aelf - A Multi-Chain Parallel Computing Blockchain Framework

V 1.7.0

October 25th, 2022

Abstract

Over the past few years, The Blockchain community has seen rapid development. Since Satoshi's Bitcoin emerged as secured, decentralized Peer-to-Peer transfer mechanism, the concept of a decentralized crypto-currency became mainstream and changed the world of finance forever. Ethereum then expanded on that idea with the successful implementation of versatile "Smart Contracts," unleashing the potential of Blockchain into numerous applications and industries. As a result, many alternative crypto assets were built upon these Blockchains. But the boundary between the Blockchain community and the business world has yet to be broken. We believe we have reached a turning point in Blockchain, with the next phase leading to the integration of Blockchain and the physical business world, and inevitably bringing in solid digital assets.

In order to enter the new paradigm of Blockchain, there needs to be a versatile Operating System designed to meet commercial needs. This Chain must address three main challenges:

1. Current Blockchains are not scalable, as the performance of one single node/mining machine determines the performance of the whole system.
2. Current Blockchains do not segregate resources for different Smart Contracts, which causes to interference between Smart Contract executions.
3. Current Blockchains do not have pre-defined Consensus Protocol to adopt updates or adapt to new technology.

This white paper introduces a highly efficient Blockchain architecture that incorporates State-of-the Art IT design principles and technologies to bring Blockchain up to a commercial standard. We envision it creates a "Linux ecosystem" for Blockchain. We focus on defining and providing the most basic, essential and time-consuming to develop components of the system and on making significant improvements for existing Chains in the market. The system allows developers to customize Chains to meet their own needs, particularly commercial requirements for various industries. It will contain the following main features:

1. MainChain and multi-layer SideChains to handle various commercial scenarios. One chain is designed for one use case, distributing different tasks on multiple chains and improving processing efficiency
2. Communication with external Blockchain systems, such as Bitcoin and Ethereum, via messaging
3. Parallel processing for non-competing transactions and cloud-based service
4. Basic components of minimum viable Block and Genesis Smart Contract Collection for each Chain to reduce data complexity and achieve high customization
5. Permission for stakeholders to approve amendments to the protocol, including redefining the Consensus Protocol; Permission for SideChains to join or exit from MainChain dynamically based on Consensus Protocol, and therefore introduce competition and incentive to improve each SideChain

CONTENT

1. Current Blockchain Systems

The Blockchain technology and its applications are developing exponentially. Many industries are migrating from traditional network architecture to a Blockchain-based network architecture. However, current Blockchain systems are not yet capable or efficient enough to function as a versatile operating system and support multiple applications. Bitcoin, the pioneering Blockchain design, is more similar to an application. Ethereum has demonstrated some characteristics of an Operating System – developers can program applications such as Smart Contracts, and the Chain provides programming language and Adaptor in the form of Solidity. However, from the perspective of a modern Operating System, Ethereum still has several drawbacks, such as the lack of decoupling between system components, the lack of customization of most modules, and insufficient system interfaces, among others.

This approach lacks the holistic design of aelf and is not yet commercially viable for cross-industry application scenarios. It greatly limits the commercial application of Blockchain technology.

1.1. General Blockchain vs. Complex Business Scenarios

The largest challenge facing commercial scale adoption of Blockchain technology is its current inability to meet the requirements of multiple, diverse, and complex business scenarios. Naturally, different scenarios often have different characteristics in terms of process and execution logic, and thus require distinct solutions. Therefore, the "one size fits all" Blockchain approach that is currently utilized by other chains is not viable if Blockchain is to succeed in the future. For example, ticket issuance requires high frequency where a high transaction rate in the system is desirable, while digital legal contracts emphasize high security and reliability over speed. It simply does not make sense for them both to be built around the same Chain.

There are two general solutions to this problem:

i. Use Blockchain as solely a database not deal with business logic. This approach aims to handle any business scenario and maintain compatibility. Many Chains similar to Bitcoin use this approach. They record business-related data and hash into a transaction output "OP_RETURN", which is stored in the Blockchain.

ii. Record various complex Smart Contracts onto one single Blockchain. These Smart Contracts serve pre-defined business models from various scenarios. Ethereum represents this type of Chain. Due to the fact that all Smart Contracts are written on one single Chain, the Blockchain becomes overly complex, requires a high maintenance cost, and lacks an effective structure to execute smart contracts.

1.2. Performance Limitation of Sequential Processing

As Blockchain becomes more and more widely used, especially for handling large scale transactions, its transaction processing capacity faces tremendous pressure when using sequential processing, which results in the bottleneck of network performance. Current Blockchain systems face multiple challenges to improving capacity, sometimes at the expense of transaction efficiency. For example, the Bitcoin transaction fee is getting more expensive as transaction volume increases and a large

backlog waits for confirmation. Ethereum faces an increasing number of congestions during token sales. However, in traditional IT architecture, modern techniques such as partitioning, sharding and decentralized architecture have proven highly effective at improving system performance.

On the other hand, the concept of parallel task processing has not been adopted to increase efficiency. When a Block contains large amount of transaction data and complex Smart Contracts, sequential transaction has hit its efficiency limitation of Block formation and verification.

1.3. Data Complexity and Redundancy

As described in Section 1.1, one universal Blockchain is used to meet the needs of different business scenarios. The drawbacks of a universal Blockchain system are overly-complex Smart Contracts and Consensus Protocols, lack of tailored solutions to specific business scenarios, and redundant data.

1.4. Dilemma of Protocol Update

Despite the increasing adoption of Blockchain, it is still in its nascent stage. Many significant improvements and innovations are yet to come. These updates are essential to evolve Blockchains and keep up with an ever-changing environment and stakeholder's interest. The large variety of stakeholders within the ecosystem usually makes it difficult reach consensus without effective governance mechanisms, leading most current Protocol updates into impasse or disputes. One vivid example is Bitcoin, as the community found it difficult to reach agreements for the introduction of many new features in recent years.

1.5. Inflation of Block

The more successful a Blockchain system is, the higher its maintenance cost. Running through a full Current Bitcoin node requires over 130G of space, and Ethereum requires over 180G. This situation will not be improved in the future. As more users adopt Blockchain and conduct more transaction activities, the inflation of Blocks will accelerate and the maintenance cost will grow even higher. Actions must be taken to alleviate this vicious cycle.

1.6. Inefficient Point-to-Point Communication Support

Existing Blockchains mainly communicate through on a broadcast network where the support for P2P communication is inefficient and insecure. One example is that if certain data only concerns a single group of users, that data should be communicated among finite nodes, not broadcasted to all nodes.

1.7. Pending Breakthrough for Cross-Chain Communication

Existing Blockchain systems have experimented with cross-chain communication to process related business logics. However, the outcomes have been unsatisfactory. Current cross-chain communication includes the centralized mechanism and the HTLC mechanism. The centralized mechanism deviates from the idea of Blockchain, and leads to lack of trust, single node failure, single node bottleneck, and is only applicable to certain scenarios. The HTLC mechanism can also only deal with specific

scenarios, such as asset exchange, and imposes strict requirements on the protocols and Consensus Protocols of communicating chains. And implementation of such a mechanism is usually complex. As a result, it is imperative to address two critical issues: Protocol compatibility and data exchanging format compatibility.

2. Key objectives of aelf

2.1. A Highly Customizable OS for Commercial Use

We envision aelf as a highly efficient and customizable OS that will become the "Linux system" for Blockchain. Take Linux as an example–Linux Kernel and various Linux versions constitute the large and successful Linux family. Linux Kernel resolves the most fundamental, critical and time-consuming parts, allowing other developers to make customized systems based on application scenarios and customer needs. This makes Linux the most popular server OS, and it supports all kinds of industries.

This same idea has been incorporated into the aelf design. First, we define and implement the aelf Kernel which includes fundamental functions of a Blockchain system, namely the minimum viable Blockchain system. Then, we develop a "shell" as the basic interactive interface for the Core. Users can either use the complete Blockchain OS, or rapidly develop a customized OS based on the Core by redefining the Core through interfaces.

2.2. Cross-Chain Interaction

aelf can interact with Bitcoin, Ethereum, and other Blockchain systems. Cross-chain interaction with mainstream Chains can be realized through messaging. It will also form an endogenous multi-level cross-chain structure based on cross-chain interaction, in order to share digital assets, users and information.

2.3. Performance Improvement

In traditional IT architecture, distributed structure is the most popular solution to debottleneck capability limitation. Blockchain systems should also support distributed parallel processing, the ability to process multiple transactions with non-competing data to improve transaction efficiency. In addition, when one chain has become too complex to be effectively processed, it should be split into parallel Chains to offload the traffic.

The initial design of an effective Blockchain should focus on solving specific business scenarios, not combining all Smart Contracts onto one single Chain. In order to deliver optimal performance based on business requirements, the Chain has to provide an effective and customized data structure, Smart Contract logic, and a Consensus Protocol can be designed specifically for the targeted objective. By doing so, the components and data within the Chain will be much simpler and easier to manage.

In addition, aelf can define a mechanism to trigger a snapshot in the system. Upon defining a cycle, it takes a snapshot of current data and trims detailed transaction data. A new Genesis Block will include all subsequent transactions. This idea has been adopted in traditional IT database architecture to alleviate system inflation.

2.4. Protocol Update

Upon the Genesis of Blockchain, the voting and update mechanism needs to be clearly defined. With introduction of the Consensus Protocol to include new features in the future, it avoids impasse and dispute over Protocol update.

