MCASHCHAIN

DECENTRALIZED AND SOCIAL ELECTRONIC PAYMENT PLATFORM

Satoshi Nguyen | David Nguyen | Nakamoto Ngo | Duc Nguyen
{snguyen, david, nngo, dnguyen}@mcash.network

MCASH Foundation | 6th Jun 2019, Hanoi

WHITE PAPER VERSION: 0.8.3
MCASH Protocol version: 0.1.0

1. INTRODUCTION

1.1 Vision

MCashChain aims to provide a solution for one of our ambitious projects which aims to challenge Ebay's Model on Blockchain. As we all know, the lengthy block confirmation time and the high gas price have made it a great challenge to auction, bid or make use of the smart contracts. Moreover, the greater adoption of cryptocurrencies require the public blockchain to achieve high throughput at low fees to enable micropayment for E-commerce and high speed for online decentralized gaming experience. We are working towards a blockchain-based platform for electronic payment which is more secure, transparent, highly scalable and available for DApps in our ecosystem and others.

1.2 Background

Masternode Systems are part of many PoS (Proof of Stake) cryptocurrencies. When a coin holder has enough coins for a masternode, he or she can send a specific amount of coins into a node's wallet, and they earn newly minted coins when blocks are produced. The system was developed to replace the Proof of Work system invented by Satoshi Nakamoto as part of Bitcoin [1].
The masternode software can be thought of like a different version of mining, with far less power consumption. The masternode communicates with other nodes in th e network and keeps a copy of the blockchain, constantly checking and updating it.
Some of the special functions that these nodes perform are:

- Increasing privacy of transactions
- Doing instant transactions
- Participating in governance and voting
- Enable budgeting and treasury system in cryptos
- Dash (2014) [3], Zcoin (2017) [4], TRON (2018) [5] are 3 of successful blockchains with Masternode that we would like to inherit their development and innovation into our platform.

1.3 Practical & Applications

The application of MCashChain is unlimited with high speed transaction, zero fees for ordinary users and high scalability in nature. Yet, the first layer of application that we will be focusing on developing are as follows:

1.3.1 Gaming

MCashDice and a dozen of existing games by GemMob Studio will be developed, converted and ported to MCashChain. MCashDice will no longer run on Tomochain.

1.3.2 Auction

Auction smart contract will be developed on MCashChain for several art and antique auction houses. Companies and startups stocks and common shares auction will first be conducted on MCashChain smart contract.

1.3.3 C-Commerce

Midashimaya will be developed on MCashChain for marketplace of both physical products and virtual assets such as virtual collectible items, virtual land ownership, virtual preferred company shares.

1.3.4 Art stock exchange (Non-Fungible Token M721)

Distributing of co-ownership titles of physical art pieces through the use of M721 token issued on MCashChain will also be studied and developed in collaboration with Chon Auction House, the largest art auctioneer in Vietnam and supported by many high networth art collectors.

1.3.5 Identification & Certification

In collaboration with Blockchain Excellence Search and Training Center, many in-depth courses relating to development on Blockchain, Smart Contract, dApp building, etc will be offered to students, developers in Vietnam and Singapore. Certification will be deployed on MCashChain.

1.3.6 Voting and Election

In collaboration with public sector, we will demo voting and election solution on MCashChain to government agencies in the ASEAN region with the help of Vietnam Chamber of Commerce in Singapore and various associations which will first use our solution for their voting cast such as Vietnamese Association in Singapore. We will also offer our solution to Singapore Business Federation and Singapore Fintech Association.

1.3.7 Gamification for Education Sector

Knowing that children are drawn towards gaming more than courses, we will work together with Arrow-HighTech and Fingerprint English to develop a series of Education Gaming using Token Economy to encourage learning experience on MCashChain. The first several learning games will be for English and Soft Skills teaching for children from 5 to 14 years old.

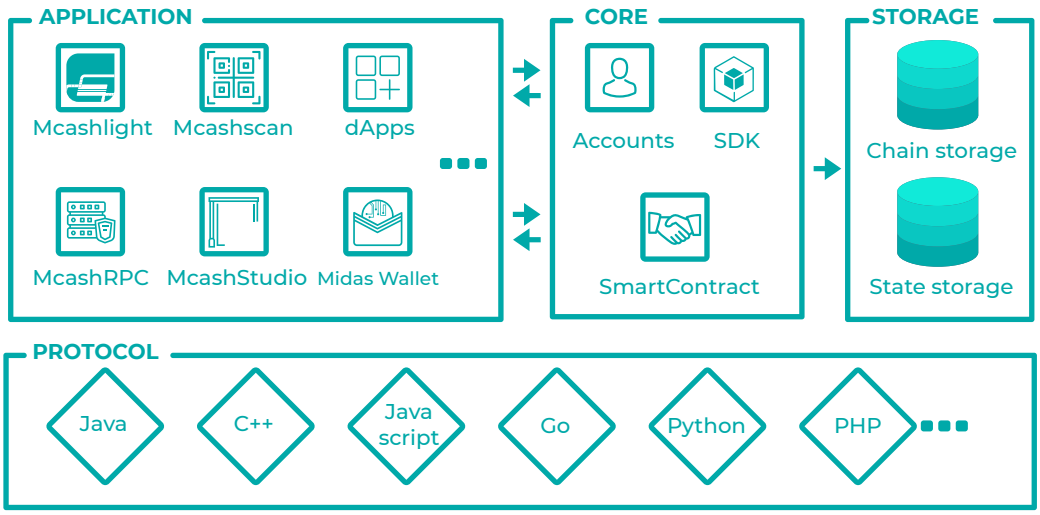
1.3.8 Blockchainization as a Service

MCashChain private testnet has been completed. Public testnet is set on 6th June 2019 and Public Mainnet is set on 26th June 2019. We design our chain for the purpose of Blockchainization Strategy. We will help converting existing apps and platforms into dApps and decentralized platforms on MCashChain. We have a good line-up of existing products and services waiting to be converted and launched on MCashChain, using MCash to fuel and to serve existing customers directly from Midas Wallet[10]. This would bring in more mainstream users, help them move from apps to dapps and create much more utility for Midas Wallet.

II. ARCHITECTURE

McashChain uses 3-layer architecture divided into Storage Layer, Core Layer, and Application Layer.

The protocol adheres to Google Protobuf, which can be extended to support a new language easily via plugins.



2.1. Core

The core layer includes consensus, account management and smart contracts. A stack-based virtual machine is implemented on McashChain with an optimized and robust instruction set.

