BOTCoin Whitepaper

2025

Summary

BOTCoin uses peer-to-peer network system technology at the bottom layer, allowing intelligent robots to become autonomous nodes in the decentralized network. It also integrates BVM to implement hardware-level smart contract prosgramming, allowing the robot's own data and programs to be executed safely and reliably. Based on the Hashgraph consensus algorithm, It is a Directed Acyclic Graph (DAG) . Supports asynchronous processing and ⅓ of faulty/malicious nodes, ensuring both network decentralization and fast block verification.The dynamic membership mechanism supports nodes joining/leaving at any time without affecting the operation of the entire network. Fast Sync supports new nodes to quickly load from the tip of the chain without loading the entire network history.Combining the above advantages, BOTCoin can be easily run on robots。

Introduction

BOTCoin is a distributed intelligent robot network that aims to fully explore the ecological value of intelligent robots, such as the surplus value of intelligent robots, safe division of labor and collaboration, etc.As the main force of social production in the next wave of technological innovation, robots will face new technical and socio-economic issues, such as robot data security and privacy, production distribution, or excess computing power of robots.Therefore, we need a blockchain node network that can run inside a robot to solve data security and privacy issues at the hardware level, solve the social distribution mechanism that does not require trust at the hardware level (in an environment where artificial intelligence replaces human work), and solve the problem of excess computing power caused by the continuous updating and iteration of robots.It is worth sharing that the BOTCoin network can simultaneously solve the above problems. The network initially relied on Babble, a consensus communication protocol suitable for deployment on low-energy mobile devices, and integrated our improved BVM, which can achieve both lightweight operation on robots and hardware-level smart contract programming, based on which the crypto world can interact directly with the real world.

BOT Introduction to Economics

Reconstruction of production relations

In the era of full BOT economy, Robots will become the main force of production, and most humans will no longer need to participate in the market division of labor. Humans only need to output ideas, and most production tasks will be completed by Robots. Most humans will change from laborers to Robots employees.。

Reconstruction of the source of surplus value

According to "Das Kapital", when you become a Robots employer, you are making a choice to maximize your benefits, and you will fully explore the surplus value of the Robots you employ. When a large number of humans compete for a small number of jobs that require humans, the cost of labor will drop significantly, and the surplus value space for human workers will not be large. Therefore, in the future, most capitalists and general employers will transfer their surplus value from workers to Robots.

Means of production and general business rules

Under the BOT economy, the three main elements of means of production are: Robots, data, and new materials.

The general business rules will be reflected as follows:

- **Robots become civilians:**

  Whether it is an **Intelligent Model** or **Robots**, it will be very cheap because the boundary cost is low and it can be imitated. Just like a computer, ordinary people can also have advanced Robots.

- **Data privatization:**

  Human creativity, commodity/product modeling, workflows, etc. will be deeply quantified and converted into data, and these data sources will be regarded as the main part of competitiveness. Enterprises and individuals will pay more attention to the privatization of core data and pay more and more attention to data security.

- **New material class:**

  New materials generally involve scarce elements on the periodic table. These scarce resources will be controlled by the big capital and power institutions in the previous economic cycle. In other words, in the current cycle, the capital elites are all buying up the "periodic table". Just like land, it was divided and controlled by the capital elites in the previous round, which resulted in the class structure we see in the current cycle.

- New business model:

  The above-mentioned will gradually form a new business model.

  1. Trustless BOT collaboration

     To complete a commodity/product, enterprises and individuals need to use one or more Robotsto participate in the collaboration. Because all parties in the collaboration must consider the security and privacy of the data, BOTCoin's trustless peer-to-peer network technology and Robots hardware-level smart contracts can be used to complete the entire collaboration process without worrying about data security and privacy issues.

  2. Fully explore the surplus value of Robots

     Employers can allow Robots to join BOTCoin and become nodes. According to the BOTCoin incentive rules, they can make full use of Robots' working time and load, obtain more surplus value from Robots, and thus complete a new form of capital transformation in the BOT Economic era.

  3. BOT consensus

     In an environment where rare new materials or core elements are gradually controlled by the rich and powerful, civilians are always passive. However, Bitcoin has given us a very good inspiration since its launch in 2009: civilians can use peer-to-peer network mechanisms to reach a new consensus and unite against traditional production mechanisms to explore "means of production" that are in line with the future, thereby completing new wealth accumulation. For example, the BOTCoin peer-to-peer network can be used to reach a new Robots consensus to explore the core means of production that are in line with the next BOT Economic era.

Innovation and Advantage

- **Asynchronous**: Participants have the freedom to process commands at different times.
- **Leaderless**: No participant plays a special role.
- **Byzantine Fault-Tolerant**: Supports one third of faulty nodes, including malicious behavior.

