Nxt 2.0 Overview

Nxt 2.0 Overview

1. Introduction

The Nxt cryptocurrency platform has been running and actively used in production for more than 2 years now. Its solid design and stability have been proven in practice, in addition to being subject to code review by independent experts.

Implemented in Java, the leading industry standard language for financial applications, the Nxt platform allows fast development and addition of new features. Being a second-generation blockchain, not a clone of Bitcoin, it has been designed from scratch to serve as a platform for much more than simple money transfer transactions. It supports trading of user-created tokens (asset exchange), listing and buying of digital goods, creation of custom currencies (monetary system), decentralized crowdfunding, a voting system with very flexible vote counting options, conditional transaction execution (phasing - analogous to multisig in Bitcoin, but much more powerful), and so on. The relative ease with which such new features have been added regularly by the development team every few months during the last two years proves the well-designed and flexible architecture of the platform.

2. Current platform limitations

The Nxt development team is constantly acquiring feedback from businesses looking to use the Nxt platform, and is trying to make the Nxt platform even more flexible and scalable in the long term.

Based on such feedback, some limitations of the current software have been identified. One limitation for example is the need for all platform features to use the same token, i.e. the "NXT" coin, not only for the payment of transaction fees to the network, but also for most transactions that need to use some token to measure value, e.g. pricing of asset exchange ask/bid orders, prices of digital goods listed on the marketplace, exchange rates of monetary system currencies. For many business needs, a custom blockchain is desired, with its own payment token, which can currently be provided by the Nxt platform only by means of offering a "clone" - i.e., a completely separate blockchain, running a modified version of the same software, but not linked in any other way to the original Nxt blockchain. The disadvantage of this solution is that a business who needs such a clone blockchain must run its own servers, generating blocks and processing transactions. In addition to being a burden for most small businesses, running a blockchain on only a few servers with few accounts creating blocks (a process known as "forging") lowers the system security, as compared to the main public Nxt blockchain currently running on hundreds of servers with many independent forging accounts. Such a clone also is bound to lag behind the current public chain software, in terms of feature additions and security bugfixes.

Another limitation of the current design, which is common for all other blockchain platforms too, is the so called blockchain bloat. It is due to the fact that every node needs to store all transactions ever created since the blockchain was started, and not only store them, but re-process all of them when it downloads the blockchain for the first time. This is a security requirement stemming from the trustless design of blockchain platforms. Being a proof-of-stake cryptocurrency, the balance of an account ("stake") at a given blockchain height determines, in a pseudo-random manner, whether this account is eligible to generate ("forge") the next block. For a node downloading the blockchain from scratch, the only way to verify that the next block it is downloading was indeed generated by a legitimate account (i.e. having sufficient stake), is to make sure it calculates and verifies each account balance as it downloads the blockchain, by processing all old transactions it encounters during the download. This represents a processing bottleneck that will only get worse as the blockchain size and the number of transactions per second increase.

While at the current transaction processing rate bloat is not yet a problem, and we have come up with several innovative solutions to reduce it even more (such as prunable data - allowing optional removal of data from the blockchain, yet when needed automatically restoring such data in a trustless manner from archival nodes), bloat is a serious issue that must be solved in a fundamental way, in order for a blockchain platform to be future-proof and scalable.

3. Proposed solution

We have identified an elegant solution that will allow us to address the above limitations of the current software, while building upon existing design and code, and reusing the time-tested platform features without having to re-implement them. The architecture of the next-generation Nxt platform, Nxt2.0, will take the system further, provide scalability, make it even more flexible, yet preserve its security and stability.

The fundamental concept of the Nxt2.0 design is a clean separation of "forging token" and "transactional token". Currently, it is the NXT coin that serves this dual purpose, being used both to determine forging stake, i.e. the right of an account to generate blocks, and to execute all kinds of value-transfer transactions, i.e. represent unit of value. A separation of these two functions is what we believe can achieve both much greater scalability, by reducing blockchain bloat, and flexibility, by allowing multiple other transactional tokens to be used, in effect allowing custom "child chains" to exist and run on the same network of nodes.

4. Implementation details

The current single blockchain will be replaced by a combination of one forging chain, on which transactions are denominated in one token, tentatively called "FXT", and multiple child chains, each having its own transactional token.

