Bridging PoW and DAG in Starcoin's BlockChain

Westar Labs

December 6, 2023

Outline

What's Starcoin

The only chain:

- POW: with enhancing Nakamoto Consensus
- Move: Support move lang as smart contract vm

Why DAG

- Parallel Block Processing: Increased Efficiency.
- Consensus Mechanism Update: Maximizing Computational PoWer Utilization.

What's FlexiDAG

- Enhancing Nakamoto Consensus Performance.
- Boosting with DAG for Performance and Usability.

Nakamoto Consensus

- Drop orphan blocks.
- Setting block time target to 10 min.
- Does't measure network congested.

Starcoin

- Commit orphan blocks header as uncle
- Update block time target by network congested situation
- Measure network congested by uncle rate

DAG

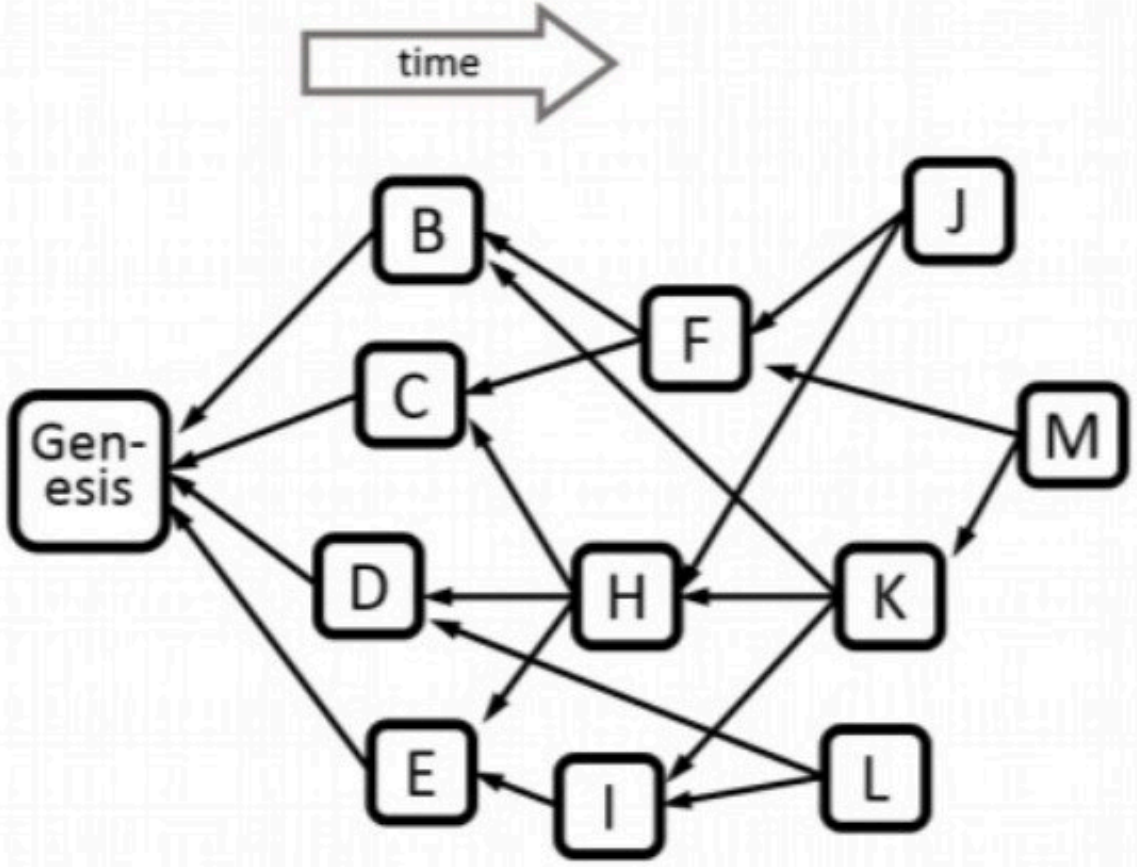
- blocks reference multiple predecessors, forming a Directed Acyclic Graph, a blockDAG.

