R5: Revisiting Highly Performant Proof-of-Work Networks

Paulo Baronceli
znx@r5labs.org
www.r5labs.org

Abstract. This paper introduces R5, a reengineered proof-of-work protocol that overcomes the performance limitations of existing mineable networks. R5 achieves breakthrough scalability by significantly increasing the maximum block size with a dynamic cap, optimizing the target block time to 7 seconds, and enhancing node efficiency through improved state caching and a larger transaction pool buffer. At its core, R5 incorporates Ethash-R5 - an evolved mining algorithm that adds extra sequential mixing rounds to narrow the performance gap between GPU and CPU mining. Parallel header verification is employed to accelerate block validation under heavy network load. Together, these innovations yield a theoretical throughput of roughly 1,000 transactions per second while retaining full compatibility with the Ethereum Virtual Machine, making R5 a robust solution for high-performance blockchain applications.

1. INTRODUCTION

** Note: This paper presumes that the reader possesses a foundational understanding of blockchain technology, especially regarding the technical implementation of the Ethereum protocol. For those less familiar with these concepts, it is advisable to begin with the Bitcoin whitepaper [2] to gain a basic conceptual understanding of blockchain, followed by the Ethereum whitepaper [3] for an in-depth exploration of smart contract protocols, the Ethereum Virtual Machine, and other pertinent topics referenced in this work.*

Proof-of-work was the cornerstone of the cryptocurrency revolution initiated by Bitcoin [2]. However, as blockchain technology has evolved, developers have increasingly favored consensus mechanisms such as proof-of-authority and proof-of-stake. These alternatives, while offering improved performance in terms of speed and efficiency, often sacrifice the inherent value proposition of resource-based token creation - a hallmark of proof-of-work systems. The economic rationale for proof-of-work lies in its ability to imbue coins with intrinsic value, much like precious metals, by making the cost of production both tangible and traceable.

R5 revisits and re-engineers the traditional proof-of-work paradigm to overcome its performance limitations. By proposing increased block sizes, reduced block time intervals, better cache handling, a larger transaction pool buffer, a much-improved consensus mechanism, and integrating an improved mining algorithm (Ethash-R5), the network aims to deliver throughput and finality that can rival, or even surpass, those of current proof-of-stake and proof-of-authority systems.

Additionally, R5 maintains compatibility with the Ethereum Virtual Machine (EVM), being fully compatible with smart contracts.

In the following sections, we detail the technical modifications and performance metrics that collectively underpin R5's design, demonstrating its potential as a highly performant and secure proof-of-work network.

2. REVALUATING POW NETWORKS

Proof-of-work underpinned the creation of the cryptocurrency industry with the launch of Bitcoin [2]. However, it has since fallen out of preference among developers in favor of novel consensus mechanisms such as proof-of-authority [13], which relies on social arbitration to keep validators honest, and proof-of-stake [13], which uses financial disincentives to promote honesty among its validators.

The reasons for proof-of-work falling out of favor among new networks can often be traced to its lower comparative performance. After all, proof-of-work requires genuine computational effort to validate blocks, which takes time and requires hardware, while both proof-of-authority and proof-of-stake operate more like glorified distributed databases, largely sustained by trust and confidence in their block validation processes. The question that remains is: How are these coins any different from fiat money if they ultimately rely on trust and confidence?

The belief that - much like gold - there is inherent commoditized value in the energy expended to create a new coin is what continues to make proof-of-work a superior choice for storing value. The traceability and tangibility of these costs give each proof-of-work coin an effective unit-of-account function, endowing it with money-like properties.

How R5 Revisits PoW Performance

We propose the following architecture and parametrization improvements to address Ethereum's comparative poor performance metrics: a) increasing the maximum gas cap for new blocks; b) operating at a lower target block time; c) working with more aggressive state cache parameters; d) doubling the transaction pool buffer size; e) adjusting the consensus mechanism to allow for better difficulty management and lower block intervals; and f) aligning minimum hardware requirements for nodes and miners with contemporary accessible hardware specs.

