MCV: Mutual Chain Voting

Asynchronous BFT Protocol

as a distributed registry   
for ULTRANET cyber community platforms

(continuously updated)

www.ultranet.org

***Disclaimer:*** *This document is for information purposes only. Ultranet Organization does not guarantee the accuracy of, or the conclusions reached in, this document, and this document is provided “as is”. Ultranet Organization does not make, and expressly disclaims, all representations and warranties, whether express, implied, statutory or otherwise, whatsoever, including, but not limited to (i) warranties of merchantability, fitness for a particular purpose, suitability, usage, title, or non-infringement; (ii) that the contents of this document are free from error; and (iii) that such contents will not infringe third-party rights. Ultranet Organization and its affiliates shall have no liability for damages of any kind arising out of the use of, reference to, or reliance on this document or any of the content contained herein, even if advised of the possibility of such damages. In no event will Ultranet Organization or its affiliates be liable to any person or entity for any damages, losses, liabilities, costs, or expenses of any kind, whether direct or indirect, consequential, compensatory, incidental, actual, exemplary, punitive, or special for the use of, reference to, or reliance on this document or any of the content contained herein, including, without limitation, any loss of business, revenues, profits, data, use, goodwill, or other intangible losses.*

# Contents

[Contents 3](#_Toc209910036)

[Glossary 4](#_Toc209910037)

[Introduction 5](#_Toc209910038)

[MCV 8](#_Toc209910039)

[Requirements 9](#_Toc209910040)

[Membership 10](#_Toc209910041)

[Vote/Block creation 10](#_Toc209910042)

[Round Aggregation 11](#_Toc209910043)

[Voting 11](#_Toc209910044)

[Quorum Randomization 12](#_Toc209910045)

[User Motivation 14](#_Toc209910046)

[Economics 16](#_Toc209910047)

[Conclusion 18](#_Toc209910048)

# Glossary

**Account** – a cryptographic DSA public/private key pair.

**Block (Vote)** – an element of a round that contains data to add to the database and a hash reference to the parent round.

**Voting round** – a round Rc is a voting of round Rp when p = c - Lag, where Lag is a constant.

**Member** – an account that declared itself as a potential block generator by placing a special transaction and notifying others that it is online. Members are eligible to generate blocks and participate in a consensus algorithm.

**Mempool** - is a blockchain node's mechanism for storing information on unconfirmed transactions, acting as a waiting room for transactions that have not yet been included in a block.

**Lag** – the number of rounds between the parent and the voting round.

**Node** – a network computer with special software that can receive and validate blocks from other nodes, send its own, relay received blocks to other nodes, and participate in the MCV consensus algorithm.

**Operation** – an atomic change in the database.

**Election round** – a round Rp is a parent of round Rc when c = p - Lag, where Lag is a constant. The rounds in the range [0… Lag] are called *genesis rounds* and have a parent round reference set to zero.

**Round** – a set of blocks that are sorted by their hashes. Each round has its own sequence index. Each member can place only one block per round.

**Roundchain** – a sequence of rounds where each block in each round has a reference to a parent round and each round refers to a previous one.

**Transaction** – a cryptographically signed data structure that contains a list of operations.

# Introduction

As the demand for decentralized systems continues to grow, consensus protocols have evolved to support faster processing, improved scalability, and more energy-efficient operations. However, existing solutions still struggle to balance performance with decentralization, resilience, and reliability. Most notable among them are Proof-of-Work (PoW), Proof-of-Stake (PoS), and a range of DAG-based and asynchronous Byzantine Fault Tolerant (ABFT) protocols.

Traditional PoW-based systems, such as those used in early blockchains, ensure security through computational work. While effective in deterring attacks, this model is inherently resource-intensive and introduces latency through competitive mining. These delays are artificial and prevent the network from scaling efficiently. Additionally, mining centralization has led to concerns over the concentration of power among a few dominant actors.

PoS protocols, while more energy-efficient, introduce new challenges. These systems rely on token-based voting, where influence is proportional to a participant’s stake. This approach raises concerns about fairness, as wealthy stakeholders can dominate decision-making and hinder new entrants. Furthermore, PoS systems often assume synchronous communication and may require additional mechanisms to ensure liveness and safety in partially connected or unpredictable networks.