- **Finality**: Babble's output can be used immediately, no need for block confirmations.
- **Dynamic Membership**: Members can join or leave a Babble network without undermining security.
- **Fast Sync**: Joining nodes can sync directly to the current state of a network.
- **Accountability**: Auditable history of the consensus algorithm's output.
- **Low energy consumption:** Supports deployment of peer-to-peer network nodes on low-energy/mobile Robots.
- **Hardware smart contract:** BVM implements hardware-level smart contracts, binds to Robots hardware, and enables direct interaction between the crypto world and the real world without going through an intermediary, meeting the needs of multi-BOTs and crypto collaboration.
- **Exploring the Surplus Value of Embodied Intelligent Robots:** Subverting the traditional capital concept, we advocate that capitalists change from exploring the surplus value of human beings to exploring the surplus value of BOT. This is in line with the new capitalist framework in the new era.

Advanced Byzantine Fault-Tolerant:

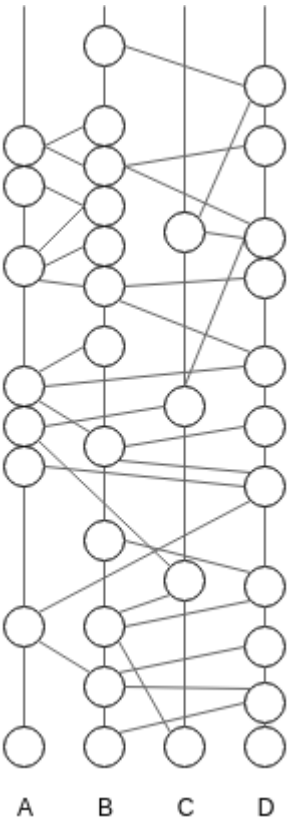
The underlying layer of BOTCoin relies on Babble consensus. The core of Babble is to extend Hashgraph (advanced Byzantine fault-tolerant consensus algorithm) to ensure that the distributed system remains available and consistent under adversarial conditions. Even if some nodes have arbitrary failures or malicious behavior, as long as a block with enough signatures (>2/3) and all previous blocks can be immediately considered valid.
However, unlike traditional blockchains, it supports nodes to submit signatures and transactions asynchronously.
So, BOTCoin ensures reliable security while having higher network efficiency.

Gossip About Gossip

One way to approximate common knowledge in this context is to use a communication protocol where participants regularly tell each other everything they know about what everyone else knows. These are usually referred to as Full Information Protocols, aka 'gossip about gossip'.

Members locally record the history of the gossip protocol in a directed acyclic graph, a DAG, where each vertex represents a gossip event and the edges connect a vertex to the immediately-preceding vertices. Roughly speaking, a member, say Alice, will repeatedly choose another member at random, say Bob, and attempt to learn what he knows that she doesn't know. She will send him a sync request saying

‘Hey, here is what I know; what do you know that I don’t know?’. Bob will compute the difference and respond with a set of events that he knows and Alice doesn’t yet know. Alice will insert these events in her DAG, and create a new event to record this sync. The newly created event includes the hashes of her last event, and Bob’s last event. Hence, the DAG is connected by a succession of recursive cryptographic hashes; like a blockchain, but two-dimensional. Each event contains the hashes of the events below it and is digitally signed by its creator. So the entire graph of hashes is cryptographically secure.



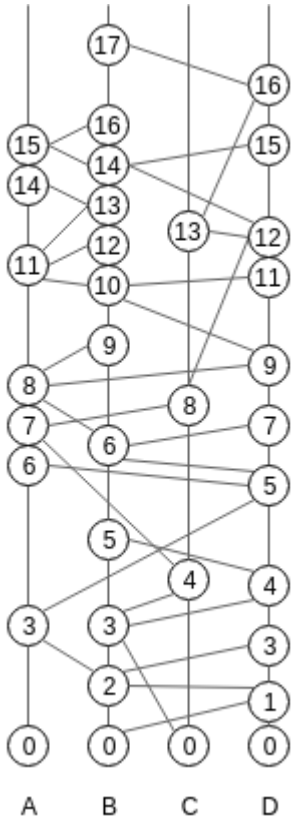
The communication graph is a very rich data structure from which we can extract all sorts of information about the history of gossip, and also derive a consistent ordering of the events, even in the presence of faulty participants. But let’s take it step by step.

Lamport Timestamps

Leslie Lamport introduced a seminal paper in 1978, entitled “Time, Clocks, and the Ordering of Events in a Distributed System”. In this paper he describes a distributed algorithm for extracting a consistent total ordering of the events in an asynchronous message passing system, using a concept of Logical Clocks.