Such modifications are further elaborated on the paper, but the expected result is a net performance gain of > 6,500% compared to Ethereum.

The theoretical maximum throughput of R5 would be of approximately 1,000 transactions per second at peak usage, and it does that while leveraging on stable and fast networking and quick finality. For comparison purposes, Ethereum averages 12 transactions per second [18], while Bitcoin averages 7 transactions per second [19].

Further Expandability

Proof-of-work networks can act as trustless settlement layers for layer-2 rollups designed for near-instant settlement - such as high-frequency trading applications. Layer-2 solutions have the potential to expand the range of use cases for layer-1 proof-of-work networks. By settling on a proof-of-work layer-1, high-performance layer-2s can take advantage of its inherent security, ensuring that transactions remain immutable and verifiable, which in turn fosters greater user trust.

3. IMPROVING PERFORMANCE

The proposed R5 protocol aims to push the boundaries of proof-of-work performance while maintaining EVM compatibility for a smooth, familiar development environment. The network's block size, block time, consensus, caching, and minimum hardware requirements were identified as key parameters for determining overall performance. Several tests were conducted to find the best configuration that maximizes feasible performance without compromising security or decentralization.

Higher Maximum Block Size

The maximum block size (or gas cap) significantly influences the network's overall scale and how easily a typical household computer can run a full node [1], thus affecting decentralization and accessibility. Larger blocks allow more transactions to be recorded in a single mining cycle, making the network more resilient to congestion during peak usage. However, they also produce larger databases, which can discourage users from running their own nodes due to increased storage and processing requirements. This reliance on third-party RPC providers can, in turn, impact overall decentralization.

Ethereum currently operates with a target block size of 15,000,000 gas units, with a maximum limit set at twice that amount [4] - a configuration implemented via EIP-1559 in August 2021. While this represents an improvement over the original 5,000,000 gas unit limit, the present configuration does not fully account for the fact that, under typical network conditions, actual gas consumption per block is significantly below the limit [5]. Consequently, increasing the gas cap under normal conditions would not adversely affect database size; however, during peak usage, the 30,000,000 gas unit ceiling proves insufficient to meet demand, leading to congestion [7].

It is important to note that larger blocks inherently require more compute to be validated, which has a direct impact on minimum hardware requirements to run network nodes.

opBNB has a minimum hardware requirement for Geth-based nodes as follows [15]: 12 CPU Cores; 10GB RAM; SSD Storage; 125MB/s stable networking. Such nodes can process up to 100,000,000 gas/s [14].

Although there are several nuances that makes this direct comparison somewhat subjective, if we use opBNB as a basis to determine compute power of Geth-based nodes in gas/s, we can infer that with similar hardware and a block time seven times larger than opBNB, allowing for 4 seconds of block propagation and other performance inefficiencies inherent to PoW networks when compared to PoS networks, we should be able to process upwards of 300,000,000 gas per block.

We propose to employ a gas cap of 147,000,000 units per block for R5, which substantially increases our maximum block size if compared to Ethereum and allows for more accessible hardware requirements if compared to opBNB.

This is dynamic cap, with a target block size half that (73,500,000u), thus allowing blocks with few or no transactions to remain smaller than full blocks.

In addition to the compute required to process and validate these blocks, the higher gas cap is also expected to result in larger storage requirements for the blockchain. Ethereum averages 70-100Kb per block [16] with a 30M maximum gas per block. We can therefore infer that our network blocks should average 342-489Kb per full block. If we assume adoption and usage equal that of Ethereum, plus an additional 30% allowance for consistent maximum bandwidth utilization, a 12,300 daily block production (considering a ~7-second block time), we should then have, in average, 3,690 blocks sized at approximately 1,262Mb, plus 8,610 blocks sized at approximately 588Mb, resulting in a total increase in storage requirements of approximately 659.5Gb per annum for full archive nodes. Full, snap, and light nodes' storage requirements will be significantly lower. You can validate the calculation above with the following equation:

$$\Delta \boldsymbol{S_{annual}} = (\boldsymbol{1} + \boldsymbol{\alpha}) \times \frac{\boldsymbol{365 \times B_d \times [f\, S_{max} + (1 - f)\, S_{min}]}}{\boldsymbol{1000000}}$$

