NEO PLUS Whitepaper

63.22.01

11 January 2022

Disclaimer

Abstract

There are more than 700 different crypto currencies available around the world. Majority of them are based on blockchain technology, which was invented in 2009. When bitcoin first came live. Blockchain technology has been going through many improvements such as proof of stake instead of proof of work, many different mining algorithms are in use and latest improvement, segwit, went live with litecoin and other coins. Still there are some major issues regarding the blockchain technology.

First issue regarding blockchain is scalability. On a bigger scale - listening transactions, putting them into the blocks and trying to get confirmation and consensus over the blocks - it just takes time. It doesn’t matter much, either it’s POW or POS, even compared to some centralized transaction platforms, such blockchain designs are too slow.

Another issue is the cost of proof of work. Today, bitcoin transaction, which initially was made to bring down the cost of the money transactions, is one of the most expensive transactions in the world. POW isn’t energy effective way to secure ledgers.

Third issue is the fact what the crypto currency community has been acknowledged that some way, majority of the miners, are holding the key of final word for the new improvements and enhancements inside the network. Miners are like a separate interest group, with their own economic incentives, besides being the inducers who are transacting inside the network. That has been the blocker to introduce new features in bitcoin blockchain. For example the activation of Segwit and Ethereum Blockchain moving from POW to POS.

1 INTRODUCTION

NPC is a decentralized cryptocurrency which is meant for wide usage mainly on USA/Asian/Europe markets but can be used all around the world.

NPC was built on top of the own network. This allows all users to secure each other's transactions by referencing earlier transactions created by other users, and also removes scalability limits common for blockchains, such as block size issue.

The consensus algorithm used to protect from double-spends is based on establishing a total order within the NPC. This is achieved by selecting a chain, called main chain, which gravitates towards units issued by commonly recognized reputable users — witnesses.

NPC is using own network chain as underlying network but it is totally new asset. All together there have been issued 21 000 000 NPCs which means 40*15 microNPCs.

By using NPC-chain, NPC is the first cryptocurrency that is targeting developing markets with the capability of being an everyday currency. Having solved the issues of scalability and keeping the cost of transactions fee-s very low, the only challenge is to spread the knowledge about crypto currencies and NPC. For this, extensive marketing campaigns will be run in social media and also in traditional marketing channels alongside with affiliate marketing.

It is important to educate the end user on what is crypto currency and what are the benefits of using it.

2 NPCCHAIN

2.1 WHAT IS BLOCKCHAIN?

Most cryptocurrencies are based on innovational blockchain technology.

A blockchain is a public ledger of all transactions that have ever been executed. It is constantly growing as 'completed' blocks are added to it with a new set of recordings. The blocks are added to the blockchain in a linear, chronological order.

To use conventional banking as an analogy, the blockchain is like a full history of banking transactions. Transactions are entered chronologically in a blockchain just the way bank transactions are. Blocks, meanwhile, are like individual bank statements.

2.2 BLOCKCHAIN CHALLENGES:

2.2.1 Transactions time

The ever-growing size of the blockchain is a problem for storage and synchronization. Bitcoin network can process 9 transactions per second and the most capable blockchains could theoretically process a few thousand transactions per second. VISA being the largest payments network is capable of processing 56 000 transactions per second. As crypto currency should help the unbanked, the need for transactions is much higher than even VISA can handle.

2.2.2 Transaction fees Transaction fees rise with the increase of transactions. Users can pay bigger fees to the miners to motivate them to confirm transactions faster. NPC have no problem regarding transaction fees because normal transaction fees pay by launching company itself. Blockchain is great for smart contracts and corporate bureaucracy, but not for everyday currency. Bitcoin is perfect as digital gold, but not as everyday currency.

2.3 NPC CHAIN

In addition to the information of the transaction, A NPC chain unit includes the signatures of one or more users who created the unit and references to one or more previous units (parents) identified by their hashes.

Units are linked to each other such way that each unit includes one or more hashes of earlier units, which serves both to confirm earlier units and establish their partial order. The set of links among units forms a NPC (directed acyclic graph). There is no single central entity that manages or coordinates admission of new units into the database, everyone is allowed to add a new unit provided that he signs it and pays a fee equal to the size of added data in bytes. The fee is collected by other users who later confirm the newly added unit by including its hash within their own units. As new units are added, each earlier unit receives more and more confirmations by later units that include its hash, directly or indirectly.

Unlike blockchains where issuing a block is a rare event and only a privileged caste of users is in practice engaged in this activity, a new unit starts accumulating confirmations immediately after it is released and confirmations can come from anyone, every time another new unit is issued. There is no two-tier system of ordinary users and miners. Instead, users help each other: by adding a new unit its author also confirms all previous units.

References to parents are what establishes the order (only partial order so far) of units and generalizes the blockchain structure. Since we are not confined to one-parent–one-child relationships between consecutive blocks, we do not have to strive for near-synchrony and can safely tolerate large latencies and high throughputs.

2.4 FIRST USES OF NPC CHAIN

2.4.1 Draft

@2017, Foxy blaze, a cryptocurrency specialist from Boston, posted his draft (drafted already in 2014) of a coin based on NPC chain. This never became a project and remained just a draft, so no actual coin was created.

2.4.2 IOTA

In the end of 2016 a cryptotoken optimized for the Internet-of-Things (IoT) industry called IOTA and upcoming is POB, was introduced. Instead of the global blockchain, the main innovation behind IOTA that they call Tangle – a blockless distributed ledger which is scalable and lightweight. IOTA is currently still in Beta.