Terminology

- A DAG of blocks is denoted G = (C, E) C: blocks, E:Edge
- past (B, G) <- C
- future (B, G) <- C
- anticone (B, G) <- C
- tips(G)

Topology

- past(H) = {Genesis,C,D,E}
- future(H) = {J,K,M}
- anticone (H) = {B, F, I, L}
- tips(G) = {J,L,M}



DAG mining protocol

1. Miner references in its new block all blocks in tips(G) where G is the DAG that the miner observes locally at the time when the new block is created.
2. Miner should broadcast its new block as fast as possible.

DAG ordering protocol

- The confirmed block DAG.
- Output the order of confirmed block DAG.

Well-connected DAG

- the set of honest blocks in B' s anticone is typically small
- consists only of blocks created in the interval [t − D, t + D].
- The proof-of-work mechanism guarantees that the number of blocks created in an interval of length 2 · D is typically below some k > 0

K-cluster

Definition 1. *Given a DAG* $G = (C, E)$*, a subset* $S \subseteq C$ *is called a* k-cluster, *if* $\forall B \in S : |anticone(B) \cap S| \leq k$.

Max K-cluster

Maximum k-cluster SubDAG (MCS_k)

Input: DAG $G = (C, E)$

Output: A subset $S^* \subset C$ of maximum size, s.t. $|anticone(B) \cap S^*| \leq k$ for all $B \in S^*$.

DAG protocol

(1) Given a block G, solve MCSk (G); let' s refer to its output as the Blue set and to its complement set as the Red one.
(2) Determine the order between Blue blocks according to some topological sort.
Then, for any Blue block B, add to the order just before B all of the Red blocks in past (B) that weren' t added to the order yet; these Red blocks too should be added in a topological manner.
But It's a NP problem

GHOSTDAG protocol

- Find a k-cluster using a greedy algorithm.
- The algorithm constructs the Blue set of the DAG by first inheriting the Blue set of the best tip Bmax.