B_d = daily blocks (~ 12,300)
f = full blocks average (30%)
S_{max} = size of averaged full blocks in Mb (1,262)
S_{min} = size of averaged remaining blocks in Mb (588)
α = allowance factor (30%)

Considering current storage costs of about USD 0.014 per gigabyte [17], a heavily used network should result in an approximate increase of about USD 9.25 per annum in storage costs, in average, for full archive nodes - which we deem to be acceptable.

We can therefore affirm that the database size resulting from the increased block size is unlikely to expand to a level that would adversely affect hardware requirements or costs to any significant extent.

Reduced Target Block Time

Target block time is one of the most critical security factors in any blockchain network, especially under proof-of-work. The Ethereum development team determined that a target of between 12-15 seconds would be optimal for maintaining security while improving speed over Bitcoin's 10-minute intervals.

Although the proposed R5 protocol leverages Ethash as a base algorithm, substantial changes were introduced to the algorithm - including four additional sequential mixing rounds appended to the Hashimoto function – as well as to other parameters such as state caching, transaction pool buffer size, and the maximum block size. As a result, further testing was conducted to understand how these modifications might influence the optimal block time for R5, specifically.

Tests were conducted at 15, 13, 10, 7, and 5-second target block times using the revised code. Nodes were run in both standalone mode and connected to two peers (one local, one over the internet), reaching block 3,000 in each configuration to allow difficulty to stabilize. No performance or security issues were noted at 15, 13, 10, or 7 seconds. However, a 5-second target block time led to more frequent uncle blocks, increased stale blocks for miners, and overall instability. Based on these observations, we propose the target block time for R5 to be set to 7 seconds.

More Aggressive State Trie Cache Parameters

The state cache in Geth is a crucial in-memory component that holds portions of the blockchain state trie - this includes account balances, contract storage, and code. During transaction execution, the EVM frequently needs to read or update this state. Without a state cache, every state access might involve expensive disk I/O and recomputation of trie paths, which would significantly slow down transaction processing.

To better utilize the hardware, we propose optimization via parametrization by increasing the default *CachesInMem* and *DatasetsOnDisk* constants by 50%.

By increasing the state cache size, we can expect a) Reduced latency in transaction execution; b) Higher throughput for EVM and non-EVM operations; c) Smoother block processing; and d) Increased RAM requirements for nodes.

Larger Transaction Pool Buffer

The transaction pool buffer is the in-memory queue that temporarily holds incoming transactions before they're processed.

We propose doubling its size from 4,096 to 8,192. The direct result of the larger transaction pool buffer is that nodes will be able to handle more transactions during spikes in network activity. This reduces the risk of dropping transactions and helps maintain smooth throughput.

The objective trade-off of a larger transaction pool buffer is increased memory usage. However, we do believe that modern hardware is able to cope with the additional memory requirement without adding any significant cost to node operators.

Improved Consensus Mechanism & Algorithm

We propose that the difficulty calculators to be tuned for a target block time of roughly 7 seconds. In these functions, the adjustment factor uses a divisor of 2 (instead of higher divisors found in older algorithms). This means the calculated "delta" between the block's timestamp and its parent is scaled more aggressively. In practical terms, when a block is mined faster than the target, the difficulty is increased more sharply; if it takes longer, the difficulty is reduced more significantly. This helps the network more quickly correct for deviations from the target block time.