DAG-based protocols represent another attempt to increase throughput by removing the linearity of blockchains. By allowing multiple blocks or transactions to coexist and reference one another, DAGs improve concurrency and speed. However, to prevent double-spending or forks, many DAG systems rely on coordinators or finality layers, which reduces decentralization. Some asynchronous models, such as Hashgraph, avoid this by using virtual voting. However, they are not permissionless, since only certified nodes are allowed to propose blocks, and their performance still tends to degrade as the network grows or traffic drops.

Across all these systems, a persistent challenge remains: the need to include external data in a trusted, decentralized way. Most protocols depend on oracles—external services that feed off-chain data into the blockchain. This reliance introduces vulnerabilities and trust assumptions that conflict with the core principles of decentralization.

In this fragmented landscape, the MCV Protocol introduces a fundamentally different approach. It avoids artificial delays, removes the need for leader selection or mining races, and enables fast, parallel block production without sacrificing consensus integrity. Instead of relying on external oracles, MCV treats network events themselves—such as node inactivity—as verifiable signals for consensus. With a fixed participant model and deterministic voting, it establishes a truly decentralized protocol designed for resilience, fairness, and full asynchronous operation.

Figure 1. The chain of block rounds

A

B

Members

C

D

E

F

Lag

Election round for Ri+4

- Vote with transactions

- Vote

Time

Lag

Lag

Ri

Ri+1

Ri+2

Ri+3

Ri+4

Ri+5

Ri+6

Ri+7

Hash references to Ri round with Lag=4

Partially filled rounds

Ri-1

Voting round for Ri

Lag

Lag

A reference to the previous round

Ri-2

Confirmed round

Confirmed round

# MCV

The MCV protocol (Mutual Chain Voting) is a voting protocol in which data is inserted in blocks. However, unlike some other protocols, there is no competition between the data transferred. The MCV is a distributed database in which all nodes insert blocks in iterations called rounds. In a round, each node has the right to insert a block of data into a common database. And if all the blocks are correct, they will all be inserted.

The MCV protocol is designed to achieve a throughput and cost close to those of DAG protocols but without sacrificing decentralization and scalability. It is based on a special data structure called Roundchain. Roundchain is similar to Blockchain, but instead of chaining single blocks, it chains sets of blocks called rounds. This allows all, or almost all, “members” to get their blocks added at each chain iteration, in contrast to a traditional blockchain, where only one “winner” can place their block onto a chain at a time. The MCV protocol, in turn, provides a simple voting mechanism that allows for reaching consensus for every round. Together, MCV and Roundchain make the technology inherently multithreaded, providing performance limited only by network throughput and without sacrificing decentralization and reliability.

A distinguishing feature of the MCV protocol is that each block represents information about the vote and also includes information to vote for in the next round.

Another feature of this protocol is the limited number of nodes. In each round, each node can propose only one block for insertion into the database.

This protocol introduces no artificial delays or leader selection mechanisms. Each node generates its block independently, and the system operates at the speed of communication between nodes—essentially, the speed of network delays. There is no need to solve complex puzzles or compete for inclusion. All correctly formed blocks are inserted in parallel, enabling high throughput without central coordination.

It is important to note that the user does not send the transaction to the mempool, but directly to the nodes. The system pseudo-randomly determines which node serves a specific address. Thus, the user cannot overload the system by sending trash transactions to several nodes at the same time. A limit of transactions per round and per node is maintained so that the time spent on a round remains adequate. This pseudo-random assignment prevents transaction flooding and ensures fair distribution of network load. As a result, users cannot manipulate the system by broadcasting identical transactions to multiple nodes.

The MCV protocol can register external events, such as a node failing to submit a block, and reach consensus on their occurrence. This mechanism allows MCV to act as a native oracle and opens the possibility for any other external data to be validated and integrated directly into the database, without trusted third parties.

By limiting the number of active participants and enforcing the removal of non-responsive or misbehaving nodes, the protocol maintains performance and reliability. Nodes are also subject to membership revocation through voting in case of protocol violations, such as forking. This ensures that consensus is always achieved among a functioning set of members and protects against monopolization or stagnation.