McashChain's consensus is based on Delegated Proof of Stake Voting (DPoSV) and many features were introduced to make the network's original objectives.

For smart contract language we choose Solidity since its widely supported by a large community & developers.

2.2. Storage

McashChain is transaction-based "state" machine, and its storage layer consists of Chain Storage and State Storage

2.2.1. Chain storage

McashChain's Chain storage uses Level DB, a great storage engine developed by Google for local storage applications.

2.2.2. State storage

The root node hashes of the transaction trie, state trie and receipts trie are stored directly in the blockchain. McashChain's State storage uses Tron's KhaosDB in the full-node memory.

2.3. Protocol

Mcashchain provides both Protobuf API and HTTP Restful API. Protobuf eases the client development, the API .proto are also available for many programming languages (C++, Java, Python, Golang, etc...). HTTP API is more ready for javascript clients (such as Nodejs).

2.5. LightningSend

LightningSend is a feature that utilizes transfer locking and supernode consensus to facilitate instantaneous transactions on McashChain. LightningSend allows MCASH to compete with existing centralized payment platforms such as VISA/Master, who offer rapid transaction times. McashChain's LightningSend technology offers this, but in a decentralized and trustless manner.

LightningSend feature uses transaction locking mechanism to prevent double-spends in the network. Currently, in order for merchants to protect against double-spends in systems such as Bitcoin, they must usually wait until a block has been confirmed, to ensure that the transaction being sent is valid. However, the limitation with this is that, on average, it takes 10 minutes for a block on the Bitcoin blockchain to be confirmed. Transaction locking is designed to improve upon the manner in which double-spends are currently being dealt with in existing cryptocurrency systems, which in turn produces significantly faster transaction times.

2.4. MCASH Virtual Machine

MVM is 100% compatible with EVM (Ethereum Virtual Machine [2]). McashChain supports all EVM-compatible smart-contracts, protocols, and atomic cross-chain token transfers. This means that any smart-contract and dApp written in Ethereum protocol can be seamlessly ported to McashChain.

2.6. Sharding

We implement sharding technique similar to DBMS (Database Management System) sharding, where rows of a database table are held separately, rather than being split into columns (which is what normalization and vertical partitioning do, to differing extents). Each partition forms part of a shard, which may in turn be located on a separate database server or physical location.

The idea is to divide the blockchain's state and transaction processing into shards, each of which is processed by a separate set of nodes. This means that individual nodes now only have to store the state of their own shard and process only a subset of the transactions, improving overall transaction throughput.

2.7. Unidex Exchange

Since the Bitcoin network came into existence in 2009, many new chains have been created. Each chain has different aims and purposes. However, this movement has created another layer of obstacle for global adoption of cryptocurrencies. Some of the research effort has resulted in the creation of wrapped cryptocurrency of one chain in order to be used on another. The typical example is WBTC (Wrapped Bitcoin), an ERC-20 token to be used on Ethereum.

In our view, each chain somehow creates a nation of its own with its ecosystem and applications. In order to bring the usefulness of cryptocurrencies to the next adoption level globally, more efforts need to be done for cross-chain atomic swap and cross-chain smart contract. This multi-dimensional interaction complexity is one of the areas of research at MCash Foundation.

We propose that, by using cross-chain smart contract to wrap MCASH, we will be able to issue n types of MCASH on n chains. On the chain i, the wrapped MCASH will be called i-MCASH. The total supply amount of all i-MCASH is still exactly equal to total supply of MCASH. One type of i1-MCASH is transferred to the cross chain smart contract, the corresponding i2-MCASH will be automatically issued out.

We select these chains for the first phase of development: NEO, ZIL, EOS and TRON with respective wrapped cryptocurrency named: z-MCASH (on ZILLIQA chain), e-MCASH (on EOS chain), n-MCASH (on NEO chain) and t-MCASH (on TRON chain). All the iMCASH would be able to convert to MCASH with the ratio 1 to 1 using hybrid cross chain smart contract. i-MCASH would be able to swap to any other MCASH on Midas wallet which support multiple chains.

2.8. ZMC (Zero Mcash)

Due to the public nature of the blockchain, users may have their privacy compromised while interacting with the network. To address this problem, third-party coin mixing service can be used to obscure the trail of cryptocurrency transactions. In May 2013, Matthew D. Green and his graduate students (Ian Miers and Christina Garman) proposed the Zerocoin protocol where cryptocurrency transactions can be anonymised without going through a trusted third-party, by which a coin is destroyed then minted again to erase its history [7]. While a coin is spent, there is no information available which reveal exactly which coin is being spent.

We aim to deploy ZMC (Zero MCash) on top of McashChain without making any changes to the base layer. We propose an analysis of the prototcol privacy promises and argue that information leakages intrinsic to the use of this protocol are controlled and well-defined, which makes it a viable solution to support private transactions in McashChain.

2.9. Implementation

The McashChain core is implemented in Java and was originally a fork from Java-Tron.

III. CONSENSUS

3.1. Delegated Proof of Stake Voting (DPoSV)

When a Bitcoin miner confirms a block, he is rewarded Bitcoin as an incentive. If the price and popularity continue to rise for Bitcoin so will the number of transactions. As the number of transaction increases, more miners will join the community. It is obvious that more mining means more energy consumption. The total power consumption of Bitcoin mining has been estimated to be equal to Ireland's power consumption, and it will be tripled in the near future.

The Proof of Stake (PoS) was created as an alternative to the Proof of Work (PoW), in order to solve energy cost problem. With Proof of Stake, owners create blocks rather than miners, and do not require power spending machines that produce as many hashes per second as possible. Because of that, the energy consumption of Proof of Stake is negligible compared to Proof of Work.

However, the problem with standard PoS is that validator influence correlates directly to the amount of tokens locked up.
This results in parties hoarding large amounts of the network's base currency wielding undue influence in the network ecosystem.

McashChain consensus mechanism uses an innovative Delegated Proof of Stake Voting system in which 64 Super Nodes (SNs) produce blocks for the network. Every 2 hours, Mcash Xnode holders can vote for a selection of SN candidates, with the top 64 candidates deemed the SNs. Voters may choose SNs based on criteria such as projects sponsored by SNs to increase Mcash adoption, and rewards distributed to voters.

SNs' accounts are normal accounts, but their accumulation of votes allows them to produce blocks.
McashChain network generates one block every 3 seconds, with each block awarding 10 MCASH (the first 2 years) MCASH to SNs and other voters (XNodes). A total of about 105,120,000 MCASH will be awarded annually in the first 2 years.

