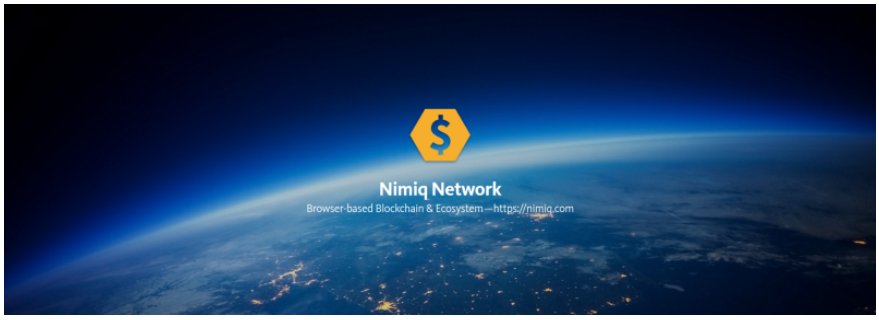
Robin Linus Follow
Creator of https://nimiq.com
Jun 14, 2017 · 7 min read

Nimiq: A Frictionless Payment Protocol Native to the Web

中文



Introduction

On June 6th we published the Nimiq Beta Testnet to get early community feedback and gather real-world data. Some people were missing the nerdy talk. So here it is:

Nimiq is a frictionless peer-to-peer payment protocol for the World Wide Web. It is a third-generation Blockchain protocol combining elements of Bitcoin and Ethereum, streamlined for the web platform. And without a doubt, it is open source and fully decentralized.

Browsers are first-class citizens in the Nimiq distributed network. They are able to establish consensus with the network, and enable true peer-to-peer payments from within, all without a trusted third party. In comparison to conventional cryptocurrencies, this browser-first approach lowers barriers of entry by orders of magnitudes for developers, customers and merchants.

Overview

The main challenge of a web-based Blockchain is to translate the core Blockchain components to the web platform:

- **Network** for establishing P2P connections.
- **Storage** for persistent keys and Blockchain data.

- **Crypto** for hashing, signing and verifying.

In addition, the protocol must be streamlined for the constraints of the web:

- **Compression** of Blockchain data to sync within seconds instead of hours.
- **Instant and scalable transactions** so over-the-counter payments are practical.
- **Simplicity** means we do only one thing and we do it better than anyone else: payments.
- **Blockchain Parameters** streamlined for our browser-first approach.
- **Cross-Chain Compatibility** with other Blockchains such as Ethereum for advanced smart contract features.

Translating the Blockchain Primitives to the Web Platform

Network

Nimiq's peer-to-peer network uses WebRTC and WebSocket connections.

There are two types of nodes in the Nimiq network: *Backbone Nodes* and *Browser Nodes*. Both types use the same isomorphic JavaScript code base.

Backbone Nodes are based on NodeJS and run on servers. They communicate with each other via WebSockets, and they act as entry point and signaling server for Browser Nodes to establish browser-to-browser WebRTC connections.

Browser Nodes are built upon browser engines and therefore they are completely installation-free. To connect to the network, they establish a WebSocket connection to at least one Backbone Node. Once they have established their first connection, they start to establish browser-to-browser connections using the Backbone Node as signaling server. Browser Nodes can also act as signaling server for further browser-to-browser connections.

In the long run, Browser Nodes will mainly be light-clients, and they won't necessarily participate as miners. Their main purpose is to establish consensus quickly to prove their accounts' balances and send transactions into the network. Serious Miners might prefer to run mainly Backbone Nodes for performance and convenience reasons, even though we want to keep the benefit of running a backbone node low in order to balance incentives in the direction of our browser-first approach. Moreover, even Browsers running light-clients will contribute resources to the network: They share the (compressed) Blockchain data with other browsers to reduce the network load on the backbone nodes.

There are some drawbacks of this approach:

1. Depending on the user's NAT configuration, direct peer-to-peer connections may not be able to be established. They would need a *TURN server* to connect to other browsers. In this case, it makes more sense to connect only to Backbone Nodes via WebSockets, because the network load on TURN servers would be unnecessary high.
2. Powerful browser APIs are restricted to secure origins. So for browsers to connect, Backbone Nodes need to provide an encrypted connection via SSL. This requires a domain and an SSL certificate. For easy and cheap access to domains, we will provide dynamic DNS and Letsencrypt packaged in an installer.
3. Depending on the user's firewall configuration, connections to non-standard ports may not be able to be established. In this case, at least some nodes need to run on the standard 443 port. To avoid running the node as root, it is a good idea to use NGINX as reverse proxy.

Storage

Browser Nodes use the IndexedDB API to store Blockchain data and keys on the user's hard drive. Since the browser can't store gigabytes of data, we compress the Blockchain with the Mini-Blockchain scheme (see *Compression*). In the beta testnet, the private key is stored unencrypted and gets deleted if the user clears his browser history.

In the mainnet we will have much higher security standards:

With Chrome's Storage Persistence API the data survives even if you clear your browser data. The private key will always be stored encrypted. Furthermore, we will provide users with a simple app to

backup their key in a printed paper wallet. We are also planning on supporting cold-wallets via Web USB or Web Bluetooth.

Crypto

Since performance is security relevant, we need performant crypto primitives. A JavaScript implementation would not be sufficient.