2.4.3 NPC

NPC is the first asset built on own platform and network. NPC evolve hand-in-hand.

3 NPC ASSETS

3.1 DEFINITION

NPC is an asset defined at git, a blockless Direct Acyclic Graph cryptocurrency. The advantage of leveraging such technology is that NPC's are not mined and therefore the whole process of transaction validating is lighter and scalable.

66 million NPCs have been defined. The asset definition though actually created begins 1017 base units, called microNPCs (µNPCs). One NPC is equivalent to one million µNPCs.

3.2 CODEBASE

The NPC codebase is for the moment made of 3 distinct projects: the client, an explorer and a faucet, respectively forks of the corresponding byteball projects.

The client is available for the following platforms: Windows, iOS, web and Android.

These forks are intended to be slimmer versions of the byteball one, where the parts of the NPCchain that are not relevant to the NPC asset are filtered out.

Some additional features are also required and were developed specifically for this asset, such as a faucet.

The most important additional feature, still under development, is aimed to address the problem of transferring the asset from one end-user to another without them having to prefund their wallet with bytes to cover the fee needed to be paid for the units to be validated and processed by witnesses.

4 THE EXCHANGE HUB NETWORK

4.1 CURRENT SITUATION

We want to allow our end users to transfer NPC without having to deal with the underlying other network. This is not easily achievable given the fact that for any transfer a new coin unit is required, whose creation requires in its turn an expense of bytes. What we need is therefore a way to provide our customers with the bytes they need transparently.

4.2 EXCHANGE HUBS AND THEIR FUNCTIONALITIES

An exchange hub is an entity available on the NPC network whose responsibility is to provide bytes (necessary for creating new transactions) to NPC clients who need or require them.

Funding hubs:

• Hubs work as an exchange office: they propose to exchange NPCs for bytes at a certain rate

• They work, similarly to witnesses, independently from each other in a way that anyone can has the possibility to become a hub. Potentially any client has this possibility.

• Their service can't be centralized or controller completely by anyone

• Offer exchange rates: client may choose the best offer.

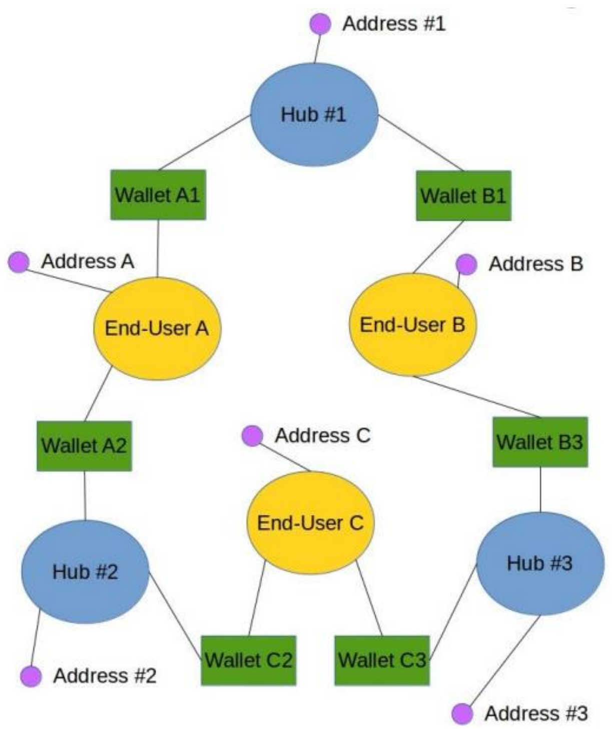
4.3 SHARED ADDRESS SOLUTION

4.3.1 Technical spec.

• It is possible to share an address among wallets

• It is possible for the owner of the address to authorize or deny transactions. If the address is shared among many clients, a transaction can be denied or authorized only by the address owner (the hub).

A hub shares a richly funded address with clients who are interested in its service. The client can use such address a fee payer for a NPC transaction upon approval from the hub owning the address.



Each customer wallet would contain one address per hub it is subscribed to. When it needs funds to create a new transaction, it uses this address as fee payer. The hub will then sign the transaction.

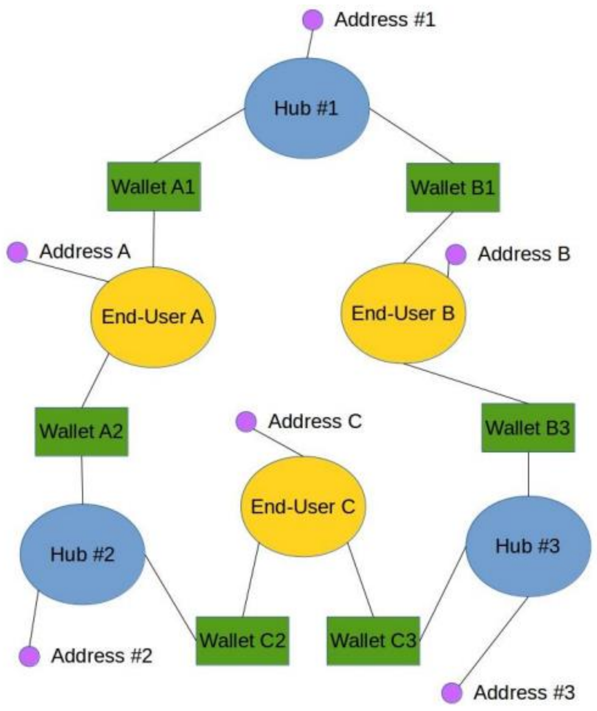
4.4 SHARED WALLET SOLUTION

4.4.1 Technical spec.

• Even in a shared wallet it is still possible to keep addresses separated and distinguish between the owners of the shared wallet

• One owner can hold his or her belongings in an address inside the wallet which no one else can manipulate.

