Caldera/Spacemesh Research

Caldera Network:

There are basically few Proofs on which blockchain works:

- Proof of Work (PoW)
- Proof of Stake (PoS)
- Proof of Space-Time(PoST)
- Proof of Elapsed Time(PoET)
- Non-interactive proof of space-time (NIPST)

There are some key Points for Caldera Blockchain:

Challenge chain: The VDF chain based on each challenge for each sub-slot, which does not infuse anything in the middle of each sub-slot. The challenges are also used for the proofs of space. The signage points in this chain are used for the SP filter.

Reward chain: The VDF chain that contains infusions of all blocks. This chain pulls in the challenge chain and optionally the infused challenge chain at the end of each sub-slot.Infused challenge chain: A VDF chain which starts at the first block infused in a slot (which is not based on the previous slot's challenge, this is called the challenge block) and ends at the end of the slot.

Slot: the list of sub-slots which contain at least 16 reward-chain blocks based on the challenge of the first sub-slot, or later sub-slots. At the end of the slot, the infused challenge chain stops, the challenge chain pulls in the result of the infused challenge chain, and the deficit is reset to 16.

Block: a block is a collection of data infused into the rewards chain which contains: a proof of space for a challenge hash with less iterations than the slot iterations, sp and ip VDFs for both chains, optional ip VDF for the infused challenge chain, and a rewards address. Some blocks are also transaction blocks. There is a maximum of 128 blocks per slot.

Transaction Block: A block that is eligible to create transactions, along with an associated list of transactions.

Challenge block: The first block to be infused in each slot, which is not based on a previous slot's challenge. The challenge block always has a deficit of 15, and always starts off the infused challenge chain.

Peak: The peak of the blockchain as seen by a node is the block with the greatest weight. Weight is the sum of the difficulty of a block and all its ancestors, which is similar to height, but a shorter chain can have heavier weight, due to difficulty adjustments.

For a block to be considered valid, it has to provide VDFs for the challenge chain and reward chain, and optionally for the infused challenge chain if it is present. Forcing all VDFs to be included means that all three chains are guaranteed to move forward at the same rate.

Roles of Nodes:

Farmers:

Farmers are nodes which participate in the consensus algorithm by storing plots and checking them for proofs of space. They communicate with a Full Node (usually on the same machine.) Farmers also communicate with one or more Harvesters which is a service that resides on the machine where plots are stored and looks up proofs of space on behalf of the Farmer process

Timelord:

Timelords are nodes which participate in the consensus algorithm by creating proofs of time and infusing blocks into their VDFs.

Full Nodes:

Full nodes can be timelords or farmers, or they can just perform the roles of a full node. This entails broadcasting proofs of space and time, creating blocks, maintaining a mempool of pending transactions, storing the historical blockchain, and uploading blocks to other full nodes as well as wallets (light clients).

We have successfully Studied Chia Network Consensus Algorithm and found the functions responsible for reward distribution and validation for Proof of Space-Time:

- **VDF(Verifiable Delay Function):** Verifies the Proof of Time.
- **Proof of Space function:** Verifies Proof of Space
- **Rewards function:** Handles reward distribution.
- **quality string(qs):** Iterates through all blocks to choose quality blocks.
- **Timelord function:** Keeps Track of time.
- **Nodes:** Various members of chain
- **Signage point:** Initialisation of reward function

Locations of these functions in code:

- chia\types\blockchain_format\vdf.py
- chia\types\blockchain_format\proof_of_space.py
- chia\types\blockchain_format\reward_chain_block.py
- chia\consensus\pot_iterations.py
- chia\timelord
- chia\full_node\signage_point.py

->Got the methods which triggers these reward functions
->Studied Various Nodes of Chia and their Functionalities
->Went through their codes, different repositories, social channels, and blogs to understand their protocol and working
->Went through Chia-Blockchain and its netspace clone major portions.

They combine proof of elapsed-time (PoET) with proof of space-time (PoST) to create non-interactive proof of space-time (NIPST)

For each of the 64 signage points, as they are released to the network every 9 seconds, or every 3.125M iterations, the farmer computes the plot filter and sees how many plots pass.

For each of the plots that pass the filter for each signage point, the farmer computes the required iterations.

In this example, the farmer only gets required_iterations < 3.125M one time in the whole sub-slot (let's say it's 2.2879M).

This is at the 14th signage point. The infusion iterations is computed as:

$infusion\ iterations$ = $signage\ point\ iterations$ + 3 * $sp\ interval\ iterations$ + $required\ iterations$ = 14 * 3. 125M + 3 * 3. 125M + 2. 2879M = 55. 4129M

Spacemesh:

As per Spacemesh Research we have got the following:

Went through Spacemesh's Full node Repository for Reward distribution and mesh's Algorithm

->Found the functions and methods responsible for distribution of rewards.

- mesh\reward.go
- api\node.go
- go-spacemesh-develop\collector\collector.go
- go-spacemesh-develop\common\types\transaction.go
- go-spacemesh-develop\mesh\mesh.go
- go-spacemesh-develop\tests\tx_generator\actions.py

->Understood the validators which verify a block for reward

To-Dos:

-> Unlike chia, we still have to find out the methods which triggers these reward functions in spacemesh and at what time interval.
->Once we find that we'll be trying to implement those algorithms in Caldera Blockchain.

Challenges:

->The major challenge in applying these changes will be that Caldera-blockchain is developed in Python and Spacemesh's Full node is developed in Go-Language. So all the codes and libraries will have to be created from scratch.
->The default dependencies checks will also have to be done if there will be any internal dependencies.

References:

- https://github.com/Chia-Network/chia-blockchain/wiki
- https://github.com/Chia-Network/chia-blockchain/wiki/Consensus-Algorithm-Summary
- https://medium.com/@chia.net/chia-vdf-competition-guide-5382e1f4bd39
- https://www.chia.net/assets/Chia-New-Consensus-0.9.pdf
- https://github.com/chianetspace/chianetspace
- https://www.youtube.com/watch?v=jvtHFOlA1GI
- https://www.youtube.com/watch?v=yTjXFb-ZhOY
- https://chiaforum.com/t/what-is-the-algorithm-to-determine-winner/3774
- https://docs.google.com/document/d/1tmRIb7lgi4QfKkNaxuKOBHRmwbVlGL4f7EsBDr_5xZE/edit
- https://chiaforum.com/t/no-rewards-for-9863-plots-for-about-a-month/8996
- https://spacemesh.io/faq/#economics
- https://pkg.go.dev/github.com/spacemeshos/go-spacemesh
- https://github.com/spacemeshos
- https://medium.com/spacemesh
- https://medium.com/spacemesh/spacemesh-for-dummies-8e2311bba2d4
- https://github.com/spacemeshos/go-spacemesh.git
- https://github.com/Chia-Network/chia-blockchain/wiki/Quick-Start-Guide
- https://github.com/Chia-Network/chia-blockchain/wiki
- https://github.com/Chia-Network/chia-blockchain/wiki/How-to-Check-If-Everything-is-Working-(or-Not)
- https://chiadecentral.com/the-beginners-guide-to-farming-chia-coin-on-windows/
- https://github.com/Chia-Network/chia-blockchain/wiki/Beginners-Guide
- https://pool.space/