2.5. Private Chain Module

A Considerable number of businesses are interested in Private Chains to leverage the advantage of Blockchain technology. These private Chains usually exist in isolation without any connection to external ecosystem or other businesses. We provide a model similar to Amazon cloud service, "AMI", where users can rapidly create an independent Chain using our Private Chain module and obtain full ownership of it.

3. Core Approaches to Realize aelf System

3.1. Performance Enhancements

The core principle of aelf is to resolve practical technical problems using solutions that have already been tested. Instead of “optimizing” the concepts of Blockchain, more attention is paid to provide a mature configuration for the stable execution of business applications.

A few performance enhancement ideas are currently being explored:

Most Blockchain sharding solutions are implemented by dividing a single consensus into numerous sub-consensus. Basically, the consensus as a whole is split, leaving several sub-consensus groups which are easier to attack than a single one. People can increase randomness to complicate the routing path, but this will impair the specialization of the production node.

PoW production nodes decrease substantially as more mining pools replace them as a specialized ledger system. These pools are able to ensure mining efficiency and timely broadcasting, slowing down the speed of block formation and keeping it steady. By leveraging experiences from the IT industry, mining pools have abandoned the standard official node software but aggregate computing power through load balancing and run smart contracts in an asynchronous parallel manner, putting their own nodes globally to improve the broadcasting efficiency. However, the performance of mining pools is still limited by the technical differences used in the pool, and by the fact that nodes are all designed equally and limited by the protocol itself. So, the upgrading of one single node does not lead to the improvement of the whole network.

aelf’s logic is that nodes are categorized according to their roles; those who provide standard services on clusters are open-sourced and work through DPoS to reach consensus of the MainChain. The delegated production nodes are able to protect SideChains and share the strong consensus of the MainChain. This method increases the pressure for each delegate, however, efficiency will improve as more SideChains are added, because delegated production nodes are capable of running in clusters. SideChains are independent of each other, thus one additional SideChain will increase the efficiency of the whole system. Moreover, the efficiency of each SideChain will also benefits from parallel processing.

3.2. Resource Segregation

To protect Smart Contracts from unnecessary mutual interference and maintain their stable running on Blockchain, aelf abandons the one-chain-fits-all solution and design a public Blockchain that is able to ensure proper running of each contract.

aelf will be a cloud-computing platform similar to AWS. No business would like to be disturbed by other businesses. For example, trades in a future market should not be interfered by the traffic generated by the Black Friday. However, this seemingly impossible interference is commonly seen in the domain of Blockchain. So the key obstacle preventing Blockchain technology being applied in real cases lies in its initial design.

3.3. Structure of Governance

Due to historical limitations, the current Blockchain governance structure is often not well defined upon creating. This problem becomes more prominent when there is a major functional upgrading or a stagnation caused by bugs. For example, Bitcoin has been stuck in scaling problems for more than two years and finally forked, and the differences on The DAO incidence between the Ethereum community and the foundation led to the birth of ETC. As a result, we clarify the method of aelf direction-setting before users entering the world of aelf:

We acknowledge the fact that aelf Token holders have the greatest right in the future of aelf, and token holders' interests are linked with the destiny of aelf, in particular those with long-term locked-in tokens.

3.3.1. Resemblance of Representative Democracy

One of the key principles of aelf is designating specialized nodes to perform specialized tasks. In aelf, vital decisions are carried out through a mechanism that resembles representative democracy. Delegated nodes must have enough votes from other stakeholders to participate in aelf governance. To some extent, production nodes constitute the health of the aelf system, so these nodes are responsible for being a ledger and handing out bonuses and feedback values to the stakeholders who entrusted them through Smart Contracts.

3.3.2. Exercise of Power by the Delegates

Foundation members realize their governance by submitting the source code and delegating production nodes to review and vote. The process goes as follows: Foundation members provide open-source code and submit new features. Delegates then choose specific features to incorporate based on their needs. If one feature is adopted by enough delegates, it gains approval by the whole system.

4. aelf System

4.1. aelf Architecture

We introduce how aelf consists of one MainChain and multiple SideChains attached to the MainChain (Figure 4.1). This differs from a traditional Single Chain system in that aelf is a "branched ecosystem" where the MainChain works as the backbone of the system and connects to multiple SideChains (this can even include multiple layers).

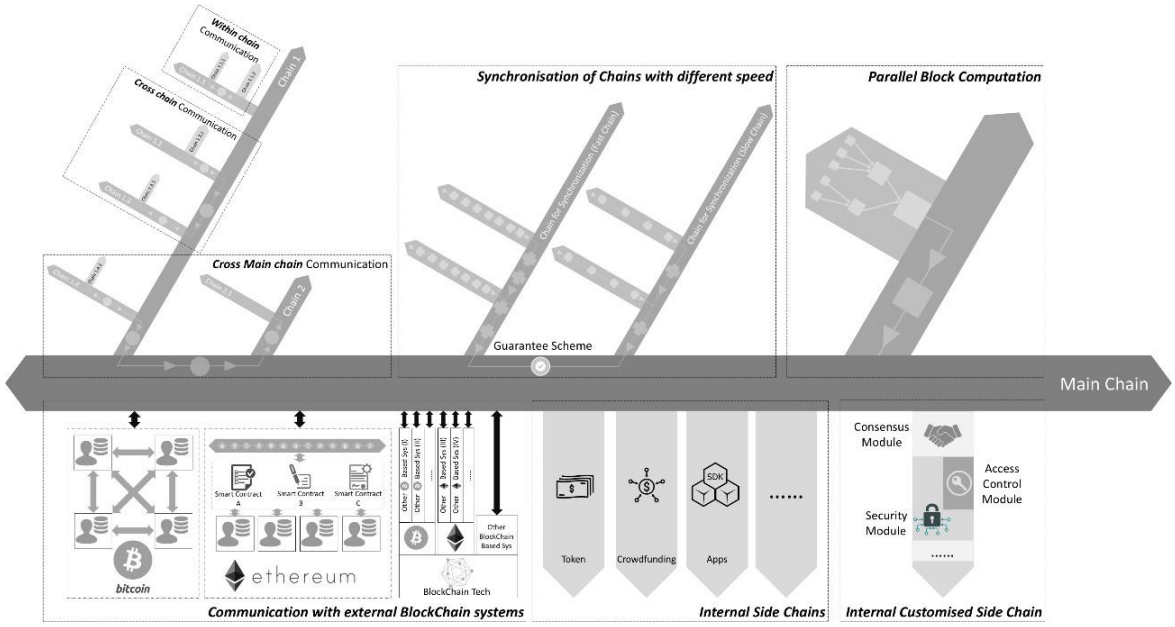
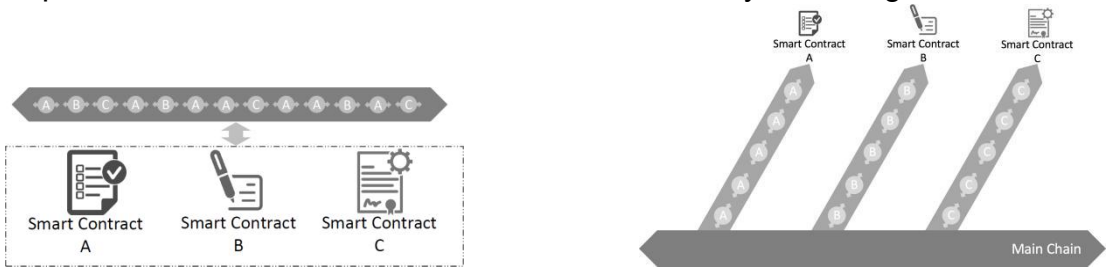


Figure 4.1: Overview of aelf Structure

aelf connects with Bitcoin, Ethereum, and other Blockchain systems via adaptor in order to be compatible with existing popular ecosystems. aelf SideChains include the System built-in aelf SideChains and other chains generated based on the aelf operating system or aelf kernel. The MainChain interacts with the SideChains by SideChain dynamic indexing.

4.1.1. One Chain One Contract

Compared with traditional structure of "one Chain to any type of Contract", aelf imposes "one Chain to one type of Contract". As illustrated in Figure 4.2 (b), each Chain is dedicated to one type of transaction and resolves one type of business problem. This makes the whole structure and data more simple and more tailored to commercial requirements. By adding new SideChains to aelf, aelf will be empowered with new functions and maintain an "easy to manage" structure.



(a) one Chain to any types of Contracts **(b) in aelf, one Chain to one type of Contract**

Figure 4.2: A Blockchain with Complex Data Structure

4.1.2. SideChain Dynamic Indexing

aelf is a dynamic system where all SideChains are attached to the MainChain. The MainChain contains the index of the system boundaries and records where the SideChains are. They interact with each other via the MainChain in the form of Merkle tree verification through external information input. As such, SideChains do not interact directly, allowing SideChains to be added to or excluded from the aelf system.

4.1.3. "Tree branch" SideChain Extension

As illustrated in Figure 4.3, aelf defines a "MainChain and SideChain structure". Theoretically speaking, any SideChain can also be connected with a few sub Chains underneath, acting as a "MainChain" in one part of the system. This creates a branch structure in the system that allows aelf to extend both horizontally and vertically. This idea is similar to that of partitioning and sharding in database architecture. It allows each shard to perform specific functions, and when one shard becomes too large to manage, it can be further broken down into multiple shards. In aelf, this corresponds to SideChains.