The algorithm follows some simple rules:

1. A process increments its counter before each event in that process;
2. When a process sends a message, it includes its counter value with the message;
3. On receiving a message, the counter of the recipient is updated, if necessary, to the greater of its current counter and the timestamp in the received message. The counter is then incremented by 1 before the message is considered received.
4. Ties are broken using an arbitrary function (eg. sort by hash)



This is a distributed algorithm. Each process independently follows these rules, and there is no central synchronizing process or central storage. Synchronization is achieved because all processes order the commands according to their timestamps, so each process uses the same sequence of commands. A process can execute a command timestamped T when it has learned of all commands issued by all other processes with timestamps less than or equal to T.

However, the resulting algorithm requires the active participation of all the processes. A process must know all the commands issued by other processes, so that the

failure of a single process will make it impossible for any other process to execute commands, thereby halting the system. BOTCoin implements Lamport Timestamps on top of the hashgraph, but with added steps for Byzantine Fault Tolerance.

Two-Phase Commit

Most distributed consensus protocols are special adaptations of this concept. There is a theoretical result that says one can't attain BFT, in the same conditions, with ⅓ of malicious participants. So, with the assumption that at least ⅔ of participants are good, the usual solution resembles something like this:

1. Someone proposes a value
2. Everyone votes on the proposal and broadcasts their vote
3. Every one confirms they have received ⅔ of votes for the same proposal, and broadcasts this confirmation.
4. When a participant collects ⅔ of such confirmations, it commits the value.

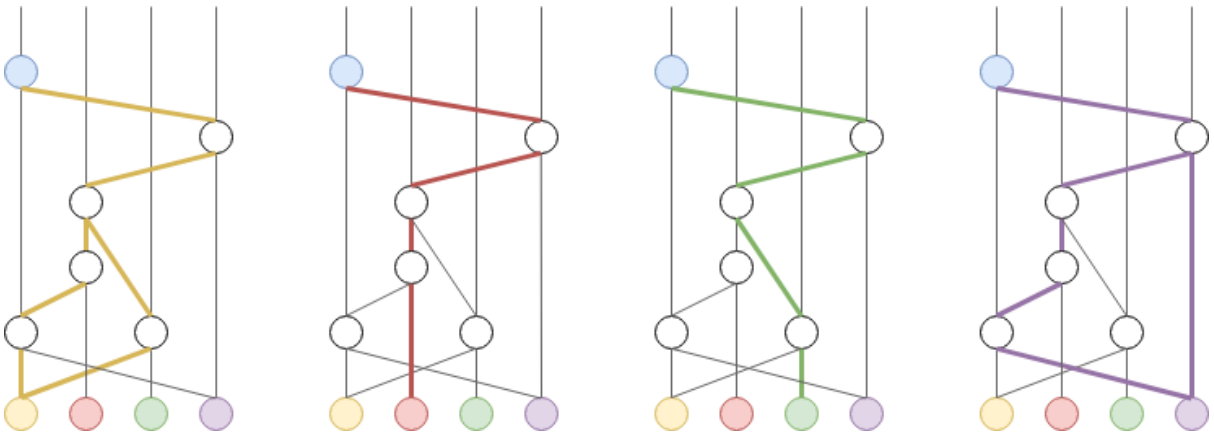
Usually, the solutions vary around who gets to propose the value - aka the leader - and how this leader is elected or changed.

Virtual Voting

A similar algorithm can be run internally thanks to the communication graph by using the concept of virtual voting. Instead of exchanging votes directly, we compute what other participants would have voted, based on our knowledge of what they know.

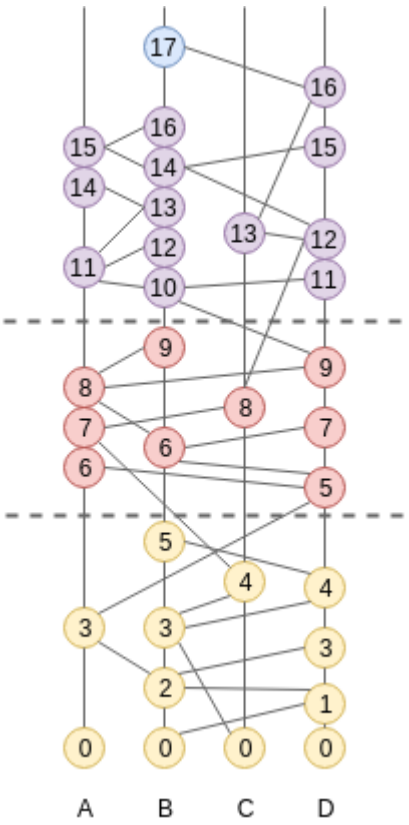
First, the Hashgraph defines a concept of *Strongly Seeing*:

"If there are n members, then an event w can strongly see an event x, if w can see more than 2n/3 events by different members, each of which can see x".