Each time an SNs finishes block production, rewards are sent to a sub-account in the super-ledger. SNs can check, but not directly make use of these tokens. A withdrawal can be made by each SNs once every 24 hours, transferring the rewards from the sub-account to the specified SN account.

The 3 types of nodes on the McashChain network are Witness Node, Full Node, and Solidity Node.
Witness nodes are set up by SNs and are mainly responsible for block production and proposal creation/voting. Full nodes provide APIs and broadcast transactions and blocks. Solidity nodes sync blocks from other Full Nodes and also provide indexable APIs.

3.2. Tiered Xnodes

Node type	Stake amount	Voting power	Bonus	Staking (incl bonus)
Masternode (Supernode Candidate)	5,000,000 MCASH	20000	20%	6,000,000 MCASH
Jedi Node	500,000 MCASH	1800	15%	575,000 MCASH
Guardian Node	50,000 MCASH	150	10%	55,000 MCASH
Warrior Node	10,000 MCASH	25	5%	10,500 MCASH
Apprentice Node	5,000 MCASH	10	0%	5,000 MCASH

Earning at each node level increases linearly with the stake amount and plus the bonus (see Appendix A). For example, the Jedi Node earns 115 times reward than Apprentice Node. We believe this system will encourage people to stake more MCASH to reach the next level, helping to create a healthy rewarding system.

IV. ACCOUNT

4.1. Types

The 3 types of accounts in the Mcash-chain are regular accounts, token accounts, and contract accounts.

- Regular accounts are used for standard transactions.
- Token accounts are used for storing M1 tokens.
- Contract accounts are smart contract accounts created by regular accounts and can be triggered by regular accounts as well.

4.3. Structure

The three different account types are Normal, Asset Issue, and Contract. An Account contains **7 PARAMETERS:**

- *ACCOUNT_NAME:* the name for this account.
- *TYPE:* what type of this account is.
- *BALANCE:* balance of this account.
- *VOTE:* received votes on this account.
- *ASSET:* other assets expected MCASH in this account.
- *LATEST_OPERATION_TIME:* the latest operation time of this account.

4.2. Creation

There are 3 ways to create a MCASH account:

- Create a new account through API
- Transfer MCASH into a new account address
- Transfer any M1 token into a new account address

An offline key-pair consisting of an address (public key) and a private key, and not recorded by the Mcashchain, can also be generated. The user address generation algorithm consists of generating a key-pair and then extracting the public key (64-byte byte array representing x, y coordinates). Hash the public key using the SHA3-256 function and extract the last 20 bytes of the result. Add 32 to the beginning of the byte array and ensure the initial address length is 21 bytes. Hash the address twice using SHA3-256 function and take the first 4 bytes as verification code. Add the verification code to the end of the initial address and obtain the address in base58check format through base58 encoding. An encoded Mainnet address begins with M and is 34 bytes in length.

V. BLOCK

A block contains one block header and some (0 to many) transactions.

5.1. Block Header

A block header contains raw_data, witness_signature, and blockID.

5.1.1. Raw Data

Raw data is denoted as raw_data in Protobuf. It contains the raw data of a message, containing 6 parameters:

- *timestamp:* timestamp of this message.
- *txTrieRoot:* the Merkle Tree's Root.
- *parentHash:* the hash of the last block.
- *number:* the block height.
- *version*
- *witness_address:* the address of the witness packed in this block.

5.1.2. Witness Signature

Witness signature is denoted as witness_signature in Protobuf, which is the signature for this block header from the witness node.

5.1.3. Block ID

Block ID is denoted as blockID in Protobuf. It contains the atomic identification of a block. A Block ID contains 2 parameters:

- *hash:* the hash of block.
- *number:* the hash and height of the block.

5.2. Transaction

5.2.1. Signing

Mcashchain's transaction signing process follows a standard ECDSA cryptographic algorithm to ensure that funds can only be spent by their rightful owners.

- *private key:* A secret number, known only to the person that generated it. A private key is essentially a randomly generated number.
- *public key:* A number that corresponds to a private key, but does not need to be kept secret. A public key can be calculated from a private key, but not vice versa. A public key can be used to determine if a signature is genuine (in other words, produced with the proper key) without requiring the private key to be divulged. Public keys are either compressed or uncompressed. Compressed public keys are 33 bytes, consisting of a prefix either 0x02 or 0x03, and a 256-bit integer called x. The older uncompressed keys are 65 bytes, consisting of constant prefix (0x04), followed by two 256-bit integers called x and y (2 * 32 bytes). The prefix of a compressed key allows for the y value to be derived from the x value.
- *signature:* A number that proves that a signing operation took place. A signature is mathematically generated from a hash of something to be signed, plus a private key. The signature itself is two numbers known as r and s. With the public key, a mathematical algorithm can be used on the signature to determine that it was originally produced from the hash and the private key, without needing to know the private key.

5.2.2. Bandwidth Model

Ordinary transactions only consume bandwidth points, but smart contract operations consume both energy and bandwidth points. There are two types of bandwidth points available. Users can gain bandwidth points from freezing MCASH, while 10000 free bandwidth points are also available daily.

When a MCASH transaction is broadcast, it is transmitted and stored in the form of a byte array over the network. Bandwidth Points consumed by one transaction = number of transaction bytes multiplied by bandwidth points rate. For example, if the byte array length of a transaction is 200, then the transaction consumes 200 bandwidth points. However, if a MCASH or token transfer results in the target account being created, then only the bandwidth points consumed to create the account will be deducted, and additional bandwidth points will not be deducted. In an account creation scenario, the network will first consume the bandwidth points that the transaction initiator gained from freezing MCASH. If this amount is insufficient, then the network consumes the transaction initiator's MCASH.

In standard MCASH transfer scenarios from one MCASH account to another, the network first consumes the bandwidth points gained by the transaction initiator for freezing MCASH. If that is insufficient, it then consumes from the free 10000 daily bandwidth points. If that is still not enough, then the network consumes the MCASH of the transaction initiator. The amount is calculated by the number of bytes in the transaction multiplied by 1000 Matoshi. Thus, for most MCASH holders who may not necessarily freeze their MCASH to participate in SuperNode voting, the first step is automatically skipped (since MCASH balance frozen = 0) and the 10000 daily free bandwidth powers the transaction.

For M1 token transfers, the network first verifies whether the total free bandwidth points of the issued token asset are sufficient. If not, the bandwidth points obtained from freezing MCASH are consumed. If there is still not enough bandwidth points, then it consumes the MCASH of the transaction initiator.