• Using funds from an address requires the authorization of the real owner.



Upon client creation, a new wallet is created and shared with a hub. In this wallet there is an address private to the customer where the customer holds his valuables (NPCs) and a shared address where the hub holds some funds.

In this image it is shown that a solution based on shared wallet would in effect create a network of shared wallets between hubs and end users.

In such wallets, the end user address would be private to the end user and the shared address public to both. Creating a transaction using funds (bytes) from the hub shared address would require the hub to cosign.

5 EXTRACTS

5.1 DATABASE STRUCTURE

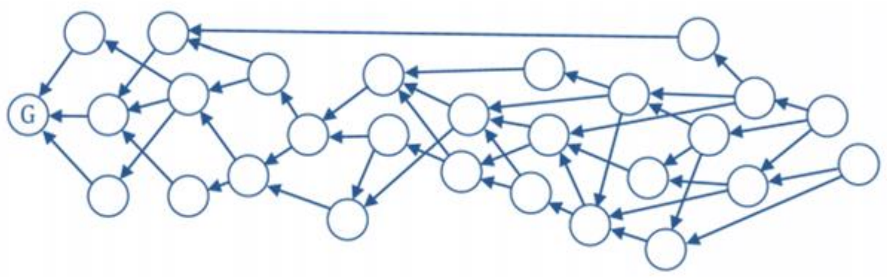
When a user wants to add data to the database; he creates a new storage unit and broadcasts it to his peers. The storage unit includes (among other things):

• The data to be stored. A unit may include more than one data package called a message. There are many different types of messages, each with its own structure. One of the message types is payment, which is used to send bytes or other assets to peers.

• Signature(s) of one or more users who created the unit. Users are identified by their addresses. Individual users may (and are encouraged to) have multiple addresses, like in Bitcoin. In the simplest case, the address is derived from a public key, again similar to Bitcoin.

• References to one or more previous units (parents) identified by their hashes.

References to parents are what establishes the order (only partial order so far) of units and generalizes the block chain structure. Since we are not confined to one-parent–one-child relationships between consecutive blocks, we do not have to strive for near-synchrony and can safely tolerate large latencies and high throughputs: we'll just have more parents per unit and more children per unit. If we go forward in history along parent-child links, we'll observe many forks when the same unit is referenced by multiple later units and many merges when the same unit references multiple earlier units (developers are already used to seeing this in git). This structure is known in graph theory as directed acyclic graph. Units are vertices, and parent-child links are the edges of the graph.



(In the special case when new units arrive rarely, the NPC will look almost like a chain, with only occasional forks and quick merges.)

Like in block chains where each new block confirms all previous blocks (and transactions therein), every new child unit in the NPC confirms its parents, all parents of parents, parents of parents of parents, etc. If one tries to edit a unit, he will also have to change its hash. Inevitably, this would break all child units who reference this unit by its hash as both signatures and hashes of children depend on parent hashes. Therefore, it is impossible to revise a unit without cooperating with all its children or stealing their private keys. The children, in turn, cannot revise their units without cooperating with their children (grandchildren of the original unit), and so on. Once a unit is broadcast into the network, and other users start building their units on top of it (referencing it as parent), the number of secondary revisions required to edit this unit hence grows like a snowball. That's why we call this design Byteball (our snowflakes are bytes of data).

Unlike block chains where issuing a block is a rare event and only a privileged caste of users is in practice engaged in this activity, in a new NPC unit starts accumulating confirmations immediately after it is released and confirmations can come from anyone, every time another new unit is issued. There is no two-tier system of ordinary users and miners. Instead, users help each other: by adding a new unit its author also confirms all previous units.

Unlike Bitcoin, where an attempt to revise a past transaction requires a large computational effort, an attempt to revise a past record in Byteball requires coordination with a large and growing number of other users, most of whom are anonymous strangers. The immutability of past records is therefore based on the sheer complexity of coordinating with such a large number of strangers, who are difficult to reach, have no interest in cooperation, and where every single one of them can veto the revision.

By referencing its parents, a unit includes the parent. It doesn't include the full content of the parent; rather, it depends on its information through the parent's hash. In the same way, the unit indirectly depends on and therefore includes the parents of the parent, their parents, and so on, and every unit ultimately includes the genesis unit.

There is a protocol rule that a unit cannot reference redundant parents – that is such parents that one parent includes another. For example, if unit B references unit A, then unit C cannot reference both units A and B at the same time. A is already, in a way, contained within B. This rule removes unnecessary links that don't add any new useful connectivity to the graph.

5.2 NATIVE CURRENCY:

Next, we need to introduce some friction to protect against spamming the database with useless messages. The barrier to entry should roughly reflect the utility of storage for the user and the cost of storage for the network. The simplest measure for both of these is the size of the storage unit. Thus, to store your data in the global decentralized database you have to pay a fee in internal currency called bytes, and the amount you pay is equal to the size of data you are going to store (including all headers, signatures, etc). Similar to pound sterling, which was equal to one pound of silver when it was first introduced, the name of the currency reflects its value. To keep the incentives aligned with the interests of the network, there is one exception in size calculation rules. For the purposes of calculating unit size, it is assumed that the unit has exactly two parents, no matter the real number.