Strongly Seeing is analogous to receiving votes from two thirds of participants in the first phase of the two-phase commit.

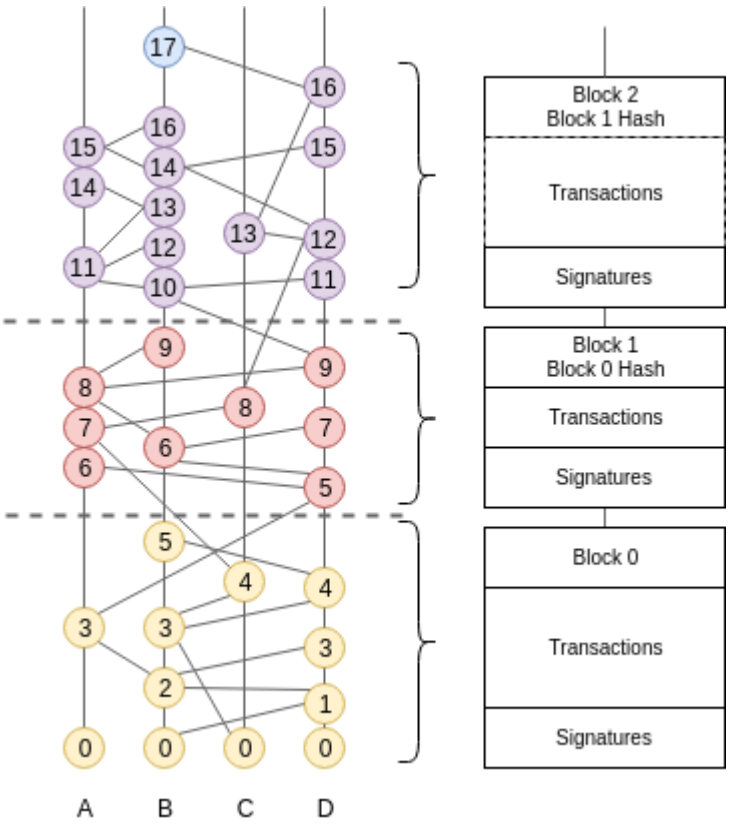
Also, we do not need a leader to propose a value. Instead, participants compute virtual cuts in the hashgraph, called rounds, which allow processing events in batches. This is also a distributed algorithm where all members end up with the same rounds. Roughly speaking, starting at round 0, when we reach a point when ⅔ of members can strongly see the cut from the previous rounds, we start a new round. When there is common knowledge about a round, attested by *Strongly Seeing*, we can decide on the order of event below that cut. The details of the algorithm are best described in the original hashgraph whitepaper.



So this algorithm doesn't need a leader. All participants run the algorithm locally, process rounds at their own speed, and end up outputting the same batches of ordered events.