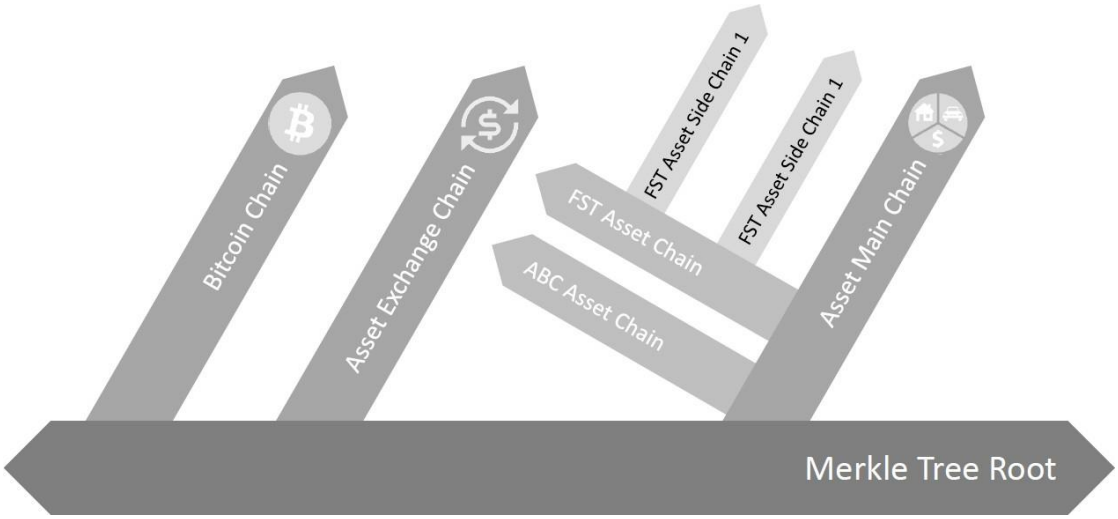


Figure 4.3: Multi-layer SideChain Structure

4.2. aelf MainChain

aelf MainChain is a Blockchain run by the aelf OS, and acts as the backbone of the whole system. The MainChain consists of a SideChain index system, Token system, and DPoS Consensus Protocol.

4.2.1. SideChain Index System

The SideChain index system connects all Chains within the aelf ecosystem. aelf indexes two types of Chains:

- External Chains of high importance which can be used to expand the boundary of aelf, e.g. Bitcoin, Ethereum
- Internal SideChains operating under aelf OS, which contribute economically to the aelf system and use the aelf Token

SideChain indexing works in following steps:

- Nodes of the MainChain read information from SideChains and form a Merkle Tree
- The header of the new Block records the Merkle Tree Root. As illustrated in Figure 4.4, if we want to confirm transaction TX1 on the 1000th Block of BTC, we only need to prove the existence of Merkle Tree for the 1000th Block of BTC as stored on the Merkle Tree Root of the MainChain, and Merkle proof of TX1 on the 1000th Block of BTC via messaging. This approach also works for other Chains such as Ethereum, as long as a Merkle Tree Root isformed

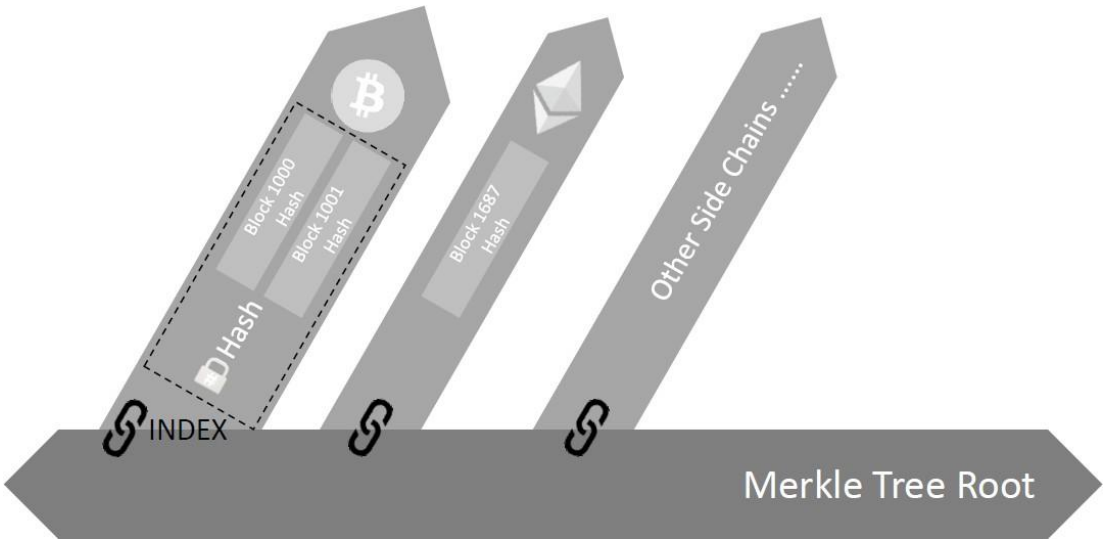


Figure 4.4: SideChain Indexing

In order to improve verification efficiency, we suggest expanding the structure of a Merkle Tree to include not only Block hashes, but the Merkle Tree Root of transactions in Figure 4.5 and their states in Figure 4.6.

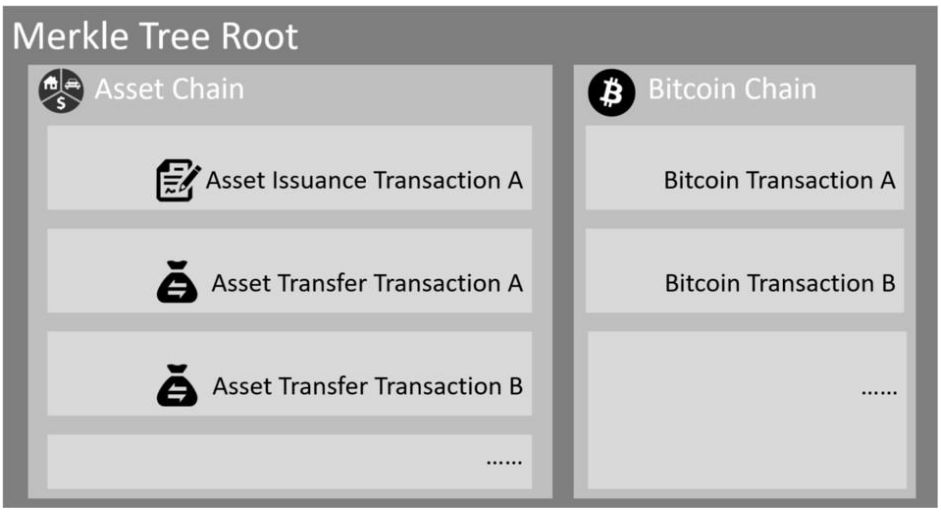


Figure 4.5: Transaction Indexing

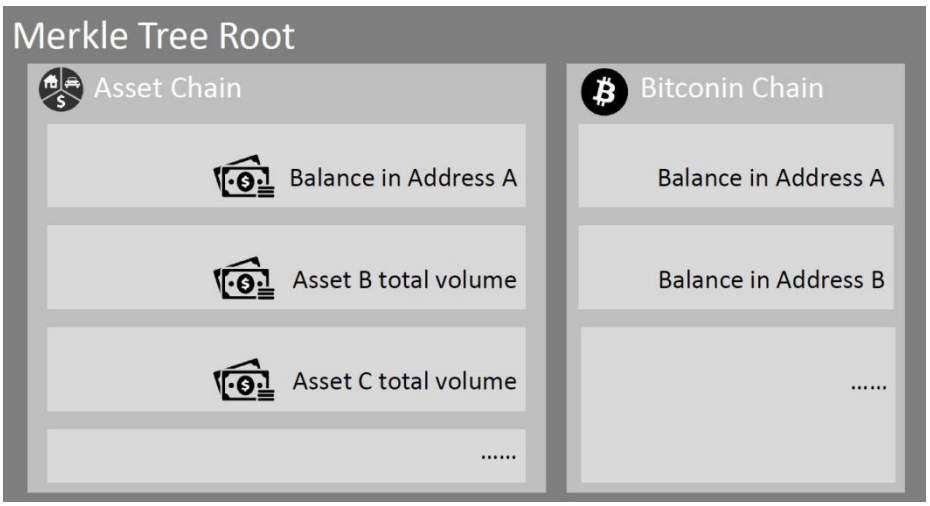


Figure 4.6: State Indexing

One key issue that arises here is the timing of SideChain indexing by the MainChain. If the MainChain frequently indexes a SideChain with a high probability to fork, it wastes efforts indexing Orphaned Blocks. Therefore, we suggest a different indexing strategy for each SideChain based on its own, unique characteristics, and this can be defined in the system. The Indexing strategy for Blockchains similar to Bitcoin can wait one minute from the time a Block is formed. This has been proven statistically, as a Block can be confirmed not to be an orphan after one minute of its formation. With aelf, if a SideChain and the MainChain adopt merge mining, real-time indexing can be conducted by the same miners.