Betanet Crypto

The crypto in the preliminary beta testnet is based on the WebCrypto APIs to reach near native performance. The WebCrypto API is not very rich and leads to too many sub-optimal design decisions. Therefore we won't use it for the mainnet.

Proof-of-Work Algorithm: We use SHA-256, because it is the only hash function supported by WebCrypto, and is sufficient for our first public testnet.

Digital Signing Algorithm: We use the NIST curve P-256, because it is the only curve supported by WebCrypto.

Mainnet Crypto (preliminary)

The cryptographic primitives in the mainnet will be based on WebAssembly for near native performance and full flexibility in our choice of crypto algorithms.

Proof-of-Work Algorithm: We will use a more sophisticated PoW in the mainnet because SHA-256 will lead to centralized mining. The mainnet PoW should be memory-hard & low energy for truly decentralized mining with regular hardware. There is no final decision on any specific algorithm yet. We are investigating multiple candidates such as Argon2 (too slow though) or ETHash, and we are very open to suggestions by the community.

Proof-of-Stake Algorithm: We would strongly prefer to switch to a Proof-of-Stake Algorithm in the long run to get to a more energy-efficient Blockchain system. We are following the research of other projects such as Ethereum, and we are investigating multiple candidates such as Ouroboros. Just as well, we are very open to suggestions by the community.

Digital Signing Algorithm: We use Ed25519 because it uses "nothing up my sleeves parameters" chosen for performance and it is becoming an industry standard.

More Fundamental Browser APIs

ES6 Classes for a clear and simple object oriented design.

Promises and the beloved async/await to escape from callback hell.

ArrayBuffer and views to serialize Blockchain data to byte level for crypto operations, network and storage.

Streamlining the protocol for the constraints of the Web Platform

Compression

It is nonsensical to have web users download gigabytes of Blockchain data to establish consensus. Especially in the case of a weak network connection, which would make this nearly impossible. So we need to compress the data such that light-clients can synchronize within seconds.

That's where the *Mini-Blockchain Scheme* comes into play: It introduces an *Accounts Tree (a Merkle Patricia Tree)* which has multiple advantages in comparison to the design of Bitcoin:

- It simplifies transactions. The end user doesn't need to deal with the concept of unspent outputs. Only an account and a balance is contained.
- The user can download only the headers of the Blockchain plus simple cryptographic proofs for account balances, which breaks down to downloading only a couple of hundred kilobytes instead of gigabytes without losing trust or security.
- Old blocks can be discarded because the complete state is stored in the Accounts Tree.
- Nano clients can use signed checkpoints to keep the headers chain constant size. Additionally, the full Blockchain up to each checkpoint will be provided for download to make public verification of these checkpoints easy. We will proceed to implement the Mini Blockchain Scheme as per development plan.
- It is easy to listen for balance changes.

Instant transactions

Based on *"The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments"* Nimiq implements *Hashed Timelock Contracts* to enable *payment channels* and within those scalable instant off-chain transactions. This solves the scalability issues beyond micro-payments. Moreover, they enable *atomic swaps* for cross-chain interoperability. There are also approaches to enable *onion routing* on top the Lightning Network to provide anonymity in a way similar to the Tor Browser.

Simplicity: No scripting language.

Simplicity is the best heuristic to building a secure system. So the only feature of Nimiq is fast and secure payments from UserA to UserB. Nimiq intentionally doesn't have a scripting language, because Ethereum already solves the smart contract problem better then we could ever do. We do not try to compete in this field. We want to be compatible with Ethereum such that Nimiq users can easily use Ethereum's smart contract features if they want to.

There is one exception to the "no smart contracts" approach: For the Lightning Network there will be a hashed timelock contract hardcoded into the protocol.

Blockchain Parameters (preliminary)

- Block time: 1 minute (inspired by the results of *On the Security and Performance of Proof of Work Blockchains*)
- Block reward: starts with 5 Nimiq (NIM); halves every ~2'100'000 blocks. This approach models Bitcoin's approach, while taking into account that Nimiq's block time is 10 times faster. (In the betanet the reward is just constant)
- Max Block size: 1 MB
- Difficulty adjustment: Every 10 blocks
- Total supply: 21 Mio Coins divisible by 10^8 (just like Bitcoin)

Cross-Chain Compatibility

Hash Timelock Contracts not just allow off-chain transactions and scalability. We will use them for cross-chain transactions to become compatible with the great work of projects like Ethereum and Bitcoin. This allows Nimiq users to use the advanced smart contract features of Ethereum and it allows users of other cryptocurrencies to easily exchange into Nimiq without an intermediary.

Conclusion

Nimiq introduced Blockchain technology native to the World Wide Web. In comparison to conventional cryptocurrencies, this approach lowers barriers of entry by orders of magnitudes and will foster mass-adoption of cryptocurrencies.

Stay Tuned

This document is work-in-progress and will be updated regularly.

To stay tuned:

- Follow our Github repository
- Join our Slack Channel
- Join our Telegram Channel
- Follow us on Twitter
- Follow us on Youtube

Read More about Nimiq

- Website
- Beta Testnet
- Whitepaper
- Team
- Contribution Terms

. . .

> *DISCLAIMER: None of the statements must be viewed as an endorsement or recommendation for Nimiq, any cryptocurrency, or investment product. Neither the information, nor any opinion contained herein constitutes a solicitation or offer by the creators or participants to buy or sell any securities or other financial instruments or provide any investment advice or service.*