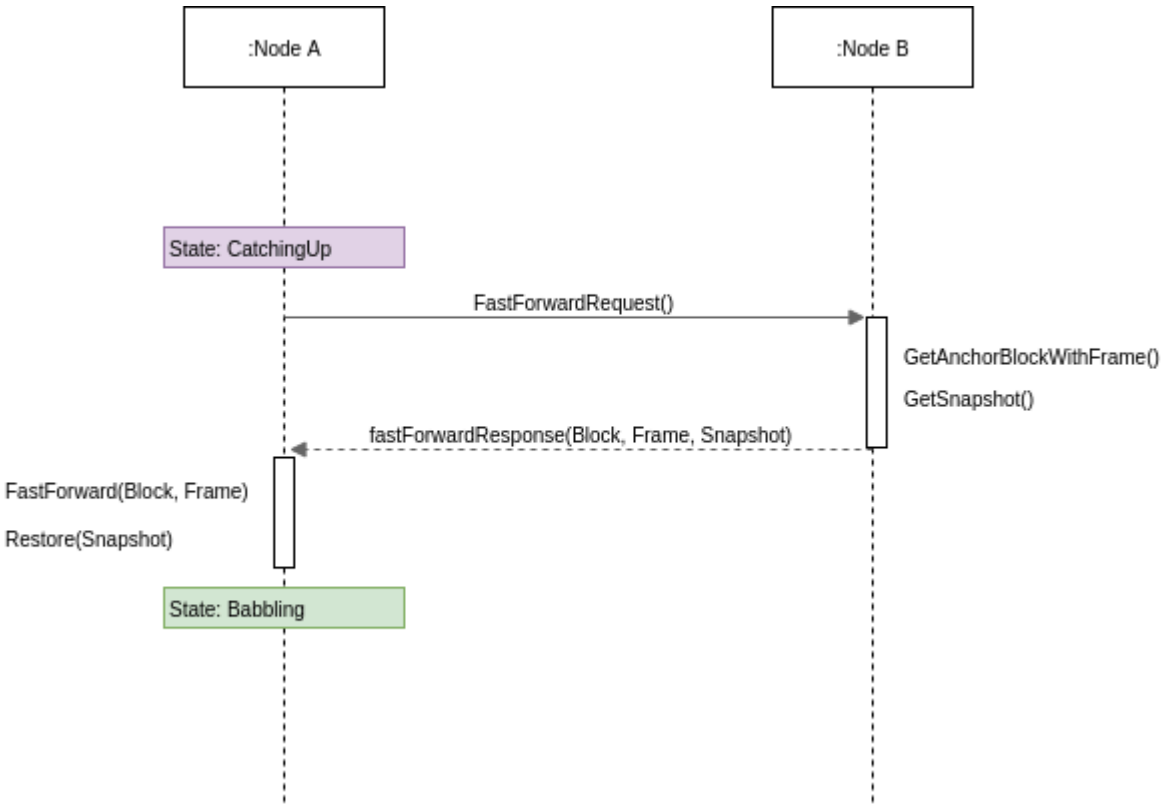
Blockchain

A blockchain is a one-dimensional data-structure made of cryptographically chained blocks. It is convenient to map our two-dimensional hashgraph onto a blockchain because the blockchain is much easier to work with when it comes to consuming and verifying the output of the consensus algorithm. The concatenation of blocks, and the transactions they contain, is recursively secured by digital signatures. A block that obtains enough signatures (>1/3) can immediately be considered valid, along with all the blocks that precede it, because it contains a signed fingerprint of the list of blocks so far.



Fast Sync

FastSync is an element of the Babble protocol which enables nodes to catch up with other nodes without downloading and processing the entire history of gossip (Hashgraph + Blockchain). It is important in the context of mobile ad hoc networks where users dynamically create or join groups, and where limited computing resources call for periodic pruning of the underlying data store. The solution relies on linking snapshots of the application state to independent and self-contained sections of the Hashgraph, called Frames. A node that fell back too far may fast-forward straight to the latest snapshot, initialize a new Hashgraph from the corresponding Frame, and get up to speed with the other nodes without downloading and processing all the transactions it missed. Of course, the protocol maintains the BFT properties of the base algorithm by packaging relevant data in signed blocks; here again we see the benefits of using a blockchain mapping on top of Hashgraph. Although implementing the Snapshot/Restore functionality puts extra strain on the application developer, it remains entirely optional; FastSync can be activated or deactivated via configuration.