## Requirements

The idea behind MCV is to create a distributed ledger algorithm with the following requirements:

* No Proof-of-Work, as it has the lowest speed and highest hardware requirements among other algorithms
* No Proof-of-Stake, as it makes the richest miners take maximal profits, so the rich get richer. It is also relatively slow without sharding
* No traditional blockchain, due to the principle of “one winner – one block”, which makes it slow and expensive
* No DAG, as despite its high speed and cheapness, it requires masternodes to prevent forks, which have a negative effect on decentralization
* No Hashgraph and similar, as it’s permissioned and can operate fast only with a very limited number of members, beyond which the confirmation time grows dramatically
* Minimal effort to create blocks
* Lowest possible transaction fees
* More than one block can be accepted at a time.

The core of MCV is a data structure called Roundchain, which represents a 2D grid of blocks where the column is a timeline of a particular member (like the blockchain of a single member) and the row (round) is an obligation for each member to place its block there. A member is a DSA public key used to prove ownership of generated votes and end-point addresses for communication. The maximum count of members is limited by an arbitrary number.

Each block refers to a hash of the parent round with some delta called “Lag”. Each round refers to a hash of the previous round. The referencing logic between rounds builds a round chain—a graph of finalized hashes that acts as both a ledger and a cryptographic checkpoint system. This chaining of rounds enhances accountability, as each round cryptographically acknowledges the integrity of prior rounds without requiring immediate sequential delivery.

Every block or vote contains information about how it votes and for what. In turn, it can also include transactions (payload) to vote for in its voting round later. Transactions are an ordered list of operations where an operation defines an atomic change in the database.

The absence of a global mempool represents a deliberate architectural choice to minimize transaction duplication, reduce propagation latency, and remove contention for block space. By pseudo-randomly assigning transaction processing responsibility to specific nodes, the protocol ensures both fairness and scalability while mitigating traditional mempool issues.

## Membership

To become a candidate, users need to send a special operation and pay an energy fee. Each new candidate is inserted at the beginning of the candidate list. Thus, there is a higher probability that this candidate is online when required, as opposed to someone who added a transaction much earlier. That is, it is more profitable for a candidate to be on the top lines to get into the system. But each declaration has a price.

Inclusion into the system takes place in the following order: at the end of the round’s confirmation, it is determined whether there is the maximum allowed number of participants at this moment in time. If not, but there are candidate applications, then the system admits them as new members. After that, they can start generating their own blocks for the round that follows the voting round of the round where they joined. This can be improved by adding new participants with some margin at the same time, since it is unknown which of them is online, and then leaving only those who successfully sent their first blocks.

## Vote/Block creation

Any particular member can add only one block per round. These restrictions prevent the grid from growing infinitely in a horizontal direction. Vertical growth for each member, in turn, is limited by accepting only those blocks whose round lies in a specific range. This limitation is needed because the order in which a node receives blocks is not predictable for recent ones due to the nature of peer-to-peer networks. The advancement of this range is driven by a voting mechanism.

If a member creates two different blocks with the same round index, then the network treats them as a cheater and cancels its membership. Also, to prevent an attacker that owns two or more members at the same time from adding the same transaction to more than one block in the same round, each transaction can only be processed by the pseudo-randomly chosen member based on the member and signer account addresses. Another transaction constraint, “Expiration,” prevents a member from deferring a transaction placement indefinitely.

## Round Aggregation

Simultaneous block generation also poses the following question: at what point do we assume that the generation of all possible blocks is complete? That all the blocks that could have arrived have arrived.

During the voting process, we allow blocks to be generated for a certain period. To ensure a cryptographic chain, it is necessary for newly generated rounds to reference previous ones. But we cannot reference the previous round, since we do not know for sure that we have received all possible blocks. Therefore, the reference is not to the hash of the previous round, but to the hash of the N-th previously generated round. The round that is referenced is called the Election, and the one that it refers to is called the Voting. The next newly generated block references the hash of the round following the one referenced by the previous one. And so on. Advancement in the system occurs under the assumption that if we received two-thirds of all possible blocks in a round, then in the round before it, we received almost all possible blocks, and in the round before that, with an even higher probability, we received all possible blocks.

During aggregation, transactions from votes are arranged in a certain order and pre-executed (without adding actual changes to the database). If no violations are detected, for example, balance overrun, access violation, or any other errors, then all operations in all transactions are assumed to execute in the database. If errors are found in an operation, the whole containing transaction is excluded from the addition, and the remaining transactions are added. Thus, the entire block does not necessarily have to be error-free. Since blocks are generated in parallel, it is impossible to guarantee their complete correctness.