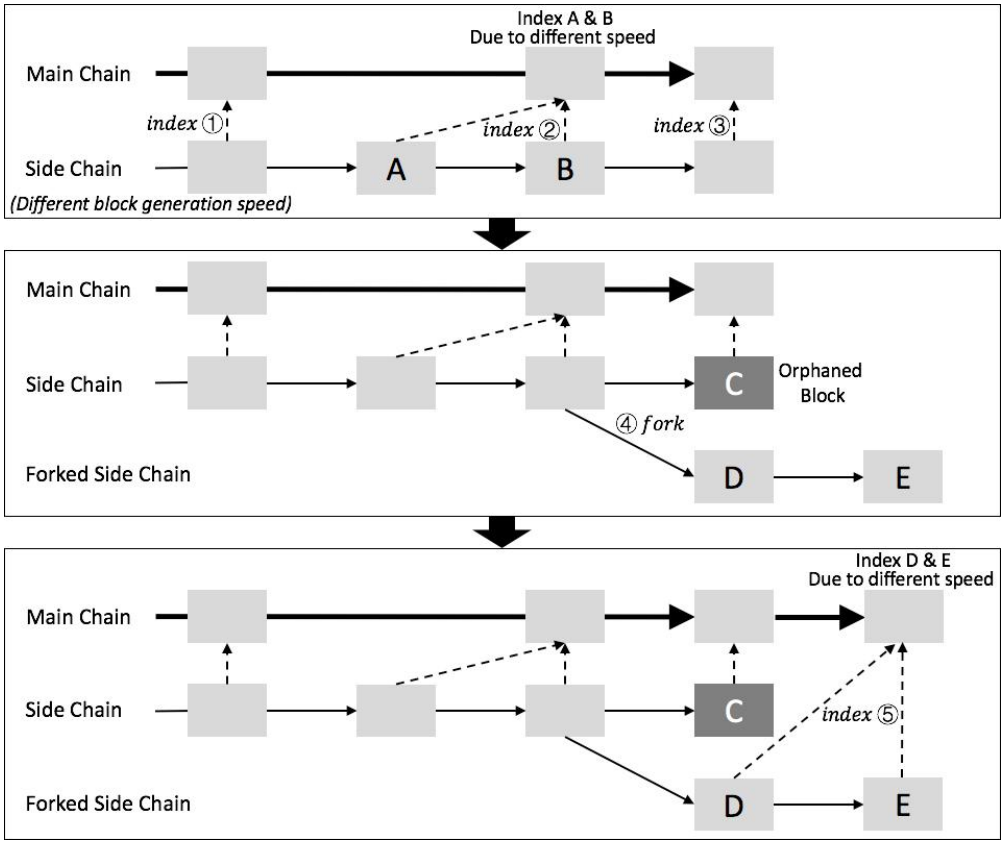


Figure 4.7 Timing of Indexing

4.2.2. aelf Token System

aelf token incentivizes honest behavior in the system. All aelf SideChains accept the aelf Token as storage of value and a means of value transfer. It can be transferred across any Chains that accept the aelf Token.

When a SideChain applies to be indexed by the MainChain, it receives some locked-in Tokens from the MainChain. When the SideChain receives transaction fees, it partially shares them with miners of the MainChain. When the MainChain finds indexing a SideChain is economically unfavorable, the MainChain has the right to terminate indexing, or permitting competition of two SideChains providing the same services.

4.2.3. Consensus Protocol

A stable and efficient Block formation mechanism is the foundation of the aelf system. The operation and maintenance of aelf is more complicated than Bitcoin and Ethereum, because aelf Block formation requires the MainChain to record information from SideChains, and aelf is designed to provide cloud-based enterprise services in a more complex structure. In addition, miners need to update information from multiple parallel Chains. The MainChain will adopt DPoS to ensure high frequency and predictability of Block formation, which will improve user experience.

4.2.4. DPoS

aelf delegates 2N+1 production nodes. N starts with 8 and increases by 1 every year. These nodes in the aelf system enforce all of consensus rules of aelf.

The purpose of these delegated production nodes is to enable transaction relay, transaction confirmation, packaging blocks and data transfer. As aelf adopts multi-SideChain architecture, production nodes have to work as miners for some SideChains.

2N+1 nodes will go through a randomized order calculation each week. The randomization process is illustrated as follows:

aelf is running along the timeline within processing units we call a “round” (horizontal arrow in Fig. 4.8 and Fig. 4.9). In a round, one production node will produce one block each time, while one node will have one extra transaction at the end of the round (vertical arrow in Fig 4.9).

Each production node ($node$) has three main properties in a specific round (t): (1) Private key, $in_{node(t)}$, which is a value inputted from the production node and kept privately by the production node itself in round t. It will become public after all block generations in round

t are completed; (2) Public key, $out_{node(t)}$, which is the hash value of $in_{node(t)}$. Every node in the aelf network can look up this value at any time; (3) Signature, $sig_{node(t)}$, which is a value generated by the production node itself in the first round. After the first round, it can only be calculated once the previous round is completed. It is used as the signature of this production node in this round and it is also opened to public at all times like the $out_{node(t)}$. See Fig 2.1 for more details.

There are two main processes in DPoS: (1) Pre-verification; and (2) order calculation within each round.

Pre-verification (Fig 4.8): before a node starts its block generation in round ($t+1$), it has to have its status verified in round (t). In round $t+1$, $in_{node(t)}$ is already published as public, and $out_{node(t)}$ can be queried at any time. So to verify the status of $node$ in round t, other nodes can check $hash(in_{node(t)}) == out_{node(t)}$.

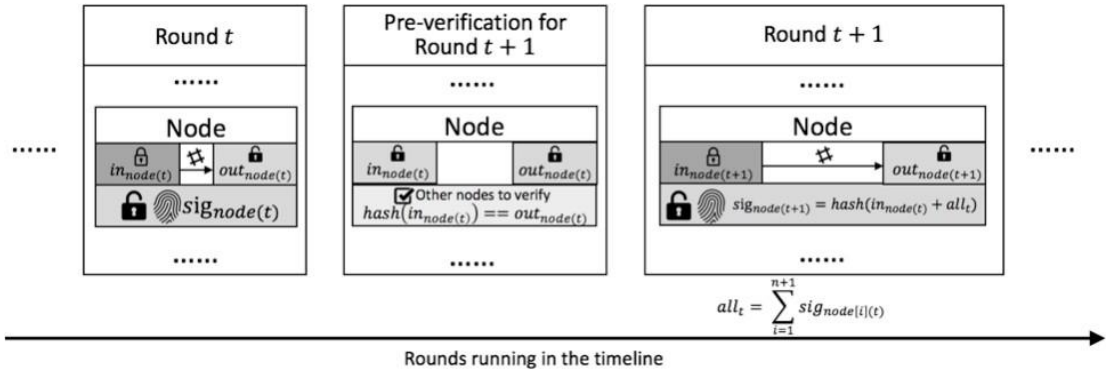


Figure 4.8 Pre-verification.

Order calculation (Fig 4.9): In Fig 4.9, we used 4 production nodes as an example to explain our order calculation strategy. In each round N, production nodes have $N+1$ block generation. In the first round (Round 1 in Fig 4.9), the ordering of block generations as well as the signature (sig) for each node are totally arbitrary. In the second round (Round 2 in Fig 4.9), the block generations are again arbitrarily ordered. However, from the second round, the signature will be calculated by

$$sig_{node(t+1)} = hash(in_{node(t)} + all_t)$$

where

$$all_t = \sum_{i=1}^{n+1} sig_{node[i](t)}$$

here, $node[i](t)$ means the node is processing the i^{th} transaction in round t.

From round 3, the ordering within a round is generated from the ordering and the node signature from the previous round. In round $t+1$, we traverse the signature of nodes at round t in order. The ordering of a node in is calculated by $t+1$

$$sig_{node(t)} mod(N) = \begin{cases} 0, & first\ place \\ 1, & second\ place \\ 2, & third\ place \\ \cdots & \\ n-1, & n^{th}\ place \end{cases}$$

For cases of conflict, i.e. results pointed to places which are not empty, we point the node to the next available place. If the node conflict is at the n^{th} place, we will find the available place from the first place.

The node that processes the one extra transaction is calculated from the signature of the node in first place of the previous round.

$$sig_{node[0](t)} mod(N) = \begin{cases} 0, & A \\ 1, & B \\ 2, & C \\ \cdots \end{cases}$$

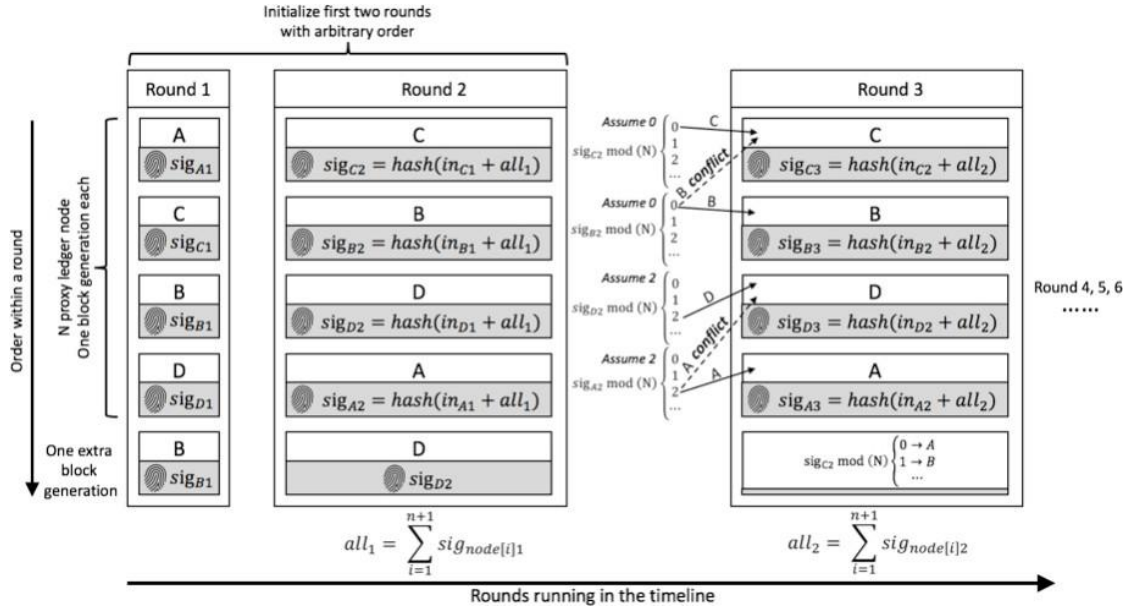


Figure 4.9 Order Calculation Details for First Three Rounds

$sig_{node[0](t)}$ is decided by: (1) all the signatures from previous round $t-1$; (1) the value in of itself in round $t-1$; (3) which node generate the extra block. So it can only be calculated after the previous round $t-1$ completed. Moreover, as it needs all the signatures from the previous round and the value in is input by each node independently, there is no way to control the ordering. The extra block generation is used to increase the randomness. In general, we create a random system that relies on extra inputs from outside. Based on the assumption that no node can know all other nodes' inputs in a specific round, no one node could control the ordering.