Therefore, the size of two hashes of parent units is always included in the unit size. This exception ensures that users will not try to include just one parent in an effort to minimize cost. The cost is the same no matter how many parents are included.

To keep the NPC as narrow as possible, we incentivize users to include as many parents as possible (as mentioned before, this does not negatively affect payable size), and as recent parents as possible, by paying part of the unit's fees to those who are first to include it as a parent. We'll define later what exactly is 'first'.

Bytes can be used not only for payment of storage fees (also called commissions), but also can be sent to other users to pay for goods or services or in exchange for other assets. To send a payment, the user creates a new unit that includes a payment message such as the following (from now on, we use JSON to describe data structures):

```
{
inputs:
        {
                 Xwalletkey: "Wallet id",
                xpassword: 2,
                xparameter: 0
        }
 outputs:
         {
                Status : 0/1
                Data : {
                        address: "RECEIVER ADDRESS",
                        amount: 15000 // in bytes
                         }
                },

 }
```

The message contains:

• An array of outputs: one or more addresses that receive the bytes and the amounts they receive.

• An array of inputs: one or more references to previous outputs that are used to fund the transfer. These are outputs that were sent to the author address(es) in the past and are not yet spent.

The sum of inputs should be equal to the sum of outputs plus commissions (input amounts are read from previous outputs and are not explicitly indicated when spending). The unit is signed with the author's private keys.

5.3 THE MAIN CHAIN

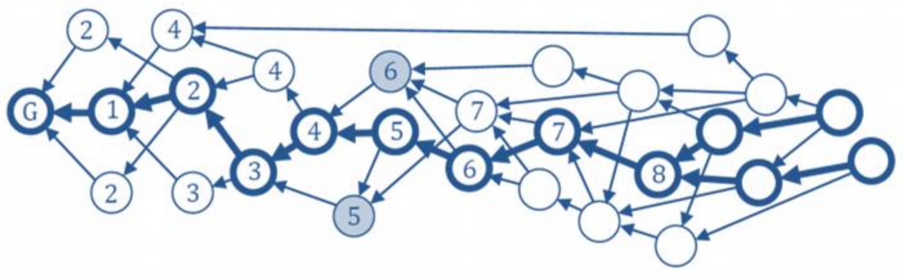
In normal use, people mostly link their new units to slightly less recent units, meaning that the NPC grows only in one direction. One can picture it as a thick cord with many interlaced wires inside. This property suggests that we could choose a single chain along child-parent links within the NPC, and then relate all units to this chain. All the units will either lie directly on this chain, which we'll call the main chain, or be reachable from it by a relatively small number of hops along the edges of the graph. It's like a highway with connecting side roads.

One way to build a main chain is to develop an algorithm that, given all parents of a unit, selects one of them as the "best parent". The selection algorithm should be based only on knowledge available 20 to the unit in question, i.e. on data contained in the unit itself and all its ancestors. Starting from any tip (a childless unit) of the NPC, we then travel backwards in history along the best parent links.

Traveling this way, we build a main chain and eventually arrive at the genesis unit.

Note that the main chain built starting from a specific unit will never change as new units are added. This is because on each step we are traveling from child to parent, and an existing unit can never acquire new parents.

If we start from another tip, we'll build another main chain. Of note here is that if those two main chains ever intersect while they go back in history, they will both go along the same path after the intersection point. In the worst case, the main chains will intersect only in genesis. Given that the process of unit production is not coordinated among users, however, one might expect to find a class of main chains that do converge not too far from the tips.



Once we have a main chain (MC), we can establish a total order between two conflicting nonserial units. Let's first index the units that lie directly on the main chain. The genesis unit has index 0, the next MC unit that is a child of genesis has index 1 and so on traveling forward along the MC we assign indexes to units that lie on the MC. For units that do not lie on the MC, we can find an MC index where this unit is first included (directly or indirectly). In such a way, we can assign an MC index (MCI) to every unit.

Then, of the two random, the one that has a lower MCI is considered to come earlier and deemed valid, while the other is invalid. If both random happen to have the same MCI, there is tiebreaker rule that the unit with the lower hash value (as represented in base64 encoding) is valid. Note that we keep all versions of the double-spend, including those that eventually lose. NPC was the first published work that suggested storing all conflicting transactions and deciding which one to treat as valid.

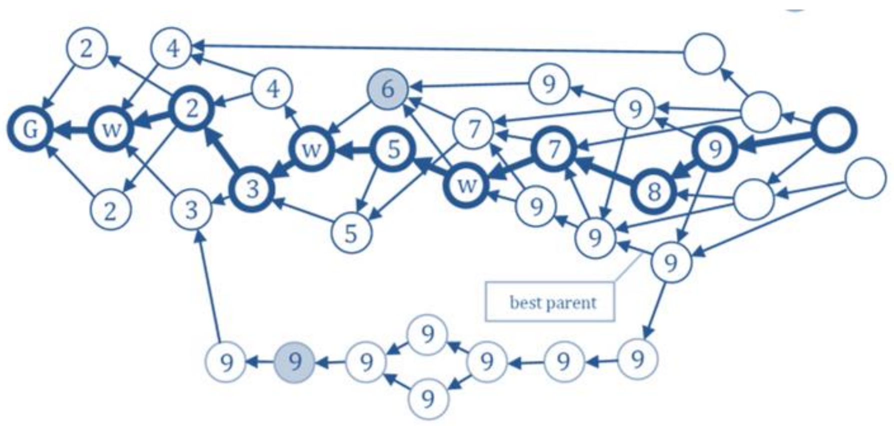
The MC built from a specific unit tells us what this unit's author thinks about the order of past events, i.e. his point of view about the history. The order then implies which random unit to consider valid, as described above. Note that by choosing the best parent among all parents of a given unit, we are simultaneously making a choice among their MCs: the MC of the unit in question will be the MC of its best parent extended forward by one link.