5.2.3. Fee

MCASH network generally does not charge fees for most transactions, however, due to system restrictions and fairness, bandwidth usage and transactions do take in certain fees.

Fee charges are broken down into the following categories:

- Normal transactions cost bandwidth points. Users can use the free daily bandwidth points or freeze MCASH to obtain more. When bandwidth points are not enough, MCASH will be used directly from the sending account. The MCASH needed is the number of bytes * 1000 Matoshi.
- Smart contracts cost energy but will also need bandwidth points for the transaction to be broadcasted and confirmed. The bandwidth cost is the same as above.
- All query transactions are free. It doesn't cost energy or bandwidth.

Mcashchain also defines a set of fixed fees for the following transactions:

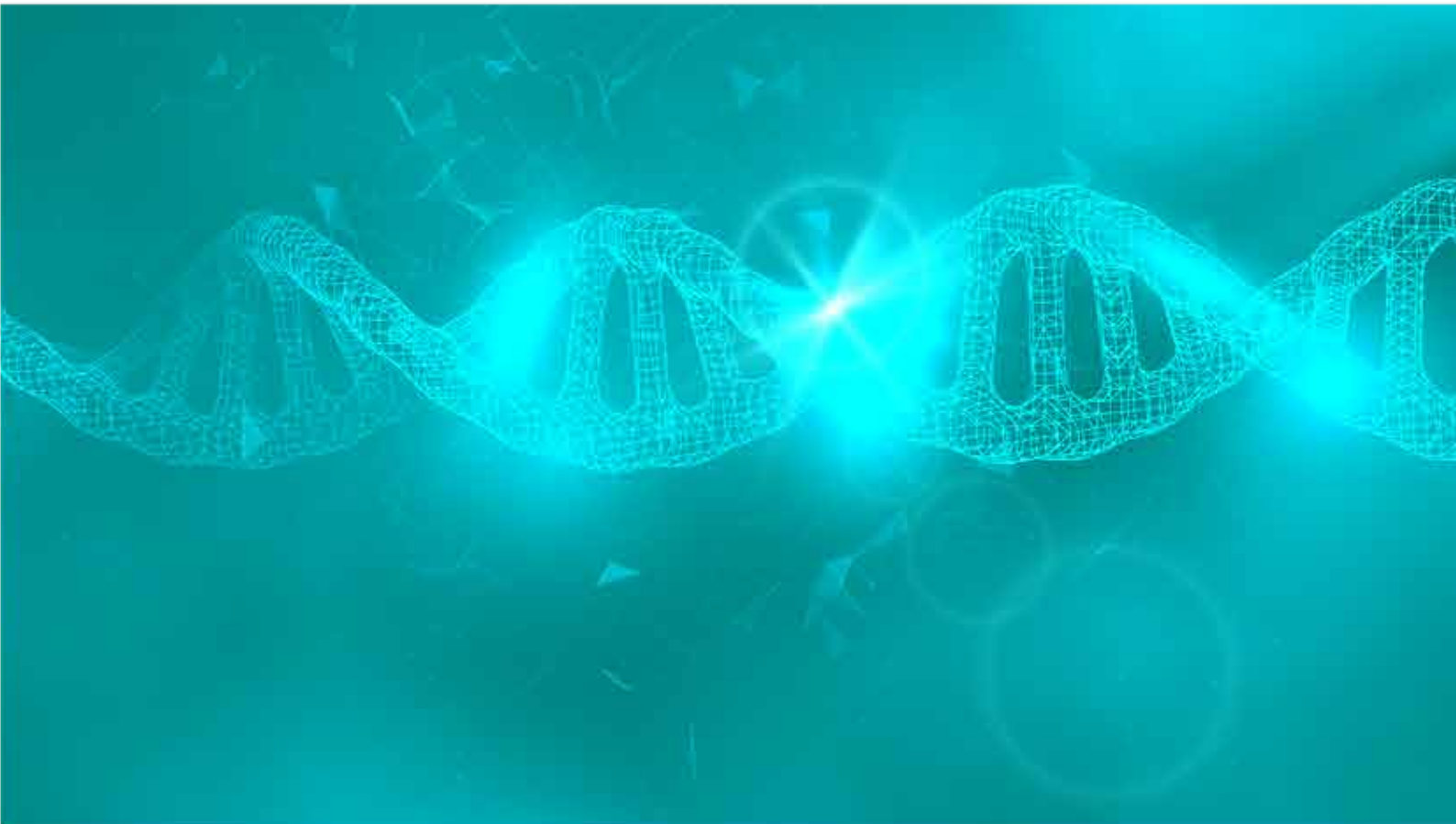
- Creating a witness node: 10000 MCASH
- Issuing a M1 token: 1024 MCASH
- Creating a new account: 0.1 MCASH
- Creating an exchange pair: 1024 MCASH

5.2.4. Transaction as Proof of Stake (TaPoS)

Mcashchain uses TaPoS to ensure the transactions all confirm the main blockchain, while making it difficult to forge counterfeit chains. In TaPoS, the networks require each transaction include part of the hash of a recent block header. This requirement prevents transactions from being replayed on forks not including the referenced block, and also signals the network that a particular user and their stake are on a specific fork. This consensus mechanism protects the network against Denial of Service, 51%, selfish mining, and double spend attacks.

5.2.5. Transaction Confirmation

A transaction is included in a future block after being broadcast to the network. After 19 blocks are mined on Mcashchain (including its own block), the transaction is confirmed. Each block is produced by one of the top 64 Super Nodes in a round robin fashion. Each block takes ~3 seconds to be mined on the blockchain. Time may slightly vary for each Super Node due to network conditions and machine configurations. In general, a transaction is considered fully confirmed after about 60 seconds.