If one node cannot generate a block in round t, it also cannot input its in for this round. In such a case, the previous in will be used. Since all production nodes are voted to be reliable nodes, such a situation should not happen often. Even if this situation does happen, the above-mentioned strategy is more than sufficient at dealing with it.

Every node only has a certain time T seconds to process transactions. Under the present network condition, T=4 is a reasonable time consideration, meaning that every node only has 4 seconds to process transactions and submit the result to the network. Any delegate who fails to submit within 4 seconds is considered to be abandoning the block. If a delegate failed two times consecutively, there will be a window period calculated as W hours ($W=2^N$, N stands for the number of failure) for that node.

In the systematic design, aelf defines that only one node generates blocks within a certain period. Therefore, it is unlikely for a fork to happen in an environment where production nodes are working under good connectivity. If multiple orphan node groups occur due to network problems, the system will adopt the longest chain since that is

the chain that most likely comes from the orphan node group with largest number of production nodes. If a vicious node mines in two forked Blockchains simultaneously to attack the network, that node would be voted out of the entire network.

DPoS production nodes are elected in a way that resembles representative democracy. The elected nodes decide how to hand out bonuses to the other production nodes and stakeholders. This mechanism will be further discussed in the following chapter.

4.2.5. Confirmation of Transactions

aelf has a faster and more predictable confirmation process than present Blockchain systems. DPoS is different than PoW because it does not need to package hashes repeatedly. This means that the time for each production node to package a block is stable and can be controlled within T (4 seconds).

aelf recommends that one fast confirmation accepted by 5 blocks is used for general transactions, and that one accepted by 15 blocks is used for substantial transactions. Thus, one general transaction will be confirmed within 20 seconds and a substantial one within 60.

Please note, that this is a conservative recommendation. Bitcoin recommends a confirmation of 6 blocks, but many users only use one or two blocks per confirmation. Advanced users are allowed to observe and collect data from their own Blockchain and tailor a confirmation time for themselves according to the average processing time of their own production nodes, and the whole network

4.3. aelf SideChain

Chains that are indexed by the aelf MainChain are considered SideChains. As mentioned before, it is recommended that each SideChain be designed to handle one specific type of transaction (Figure 4.8).

When a new SideChain is created via aelf OS, it is recommended they merge mining with the MainChain and establish their own Consensus Protocols. To contribute to the aelf ecosystem, SideChains should reserve a certain amount of aelf Tokens and share partial transaction fees with the MainChain.

When a SideChain needs to verify information from another SideChain, it must include the Block header information of the aelf MainChain. SideChains do not interact directly with each other. Verification is done through the Merkle Tree Root provided by the MainChain.

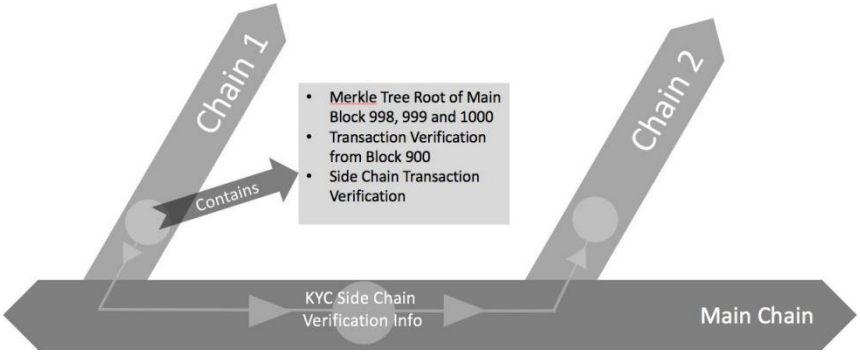


Figure 4.8: Messaging Interaction between Two SideChains

Obtaining state information from UTXO systems such as Bitcoin is a highly complex process; for example, it is very difficult to obtain the available balance from a Bitcoin address. Cross-Chain communication can be addressed via a Blockchain Adaptor, where it creates a compatible Block header including Merkle Tree with Bitcoin. aelf adopts such an adaptor and intends to establish a fully compatible Bitcoin SideChain using aelf OS to cooperate with the widely used Bitcoin and interact with its assets.

4.4. The Economics of aelf

A sound economy is the foundation of a sustainable aelf ecosystem.

For PoS and DPoS, any stakeholder can sell their Tokens and exit from the Eco system in a short timeframe (PoS has a certain lock-up period). One challenge that PoS and DPoS face is the fact that Exchanges can hold large amount of Tokens in the system, and earn rewards at almost zero cost.

For PoW, miners face a more complex consideration before exiting. Their exit is constrained by external factors such as electricity cost, mining machine depreciation, land lease, and human resources.

aelf will use DPoS on the MainChain to incentivize large Stakeholders to maintain a stable system. In the aelf system, the Consensus Protocol on each Chain can be customized to achieve specific objectives.

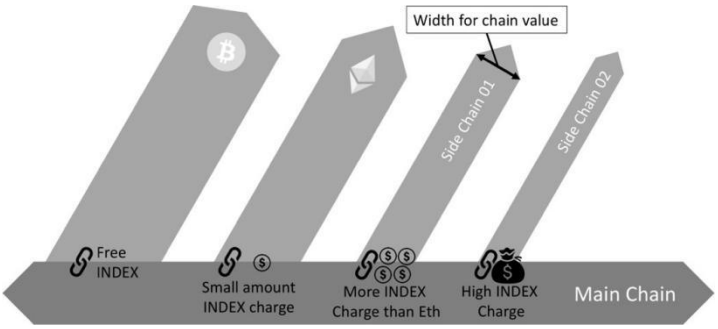


Figure 4.9: Fee Mechanism of Indexing aelf SideChains

After a SideChain is included in the aelf ecosystem, it will pay a certain transaction fee to the MainChain for indexing. aelf adopts a dynamic transaction fee strategy to reflect the different contribution level of each SideChain to the aelf ecosystem. For instance, aelf will charge a lower transaction fee for a SideChain with high levels of contribution such as Bitcoin, which has wide adoption and many associated assets. On the other hand, a SideChain with little value to the ecosystem and that consumes many resources from other Chains will be charged a higher transaction fee.

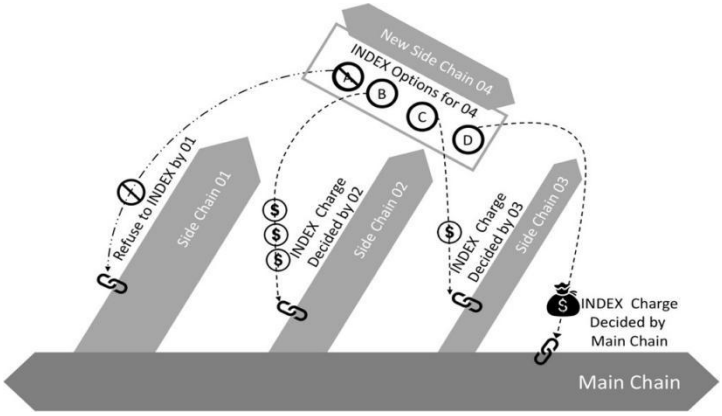


Figure 4.10.: Sub-Chain Indexing

The Ecosystem of each SideChain votes to determine the indexing strategy for Sub-Chains independent of the MainChain. Its own strategy includes, but is not limited to, the business scope (e.g. An insurance Chain will only include sub-Chains that are also related to the insurance industry) and fee schemes of Sub-Chains. Any Chain can also decide not to include any Sub-Chain or actively invite a Chain to become a Sub-Chain as a means to enrich its ecosystem. Within the aelf ecosystem, any Chain can apply to become a Sub-Chain of another Chain or even multiple Chains.

4.5. System Built-in aelf SideChains

aelf Node Topology consists of an interlinking P2P network between full nodes of the MainChain, light nodes, and nodes of SideChains. Non-production nodes are usually Light Nodes. Similarly, ledger nodes are full nodes. Nodes of SideChains are distributed by aelf Node Topology based on their indexing relationship with the MainChain. SideChains can be developed under the guidance of this Foundation. We believe it is necessary to build a system like this. aelf does not aim to build a SideChain itself, but will provide a development template and infrastructure for a SideChain, and facilitate communications between the SideChains.

