THE CAASIOPE NETWORK
The first Cryptocurrency-as-a-Service blockchain

Guillaume Bonnot
www.caasiope.net
August 29th, 2018

Abstract. This document describes a layer 2 protocol using a light-weight blockchain to secure and track ownership of external assets under custody. The blockchain uses a federated consensus where special nodes are in charge of validating each transaction. The peer to peer network relies on public key infrastructure to maximize security between nodes and can be used as a transportation layer to support additional protocols. The ledger contains both the state and the history of the blockchain to facilitate light-weight clients and data manipulation. The protocol is designed to enable multi-currency transactions, to provide advanced features via complex account types and to support cross-blockchain interoperability.

1. Consensus

The choice of the consensus protocol is made taking multiple factors into considerations. Being a second layer protocol, certain types of consensus are de facto eliminated, while other types can leverage the fact that the custodian already requires a certain level of trust. Finally, the consensus should provide high throughput and low cost without compromising security.

Proof of Work. Proof of Work is an open consensus algorithm that emphasizes maximum security using the real world physical limitation of computing power and the introduction of economic incentives. Proof of Work is slow and costly by design. While the speed can be increased at the cost of stability and reliability, it also increases the probability of orphaning a block.

Proof of Stake. Proof of Stake is an open consensus algorithm that uses economic incentives, based on the value of a native token, to ensure relative security without requiring the wastage of computational resources. Proof of Stake is designed to be fast and cost-efficient.

Game Theory. The Consensus algorithms used by native chains often rely on the value of their native coin to incentivize good behavior using game theory, where attacking the network is made more costly than the potential reward.

Side Chain. Sidechains are layer 2 protocols that allow external assets to be securely used on their blockchain. A two-way peg mechanism must be used to move the asset in and out from the sidechain to the asset's native blockchain.

Layer 2 Consensus. The Consensus algorithms used by side chains cannot rely on the value of their native coin to incentivize good behavior, considering that the combined value of the external assets could outweigh the total value of the native coins.

Federated Consensus. The Federated Consensus is a closed consensus protocol where arbitrarily selected nodes, called validators, are in charge of validating the transactions and producing the next block.

Reaching Consensus. When the validators reach consensus on which transactions are to be included in the new block and the new state of the ledger, a new block is created and the ledger is updated and broadcasted. A consensus is reached only when the required threshold of validators, the quorum, agree on the same proposal.

Quorum. The optimal number of validators participating in a federated consensus is still to be debated, as we are looking for a fine tune between speed and security. While increasing the number of validators will decrease points of failure, the cost of synchronizing all of them will rise exponentially. We would recommend no more than 10 validators participating in the federated consensus, as any further addition would not significantly improve the relative security of network.

2. Post Synchronous Federated Byzantine Fault Tolerant Consensus

The Caasiope Network is using its own implementation of the federated consensus protocol which is byzantine fault tolerant and optimized to process transactions at scale using post synchronous consensus agreement.

Byzantine Fault Tolerant. The Caasiope Network consensus protocol is Byzantine Fault-Tolerant and meets the 3 following criteria: safety, liveness and fault tolerance.

Federated consensus. Under the federated consensus protocol, the new ledger state is considered updated once a specified quorum of validators has signed it.

Bottleneck. While the federated consensus protocol is a high-performance protocol, the time lost by validators synchronizing between each other in order to get the same output appeared to be a real performance bottleneck. Optimizing the way validators synchronize substantially increases the blockchain's throughput.

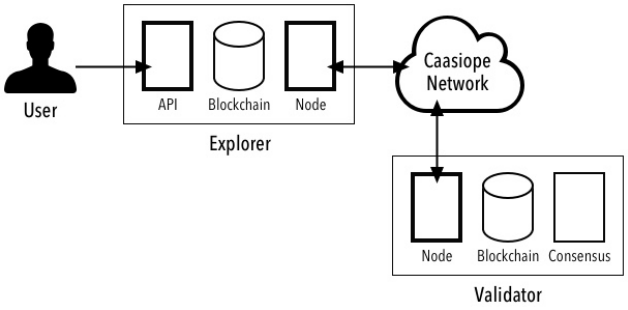
Post State. The Caasiope Network consensus protocol avoids this bottleneck by separating the transaction validation process from the federated consensus process. Each validator is validating transactions in real time and maintaining its own post state of the ledger.

Post-Synchronous. After the fact, validators reach consensus to produce a block containing the transactions that were validated by the quorum, and finalize the new ledger state by signing it. As a result, the transactions included in a block n produced at time t; were validated at time t-x, where $x = t(n) - t(n-1)$ is the time elapsed since the last produced block.

Performance. The Caasiope Network has developed a high-performance consensus that enables the blockchain to process more than 1000 transactions per second and produce a block every second.

3. Network Topology

Node. The Caasiope Network peer-to-peer infrastructure is composed of nodes, whose role is to store the blockchain and relay transactions.



Explorer. The explorers are public nodes where the blockchain data is exposed to the user via an API.

Validator. The validators are private nodes where transactions are validated and take part in the consensus.