Recognizing that many (or even all) parent units might be created by an attacker, and remembering that the choice of best parent is essentially the choice among versions of history, we should require from our best parent selection algorithm that it favors histories that are "real" from the point of view of the child unit. We hence need to devise a "reality test" that our algorithm would run against all candidate MCs to select the one that scores best.

5.4 WITNESSES

Looking for a "reality test", observe that some of the participants of our network are non-anonymous reputable people or companies who might have a long established reputation, or they are businesses interested in keeping the network healthy. We'll call them witnesses. While it is reasonable to expect them to behave honestly, it is also unreasonable to totally trust any single witness. If we know the addresses of several witnesses, and also expect them to post frequently enough, then to measure the reality of a candidate MC one might travel along the MC back in time and count the witness-authored units (if the same witness is encountered more than once, he is not counted again). We would stop traveling as soon as we had encountered the majority of witnesses. We would then measure the length of the longest path on the graph from the point at which we stopped to the genesis. We'll call this length the level of the unit where we stopped, and the witnessed level of the parent whose MC we are testing. The candidate MC that yields the greater witnessed level is considered more "real", and the parent bearing this MC is selected as best parent. In case there are several contenders with a maximum witnessed level, we would select the parent whose own level is the lowest. If the tie persists, we would select the parent with the smallest unit hash (in base64 encoding).

This algorithm allows the selection of the MC that gravitates to units authored by witnesses, and the witnesses are considered to be representative of reality. If, for example, an attacker forks from the honest part of the network and secretly builds a long chain of his own units (shadow chain), one of them containing a double-spend, and later merges his fork back into the honest NPC, the best parent selection algorithm at the merger point will choose the parent that drives the MC into the honest NPC, as this is where the witnesses were active. The witnesses were not able to post into the shadow chain simply because they didn't see it before the merger. This selection of MC reflects the order of events as seen by the witnesses and the user who appointed them. After the attack is over, the entire shadow chain will land on the MC at one point, and the double-spend



Contained in the shadow chain will be deemed invalid because its valid counterpart comes earlier, before the merger point.

This example shows why the majority of witnesses have to be trusted to post only serially. The majority should not collude with the attacker and post on his shadow chain. Note that we trust the witnesses only to be signs of reality and to not post random units on any shadow chains. We are not giving any of them control over the network or any part thereof. Even for this small duty, it is users who appoint the witnesses and they can change their decisions at any time.

The idea of looking at some known entity as a sign of reality is not new. It has long been known, and some companies have engaged in such activity, that to prove that some data existed before a specific date, one can hash the data and publish the hash in some hard-to-modify and widely witnessed media, like printed newspaper.

Witnesses serve the same function as the newspaper. Like newspapers, they are well known and trusted. As for newspapers where trust is limited to trusting them to publish the data they are given, witnesses in only trusted to post serially, and not much more. Like newspapers, witnesses don't know what's behind the hashes they are witnessing and have few reasons to best parent care.

Newspapers are hard to modify (but possible, and in 1984 they do it), while everything produced by witnesses is protected by digital signatures, which makes any modifications impossible. For reliability, we have several witnesses, not just one, and for speed and convenience, these are online.

Having decided on a list of witnesses, we can then select best the parent and the corresponding history that best fits the definition of reality as "somewhere where these witnesses live". At the same time, the parents themselves might have different witness lists and consequently different definitions of reality. We want the definitions of reality, and histories that follow from them, to converge around something common. To achieve this, we introduce the following additional protocol rule.

The “near-conformity rule”: best parents must be selected only among those parents whose witness list differs from the child’s witness list by no more than one mutation. This rule ensures that witness lists of neighboring units on the MC are similar enough, therefore their histories mostly agree with one another. The parents whose witness list differs by 0 or 1 mutation will be called compatible (with the unit that includes them directly), while the others are incompatible. Incompatible parents are still permitted, but they have no chance of becoming best parent. If there are no compatible potential parents among childless units (an attacker could flood the network with his units that carry a radically different witness list), one should select parents from older units.

The above means that each unit must list its witnesses so that they can be compared. We require that the number of witnesses is exactly 12. This number 12 was selected because:

• It is sufficiently large to protect against the occasional failures of a few witnesses (they might prove dishonest, or be hacked, or go offline for a long time, or lose their private keys and go offline forever);

• It is sufficiently small that humans can keep track of all the witnesses to know who is who and change the list when necessary;

• The one allowed mutation is sufficiently small compared with the 11 unchanged witnesses.

In case a user thinks that any of the witnesses has lost his credibility, or there are just better candidates, the user can replace the witness with a new witness in his list, bearing in mind that his witness list may not differ from that of other units by more than one position. This means that any changes can happen only gradually, and a general consensus is required for a change bigger than one position.