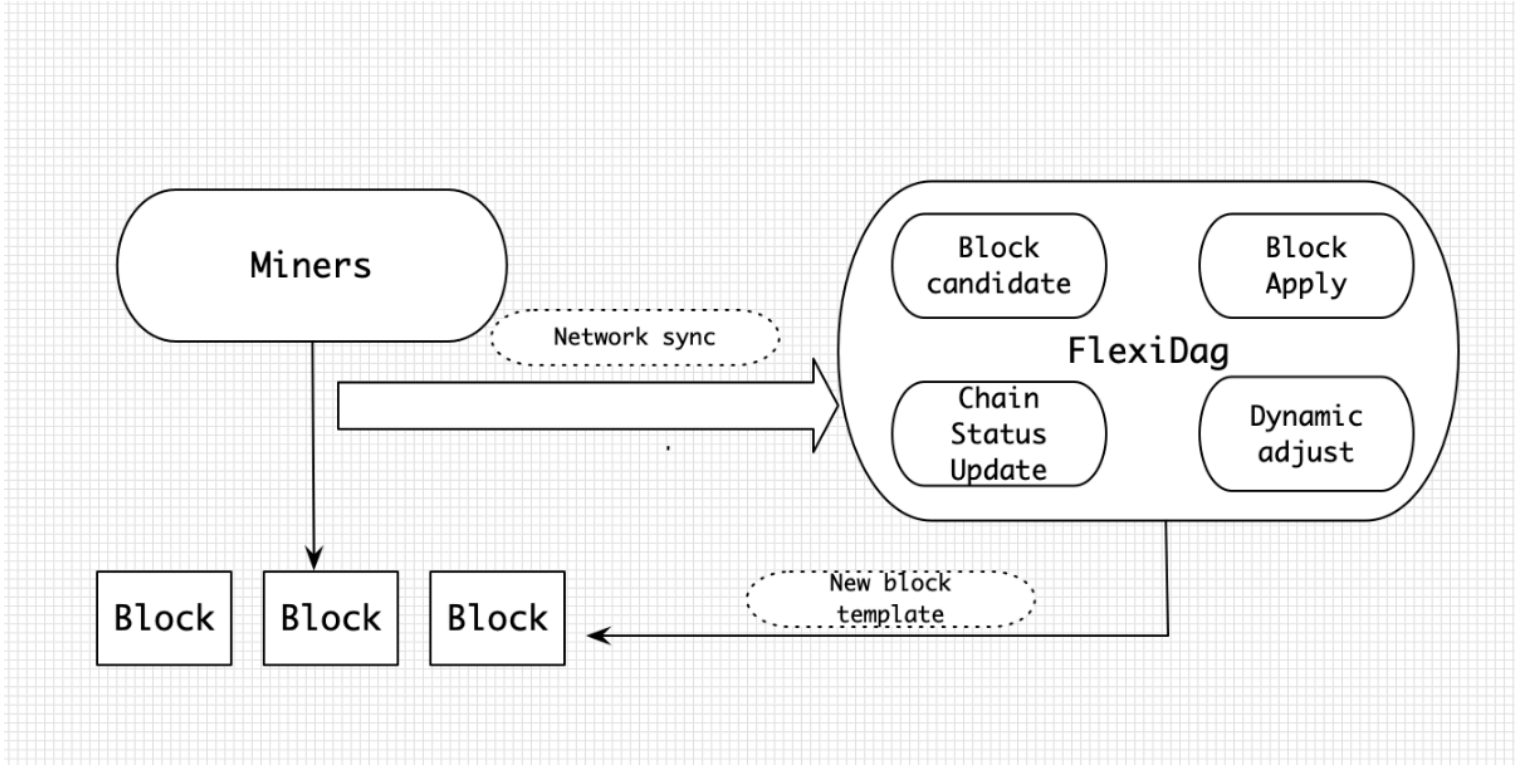
Greedy algorithm

Input: G – a block DAG, k – the propagation parameter
Output: $BLUE_k(G)$ – the Blue set of G; ord – an ordered list containing all blocks in G

1: **function** ORDERDAG(G, k)
2: **if** $G == \{genesis\}$ **then**
3: **return** $[\{genesis\}, \{genesis\}]$
4: **for** $B \in tips(G)$ **do**
5: $[BlueSet_B, OrderedList_B] \leftarrow$OrderDAG($past(B), k$)
6: $B_{\max} \leftarrow \arg\max\{|BlueSet_B| : B \in tips(G)\}$ (break ties according to lowest hash)
7: $BlueSet_G \leftarrow BlueSet_{B_{\max}}$
8: $OrderedList_G \leftarrow OrderedList_{B_{\max}}$
9: add $B_{\max}$ to $BlueSet_G$
10: add $B_{\max}$ to the end of $OrderedList_G$
11: **for** $B \in anticone(B_{\max}, G)$ **do** in some topological ordering
12: **if** $BlueSet_G \cup \{B\}$ is a k-cluster **then**
13: add B to $BlueSet_G$
14: add B to the end of $OrderedList_G$
15: **return** $[BlueSet_G, OrderedList_G]$

Starcoin 1.0 with DAG

Overview


Miners
Network sync
Block candidate
Block Apply
FlexiDag
Chain Status Update
Dynamic adjust
Block
Block
Block
New block template

Make the Uncle block executable

- Incorporate uncle blocks as part of the main chain.
- Ensure transaction executability.
- make sure the txn order

Block construction consensus

1. New Block Creation: Form new blocks pointing to eligible leaves in the DAG.
2. Optimal Parent Node: Identify the parent node with the highest workload as the optimal choice.
3. Consensus Election: Select anti-cone nodes that maintain DAG connectivity to expand the block candidate set.
4. Node Application: Sort and apply selected nodes in sequence after the optimal parent.
5. Chain State Update: Determine the latest block by choosing the longest chain from forks.
6. Solution Validity: Ensure the intersection of the anti-cone and confirmed block sets does not exceed limit K.
7. Integration of Uncle Blocks: Uncle blocks from Starcoin 1.0 are elected into the blockchain, allowing concurrent block issuance.

Transaction and Block index

- block hash -> main block id
- main block id -> (block accumulator info, txn accumulator info)
- main block id -> block info
- main block id -> (txn id, txn info id)
- main block id -> txn info ids

Security Analysis

- Double Spending Attack
- Selfish Mining or Long-Range Attacks
- Sybil Attack
- Denial of Service (DoS) Attack

VS Kaspa

- Utxo vs account
- Move smart contract vs script
- Dynamic adjust vs constant consensus