The forging chain will support a very limited set of allowed transaction types, such as transfer of FXT from one account to another, trading of FXT to each of the child chain tokens and back, and a special type of "ChildChainBlock" transactions. The forging power of each account will depend on its FXT balance, in exactly the same way it depends currently on the NXT balance. All transactions that change FXT balances will be recorded on the forging chain, and therefore downloading and re-processing the transactions from the forging chain will provide exactly the same proof-of-stake security as the current Nxt platform. But all transactions that modify child chain token balances only, or any other account holdings, will not be recorded on the forging chain, but only on their corresponding child chains. Thus, removal ("pruning") of those child chain transactions after they are no longer needed, will not affect the blockchain security, as validity of FXT account balances can always be verified in a trustless manner by each node.

Of course, the validity of child-chain transactions and account balances (in native tokens) must also be ensured by the platform, and this will be done by anchoring them to the forging chain by means of the ChildChainBlock transactions. This special transaction type will contain as attachment a list of transactions belonging to a single child chain, i.e. denominated in that child chain native token and their execution affecting only account balances and holdings on that chain. In effect, such an attachment represents a "block" on the child chain, although no actual forging (block generation) is done on child chains. But those attachments will be linked to the ChildChainBlock transaction by means of a cryptographic hash only, thus allowing its signature verification to be performed even after the actual content of the attachment has been pruned after some time. This is building upon design and technology already implemented and in production use on the current Nxt platform - in the form of prunable messages, and prunable data ("data cloud"), and a network of special purpose archival nodes to store them.

Each node running the blockchain will validate the transactions from all child chains, before they are pruned. A node downloading the blockchain from scratch will no longer be able to fully validate child chain transactions that have already been pruned (it will only verify their hashes and ChildChainBlock transaction signature), but this does not lower the overall blockchain security as it can still verify that the accounts who forged the blocks containing them were eligible to forge at that time, and therefore those transactions must have been validated by all up to date nodes while their data was still available, in order to get included in the currently winning (best difficulty) blockchain fork selected by the majority of nodes.

All transactions from all chains must be processed by all nodes. All nodes will carry all child chains for the last 1440 blocks at least. Archival nodes can opt to store one or more child chains longer, or indefinitely. Transactions on the child chains will be pruned completely after 1440 blocks on nodes not configured to archive them longer.

In addition to forging chain transactions and blocks, each node will need to store the current state of all accounts, as represented by balances in native child chain tokens, asset and monetary system balances, account properties, aliases, and all other objects and account holdings that are created as result of transactions. Any state that might be needed for validation of future transactions must be kept. But once the rolling fork resolution limit of 720 blocks has passed, older state (i.e. the values of such balances and holdings) no longer needs to be kept. Such state is being removed ("trimmed") even now, however in the current system a node that downloads the blockchain from scratch is reprocessing all past transactions, thus re-creating each past state and trimming it as it goes along. In the 2.0 design, those old transactions will also have been pruned, so re-creating past state in order to arrive at current state will no longer be possible. To solve this issue, snapshots and snapshot propagation will be implemented.

Periodically, each node will calculate a snapshot of the state of all derived objects, and a hash of this snapshot will be included in the current block by its forger. All nodes that are up to date and on the same fork already have exactly the same state, and thus will be able to verify this hash (and reject the block if invalid). A protocol will be defined by which nodes that are out of date, or downloading the blockchain from scratch, are able to request the latest snapshot from up to date nodes, validate it based on its hash being included in the blockchain, and download it in a decentralized manner. In this way such nodes will catch up with the latest system state, bypassing the need to re-process all old transactions, that are already pruned.

The snapshot data itself does not need to propagate through the network when the snapshot hash is calculated. Each node that is up to date already has the state of all child chains, so it can generate such a snapshot for itself. It must only validate that the hash the forger calculated for the snapshot indeed matches its own snapshot.

Fee on child chains will be denominated in the chains native tokens, but the forging chain block forgers will still accept fees in the forging token (FXT) only. To convert fees collected in native tokens to FXT, the role of "bundlers", or ChildChainBlock creators, will be introduced. Any account can serve as a creator of a childchain block, provided it is willing to accept the fees (in native token) collected from the transactions in the childchain block, and in exchange pay the required fee (in FXT) to the forging chain block forger. This will establish a market rate for child chain token to FXT token. If the fee in native token offered by a transaction sender is too low by current market rate, when converted to FXT, no-one will be willing to bundle such transaction into a ChildChainBlock, and the sender must resubmit the transaction with a higher fee. If a child chain token loses value completely and no-one is willing to exchange it to FXT, transaction processing on that child chain will naturally stop - unless someone interested in keeping it alive is willing to subsidize it, creating ChildChainBlocks and paying the expected FXT fees to the forgers, while getting worthless (by free market rate) native tokens in return.

Child chains will compete with each other for inclusion into a block, since at the end the forgers will still look at the fee/size ratio for each transaction and will want to maximize their forging profits, subject to main chain block size and transaction numbers limits.