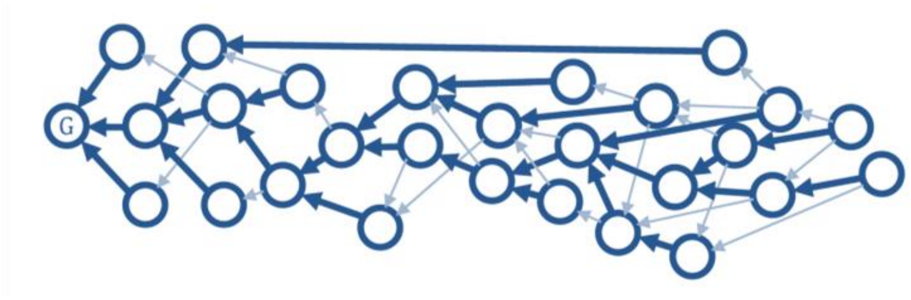
5.5 FINALITY

As new units arrive, each user keeps track of his current MC which is built as if he were going to issue a new unit based on all current childless units. The current MC may be different at different nodes because they may see different sets of childless units. We require that the current MC be built without regard of witness lists, i.e. the user’s own witness list doesn’t matter and even incompatible parents can be selected as best parents. Those means that if two users have the same set of childless units, but have different witness lists, their current MCs will still be identical. The current MC will constantly change as new units arrive. However, as we are about to show, a part of the current MC that is old enough will stay invariant.

We expect witnesses (or rather the majority thereof) to behave honestly, therefore necessarily include their previous unit in the next unit authored by the same witness. This means that when a witness composes a new unit, only recent units are candidates to be chosen as parents. We might expect, therefore, that all future current MCs will converge no farther (when traveling back in time) than a particular stability point. Indeed, the genesis unit is a natural initial stability point.

Assume we have built a current MC based on the current set of childless units, and there was some point on this MC that was previously believed to be stable, i.e. all future current MCs are believed to converge on or before this point (again, when traveling back in time), and then travel along the same route. If we can find a way of advancing then this point forward (away from the genesis), we can prove by induction that a stability point exists.

Note that if we forget about all parents except the best parent, our NPC will be reduced to a tree that consists only of best parent links. Obviously, all MCs will go along the branches of this tree. We then need to consider two cases – when the tree does branch in the current stability point and when it does not – and decide if we can advance the stability point to the next MCI.



First, assume the tree does not branch. We then need to consider the possibility that a new branch will still be added and somehow supported by the witnesses so that it outgrows the existing branch. The other possibility is that the witnesses put so much weight in support of the existing branch, that the requirement of including one's previous unit leaves them no options but continue supporting the existing branch. Let's quantify the latter possibility.

Remember that best parent is selected as the parent with the greatest witnessed level. Let's travel back in time along the current MC from the tip until we meet the majority of witnesses (we are referring to witnesses as defined by the unit lying on the current stability point). If at least one of them lies earlier than the current stability point, we do not try to advance the stability point, otherwise we proceed. In this case, all these witnesses are already "invested" into the current MC. Among these witnesses, we find the minimum witnessed level min_wl. When any of these witnesses posts a new unit, this unit might have parents whose MC leads to the current MC and parents whose MC leads to a competing branch, and the parent with the highest witnessed level will be selected as best parent and will define the direction of the next current MC. Since the witness has to include its previous unit, the witnessed level of the parent leading to the current MC will be at least min_wl.

The witnessed level of any parent leading to the alternative branch will never exceed the level of the current stability point, even if all remaining (minority) witnesses flock to the alternative branch. Therefore, if the current MC grows far enough so that min_wl is greater than the level of the current stability point, the majority of witnesses will have to increase support for the existing current MC, the alternative branch has then lost all chances to win, and we can move the stability point forward to the next MCI.

Next, assume the tree does branch. We need to find a condition where the alternative branches will lose any chance to outgrow the current MC. Let's start by defining min_wl as in the previous case. Among all units on the alternative branches, we then select those that increase the witness level, i.e. their own witnessed level is greater than that of every parent. Among these, we find the maximum level. Then, even if all the remaining (minority) witnesses gather on the alternative branches, the witnessed level on the alternative branches will never exceed this maximum level. Therefore, if this maximum level is less than min_wl, game is over for the alternative branches, and we can advance the stability point along the current MC.

Thus, there is a point on the current MC before which the MC will never change (assuming the majority of witnesses don't post nonserial units). The total order defined relative to this MC is therefore also final. If we had nonserials, our decision about which one of them is valid, is final as well. If a new nonserial ever appears that conflicts with anything already on the stable MC, the new nonserial unit will definitely be ordered after the old counterpart, and the new one will be deemed invalid. Therefore, any payment made in the unit included on the stable MC is already irreversible. Unlike Bitcoin where transaction finality is only probabilistic, this is deterministic transaction finality.

Every user builds his own (subjective) current MC based on the units that he sees. Since the propagation of new units is not instant, and they may arrive in different order to different users, the users will have different current MCs and different opinions about the last stable point of the MC at any given time.

However, since the current MC is defined solely by the set of units known to the user, in case user B hasn't yet advanced his stability point to the same MCI as user A, he will inevitably do that later – i.e. as soon as he receives the same units as user A, or more. Thus the opinions of different users about the state of any given unit are eventually consistent.