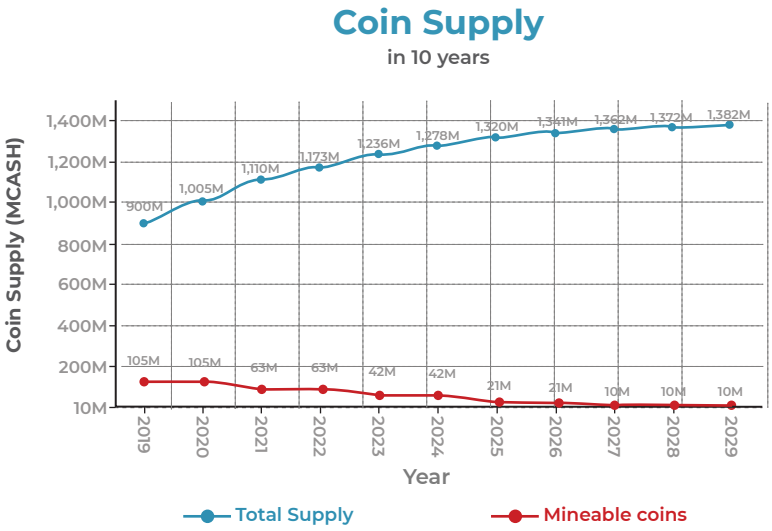


VI.TOKEN EMISSION SCHEDULE

6.1. Token distribution

The total amount of tokens at the genesis block is 900 million MCASH tokens in circulation: 500 million are reserved for MAS snapshot distribution over the next 3.5 years; 100 million are reserved for the team vested over the next 5 years; 150 million to airdrop for acquiring users from similar chains (Ethereum, Tron, Zilliqa, Tezos, Zcoin, etc...); and 150 million are reserved for Mcash Foundation which will run the first 30 Supernodes (profit from running nodes will use to develop an ecosystem and strategic partner fund).

After the mainnet launching, the block reward for the 1st and 2nd year is 10 MCASH (105 million MCASH annually); the block reward for the 3rd, 4th year is 6 MCASH (63 million MCASH annually); 5th and 6th year is 4 MCASH (42 million MCASH annually); 7th and 8th year is 2 MCASH (21 million MCASH annually). From 9th year onward the block reward will fix at 1 MCASH (10 million MCASH annually).



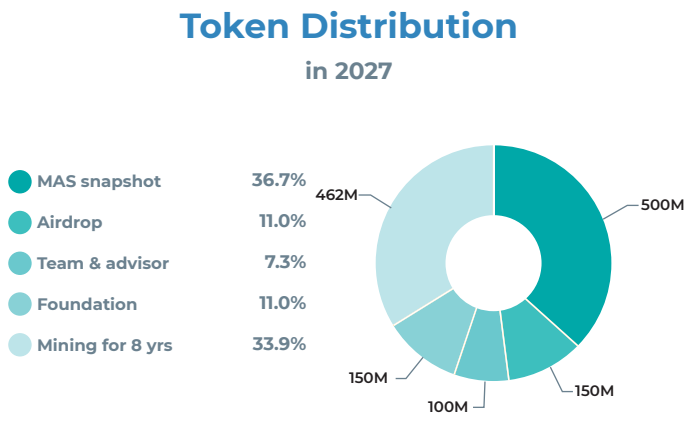


Figure 3. Coin Supply in 10 years

Figure 2. *Token Distribution after 8 years (unit: million MCASH)*

6.2. Implementation

Each epoch consists of 2400 blocks, which will reward a total of 24,000 MCASH in the first 2 years

The portion of 20% (4,800 MCASH), called “Infrastructure Reward” will be divided to all the SNs proportional to the number of blocks they produced during the epoch

The portion of 80% (19,200 MCASH), called “Staking Reward” will be shared proportionally (with bonus -Section 3.2) based on the token staking amount.

VII. GOVERNANCE

7.1. Supernode (SN)

Every account can apply and have the opportunity to become a Supernode (denoted as SN). Every other node can vote for SN candidates (see 3.2 for other nodes' voting power). The top 64 candidates with the highest votes will become SNs with the right and obligation to produce blocks. The votes are counted at the end of every epoch (2400 blocks - approx 2 hours) and the SNs will change accordingly.

If an SN performs badly, they will get Slashed and a minor penalty (to not produce blocks in the next 6 epoches). At the epoch after a minor penalty, if the node performance is not improved, a major penalty (24 epoches) will be given.
To prevent malicious attacks, there is a cost to becoming an SN candidate. When applying, 10,000 MCASH will be burned from the applicant's account. Once successful, such account can join the SN election.

7.2. Committee

The committee is used to modify MCashChain dynamic network , such as block generation rewards, transaction fees, etc. The committee consists of the 64 SNs in the current round. Each SN has the right to propose and vote on proposals. When a proposal receives 33 votes or more, it is approved and the new network parameters will be applied in the next maintenance period (3 days).

7.2.1 Dynamic Network Parameters

0. MAINTENANCE_TIME_INTERVAL

Description	Modify the maintenance interval time in ms. Known as the SN vote interval time per round.
Example	[2 * 3600 * 1000] ms - which is 2 hours.
Range	[3 * 64 * 1000, 24 * 3600 * 1000] ms.

1.ACCOUNT_UPGRADE_COST

Description	Modify the cost of applying for SN account.
Example	[999,900,000,000] Matoshi - which is 9,999 MCASH.
Range	[0,10 000 000 000 000 000 000] Matoshi.

2. CREATE_ACCOUNT_FEE

Description	Modify the account creation fee.
Example	[100,000,000] Matoshi - which is 1 MCASH.
Range	[0,10 000 000 000 000 000 000] Matoshi.

3. TRANSACTION_FEE

Description	Modify the amount of fee used to gain extra bandwidth.
Example	[10] Matoshi/byte.
Range	[0,10 000 000 000 000 000 000] Matoshi/byte

4. ASSET_ISSUE_FEE

Description	Modify asset issuance fee.
Example	[102,400,000,000] Matoshi - which is 1024 MCASH.
Range	[0,10 000 000 000 000 000 000] Matoshi.

5. WITNESS_PAY_PER_BLOCK

Description	Modify SN block generation reward. Known as unit block reward.
Example	[3,200,000,000] Matoshi - which is 32 MCASH.
Range	[0,10 000 000 000 000 000 000] Matoshi

6. CREATE_NEW_ACCOUNT_FEE_IN_SYSTEM_CONTRACT

Description	Modify the cost of account creation. Combine dynamic network parameters #7 to get total account creation cost: CREATE_NEW_ACCOUNT_FEE_IN_SYSTEM_CONTRACT × CREATE_NEW - _ACCOUNT_BANDWIDTH_RATE
Example	[0] Matoshi.
Range	[0,10 000 000 000 000 000 000] Matoshi.

7. CREATE_NEW_ACCOUNT_BANDWIDTH_RATE

Description	Modify the cost of account creation. Combine dynamic network parameters #8 to get total account creation cost: CREATE_NEW_ACCOUNT_FEE_IN_SYSTEM_CONTRACT × CREATE_NEW - _ACCOUNT_BANDWIDTH_RATE
Example	[1].
Range	[0,10,000,000,000,000,000,000].