After that, unnecessary information is removed and all valid transactions are hashed. Then a vote that refers to a newly aggregated round can be created (with its own transactions if any) in this way proposing its own version of the parent round to other members. Accordingly, all the other member nodes generate their own votes with their version of the parent hash.

Additionally, if we believe that we have not received a block in a round, then in the vote in a special field, we mark that we propose to cancel membership for this node due to its inactivity.

## Voting

Voting occurs by pseudo-randomly selected blocks (a maximum of 21). The majority of them are selected based on hash values. If the condition for the required number of matching hashes is met in a voting round, then this means that the network has reached a consensus on the state of the parent round. That is, this random voting leads to consensus, and consensus in turn leads to confirmation of the round from which the hash for voting was calculated. Confirmation of the round means that these transactions will be executed and changes applied to the database of all participants.

If, during a round, a node has proposed a hash that is different from the majority, it means that, for some reason, it’s computed from a different list of transactions of the parent round. Considering that the transaction generation logic is the same for all blocks, this node most likely did not receive all the data and must perform the resynchronization procedure. During this process, it temporarily falls out of consensus but can rejoin in subsequent tries once resynchronization is complete.

To maintain liveness and prevent indefinite deadlocks, the protocol introduces a retry mechanism based on a series of "try" attempts. In each new try, the set of voting blocks is reselected using a pseudo-random function that takes the try number into account. This dynamic reshuffling ensures that even in the presence of communication delays, temporary desynchronization, or missing data, honest nodes can eventually re-align with the network’s majority state. Each try provides a fresh opportunity for consensus, improving resilience under unstable conditions. However, if a node consistently produces diverging hashes across multiple tries, it is flagged as potentially faulty or non-compliant and may be subject to removal through the protocol’s voting-based exclusion process.

At the same time, well-behaved nodes are rewarded with a higher rating, reflecting their reliability. But to prevent such participants from remaining in the system indefinitely, this rating gradually decreases over time, simulating “aging” and ensuring natural rotation. This design guarantees that even in adverse network conditions, the system remains agile, self-healing, and capable of progressing without centralized intervention.

## Quorum Randomization

How are the blocks that will vote selected? To enhance robustness and fault tolerance, the protocol’s dynamic pseudo-random selection mechanism adapts to fluctuating network conditions and node availability. We select a pseudo-random maximum of 21 votes based on the state of the last confirmed round and the current try number. When the try parameter changes, the randomness also changes, since it takes try into account when calculating. Thus, different sets of voters are obtained with different tries.

This is done to minimize the number of votes required to reach consensus and avoid situations where consensus cannot be reached because a lot of participants have gone offline. And by changing a set of required votes, we minimize the probability of such an outcome. If the network is deployed and working, then at least 2/3 of the maximum of 21 votes (ie, 14 votes) with the same parent hash is enough to reach consensus.

By continuously varying the voting subset across retries, the system mitigates the risk of network partitions or temporary node outages impeding consensus progress. This approach also prevents adversarial manipulation of the voting set, since the selection is unpredictable and dependent on both the round data and the try number. Moreover, the fixed size of 21 voting blocks represents a practical compromise between ensuring sufficient diversity for Byzantine resilience and maintaining efficient communication overhead. The feature enables the network to gracefully tolerate even more than one-third of faulty or offline participants. Together, these design choices enable the protocol to scale effectively in large, decentralized deployments and maintain liveness even in volatile or partially adversarial environments.

For each round, the algorithm expects all or the top Mmax members to either create a vote with or without transactions, either of them acting as a vote for a parent round. In this way, members can vote for a particular candidate (a subset of blocks) of a parent round if more than one exists. The parent round with the most votes is considered to be the winner. There must be M\*2/3 of blocks of chosen members collected by any round Ri to elect round Ri-Lag. As soon as the Ri+Lag round voted, which means Ri is elected and all [R0 … Ri-‍1] rounds are confirmed, then round Ri is also considered as confirmed. In other words, for any round elected by a corresponding voting round, if all previous rounds are confirmed, then this round is also flagged as confirmed.