For example, there could be a content-based network where users are able to buy contents with the token of this network. When a decentralized “Twitter” joins aelf, aelf will help users share contents through this network, distribute network resources, and swap this “Twitter” token with tokens of content-distribution network on a decentralized exchange.

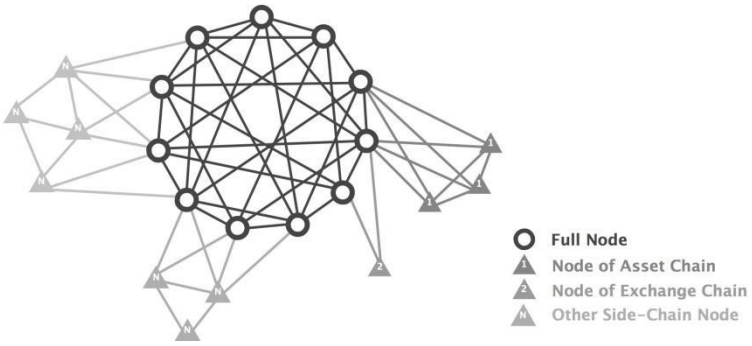


Figure 4.11: Illustration of aelf Node Topology

4.5.1. Information Registration and Authentication SideChain

Information registration and authentication SideChain create great value to industries both online and offline. O2O business such as e-commerce, car hailing and delivery

have already adopted this technology. Huge opportunities have yet to be unleashed in businesses such as supply chain finance, logistics, credit scoring and more, where their large information assets can be migrated onto this SideChain in the future.

4.5.2. Digital Asset Ownership SideChain

The main function of this SideChain is to store digital assets and wallet ownership information.

4.5.3. Asset Initial Distribution SideChain

The main function of this SideChain is to facilitate asset initiation (First Coin Sales). Once the distribution has been completed, assets will be moved to the Digital asset ownership SideChain. The advantage is that normal transactions will not be interrupted during a large scale First Coin Sale.

4.5.4. Decentralized Exchange SideChain

A decentralized transaction SideChain functions as an Exchange. It enables KYC, asset transfer, order placement/ withdrawal and execution.

4.6. aelf Cross-Chain Optimization

Cross-chain transactions need to be optimized to match the block formation speed between different chains. aelf has two mechanisms to address this. The first is a hierarchical SideChain mechanism. We categorize chains into different levels according to the block formation speed of the chain and provide a dedicated adapting SideChain or adapter module to carry out the same level of cross-chain transactions for each level of the chain. Part of this solution is a cross-level guarantee mechanism. For cross-chain transactions at different levels, the MainChain provides a guarantee for the slower chain, but this is only an optional mechanism if required. These two mechanisms effectively enhance aelf's cross-chain transaction speed.

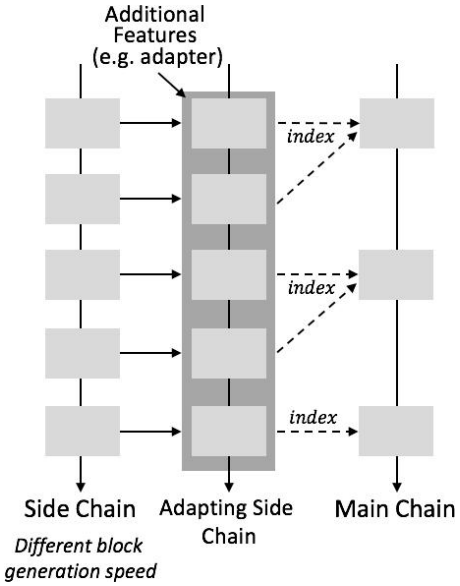


Figure 4.12: Illustration of aelf Node Topology

5. aelf Operating System

5.1. Definition of Minimum Viable Blockchain System

The aelf system creates highly specialized and efficient Chain structures to handle all types of business scenarios. It also enables "Chain splits" to address capacity issues when demand increases. To further enhance aelf's commercial potential, we lay out the system's most fundamental block and infrastructure for developers and the community. The following Chapters discuss the minimum viable Blockchain system and aelf Operating System as the foundations for achieving high customization and efficiency.

Block: A Block is used to record a state in the system. The transition from the last Block to the current Block is defined by the transactions included in current Block.

Transaction: Transaction logic is defined as Smart Contract. A Smart Contract is essentially a Protocol. It always gives the same output from the same input.

Account: An account is used to distinguish the boundaries of data storage. It consists of public key and private key systems.

P2P network communication: Data transmission between nodes is done through the underlying P2P network.

Consensus Protocol: A Consensus Protocol defines the rules and authority to update a state within the Blockchain.

5.2. aelf Kernel

5.2.1. Built-in Minimum Viable Blockchain System

These are the foundational components of the Blockchain system operating within the aelf Kernel. They are linked with relevant interfaces to define the customizable parts of Smart Contracts, Consensus Protocols, Blockchain headers.

5.2.2. Unified Account System

The Bitcoin system introduced public and private keys into the concept of an account. The Pay to Script Hash gives transaction authority to a Smart Contract. While Ethereum defines externally owned accounts and contract accounts, aelf Kernel defines both types of accounts as Smart Contracts.

5.2.3. Parallel Transactions Processing Within a Block

aelf analyzes the static state of transactions and assesses the impacted data range of each transaction. As illustrated in Figure 5.1, Multiple transactions without read/ write conflicts can then be processed in parallel, without affecting the output of each transaction. During the process of Block formation, nodes assign transactions to different groups based on the mutex of the transactions. Transactions within a group will be processed in sequence, while all groups will be processed simultaneously.

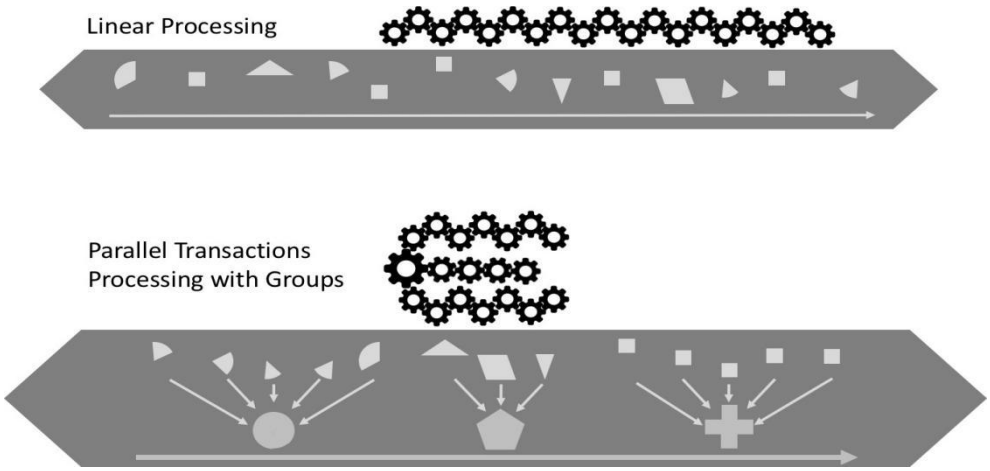


Figure 5.1: Parallel transactions processing within a block

There are special transactions that cannot be processed in parallel because they impacted data ranges change while other transactions are being processed. For these circumstances, nodes will prioritize transactions that can be processed in parallel. With sufficient transaction fees, these special transactions in a nonparallel group will be processed in sequence. Otherwise, nodes can reject these transactions. It is to be noted that, when a faulty node accepts a transaction that cannot be processed in a parallel and time-consuming manner, the probability that other nodes reject this Block increases.

Amdahl's law is an empirical rule in computer architecture named after Computer Scientist Gene Amdahl that gives the theoretical speedup in efficiency when using parallel processing. Think about a program that runs on a single processor. In terms of the execution time, “f” is the proportion of the execution time that the part benefiting from improved resources originally occupied, so (1-f) is the proportion of execution time that is fixed for sequential processing. If there are “m” (numbers) of processors that run in parallel, then the theoretical speedup of this program will be calculated as follows:

$$SpeedUp_{Amdahl} = \frac{1}{(1-f) + \frac{f}{m}}$$

Two major conclusions are conducted:

- (1) Speedup hardly improves when f is at minimum.
- (2) As m rises to the maximum, speedup is limited by 1/(1-f).

Amdahl's law is a fixed-size mode, which means it will solve problems of a fixed size with a fixed proportion of execution in parallel.

Most Blockchain transactions are not correlated. Using Amdahl’s law, data execution can be greatly sped up. However, most present Blockchain systems execute in sequence, and all nodes carry out the same set of computing. This wastes resources and hinders transaction speed. EVM, for instance, does not only process transactions sequentially, but also has requirement for gas fees, resulting in extremely low performance efficiency.

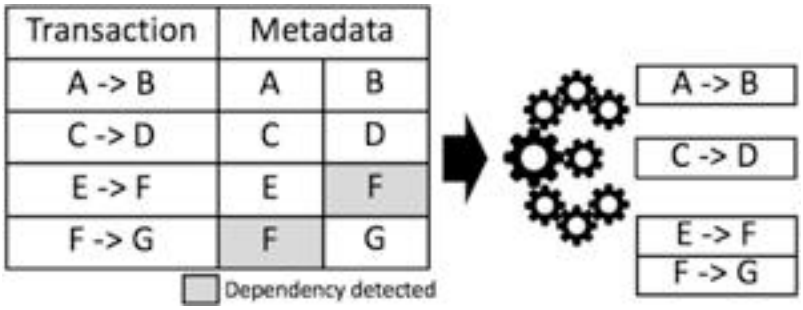


Figure 5.2

To solve complex Blockchain problems, low-speed transactions are simply not a viable option. aelf aims to build a Blockchain system with high on-chain TPS through parallel processing. The key here is to separate transaction data and computational dependency, which solves data hazard issues. We could refer to the architecture of Intel micro-processor, where a reservation station separates electrocircuit dependency along with other technics such as register renaming to deal with the large data hazards that frequently occur in RAW, WAW, and WAR and execute ALUs in parallel.