8. ALLOW_CREATION_OF_CONTRACTS

Description	To turn on MCASH Virtual Machine (MVM).
Example	True.
Range	True/False.

9. REMOVE_THE_POWER_OF_THE_GR

Description	Remove the initial GR genesis votes.
Example	True.
Range	True/False - Notice: cannot set back to False from True.

10. ENERGY_FEE

Description	Modify the fee of 1 energy.
Example	20 Matoshi.
Range	[0,10 000 000 000 000 000 000] Matoshi.

11. EXCHANGE_CREATE_FEE

Description	Modify the cost of trading pair creation. Known as the cost of creating a trade order.
Example	[102,400,000,000] Matoshi - which is 1024 MCASH.
Range	[0,10 000 000 000 000 000 000] Matoshi.

12. MAX_CPU_TIME_OF_ONE_TX

Description	Modify the maximum execution time of one transaction. Known as the timeout limit of one transaction.
Example	50 ms.
Range	[0, 1000] ms.

13. ALLOW_UPDATE_ACCOUNT_NAME

Description	Modify the option to let an account update their account name.
Example	False
Range	True/False - Notice: cannot set back to False from True.

14. ALLOW_DELEGATE_RESOURCE

Description	Modify the validation of allowing to issue token with a duplicate name, so the tokenID of the token, in long integer data type, would be the only atomic identification of a token.
Example	False
Range	True/False - Notice: cannot set back to False from True.

15. TOTAL_ENERGY_LIMIT

Description	Modify the whole network total energy limit.
Example	[5,000,000,000,000,000,000] Matoshi - which is 50,000,000,000 MCASH.
Range	[0,10 000 000 000 000 000 000] Matoshi.

16. ALLOW_MVM_TRANFER_M1

Description	Allow M1 token transfer within smart contracts. ALLOW_UPDATE_ACCOUNT_NAME, ALLOW_SAME_TOKEN_NAME, ALLOW_DELEGATE_RESOURCE proposals must all be approved before proposing this parameter change.
Example	False.
Range	True/False - Notice: cannot set back to False from True.

7.2.2. Create Proposal

Only the Super Node accounts have the right to propose a change in dynamic network parameters.

7.2.3. Vote Proposal

Only committee members (SNs) can vote for a proposal and the member who does not vote in time will be considered as a disagree. The proposal is active for 3 days after it is created. The vote can be changed or retrieved during the 3-days voting window. Once the period ends, the proposal will either succeed (33+ votes) or fail and end.

7.2.4. Cancel Proposal

The proposer can cancel the proposal before it becomes effective.

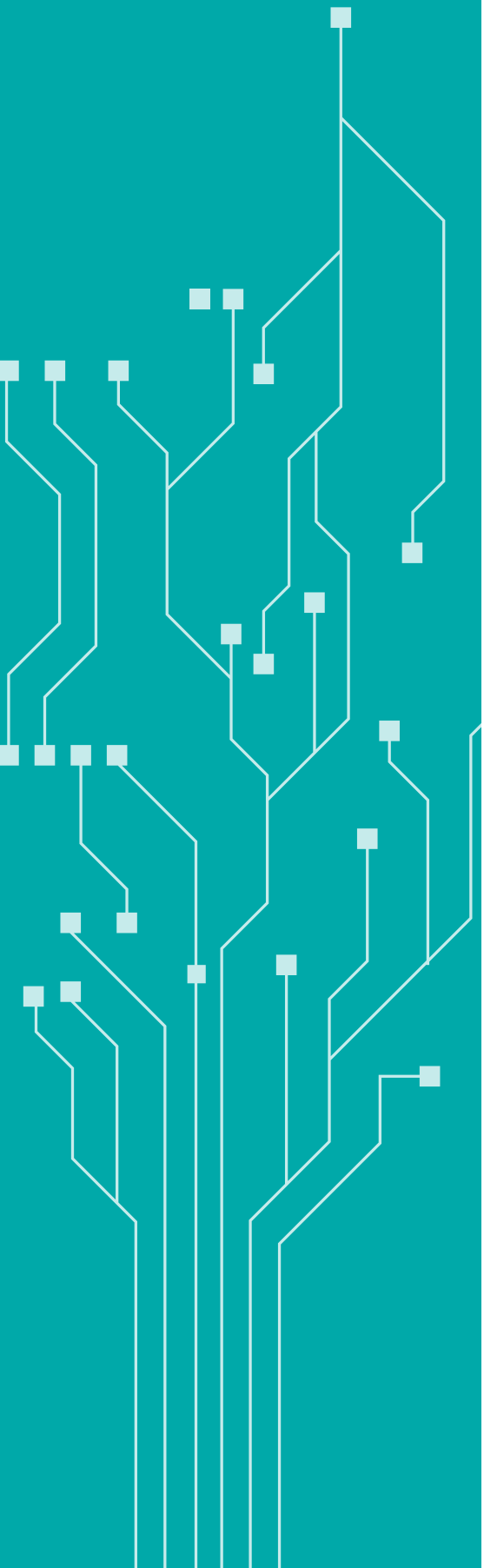
7.3. Structure

SNs are the witnesses of newly generated blocks. A witness contains 8 parameters:

- address: the address of this witness.
- voteCount: number of received votes on this witness.
- pubKey: the public key for this witness.
- url: the url for this witness.
- totalProduced: the number of blocks this witness produced.
- totalMissed: the number of blocks this witness missed.
- latestBlockNum: the latest height of block.
- isjobs: a boolean flag.

VIII. SMART CONTRACT

McashChain smart contracts are written in the Solidity language. McashChain Solidity is a fork from Ethereum's Solidity language.



8.1. Energy Model

The maximum energy limit for deploying and triggering a smart contract is a function of several variables:

- Dynamic energy from freezing 1 MCASH is 50,000,000,000 (Total Energy Limit)/(Total Energy Weight)
- Energy limit is the daily account energy limit from freezing MCASH
- Remaining daily account energy from freezing MCASH is calculated as Energy Limit - Energy Used
- Fee limit in MCASH is set in smart contract deploy/trigger call
- Remaining usable MCASH in the account

8.2. Deployment

The Solidity compiler automatically generates a JSON file, the contract metadata, that contains information about the current contract. It can be used to query the compiler version, the sources used, the ABI documentation in order to more safely interact with the contract and to verify its source code.