5.6 STORAGE OF RANDOM UNITS

When we decide that a unit is a random, we still have to store it. However, part of its data is replaced with a hash of the data. This rule serves two purposes. First, to reduce storage consumed by a unit that nobody paid for (the entire content of the random unit is deemed invalid, including its payment of commissions). Second, to reduce the utility of the random to the user who sent it, because the hash replaces all useful data that the author wanted to store (for free). This prevents attackers from abusing random as a way to store large amounts of data for free.

The hash that is stored instead of the full content still has some utility to the attacker, as he can store the original data himself and use the hash to prove that the data existed. But remember that:

1. He still has to pay for one unit that is deemed valid

2. If the attacker is already internally storing metadata that is necessary to interpret Byteball data, he could do equally well by just combining all his data into a Merkle tree (Hash tree) and using Byteball to store only its Merkle root for the cost of one small unit.

Under this design, there is therefore no self-interest in trying to send random.

It ought to be mentioned that we cannot just reject random the first time we see them. If we did, an attacker could send his random to different users in different order. Different users would then stick to the versions they first received and reject everything based on the other version, so the attacker would succeed in partitioning the network. That's why we have to store both versions and then decide on their order. Even more, users should forward random to peers just like any other units, as the sooner peers learn about the random the better.

We still try to avoid including random if possible: the parent selection algorithm excludes random as long as they are childless. For this reason, it's desirable to help peers learn about random as soon as possible.

5.7 COMMISSIONS

As mentioned before, the cost to store a unit is its size in bytes. The commission is split into two parts: headers commission and payload commission. Payload commission is equal to the size of messages; headers commission is the size of everything else. The two types of commissions are distributed differently.

Headers commission goes to one of the future units which take the payer unit as parent. The receiver is selected only after both the payer unit's MCI and the next MCI become stable. To determine the receiver, we take those children whose MCI is equal to or 1 more than the MCI of the payer. The hashes of each of these children are concatenated with the hash of the unit lying on the next MCI (relative to the payer), and the child with the smallest hash value (in hex) wins the headers commission. This hashing with the next MC unit is designed to introduce unpredictability (the next MC unit is not known beforehand) and render useless any attempts to improve one's chances of receiving commission by playing with one's own unit hash. At the same time, restricting candidates to those whose MCI is no more than 1 greater than the MCI of the payer, incentivizes the selection of the most recent units as parents. This is useful to keep the NPC as narrow as possible.

We pay only the headers commission and not the entire commission to those who are quick to pick our unit as parent, for the following reason. If we did pay the entire commission, we would have incentivized abusive behavior: split one's data into several chunks and build a long chain of one's own units storing one chunk per unit. All the commissions paid in a previous unit would then be immediately collected by the same user in the next unit. As we pay only the headers commission, such behavior is not profitable because to produce each additional element of the chain one has to spend additional headers commission – roughly the same as one earns. We use the remaining (payload) commission to incentivize others whose activity is important for keeping the network healthy.

Payload commission goes to witnesses. To incentivize witnesses to post frequently enough, we split payload commission equally among all witnesses who are quick enough to post within 100 MC indexes after the paying unit (the faster they post, the faster this unit becomes stable). If all 12 witnesses have posted within this interval, each receives 1/12 of the payload commission. If only one witness has posted, he receives the entire payload commission. In the special case that no witness has posted within this interval, they all receive 1/12 of payload commission. If the division produces a fractional number, it is rounded according to mathematical rules. Because of this rounding, the total commission paid out to witnesses may not be equal to the total payload commission received from the unit's author(s), so the total money supply will change slightly as well. Obviously, the distribution happens only after MCI+100 becomes stable, where MCI is the MCI of the paying unit. To spend the earned headers commissions or witnessing commissions, the following input is used:

inputs: [{ type: "headers_commission", from_main_chain_index: 123, to_main_chain_index: 205 }, { type: "witnessing", from_main_chain_index: 60, to_main_chain_index: 205 }, …]

Such inputs sweep all headers or witnessing commissions earned by the author from commission paying units that were issued between main chain indexes from_main_chain_index and to_main_chain_index. Naturally, to_main_chain_index must be stable.

When a unit signed by more than one author earns headers commission, there is so far ambiguity as to how the commission is split among the authors. To remove the ambiguity, each unit that is signed by more than one author must include a data structure that describes the proportions of revenue sharing:

unit: { … earned_headers_commission_recipients: [{address: "ADDRESS1", earned_headers_commission_share: 30}, {address: "ADDRESS2", earned_headers_commission_share: 70}], … }

The addresses who receive the commissions needn’t be the same as the author addresses the commission can be sent to any address. Even if the unit is signed by a single author, it can include this field to redirect headers commissions elsewhere.

5.8 CONFIRMATION TIME

Confirmation time is the time from a unit entering the database to reaching stability. It depends on how often the witnesses post, since to reach stability we need to accumulate enough witness-authored units on the MC after the newly added unit. To minimize the confirmation period, the witnesses should post frequently enough (which they are already incentivized to do via commission distribution rules) but not too frequently. If two or more witnesses issue their units nearly simultaneously (faster than it typically takes to propagate a new unit to other witnesses), this may cause unnecessary branching of the tree composed of best-parent links, which would delay stability. For this reason, the best confirmation times are reached when the witnesses are well connected and run on fast machines so that they are able to quickly validate new units. We estimate the best confirmation times to be around 30 seconds; this is only reachable if the flow of new units is large enough so that the witnesses earn more from witnessing commissions than they spend for posting their own units.