The aelf Parallel Execution Scheduler (GPES) adopts a similar approach. In the regular internal test, aelf separates computational dependency and data dependency in Blockchain from the memory pool. GPES also has a set of pretreatment (prediction on computing time span), pre-indexing of code segments that are able to be processed in parallel, initiate the pipeline, and execute parallel processing in multiple dimensions.

This set of indexing language can be used to solve more complicated parallel logic problems.

aelf's pipeline is also an important method to increase speed. It has been widely adopted for cases such as CPU and meta function (map, aggregate first, and contains) processing. This set of Turning incomplete language is perfect for processing data streams (or simple transaction streams). Parallel processing functions and context free/immutable computing will make full use of cores and nodes.

In general, parallel processing is a comprehensive strategy to significantly increase the speed of Blockchain transactions.

5.2.4. Transactions Marked by Blocks

A valid transaction in the period between broadcasting and confirmation is considered to be in a "pending" state. Usually, transactions will be quickly packaged and confirmed. However, there are also cases where transactions are left unconfirmed for a relatively long period; for instance, during times of Bitcoin network congestion or when a majority of miners are unsatisfied with the gas fees. When a transaction is neither confirmed nor able to be withdrawn for a relatively long period, it will be considered in a state of "chaos".

aelf requires that the broadcast for each transaction is labeled with a "mark", which is the hash header of the the latest block when the transaction happens. The production node will then only process the hash header of the most recent 64 blocks. If a transaction is not confirmed after 64 blocks are generated, then this transaction is deemed expired. Simply put, a transaction that is not confirmed within 5 minutes can be rebuilt by token holders.

Another function of marking transactions is to effectively render Blockchain forking obsolete. One node successfully marks a transaction when the the hashes of this transaction are included the latest 64 blocks. If a node receives a large amount of invalid marking hashes from the highest chain and is not able to package these transactions, then it is likely working on a forked chain. If nodes receive a large amount of transactions with invalid marking, there's a high possibility that this Blockchain has forked. At this moment, nodes can suspend trading to avoid risks.

5.2.5. Smart Contract Collection

The aelf Chain Contract has a collection of Smart Contracts that are defined during the Genesis. This collection is named the Genesis Smart Contract Collection, in honor of Satoshi. The essence of Smart Contract Collection is a class that defines the main functions, the Consensus Protocol of the chain, and the update mechanism of the collection.

5.2.6. Smart Contract Update

The functions of aelf are defined by the Smart Contract Collection. Therefore, updating the collection impacts the functions of the whole Chain. The update mechanism of the collection is defined by the previous collection. For example, if an 80% majority votes for a new Smart Contract Collection in the most recent 100th Block, it is confirmed by the consequent 2000 Blocks, and a new collection will replace the original one. Nodes that do not update the collection will have their work terminated.

5.2.7. Customizable Consensus Protocol

For specific business scenarios, the Consensus Protocol has a major impact on participants' decisions. For a private chain with a high trust level, PBFT is a popular Consensus Protocol. It creates high performance with a small number of preassigned miners. In an environment with low trust, the stability of a Blockchain is maintained via Consensus Protocols such as PoW, PoS and DPoS.

aelf defines Consensus Protocols as part of the Smart Contract collection and can implement any type of Consensus Protocol based on any business scenario. We will use Bitcoin and Peercoin examples to illustrate the considerations of choosing the Consensus Protocol.

PoW used by Bitcoin authenticates the Blockchain solely based on information from the Block header without any forms of input. On the other hand, PoS used by Peercoin requires data from stake transactions within the Block–its own authentication of the transaction besides Block header. We recommend future users pursue a Consensus Protocol that only requires Block header information in order to achieve timely authentication. In addition, for highly specific scenarios, customizable Consensus Protocols can be implemented.

5.2.8. Customizable Block Header

To facilitate the recommendation that the Consensus Protocol only uses Block header information, we introduced a customizable block header. The Block header of Peercoin does not contain information that verifies the legitimacy of a Block, which means that a stake Block cannot verify Block legitimacy by itself. aelf Kernel allows users to customize Block header structure during the creation of a Chain. Self-proof based on Block header can be done by verifying unspent transaction Merkle Tree with

Hash (TxID + N + Value) and calculating the stored Root to obtain TxID, N and Value and Merkle Tree verification.

5.3. aelf Operating System Customer Interface

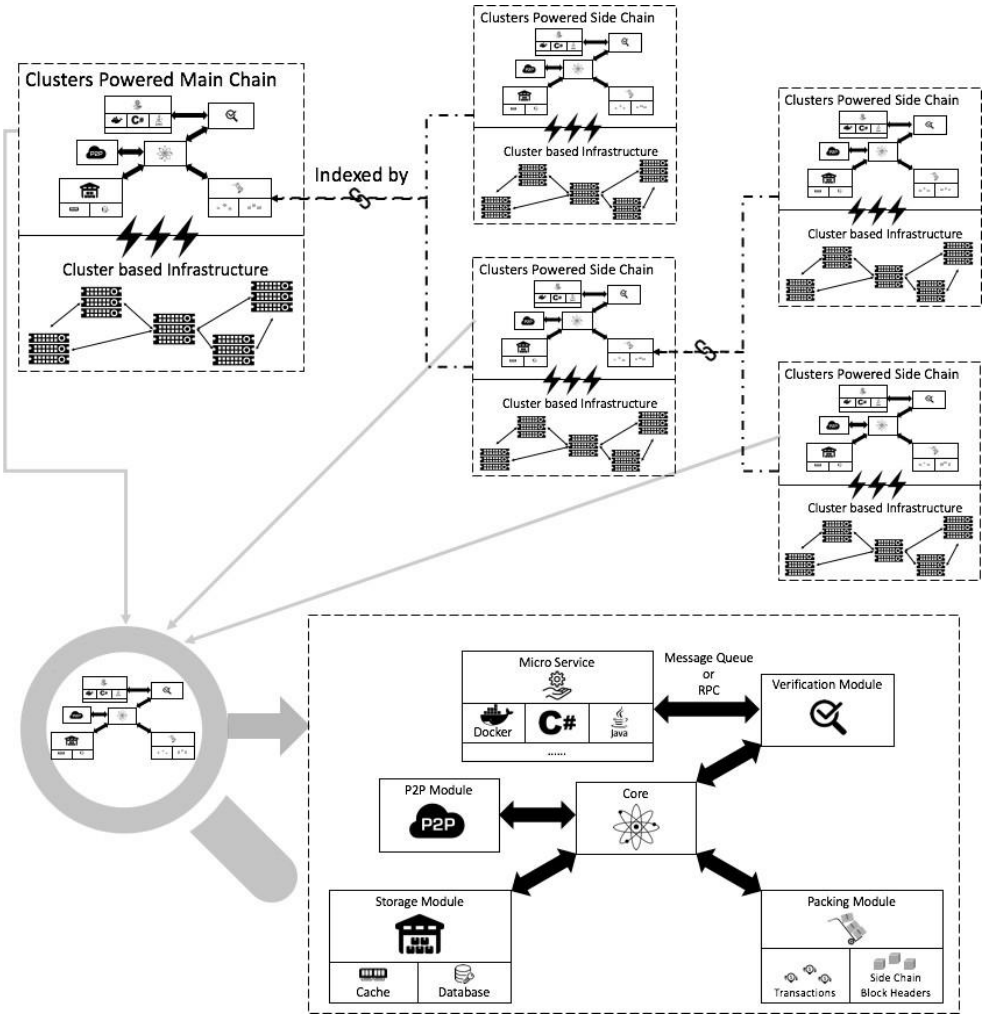


Figure 5.3: aelf Operating System Interface

5.3.1. Smart Contract Execution

The aelf Operating System defines Smart Contracts as Protocols. They can be executed in any forms of service realization.

The aelf Operating System prefers Docker, but also supports native programming languages such as Java, C#, Go, Javascript, LUA.

For Docker, aelf provides internal RPC services to grant access to read variables and user accounts during Smart Contract realization. For native programming languages, aelf provides respective SDKs to execution functions.

5.3.2. Micro-service

Smart Contracts are defined as micro-service in aelf. This makes Smart Contracts independent of specific programming language. Consensus Protocol essentially becomes a service, because it is defined through Smart Contract.

5.3.3. Cloud Base

Through the micro-service approach, aelf Kernel extends parallel processing to a cloud, thus enabling cloud-based contract execution.

aelf Kernel has defined data structure and standards, therefore hot data can be stored in RAM. By utilizing a mature and decentralized database service, this effectively improves IO performance of the system.