IX. TOKEN

9.1. M1 Token

McashChain accounts can spend 1024 MCASH to issue an M1 token with the following parameters:

- token_name
- total_capitalization
- exchange_rate: to MCASH
- circulation_duration
- total_supply
- total_lock_amount
- locking_period: in days
- description
- total_bandwidth_consumption
- max_bandwidth_consumption_per_account
- token_frozen_amount

9.2. M20 Token

Fully compatible with ERC-20. M20 interface is:

```
contract M20Interface {
function totalSupply() public constant returns (uint);
function balanceOf(address tokenOwner) public constant returns (uint balance);
function allowance(address tokenOwner, address spender) public constant returns
(uint remaining);
function transfer(address to, uint tokens) public returns (bool success);
function approve(address spender, uint tokens) public returns (bool success);
function transferFrom(address from, address to, uint tokens) public returns (bool
success);
event Transfer(address indexed from, address indexed to, uint tokens);
event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}
```

9.3. M721 Token (NFT)

The M721 token standard helps create non-fungible tokens. In many ways, it is pretty similar to M20 in functionality. This similarity exists for two reasons:
Firstly, it is easier for developers to make the transition.
It makes life much easier for users who can store these tokens in ordinary wallets [7] and trade them on exchanges or c-commerce platforms [8]
M721 gains its non-fungible properties by capturing the ownership of that particular token.

X. DEVELOPER RESOURCE

10.1. APIs

The McashChain provides HTTP API gateways for connecting and interacting with the network via Full and Solidity Nodes

10.2. Networks

McashChain has a public Testnet and a Mainnet.

10.3. Tools

10.3.1. Mcashscan.io

Mcashscan allows you to explore and search the McashChain for transactions, addresses, tokens, prices and other activities taking place on McashChain.

10.3.2. McashLight

McashLight is the MCASH Chrome/-Firefox extension wallet beside in addition to the Midas Wallet for IOS/Android/Mac/PC platforms. Its main functions are sending and receiving MCASH, M1, M20 and M721 tokens; integrating smart contract calls on developers' site; using Mcash dApps within the browser.

10.3.3. McashRPC

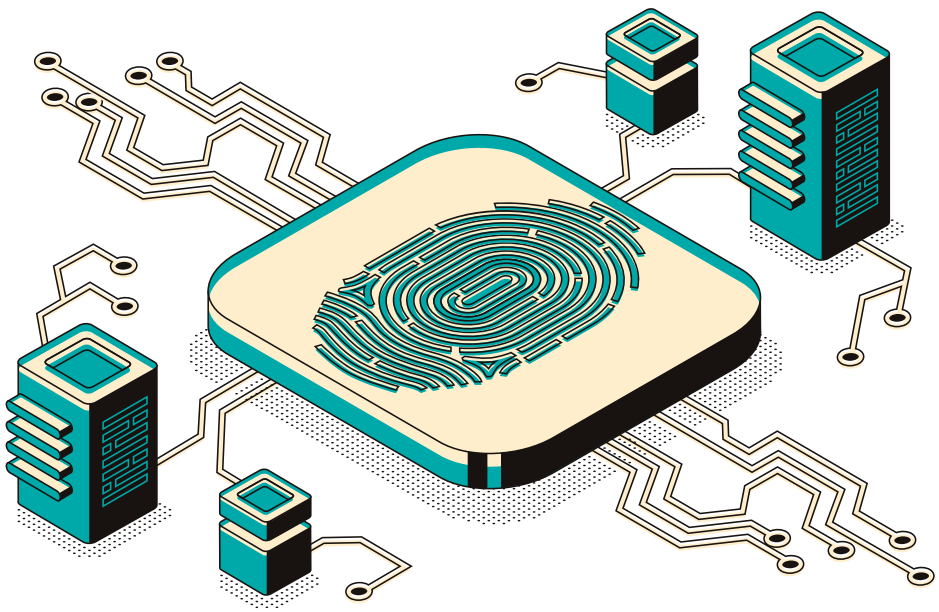
McashRPC allows developers to access the chain without having to run their own node. McashRPC offers access to both the public testnet and the mainnet.

10.3.4. McashStudio

McashStudio is an IDE for developing, deploying, and debugging smart contracts based on MVM. It uses gRPC to register accounts, deploy, and trigger smart contracts.

10.3.5. McashJs library

McashJs is a comprehensive JavaScript library containing API functions that enable developers to deploy smart contracts, change the blockchain state, query blockchain and contract information.



XI. CONCLUSION

In conclusion, McashChain, by learning from all the existing dPos Chains, eliminating the pain points and the instability of previous chains developed, will have the most desired characters for high throughput dApps required by Midas Ecosystem and our dApps development partners. Tens of thousands of TPS, Lightning network enabled since the 1st day, 3-second block confirmation time, zero-fee for ordinary users, fully integrated with Midas Wallet, listed on Vinex Network and UniDex. McashChain also give users the opportunity to earn passive income through 5 levels of nodes, run and monitored directly on Midas Wallet and 1 supernode for expert users.

XII. REFERENCE

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronics cash system. 2008.

[2] Ethereum Foundation. Ethereum's White Paper. http://github.com/ethereum/wiki/white-paper, 2014.

[3] Evan Duffield, Daniel Diaz. Dash's whitepaper. https://github.com/dashpay/dash/wiki/Whitepaper, 2014.

[4] Zcoin Foundation. Zcoin project. https://zcoin.io/, 2016.

[5] Tron Foundation. Tron project. https://tron.network/, 2018.

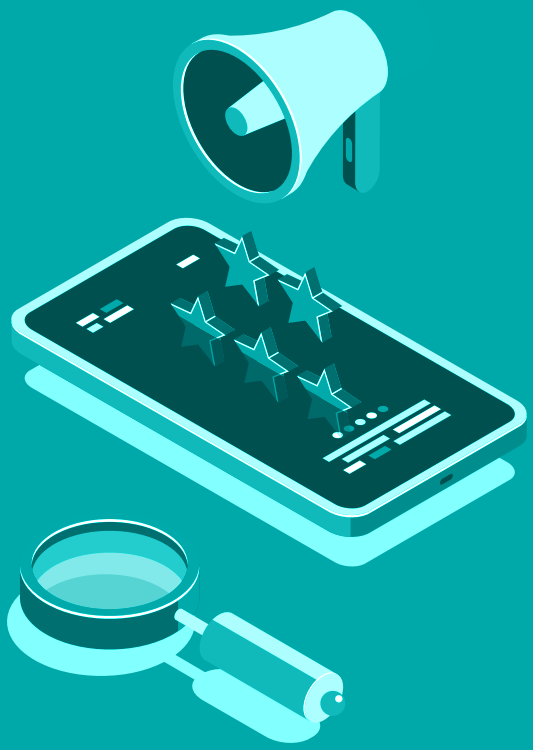
[6] VadimArasev. Proof of Authority. https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper, 2017.