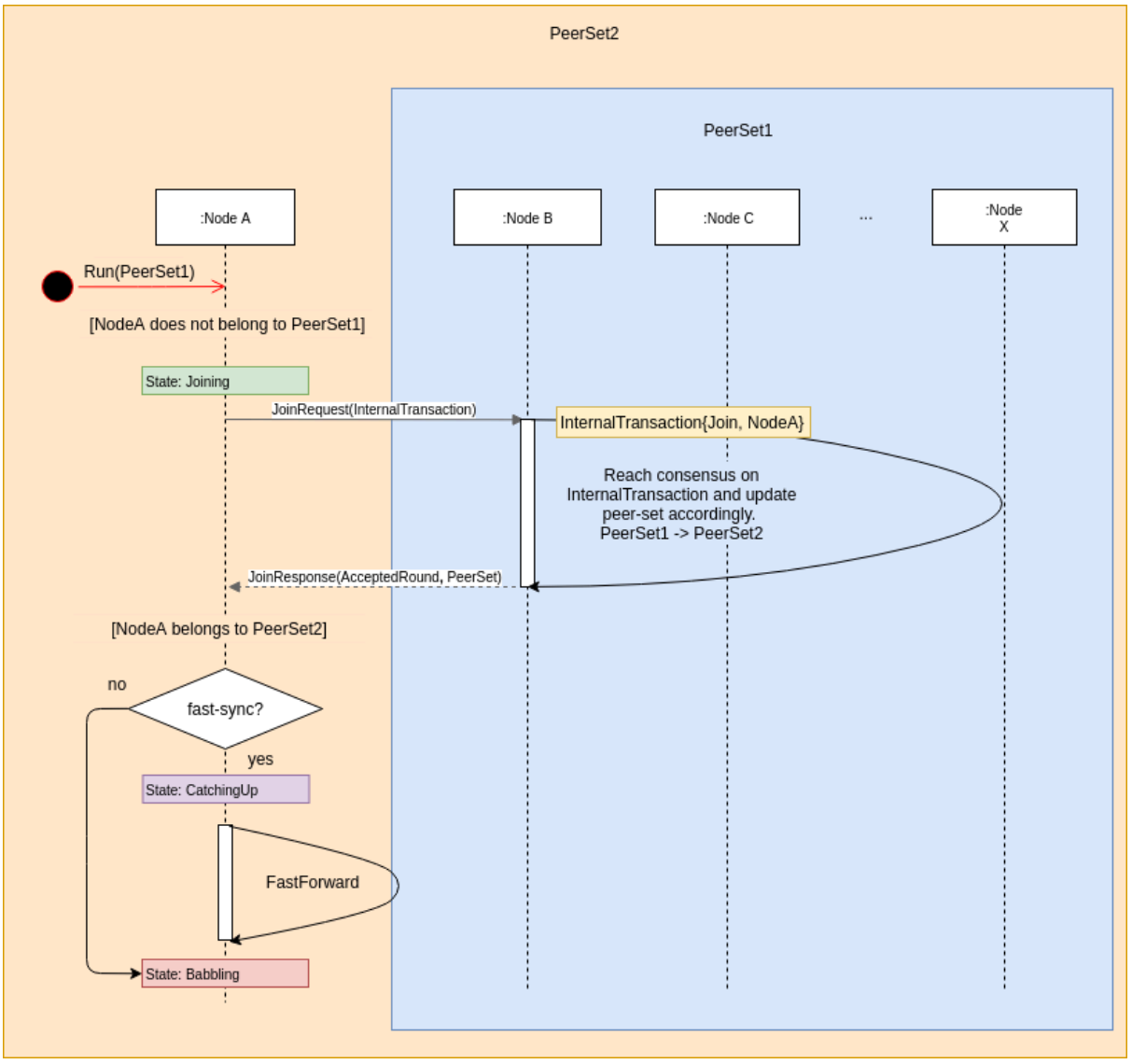
The BOTCoin node is implemented as a state-machine where the possible states are: **Babbling**, **CatchingUp**, **Joining**, **Leaving**, and **Shutdown**. When a node is started and belongs to the current validator-set, it will either enter the **Babbling** state, or the **CatchingUp** state, depending on whether the **fast-sync** flag was passed to BOTCoin.

In the **CatchingUp** state, a node determines the best node to fast-sync from (the node which has the longest hashgraph) and attempts to fast-forward to their last consensus snapshot, until the operation succeeds. Hence, FastSync introduces a new type of command in the communication protocol: *FastForward*.

Upon receiving a FastForwardRequest, a node must respond with the last consensus snapshot, as well as the corresponding Hashgraph section (the Frame) and Block. With this information, and having verified the Block signatures against the other items as well as the known validator set, the requesting node attempts to reset its Hashgraph from the Frame, and restore the application from the snapshot. The difficulty resides in defining what is meant by *last consensus* snapshot, and how to package enough information in the Frames as to form a base for a new/pruned Hashgraph.

Dynamic Membership

Dynamic Membership is an extension to the Babble protocol, which enables peers to join or leave a live cluster via consensus. Until now, we had only considered fixed peer-sets, where the list of participants was predetermined and unchanged throughout the life of a BOTCoin network. This was an important limitation, hindering the formation of ad-hoc blockchains as we envision them. Here we present our solution, and its implementation, which relies on previous work regarding the Hashgraph-to-Blockchain projection, and FastSync.



A BOTCoin node is started with a genesis peer-set (genesis.peers.json) and a current peer-set (peers.json). If its public-key does not belong to the current peer-set, the node will enter the `Joining` state, where it will attempt to join the peer-set. It does so by signing an `InternalTransaction` and submitting it for consensus via a `JoinRequest` to one of the current nodes.

The InternalTransaction is added to an Event, and goes through BOTCoin consensus, until it is added to a block and committed. However, unlike regular transactions, the InternalTransaction is actually interpreted by Babble to modify the peer-set, if the application-layer accepts it. We shall see that, according to Hashgraph dynamics, an accepted InternalTransaction, committed with round-received R, only affects peer-sets for rounds R+6 and above.