5.3.4. Light Node

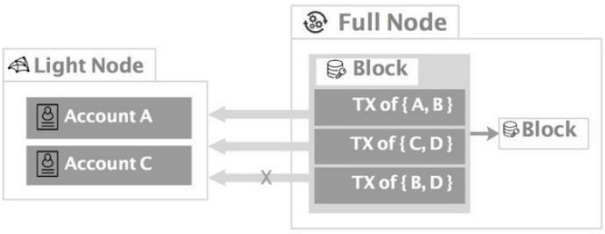


Figure 5.4: Illustration of Light Node Data Structure

Each node within aelf handles only relevant information within the system, which is accomplished through customization and an internal Merkle Tree verification mechanism. This enables nodes to be lighter and significantly increases compatibility with light desktop and mobile terminals.

5.3.5. Optional Modules

5.3.5.1. Data Cleansing Mechanism

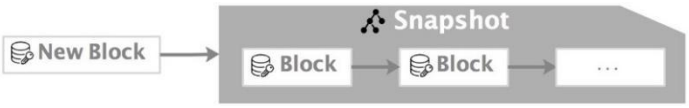


Figure 5.5: Illustration of Data Cleansing Mechanism

The aelf system adopts a snapshot mechanism that resets Block formation with the addition of original data to the new Genesis Block. However, the aelf system does not rely on historical data, but rather focuses on new data that needs to be processed. This is similar to human history, where despite the fact that many historical details have been lost, it does not affect the decisions of people in the present. Likewise, the aelf system has the ability to abandon historical data if it becomes too bulky to record.

5.3.5.2. Data Tunnel

The data tunnel is a mechanism to execute P2P transfer. These data are not recorded in the Block. Data tunneling is only applicable to encrypted P2P data transfer. For example, if A purchases data from B, B transfers data to A while A transfers assets to B, both through the data tunnel. This design aims to enable data transfer between two nodes directly. In present Blockchain systems, the only strategy is to broadcast transactions so that all nodes need to process the transaction. This is a waste of resources and will limit the volume of processed transactions as well. Data tunnels can be realized through a plug-in protocol, but this will require approval by all the nodes. If this does not occur, things will become intractable. For example, developers

often get into trouble when Internet Explorer does not support features from Chrome). With this protocol, aelf will support more applications, such as a data purchase contract, among others. (see further below).

5.3.5.3. Rapid Confirmation Model

aelf permits rapid transaction confirmation only if the recipient has been authorized by the sender. The authorization is only valid for a certain type of transaction during a certain period and between assigned addresses. For example, if A wants to initiate a rapid confirmation model with B during an asset transfer, then A needs to initiate a transaction with a certain amount of assets reserved for the transaction, and specify B as the counterparty. During the actual transaction process, A will send the signed transaction to B via a Data Tunnel. B instantly confirms the transaction upon receival. Affected assets will then be transferred to B after B signs on the transaction with their address. A will then receive the remaining assets. The Data Tunnel is terminated after the transaction.

5.3.5.4. Token Module

The Token module defines all logics and algorithms for the value carrier (Token). It specifically serves scenarios such as payments for resource allocation, or rewards for maintaining the stability of aelf.

In most public chains, the token mechanism is indispensable. It is used to incentivize healthy development of the whole network, and settle the contributions of the different roles. aelf designed a token module where every SideChain recognized by the aelf Operation System is allowed to accept the aelf token.

5.3.5.5. Customization

aelf enables developers to rapidly customize the system by redefining parameters on each module, and to implement SideChains by the aelf Operation System. aelf’s core principle is that "one chain serves one specific business scenario", and we established a highly abstract and modular architecture. For enterprise users and entrepreneurs, this accelerates the process of implementing their business ideas. For more advanced users, it permits high customization for their own Chains and unleashes the full potential of Blockchain.

6. aelf Ecosystem development

Any new technology will not succeed without commercial adoption and a sustainable ecosystem. aelf has proposed a technical blueprint with commercial application instilled throughout the whole design. It is crucial to establish an aelf ecosystem including internal and external resources. We pursue this goal by striving concurrently in three dimensions: technology, business, and capital.

6.1. Technology

The chapters above laid out the key technical features of aelf. The aelf team has years of Blockchain development experience, with particular involvement in a few commercial-focused enterprise projects. The proposed aelf technical solution intends to resolve the most pressing obstacles for commercial adoption of Blockchain, such as scalability, security, customization, and interoperability. It provides a highly efficient infrastructure to adopt new protocols and support all kinds of commercial scenarios in the future.

6.2. Business applications

Ultimately, aelf is intended to become a new “Internet infrastructure” to support the next generation of “digital businesses.” The team and their advisors have worked with numerous Blockchain projects in the past, and we see the industries that will be the “early adoptors” and “Blockchain stars” of aelf:

Figure 6.1: Illustration of aelf Business Applications

1) Financial services

   Blockchain has drawn a lot of attention from the financial services industry as it significantly reduces intermediaries and ensures secure transactions. Multiple chains can be developed on aelf specifically for financial services, including cross-border payments, trade finance, and supply chain financing. The parallel processing feature is capable of handling business transactions on an international scale, and the inter-chain communication feature allows smooth coordination from asset registration, account management, and real-time transaction.

2) Insurance

Insurance is another well established and attractive field that can be disrupted by Blockchain. A dedicated aelf SideChain for insurance will integrate various DApps and transform the whole industry value chain. These chains will encompass everything from from user identity, insurance contract execution, and claim handling.

3) Digital identity and IPs

aelf's multi-chain structure has a built-in chain for digital identity. This ensures the performance of this SideChain if another SideChain is busy, such as when a new token is issued on that other SideChain. Within aelf, digital identity can be used by other SideChains via "messaging". Using the adaptor, aelf is also capable of retrieving information and data from other established chains, such as Bitcoin and Ethereum.

4) Smart City

Governments would be able to use aelf to securely and conveniently run certain administrative tasks. Governments or organizations can customize the consensus protocol to meet national security requirements. Activities, such as utility recording, citizen identities, government agency information disclosure and polling can be realized on aelf with great transparency and efficiency. A few countries are experimenting in this field, including Estonia, Singapore, and China.

5) Internet of Things

aelf supports light node and cloud service, which reduces the computational requirement for connected devices while maintaining high performance. This is critical in order to manage billions of devices and enables micro-payments across them to link Internet of Things.

aelf has laid out a strong foundation for the above industries and more to thrive, and we actively identify new business opportunities and DAPPS to be part of aelf ecosystem.

1) Interoperate with existing DApps on existing chains

There are already some proven DApps on existing Chains, such as on Bitcoin and Ethereum. aelf will leverage its interoperability feature to connect with these DApps in order to allow asset exchange and capture the transaction data coming from those DApps

2) Nurture new start-ups ideas

The development team and their advisors are deeply involved in new idea formation and commercialization in the global Blockchain community. New startups have approached us for technical and commercial advice. We will leverage these strong connections to nurture the start-ups and include them in the aelf ecosystem. Together with VCs, we are confident we can identify and bring the most promising projects to launch themselves on aelf.

3) Transform established companies to “Blockchain savvy”

   Established companies pose another opportunity to benefit from and join the aelf ecosystem. They already possess large customer bases and have proven value in their current business. aelf can help them become even more powerful with strong incentives and rewards to customers, resolving certain pain points within various industries as described above. The aelf team has been in discussion with Internet companies and traditional corporates on disruptive business model on aelf. We foresee a few exciting announcements will be made in near future. In addition, the team intends to collaborate with global strategy consulting firms to push the boundary of next generation business models on aelf ecosystem.

6.3. Capital

Building an ecosystem requires a large amount of capital. Besides leveraging the funds raised during the Token sale, the team and its advisors have established strong alliances with leading crypto funds globally. We have advised numerous Token sale projects to successfully raise funds and have helped them overcome the many obstacles they face. This international capital network and our reputation ensures a strong financing capability, and we are building a pipeline to support these businesses globally.

References

1. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
2. Vitalik Buterin. Ethereum White Paper: A Next Generation Smart Contract and Decentralized Application Platform. 2013.
3. Melanie Swan. Blockchain: Blueprint for a new economy. ” O’Reilly Media, Inc.”,2015.
4. Frederick P. Brooks. The Design of Design: Essays from a Computer Scientist. “Addison-Wesley”, 2010.
5. Andrew S. Tanenbaum. Modern Operating Systems “Pearson”, 2007.
6. Joseph Poon and Thaddeus Dryja, The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. 2016.
7. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. 2014.
8. Hyperledger Whitepaper. 2016.
9. Muhammad Saqib Niaz and Gunter Saake. Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data. 2015.
10. Robert McMillan. The inside story of mt. gox, Bitcoin’s 460 dollar million disaster. 2014.
11. Sunny King, Scott Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. 2012.
12. David Schwartz, Noah Youngs, and Arthur Britto. The ripple protocol consensus algorithm. Ripple Labs Inc White Paper, 5, 2014.

13. Leslie Lamport. The Part-Time Parliament. *ACM Transactions on Computer Systems*, 21(2):133–169, May 1998.

14. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

15. Leslie Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, Jul 1978.

16. Paul Tak Shing Liu. Medical record system using Blockchain, big data and tokenization. *Information and Communications Security*, pages 254–261. Springer, 2016.

17. Robert Love. Linux Kernel Development. “Addison-Wesley”, 2010.

18. Shawn Wilkinson and Tome Boshevski, Storj: A Peer-to-Peer Cloud Storage Network. 2016.

19. Contract. URL *https://en.Bitcoin.it/wiki/Contract,* 2014.

20. Mandatory activation of segwit deployment, UASF, BIP 0148. URL *https://github.com/Bitcoin/bips/blob/master/bip-0148.mediawiki,* 2017.

21. Smart Property. URL *https://en.Bitcoin.it/wiki/Smart_Property*, 2016.