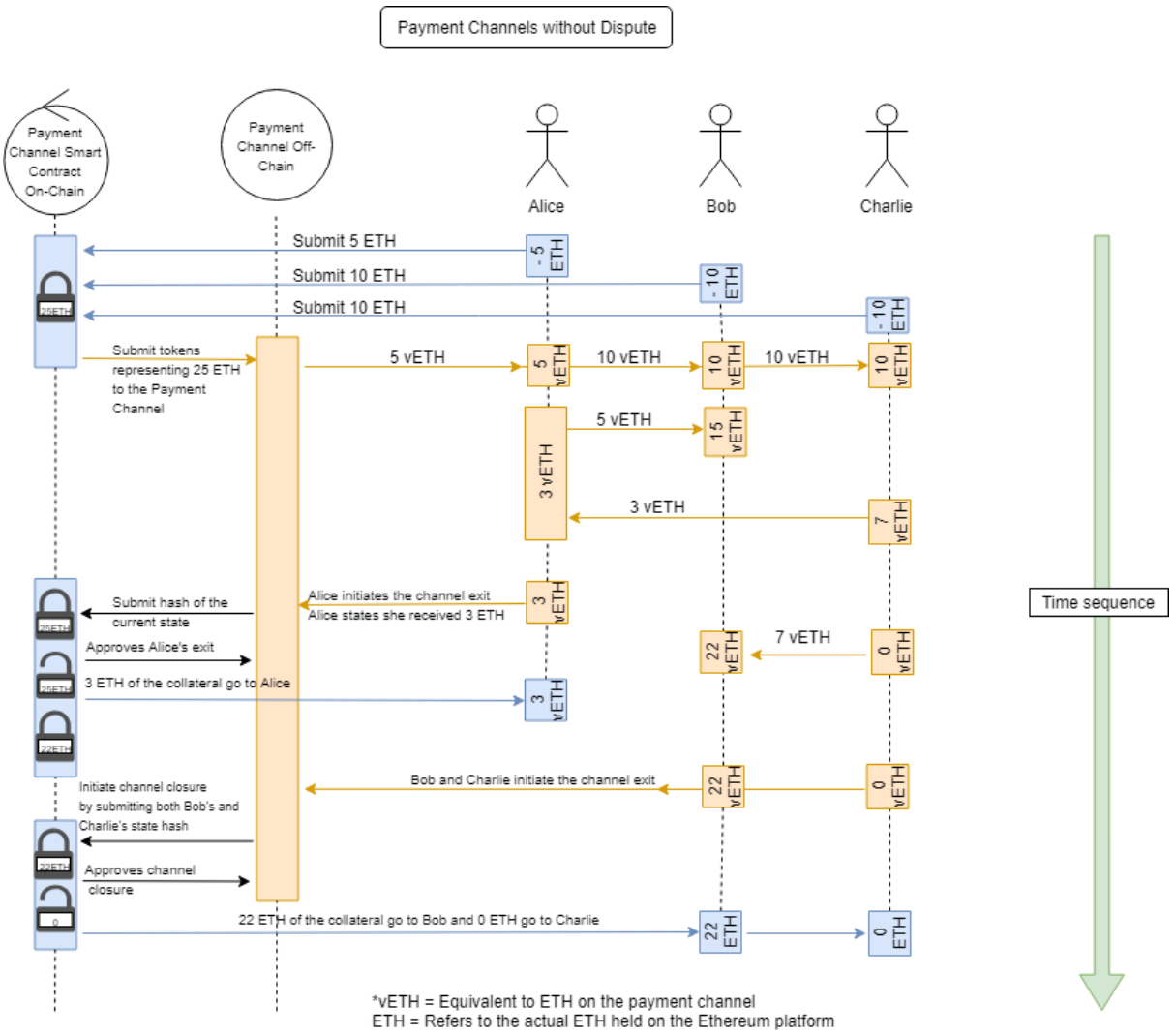

Figure 5 Payment channel

You can learn more about watchtowers here:

- [Pisa: Arbitration Outsourcing for State Channels]
- [The solution by Celer Network]
- [Towards Secure and Efficient Payment Channels]

Main Points

- Replicated ledgers have to be linear scalable to provide for high transaction throughput. Currently, most public blockchains are only able to process a dozen transactions per second.
- The network throughput may reference the number of transactions, which the network can process within a specific time period. The latency refers to the time required to process individual transactions.
- The effect of the throughout and latency of a system is design dependent. Systems aim to optimise for high throughput and low latency.
- The design of public ledgers are intended to prevent a central authority or malicious node from tampering with previous blocks.

- The security and decentralisation of a public ledger depends to some extent on the number of verifier nodes that participate in its consensus protocol.
- While Layer 1 scaling solutions aim to preserve the security provided by the main chain by keeping the network together as much as possible, Layer 2 scaling solutions foster direct connections between peers in a network to take transaction off chain until a certain condition is met.
- Sharding allows the ledger to operate several chains that are interconnected to process and append transactions. Many sharded ledgers depend on a beacon chain, which is responsible to provide cross-shard communication, assign validators and retrieve snapshots from shards.
- While a sharded ledger may not have a main chain or a beacon chain that shards depend on, side chains are blockchains that are still dependent on the main chain. Overall, the ledger may have a main chain, which processes all transactions like a normal chain but then has several chains running in parallel.
- Plasma is built on a Plasma smart contract that runs on the main chain. A Plasma smart contract provides the access to a side chain, called the child chain. Each child chain may reference additional child chains within.
- State Channels are not separate blockchains such as described earlier. Instead, a predefined number of participants enter into the payment channel through a smart contract on the main chain. It does also allow participants to interact with each other on dApps.
- Raiden Networks are payment channels that connect multiple nodes with each other to transact value.