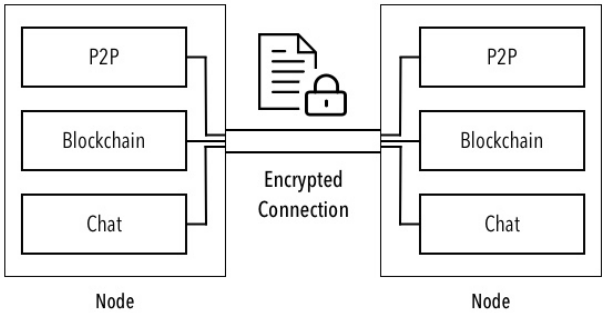
4. Secure Network Infrastructure

Identity. The Caasiope Network is using top grade cryptography in order to secure the peer to peer network. Each node is identified using the Public Key Infrastructure X.509 standard.

Encryption. Communications between nodes are fully encrypted using the TLS protocol.

Multiplexing. Each connection between peers is optimized via multiplexing, allowing different protocols to run on a single socket using a dedicated channel.

Flexibility. The Caasiope Network communication layer is robust and flexible so more services can be plugged-in in the near future.



5. Hybrid SQL & NoSQL Database

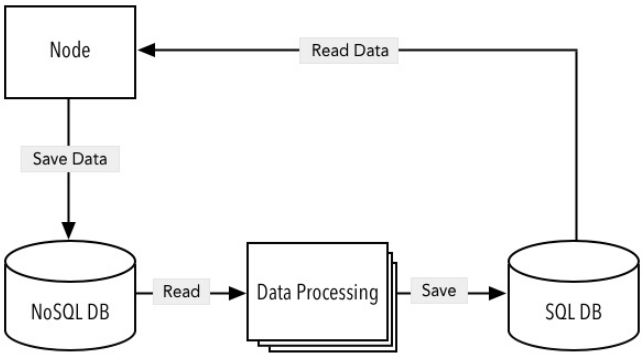
The first generation of blockchain, based on bitcoin's code, is using a file database system to save three fundamental bits of information : the blockchain itself in the form of a full block history, the list of the current unspent transaction outputs which are required to optimize loading, and finally, the difference between each block, used to optimize rollbacks

NoSQL Database. While file databases provide excellent performance for saving operations, this data structure (NoSQL) doesn't allow advanced data manipulation that is required by most businesses to operate.

SQL Database. The use of a relational database (SQL) is definitely a plus when it comes to integrating a blockchain into an existing information system infrastructure. However, using a relational database requires additional processing time.

Performances. Furthermore, next-generation blockchains need to process an increasing number of transactions per second and the time spent saving data to the disk has become a performance bottleneck.

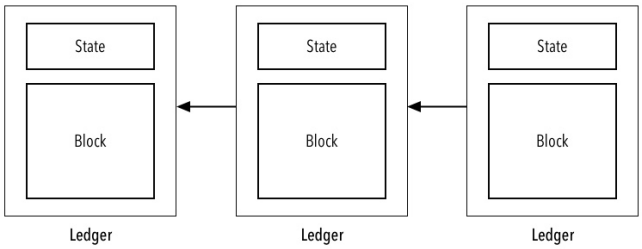
Hybrid Solution. In order to combine the fast saving speed of a NoSQL database and the ease of use of a SQL database, Caasiope is using a hybrid system where new blocks are first saved in a NoSQL database; and then saved in the SQL database by several threads working in parallel.



6. Ledger

The ledger represents the state of the blockchain at a given point in time. It contains both the history and the state of the network, both the list of the validated transactions and the list of all the accounts and balances.

Chain. Every ledger has a cryptographic reference to the previous one, effectively chaining blocks together to create an immutable history that goes back to the genesis block.



Header. The header of every ledger contains the required information to locate it within the blockchain's history, the ledger height to uniquely identified in the blockchain history, the timestamp for when it was created, the hash of the block it contains, the hash of the updated ledger state and the hash of the last ledger.

Ledger	
Timestamp	2017/10/01 07:23:15
Heigth	1786
Block Hash	XXXXXXXXXXX
State Hash	XXXXXXXXXXX
Last Ledger Hash	XXXXXXXXXXX

Blocks. When consensus is reached, a new block is produced, which contains all the transactions being confirmed. The block contains all the necessary information for the transition from the last ledger state to the current ledger state : state(h) + block(h+1) = state(h+1).

Merkle Tree. The block hash and the state hash are created using a Merkle tree. Merkle proofs can then be used to securely verify the inclusion of a transaction or the balance of an account, making Simple Payment Verification a trivial operation

Simple Payment Verification. Simple Payment Verification is a feature that allows a client to verify if a particular transaction is included in a block without downloading the entire ledger history.

7. Transactions

Transactions are used to record atomic events on the blockchain. Transactions are submitted to the network in order to modify the state of the ledger. When a transaction has been successfully validated, the transaction is included in a block and is considered irreversible once the block has been confirmed.