Despite the period of full confirmation being rather long, a node that trusts its peers to deliver all new units without filtering may be reasonably sure that once a unit was included by at least one witness, plus a typical latency has elapsed (the time it takes a new unit to travel from peer to peer), the unit will most likely reach finality and be deemed valid. Even if a double-spend appears later, it will be likely ordered after this unit.

5.9 LIGHT CLIENTS

Light clients do not store the entire NPCchain database. Instead, they download a subset of data they are interested in, such as only transactions where any of the user's addresses are spending or being funded. 33 Light clients connect to full nodes to download the units they are interested in. The light client tells the full node the list of witnesses it trusts (not necessarily the same witnesses it uses to create new units) and the list of its own addresses.

The full node searches for units the light client is interested in and constructs a proof chain for each unit in the following way:

1. Walk back in time along the MC until the majority of requested witnesses are met. Collect all these MC units.

2. From the last unit in this set (which is also the earliest in time), read the last node.

3. Starting from this last ball, walk back in time along the MC until any ball with a skip list is met. Collect all these balls.

4. Using the skip list, jump to an earlier ball referenced from the skip list. This ball also has a skip list, jump again. Where there are several balls in skip list array always jump by the largest distance possible, so we accelerate jumping first by 10 indexes, then by 100, then by 1000, etc.

5. If the next jump by the skip list would throw us behind the target ball, decelerate by jumping by a smaller distance. Ultimately, leave the skip list and walk along the MC one index at a time using just parent links.

This chain has witness-authored units in the beginning, making it trustworthy from the light client's point of view. All the elements of the chain are linked by either parent unit links (while accumulating the witnesses), or by last ball reference, or by parent ball links, or by skip list links. At the end of the chain, we have the unit whose existence was to be proved.

5.10 ADDRESSES

Users are identified by their addresses, transaction outputs are sent to addresses, and, like double Bitcoin, it is recommended that users have multiple addresses and avoid reusing them. In some circumstances, however, reuse is normal. For example, witnesses are expected to repeatedly post from the same address.

An address represents a definition, which is a Boolean expression (remotely similar to double Bitcoin script). When a user signs a unit, he also provides a set of authentifiers (usually ECDSA signatures).

The definition indicates that the owner of the address has a private key whose public counterpart is given in the definition (in base64 encoding), and he will sign all units with this private key. The above definition evaluates to true if the signature given in the corresponding authentifier is valid, or otherwise false. The signature is calculated over all data of the unit except the authentifiers.

Given a definition object, the corresponding address is just a hash of the initial definition object plus a checksum. The checksum is added to avoid typing errors. Unlike usual checksum designs, however, the checksum bits are not just appended to the end of the unchecksummed data. Rather, they are inserted into multiple locations inside the data. This design makes it hard to insert long strings of illegal data in fields where an address is expected. The address is written in base32 encoding. The above definition corresponds to address <address 64/B>.

When an address is funded, the sender of the payment knows and specifies only the address (the checksummed hash of the definition) in the payment output.

The definition is not revealed and it remains unknown to anyone but the owner until the output is spent.

If the user sends a second unit from the same address, he must omit the. He can send the second unit only after the definition becomes stable, i.e. the unit where the definition was revealed must be included in the last ball unit of the second unit.

Users can update definitions of their addresses while keeping the old address. For example, to rotate the private key linked to an address. Here, definition_chash indicates the checksummed hash of the new address definition (which is not revealed until later), and the unit itself must be signed by the old private keys. The next unit from this address must:

• include this address_definition_change unit in its last ball unit, i.e. it must be already stable;

• reveal the new definition in the authors array in the same way as for the first message from an address.

After the change, the address is no longer equal to the checksummed hash of its current definition. Rather, it remains equal to the checksummed hash of its initial definition.

5.10.1.1 Delegation to other addresses

An address can contain reference to another address:

["and", [["address", "ADDRESS 1 IN BASE64"], ["address", "ADDRESS 2 IN BASE64"]]]

which delegates signing to another address and is useful for building shared control addresses (addresses controlled by several users). This syntax gives the users the flexibility to change definitions of their own component addresses whenever they like, without bothering the other user.

5.10.1.2 Signatures and authentifiers

In most cases, a definition will include at least one signature.

This can be useful for cross-chain exchange algorithms. In this case, the hash pre-image is entered as one of the authentifiers.

5.11 CONCLUSION

We have proposed a system for decentralized immutable storage of arbitrary data, including data of social value such as money. Every new unit of data implicitly confirms the existence of all previous units. Revision of past records similar to that in 1984 becomes impossible, as every new unit also implicitly protects all previous units from modification and removal. There is an internal currency that is used to pay for inclusion of data in the decentralized database. The payment is equal to the size of the data to be stored, and other than this payment there are no restrictions on access to the database. Other assets can also be issued and their ownership can be tracked on the database.

When tracking payments in the internal currency and other assets, double-spends are resolved by choosing the version of history that was witnessed by known reputable users. Settlement finality is deterministic. Assets can be issued with any rules that govern their transferability, allowing regulated institutions to issue assets that meet regulatory requirements. At the same time, transfers can be hidden from third parties by sending their content privately, directly from payer to payee, and publishing spend proofs to ensure that each coin is spent only once.

End of the Document