The new improved algorithm distinguishes between blocks with and without uncles by adjusting the factor slightly (subtracting from 1 in the absence of uncles and from 2 when an uncle is present). This nuanced handling helps maintain a consistent block time even when uncle blocks (which are allowed but not rewarded) appear.

Parallel Header Verification

The *VerifyHeaders* function is to offer concurrency into the header verification process. By using Go's *goroutines* (with the number of workers tied to available CPU cores via *runtime.GOMAXPROCS*), the system can validate multiple block headers in parallel.

In a purely sequential approach, the total verification time would be:

$$T_{seq} = n \times t_v$$

With parallel header verification, the ideal total time is:

$$T_{par} = \frac{n \times t_v}{P} + T_{overhead}$$

The speedup (improvement factor) achieved by parallelization can then be expressed as:

$$Speedup = \frac{T_{seq}}{T_{par}} = \frac{n \times t_v}{\frac{n \times t_v}{P} + T_{overhead}}$$

This design reduces the latency in processing batches of headers, improves throughput (particularly under heavy network load), and ensures that consensus can be reached faster, even when many blocks are being processed simultaneously.

Together, the improvements proposed to the algorithm and consensus will help the network with:

- **Consistency:** With a more sensitive difficulty adjustment mechanism, the network can maintain a steady 7-second block interval even in the face of rapid changes in the mining power.
- **Efficiency:** Concurrent processing and refined consensus checks, lower latency and improve throughput, ensuring that the network remains robust under heavy usage.
- **Security and Stability:** By more efficiently controlling difficulty adjustment and validating headers, the network should be less prone to disruptive fluctuations, which in turn will help preserve the integrity of the blockchain.

Expected Throughput

A raw network transaction that does not interact with the EVM has a fixed gas cost *(B)* of 21,000u, and the maximum block size *(A)* is 147,000,000u. This allows approximately 7,000 transactions per block. With a target block time *(C)* of 7 seconds, the network can theoretically reach a throughput *(X)* of around <u>1,000 transactions per second</u>.

$$X = \frac{\frac{A}{B}}{C} = \frac{A}{BxC}$$

There are, however, important caveats to these figures - particularly regarding performance for EVM transactions, such as smart contracts, tokens, and NFTs - where the virtual machine itself can become a bottleneck and reduce throughput.

It is also worth noting that, although the theoretical throughput is mathematically feasible, the likelihood of an entire block consisting solely of non-EVM transactions on an EVM-compatible network is minimal at best.

Differentiating Speed and Performance

It is important to note that with a ~7-second block time, transactions are expected to take as long, in average, to be confirmed, even with lower network usage. To illustrate the difference between the proposed architecture's throughput and that of non-proof-of-work networks, consider an analogy to water pipes: the protocol serves as the pipe, while transactions represent the water flowing through it. In this analogy, reducing the pipe's diameter may increase the speed of water flow due to higher pressure, but it limits the overall volume that can pass through. On the other hand, increasing the diameter may reduce flow speed but allows a greater volume to be transported.

For example, TRON [8] (a proof-of-stake network) can process up to 2,000 transactions per second [11] with a block time of approximately 3 seconds [9]. In a hypothetical scenario where transaction demand spikes to 3,000 per second, the blocks would become congested, leading to increased confirmation times. Similar issues have been observed on the Fantom Network [10] (now Sonic), which, despite having a 1-second block time, has experienced confirmation delays of several minutes, several times in the past years, during peak usage.

In contrast, the proposed architecture prioritizes resilience; its ~7-second confirmation time is designed to remain stable even under significant demand spikes, functioning as a larger pipe that can handle a higher volume of transactions without compromising processing speed.