Transaction					
Expiration			2017/10/01 07:23:15		
Inputs			Outputs		
Alice	BTC	0.1	Bob	BTC	0.1
Bob	ETH	50	Alice	ETH	50
Message			XXXXXXXXXXX		
Fees			Alice	GAS	10

Transaction Hash	XXXXXXXXXXX
Signatures	
Alice	XXXXXXXXXXX
Bob	XXXXXXXXXXX

Hash. The hash of a transaction is a unique identifier that is computed whenever the transaction is created. The hash is computed from the data of the transaction fields and can be used to track the status of this transaction.

Expiration. Each transaction must include an expiration timestamp and cannot be included in a block after this time. This field prevents transactions to be stuck in limbos waiting to be included.

Declarations. Declarations can be included in a transaction, in order to create a special account or reveal special information required to spend from a special account.

Inputs Outputs. Every transaction can include inputs and outputs in order to transfer value from accounts to accounts. The main restriction is that the total amount in the inputs must match with the outputs.

Message. The message field can be used to attach limited data to the transaction, which is read by external services.

Network Fees. Fees are used to pay for the service provided by the network and must be paid in the native token (CAS). Transaction fees also help prioritize which transaction is included in a block first.

Signatures. In order to be valid, a transaction must include the signatures, with their associated public keys, required to spend the inputs. The private key is used to sign the transaction hash and must remain secret.

Native Currency. The native currency of the network (CAS) is used to pay network fees and the total supply will be issued at the genesis block.

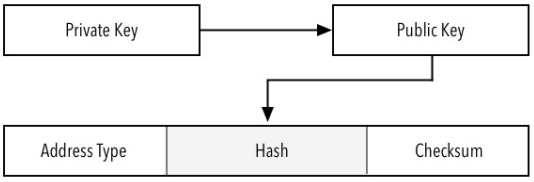
External Currencies. The Caasiope Network can support almost any currency; more currencies will be supported in the near future.

8. Accounts

Accounts are represented by an address and can hold a balance of different currencies. Special account types have different conditions to be able to spend their funds.

Account	
Address	XXXXXXXXXXX
Balances	
BTC	10
ETH	50

Addresses. The Address32 format is an address format derived from the BECH32 proposal for Bitcoin that is human readable and contains error checking and error correction.



Encoding. The address is encoded in base 32 and contains the address type, the hash, and a checksum.

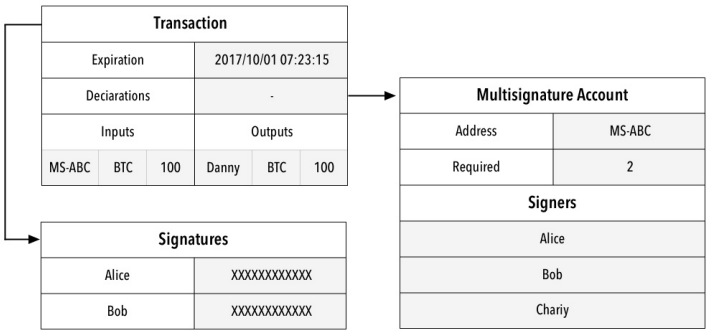
Part	Characters	Bits
Address Type	2	10
Hash	32	160
Checksum	6	30

The address type is using 2 chars; but will be a single byte. The total length of an address is 40 chars or 25 bytes, containing a 20 bytes hash like Ethereum and Bitcoin.

ECDSA Account. The simplest type of addresses is using the SHA3 hash of the public key associated with the ECDSA private key for this account.

Scripting. The protocol doesn't support scripting, but equivalent built-in functionalities will be implemented to satisfy the most popular scripting use cases.

Multisignature Accounts. The protocol supports built-in X out of Y multisignature accounts, where X signatures from the list of Y signers are required order to spend from the multisignature account. The list of the signer's addresses and the number of signatures required must be declared before you can spend from this account.



Account Declarations. Special account types like multi-signature accounts must be declared before the funds held can be spent. Account declarations must be included in the declarations field of a transaction.

Undeclared Accounts. Special accounts will produce a deterministic address that can be used even if the declaration is not published yet. Anyone can send funds to this address, but those funds cannot be spent until the declaration is published.

Partial Reveal. Multi-signature accounts allow spending even if one of the signer account has not been declared, as long as enough signatures can be gathered from known signer accounts.

Hash Lock Accounts. Hashlock is a type of account that restricts spending from this account until the associated secret is revealed.

Time Lock Accounts. Timelock is a type of account that restricts the spending from this account until a specified future time.

Hashed Time Lock Accounts. The protocol supports hashed timelock accounts using a combination of multi-signatures, hashlock and timelock account. In order to spend from the hashed timelock account, the transaction needs to be signed by the receiver and contains the secret that corresponds to the hash or be signed by the sender after lock time is elapsed.

Cross Chain Atomic Swap. The protocol supports cross chain atomic swaps that allow users to exchange assets located on two different networks without having to trust a third party.

Payment Channel. The protocol supports payment channels to allow users to make multiple transactions without committing all the transactions to the blockchain.

Lightning Network. The protocol supports external layers like the Lightning Network to operate on top of the blockchain, for even faster cross chain transactions.