[7] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin, 2013.

[8] Jens Groth, Markulf Kohlweiss. One-out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin, 2015.

[9] Gavin Wood. Solidity official documentation. https://solidity.readthedocs.io/, 2014.

[10] D. Nguyen, S. Nguyen, P. Phung, L. Phan, Midas Protocol's White Paper. 2018.



XIV. APPENDIX A. REWARD CALCULATION

Notations:

- N: the current number of supernodes, max{N} = 64
- SN1, SN2, .., SNn: the set of supernodes (SN) in the current epoch
- B1, B2, .., Bn: the number of block an SN has produced
- N1, N2, ..., Nn: the amount of staking amount (including bonus) by a Xnode
- TLS = N1 + N2 + ... + Nn: the total amount of token locked and staking
- X: the total reward per epoch for all nodes

Note that a supernode will earn both RI (for infrastructure reward) and RS (for staking reward).

Reward per epoch:

- Infrastructure reward divided to Supernode SNi:

$$RI_i = \frac{B_i}{\sum B} \times X \times 0.2$$

- Staking reward divided to Xnode Ni:

$$RS_i = \frac{N_i}{TLS} \times X \times 0.8$$

Reward per month (12 * 30 = 360 epochs):

- Infrastructure reward divided to Supernode SNi:

$$RI_i = 360 \times \frac{B_i}{\sum B} \times X \times 0.2$$

- Staking reward divided to Xnode Ni:

$$RS_i = 360 \times \frac{N_i}{TLS} \times X \times 0.8$$

Reward per year (12 * 365 = 4380 epochs):

- Infrastructure reward divided to Supernode SNi:

$$RI_i = 4380 \times \frac{B_i}{\sum B} \times X \times 0.2$$

- Staking reward divided to Xnode Ni:

$$RS_i = 4380 \times \frac{N_i}{TLS} \times X \times 0.8$$

Applying the reward calculation formula to specific scenarios with assumption that all supernodes has produced blocks equally. Therefore they all will receive the same infrastructure reward.

Scenario 1: 30 Supernodes, 240 million token locked and staking

N = 30

TLS = 240000000 MCASH

X = 2400 * 10 = 24000 MCASH

B1 = B2 = ... = Bn = 80

Reward per epoch:

- Infrastructure reward = 80/2400 * 24000 * 0.2 = 160 MCASH
- Staking reward:

Master Node = 6000000/240000000 * 24000 * 0.8 = 480 MCASH

Jedi Node = 575000/240000000 * 24000 * 0.8 = 46 MCASH

Guardian Node = 55000/240000000 * 24000 * 0.8 = 4.4 MCASH

Warrior Node = 10500/240000000 * 24000 * 0.8 = 0.84 MCASH

Apprentice Node = 5000/240000000 * 24000 * 0.8 = 0.4 MCASH

Reward per month:

- Infrastructure reward = 360 * 160 = 57600 MCASH
- Staking reward:

Master Node = 360 * 480 = 172800 MCASH

Jedi Node = 360 * 46 = 16560 MCASH

Guardian Node = 360 * 4.4 = 1584 MCASH

Warrior Node = 360 * 0.84 = 302.4 MCASH

Apprentice Node = 360 * 0.4 = 144 MCASH

Reward per year:

- Infrastructure reward = 4380 * 160 = 700800 MCASH
- Staking reward:

Master Node = 4380 * 480 = 2102400 MCASH

Jedi Node = 4380 * 46 = 201480 MCASH

Guardian Node = 4380 * 4.4 = 19272 MCASH

Warrior Node = 4380 * 0.84 = 3679.2 MCASH

Apprentice Node = 4380 * 0.4 = 1752 MCASH

Scenario 1's Return-on-investment:

Scenario 1	Investment	Return	ROI
Supernode	5,000,000 MCASH	2,803,200 MCASH	56%
Master Node	5,000,000 MCASH	2,102,400 MCASH	42%
Jedi Node	500,000 MCASH	201,480 MCASH	40%
Guardian Node	50,000 MCASH	19,272 MCASH	39%
Warrior Node	10,000 MCASH	3,679 MCASH	37%
Apprentice Node	5,000 MCASH	1,752 MCASH	35%

Scenario 2: 64 Supernodes, 600 million token locked and staking

N = 60

TLS = 600000000 MCASH

X = 2400 * 10 = 24000 MCASH

B1 = B2 = ... = Bn = 80

Reward per epoch:

- Infrastructure reward = 80/2400 * 24000 * 0.2 = 160 MCASH
- Staking reward:

Master Node = 6000000/600000000 * 24000 * 0.8 = 192 MCASH

Jedi Node = 575000/600000000 * 24000 * 0.8 = 18.4 MCASH

Guardian Node = 55000/600000000 * 24000 * 0.8 = 1.76 MCASH

Warrior Node = 10500/600000000 * 24000 * 0.8 = 0.336 MCASH

Apprentice Node = 5000/600000000 * 24000 * 0.8 = 0.16 MCASH

Reward per month:

- Infrastructure reward = 360 * 160 = 57600 MCASH
- Staking reward:

Master Node = 360 * 192 = 69120 MCASH

Jedi Node = 360 * 18.4 = 6624 MCASH

Guardian Node = 360 * 1.76 = 633.6 MCASH

Warrior Node = 360 * 0.336 = 120.96 MCASH

Apprentice Node = 360 * 0.16 = 57.6 MCASH

Reward per year:

- Infrastructure reward = 4380 * 160 = 328500 MCASH
- Staking reward:

Master Node = 4380 * 192 = 840960 MCASH

Jedi Node = 4380 * 18.4 = 80592 MCASH

Guardian Node = 4380 * 1.76 = 7708.8 MCASH

Warrior Node = 4380 * 0.336 = 1471.68 MCASH

Apprentice Node = 4380 * 0.16 = 700.8 MCASH

Scenario 2's Return-on-investment:

Scenario 2	Investment	Return	ROI
Supernode	5,000,000 MCASH	1,169,460 MCASH	23.4%
Master Node	5,000,000 MCASH	840,960 MCASH	16.8%
Jedi Node	500,000 MCASH	80,592 MCASH	16.1%
Guardian Node	50,000 MCASH	7,708 MCASH	15.4%
Warrior Node	10,000 MCASH	1,471 MCASH	14.7%
Apprentice Node	5,000 MCASH	700 MCASH	14.0%