4. ETHASH-R5

We propose to employ, and have developed, an enhanced version of the Ethash algorithm [6] for mining, known as Ethash-R5. While preserving the original cache and dataset generation as well as the Hashimoto function for proof-of-work, Ethash-R5 adds four extra sequential mixing rounds to the standard Hashimoto process. Specifically, after computing the usual Ethash digest, the algorithm performs four additional loops that rehash the working state ("extra") using Keccak256 combined with an FNV mixing step. This design deliberately introduces additional sequential dependency.

$$\boldsymbol{(finalDigest, finalHash)}$$
$$\left(\begin{array}{c} \boldsymbol{(Keccak256 \circ FNV_{Mix})^4(digest),} \\ \boldsymbol{Keccak256}\left(\begin{array}{c} \boldsymbol{seed \parallel (Keccak256 \circ FNV_{Mix})^4} \\ \boldsymbol{(digest)} \end{array} \right) \end{array} \right)$$

Converting to Go code *(indentation ignored)*:

```
const extraRounds = 4
extra := digest
for round := 0; round < extraRounds; round++ {
for i := 0; i < len(extra); i += 4 {
word := binary.LittleEndian.Uint32(extra[i:])
seedWord := binary.LittleEndian.Uint32(seed[((i/4)%10)*4:])
word = fnv(word, seedWord)
binary.LittleEndian.PutUint32(extra[i:], word)
}
extra = crypto.Keccak256(extra)
}
finalDigest := extra
finalHash := crypto.Keccak256(append(seed, finalDigest...))
return finalDigest, finalHash
}
```

Compatibility with Existing Mining Software

Because the extra mixing rounds are not part of the canonical Ethash specification, standard Ethash miners that are unaware of these additional steps will compute a different final hash and therefore will be incompatible. This change acts as a fork from standard Ethash, effectively requiring customized mining software or at least an update to existing miners to support the modified algorithm.

Built-in Miner

The built-in mining software is to be updated to support the new algorithm, allowing users to mine R5 with their CPU through the R5 JS Console. Standalone miners are also expected to be developed, both internally and by external contributors, to further facilitate the mining process.

Performance Implications and GPU Mining

The extra sequential rounds are designed to limit the advantage of hardware optimized for massive parallelism, such as GPUs. GPUs typically achieve high throughput by processing many independent operations at once. By introducing additional sequential computations, the modified algorithm reduces the level of parallelism available, which narrows the performance gap between GPU and CPU mining. While not "GPU-proof," it does lessen the disparity in mining performance across different hardware types.

5. SMART CONTRACTS

The proposed network inherits its smart contract functionality from Ethereum and the Ethereum Virtual Machine (EVM). The deployment of smart contracts can be done using any Ethereum-compatible IDEs, such as Remix or Hardhat.

It is important to note that the R5 codebase leverages the *Berlin* version of the EVM, which may require some fine tuning to ensure full compatibility with smart contracts built on EVM versions employed after Ethereum's move to PoS.

6. GAS PARAMETRIZATION

Part of our testing involved tweaking gas parametrization and nominal gas consumption fees to better understand how these could result in lower gas costs to the end-user.

It is unquestionable that tweaking such variables have no effect on the actual quantitative compute required to process transactions and validate blocks, however, it has a real impact on nominal gas consumption and the overall affordability of the network.

In our testing, we have managed to successfully run the network with lower nominal gas fees, however, the changes in the way gas consumption is calculated meant that most existing EVM applications could not efficiently understand how to properly calculate transaction costs, resulting in breakages in compatibility.

Metamask, specifically, could connect to our nodes, display balances correctly, but were not able to send transactions.

Due to the significant compatibility issues resulting from such changes, we have decided to retain the original gas calculation parametrization inherited from Ethereum.

The option of reviewing gas calculation and parametrization can be revisited in the future.

7. CONCLUSION

We have proposed a highly performant proof-of-work network that can rival proof-of-stake and other emerging consensus mechanisms, while retaining the inherent value proposition associated with the resource expenditure required for token creation.

The network achieves enhanced performance through increased block sizes, reduced block time intervals, better cache handling, a larger transaction pool buffer, and a much-improved consensus mechanism. Additionally, an improved variant of Ethash - termed Ethash-R5 - introduces a more efficient algorithm.