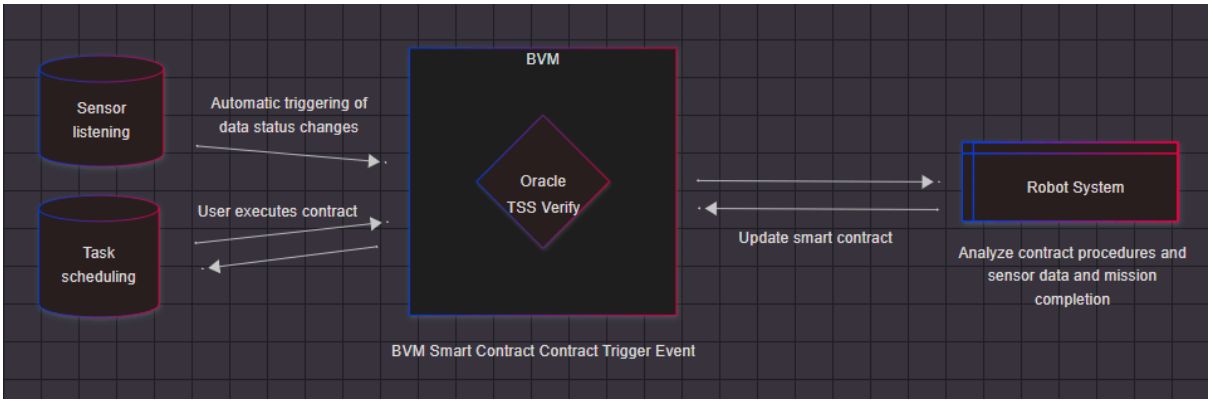
If the JoinRequest was successful, and the new node makes it into the peer-set, it will either enter the `Babbling` state directly, or the `CatchingUp` state, depending on whether the `fast-sync` flag was provided. In the former case, the node will have to retrieve the entire history of the hashgraph and commit all the blocks it missed one-by-one. This is where it is important to have the genesis peer-set, to allow the joining node to validate the consensus decisions and peer-set changes from the beginning of the hashgraph.

The functionality for removing peers is almost identical, with the difference that there will be an automatic way of deciding when nodes should be removed, based on a minimum level of activity (ex: 10 rounds with no witnesses). As of today, a node submits a LeaveRequest for itself upon capturing a SIGINT signal when the BOTCoin process is terminated cleanly.

BVM (Hardware-level programmable smart contracts)

BVM refers to the "BOTCoin Virtual Machine". This is a decentralized virtual environment that executes code in a secure and consistent manner on all BOTCoin nodes. Nodes run BVM to execute smart contracts, using "Fuel" to measure the computational work required to perform operations, thereby ensuring efficient resource allocation and network security. BVM is an improvement based on Ethereum EVM.

BVM is different from traditional "VM". Traditional "VM" only allows code to be executed within the program. BVM supports, on this basis, the expansion of the oracle protocol that can interact with hardware, so that operators (including users or BOTs) on BOTCoin can directly realize automated and secure interaction with the BOTs system hard code through BVM smart contracts.



BVM硬件交互过程

Accounts

BOTCoin uses the same account model as Ethereum. Accounts represent identities of external agents and are associated with a balance (and storage for Contract accounts). They rely on public key cryptography to sign transactions so that the BVM can securely validate the identity of a transaction sender.

The following is mainly inspired by Ethereum:
Accounts are objects in the BVM state. They are of two types: externally owned accounts and contract accounts. Externally owned accounts have balances, contract accounts have balances and storage space. The BVM state is the state of all accounts and is updated with each transaction. The underlying consensus engine ensures that every participant in the BOTCoin network processes the same transactions in the same order, thus reaching the same state. Using contract accounts in the BVM allows the deployment and use of smart contracts.

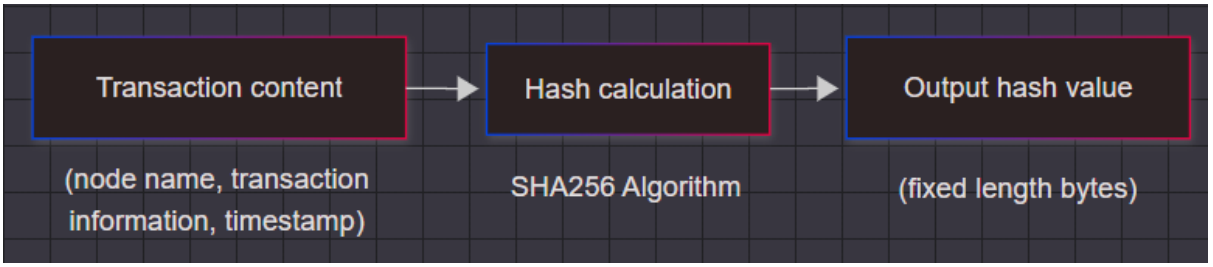
Transaction

A transaction is a signed data package containing BVM instructions. It can contain instructions to transfer coins from one account to another, create a new contract account, or call an existing contract account. Transactions are encoded using a custom Ethereum scheme, RLP, and contain the following fields:

- The recipient of the message.
- A signature identifying the sender and proving their intent to send the transaction.
- The amount of coins transferred from the sender to the receiver.
- An optional data field that can contain the message sent to the contract.
- A STARTGAS value, indicating the maximum number of computational steps allowed for the transaction execution.
- A GASPRICE value, indicating the fee the sender is willing to pay for gas. One unit of gas corresponds to the execution of one atomic instruction, i.e. one computational step.

Privacy

Hashgraph is a beautiful consensus algorithm based on a homonymous data structure. It is a Directed Acyclic Graph (DAG) from which the order must be extracted via some complex consensus functions. To verify the consensus index of a given transaction, one has to re-compute the consensus methods on a subset of the hashgraph. On the other hand, blockchains do not need any further processing to extract the ordered list of transactions and simple cryptographic primitives are sufficient to validate blocks.



Transaction content encryption

```
Block: {
    Body:{
        Index                      int                          // block index
            RoundReceived              int                          // round received of corresp
            Timestamp                  int64                        // unix timestamp (median of
            StateHash                  []byte                       // root hash of the applicat
            FrameHash                  []byte                       // hash of corresponding has
            PeersHash                  []byte                       // hash of peer-set
            Transactions               [][]byte                     // transaction payload
            InternalTransactions       []InternalTransaction        // internal transaction payl
            InternalTransactionReceipts []InternalTransactionReceipt // receipts for internal tra
    }
    Signatures: map[string]string
}
```

Blocks contain a body and a set of signatures. Signatures are based on the hash of the body; which is enough to verify the entire block because it contains a digital fingerprint of the body.

Inspire activists

The main criterion for whether a peer-to-peer network has a promising future is whether there are more active participants. Therefore, we combine the BFT consensus and propose the Proof-of-Active (POA) incentive layer consensus mechanism. The role of POA is to continuously motivate active participants to participate in the network consensus and promote active participants to be more active, so as to ensure that the network can operate efficiently without changing the degree of decentralization.

Active quantification is based on: validators, Stakers, cluster nodes, etc.

There are two sources of rewards: core coin rewards and gas fee rewards.

BOC award

Rewards are issued every 7,200 block of generation, and each reward is distributed according to the following rules:

- Validator rewards: 60% of rewards
    - Evenly distributed to all validators in the validator set
    - Requires the last time you join the network, the number of verifications must be greater than 7,200 blocks
- Staker Rewards: 35% of the rewards
    - Distributed according to Stake ratio
    - Minimum Stake requirement: 100,000 BOC
- Historical stability reward: 5% of the reward
    - The purpose of this reward is to encourage relatively long-term stable nodes, because this means that the node has recorded a long block history Based on the block height when the node last joined the network, calculate the height difference and the total height difference of the entire network

Gas Fee Rewards

Each transaction applied to the BVM is associated with a BaseCoin address (possibly empty) of up to 1 validators, which receives the transaction fees. In BOTCoin, we implemented a system to distribute fees fairly and securely among validators.

After the block validation is completed, we get the corresponding validator set from BOTCoin. Then we use the block hash to get a pseudo-random number and use it to select up to 1 peers from the validator set. This peer will receive all transaction fees for that block. The system is fair and secure because the selection process is evenly distributed and it is impossible for a malicious validator to cheat it by manipulating the block hash.

BOC Output rules

BOC definition

BOTCoin is referred to as BOC，BOC is the core circulation asset of the BOTCoin network。
BOCTotal amount: 21000000000,
BOC The total amount is constant.

Output rules

All participants enjoy fairness without any pre-production.
According to the "Active Incentive Rules", a reward is issued every 7,200 blocks (about 8 hours). (Each block is usually generated in 5 to 6 seconds).
Each reward is 5,000,000 BOC.
Every 15,000,000 blocks (about 3 years), the reward is halved, that is, starting from block 15,000,001, each reward is 2,500,000 BOC. And so on. Until the total amount of BOC not produced is 0.
Of course, we don’t need to worry about the total unproduced balance being 0, which will cause the BOTCoin network to collapse due to no nodes participating in the consensus. First, there are still decades before the balance is produced. Secondly, the BOTCoin network community will also follow the development and changes of society and promote BOTCoin to upgrade to adapt to society.

Open source

Project open source GitHub address:
https://github.com/BOTCoinNetwork/botcoin

From Service@BOTCoin.network