## User Motivation

The users’ motivation to participate in the system is based on their main resource – the stability of the Internet connection, which cannot be fully guaranteed. Unlike some other consensuses, where everything is quite predictable: you can calculate and capture the hash rate or buy tokens and reserve the probability. Here, everything is more chaotic, since the slightest failure leads to the impossibility of sending a block and the interruption of membership. With a large queue of candidates, it will be just as difficult to return. It turns out that this consensus attracts enthusiasts who can afford downtime and waiting in line.

Participants who run multiple nodes will be able to minimize the probability of failures, since the nodes will back each other up. However, it is impossible to reduce it to zero. Therefore, it is impossible to establish a status quo or monopoly in this consensus. The protocol can also include a mechanism of forced rotation, allowing the removal of participants who have remained in the system for too long.

Now there are motivating factors in the form of ratings, which increase stability. But a demotivating factor will appear in the form of unpredictable participation.

It should be noted that the system does not store the entire transaction log, which significantly reduces the size of the database. Only relevant data needs to be retained, while historical records can be stored optionally.

Figure 2. Voting and Confirmation

Ri

Ri+1

Ri+2

Ri+3

Ri+4

Ri+5

Ri+6

Ri+7

Ri+8

Ri+9

Ri+10

Ri+11

Ri+12

Ri+13



Lag = 8

Voted for Ri-8

Confirmed by Ri+8

Voted for Ri-7

Voted for Ri-5

Collecting votes

Collecting votes

Collecting votes

Voted for Ri-6

Confirmed by Ri+9

Confirmed by Ri+10

Voted for Ri-4

Voted for Ri-3

Voted for Ri-2

Voted for Ri-1

Voted for Ri

Voted for Ri+1

Voted for Ri+2

# Retrieving Data from the Outside World

The peculiarity of this protocol is the opening of possibilities, thanks to the removal of members. The removal of a participant occurs because they did not send a block, which can be considered as an external event, or some information about the outside world. Most blockchains do not have such capabilities. Namely, the presence of the asynchronous voting mechanism for an external event in the protocol allows importation of any other external data into the network.

This ability treats node behavior—such as silence, failure to vote, or invalid participation—as an external signal that effectively transforms the protocol into a built-in oracle system. Rather than depending on third-party services to inject off-chain data, MCV uses internal consensus to detect and validate real-world events observable through network activity. As a result, the network itself becomes a trustless interpreter of external conditions, enabling decentralized automation of smart contracts and interchain communication without introducing external dependencies or oracle vulnerabilities. This approach redefines how external truth can be integrated into a blockchain, making the system inherently extensible and autonomous. This also allows two similar networks to communicate with each other, reach consensus, and exchange data without any mediator.

# Economics

The protocol introduces the concept of a token which is not tied to financial value but to the resources of the network itself. These resources are of two types: the energy required to execute transactions, and the storage space measured in byte-days, referred to as space-time. Every transaction consumes Energy, with the current rate set at 1 operation = 1 Energy. The unit of Energy is deliberately small, ensuring that transaction costs remain minimal while still limiting excessive applications and preventing spam.

# Conclusion

The current landscape of consensus protocols remains a patchwork of partial solutions, each addressing performance, scalability, or decentralization—but rarely all three simultaneously. From energy-intensive Proof-of-Work models to oligarchic tendencies in Proof-of-Stake systems, and from the structural complexity of DAG protocols to the inherent limitations of oracle-based architectures, the industry continues to face unresolved technical and systemic challenges. Many of these protocols require trade-offs that compromise reliability, inclusiveness, or true decentralization, leaving the market far from optimized or mature.

Against this backdrop, the MCV Protocol offers a novel and robust alternative. Its core mechanisms—parallel block production, deterministic voting, native integration of external signals, and leaderless consensus—demonstrate that it is possible to design a system that is not only scalable and efficient, but also fair, resilient, and genuinely decentralized. By removing unnecessary layers of complexity and rejecting outdated assumptions, MCV introduces a model that simplifies the path to consensus while preserving system integrity and transparency.

However, the emergence of technically sound protocols is only one part of the solution. For such innovations to impact the industry at scale, they must be effectively communicated, implemented, and adopted. The role of strategic development and market-facing education cannot be overstated. In a space often dominated by hype, marketing-driven design, and speculative priorities, protocols like MCV represent a return to foundational, well-engineered principles.