The network retains compatibility with the Ethereum Virtual Machine, with full support for smart contracts.

Collectively, these improvements demonstrate that R5 is not only scalable and efficient but also fully compatible with an existing ecosystem of applications, paving the way for next-generation decentralized applications.

8. REFERENCES

[1] Bitstamp, **"What Is Block Size?,"** https://www.bitstamp.net/en-gb/learn/crypto-101/what-is-block-size/, accessed 24-02-2025.

[2] Bitcoin.org, **"Bitcoin: A Peer-to-Peer Electronic Cash System,"** https://bitcoin.org/bitcoin.pdf, accessed 24-02-2025.

[3] Ethereum.org, **"Ethereum Whitepaper,"** https://ethereum.org/en/whitepaper/, accessed 24-02-2025.

[4] YCharts, **"Ethereum Average Block Size,"** https://ycharts.com/indicators/ethereum_average_block_size, accessed 26-02-2025.

[5] Etherscan, **"Ethereum Transactions Chart,"** https://etherscan.io/chart/tx, accessed 26-02-2025.

[6] Ethereum Yellowpaper, **"Ethereum Yellowpaper,"** https://ethereum.github.io/yellowpaper/paper.pdf, accessed 26-02-2025.

[7] Ethresear.ch, **"On Block Sizes, Gas Limits, and Scalability,"** https://ethresear.ch/t/on-block-sizes-gas-limits-and-scalability/18444, accessed 27-02-2025.

[8] Wikipedia, **"Tron (blockchain),"** https://en.wikipedia.org/wiki/Tron_(blockchain), accessed 27-02-2025.

[9] TRON Protocol Documentation, **"dPOS Overview,"** https://tronprotocol.github.io/documentation-en/introduction/dpos/#:~:text=Slot%3A%20In%20TRON%2C%20every%20three,TRON%20is%20approximately%20three%20seconds, accessed 27-02-2025.

[10] Reddit, **"Question About Network Congestion (Fantom Network),"** https://www.reddit.com/r/FantomFoundation/comments/sa9bkm/question_about_network_congestion/, accessed 27-02-2025.

[11] TRON Developers, **"TRON Block Documentation,"** https://developers.tron.network/docs/block, accessed 28-02-2025.

[12] MetaMask, **“HOW TO ESTIMATE THE GAS FEE,”** https://support.metamask.io/more-web3/learn/how-to-estimate-the-gas-fee, accessed 28-02-2025.

[13] Cointelegraph, **“PROOF-OF-AUTHORITY VS PROOF-OF-STAKE,”** https://cointelegraph.com/learn/articles/proof-of-authority-vs-proof-of-stake-key-differences-explained, accessed 28-02-2025.

[14] MegaETH, **“MegaETH Research,”** https://www.megaeth.com/research, accessed 03-03-2025.

[15] BNB Chain, **“opBNB Harware Requirements,”** https://docs.bnbchain.org/bnb-opbnb/developers/cheat-sheet/, accessed 03-03-2025.

[16] BitInfoCharts, **“Ethereum Average Block Size,”** https://bitinfocharts.com/comparison/ethereum-size.html#alltime, accessed 03-03-2025.

[17] Backblaze, **“Hard Drive Cost Per Gigabyte,”** https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/, accessed 03-03-2025.

[18] Bitquery.io, **“Understanding Blockchain Demand: Analyzing TPS and Throughput Needs,”** https://bitquery.io/blog/understanding-blockchain-demand-tps-throughput-analysis#:~:text=Ethereum%20has%20consistently%20maintained%20its,capacity%2C%20averaging%20around%2012%20TPS., accessed 03-03-2025.

[19] Wikipedia, **“Bitcoin scalability problem,”** https://en.wikipedia.org/wiki/Bitcoin_scalability_problem, accessed 03-03-2025.

[20] Metamask, **“Metamask,”** https://metamask.io/, accessed 03-03-2025.