5. Gains and benefits

The scalability benefit provided should be obvious. A new node downloading the blockchain only needs to download and process the transactions from the forging chain, and the latest state snapshot. No validation and processing of any old child chain transactions will be needed, resulting in huge performance gains and storage space savings. Old transactions from all child chains can be pruned, and kept only on archival nodes, which in the future can specialize as commercial service providers, to provide this archival function for a fee, yet in a trustless way.

Each child chain will have its own native transactional token, which will have independent market value, or could be pegged by the child chain creator to an external unit of value (e.g. BTC, a fiat currency, or some other asset).

In a transparent way, this makes it possible to conduct transactions denominated in the transaction sender's token of choice, by just designating the transaction to belong to the specific child chain available for that token. For example, assets can be traded not only for NXT, but for BTC or EUR, as long as such a child chain with a token pegged to that currency exists. Digital goods listed on a specific child chain automatically have their prices denominated in that chain token, and so on.

As transactions on each child chain pay fees in their native token, users of a specific child chain do not need to acquire and deal with NXT coins only to pay fees. End users do not even need to be aware of the existence of the forging chain which takes fees in FXT only. A child chain creator can sponsor their chain, by covering fees for the users, even for a token that otherwise does not have a market value.

Since all child chains are run by the same code, they all can support the same features (transaction types), and at launch those will be all the features of the current Nxt platform. But a chain can optionally be limited to support only a subset of the globally available transaction types, thus excluding features that are not needed by the specific child chain creator business, are undesireble, or have legal restrictions in their jurisdiction.

Child chains can enforce further rules on transactions denominated in their token, such as permissioning, limiting which accounts are authorized to issue specific transaction types, in order to e.g. comply with KYC rules for a child chain pegged to a fiat currency, or assets marketed to a jurisdiction imposing additional restrictions on who can trade them.

Even though they have their own tokens, all child chains will gain security from the fact that all nodes validate all transactions on all chains. As there is no forging on child chains themselves, it doesn't matter if a child chain has only a few active users and not many transactions per day. It will be fully secured by the vast network of nodes running the global Nxt platform. A small business that needs a blockchain no longer needs to run their own servers and forging nodes. The forging chain guarantees security for all child chains, and collects fees from them. In return, each of the child chains gets the ability to be pruned. Child chains no longer need to keep all their old data going back to genesis in order to be secure, because they do not forge.

Since all nodes run the same software, new features, bugfixes, and security patches, will be automatically available to all child chains. This is a significant improvement over the current cloned private blockchain solution, which requires custom software for each chain, that can easily get out of date and out of sync with the main Nxt platform.

6. Implementation phases

The full rollout of the proposed 2.0 design does not need to happen all in a single step, but as common in large software projects can be implemented in several phases.

A. First, a system consisting of the forging chain and a few hardcoded child chains will be created. Each child chain will be using its own native token, and child chain transactions will be bundled in ChildChainBlock transactions on the forging chain. However, while future pruning of those transactions will be possible, it will not yet be implemented, and as the system is just starting and total transaction count will be low, pruning will not yet be needed. Snapshot calculation and propagation will also not yet be implemented. Since there is no pruning, all nodes will store all child chain data in this phase.

B. Next, pruning, snapshot calculation, and snapshot propagation will be implemented. It will be possible to prune child chain transactions, even retroactively those already created in phase A, because they will have been created as prunable by design. Each node will be storing the transactions of only the child chains the node owner is interested in, with archival nodes providing the service of making old transactions available for download by others. Child chain creation will still be a manual process.

C. Full automation of the child chain creation lifecycle. Users will be able to create their own child chains, without depending on the Nxt core development team. At this stage, we will have much better understanding of the actual needs of child chain creators, the resources a child chain consumes and parameters it needs defined, and thus will be able to automate the process by adding the required transaction types to make it all self-serve.

D. Advanced concepts of the so-called "transparent forging" design can be put in place, allowing reduction of block times and increase of transaction processing throughput based on prediction of next forger, penalty for forgers missing their turn to forge, sending of transactions directly to the forging hub of the next forger, "economic clustering" by marking of transactions with the block id of a recent block, and so on. Many of those potential improvements are only needed when there is a demand for high transaction processing capacity, but this demand can be satisfied only after the scalability and bloat reduction features have been provided by the prunable child chain design in the previous phases.

In an agile manner, the precise ordering of those implementation steps is subject to change and adjustment, e.g. whether automating the lifecycle of child chain creation (phase C), or pruning and snapshot propagation (phase B) should be done first, will be decided depending on actual market needs after phase A has been achieved.