Bitcoin Blockchain: Fast and Secure Transactions

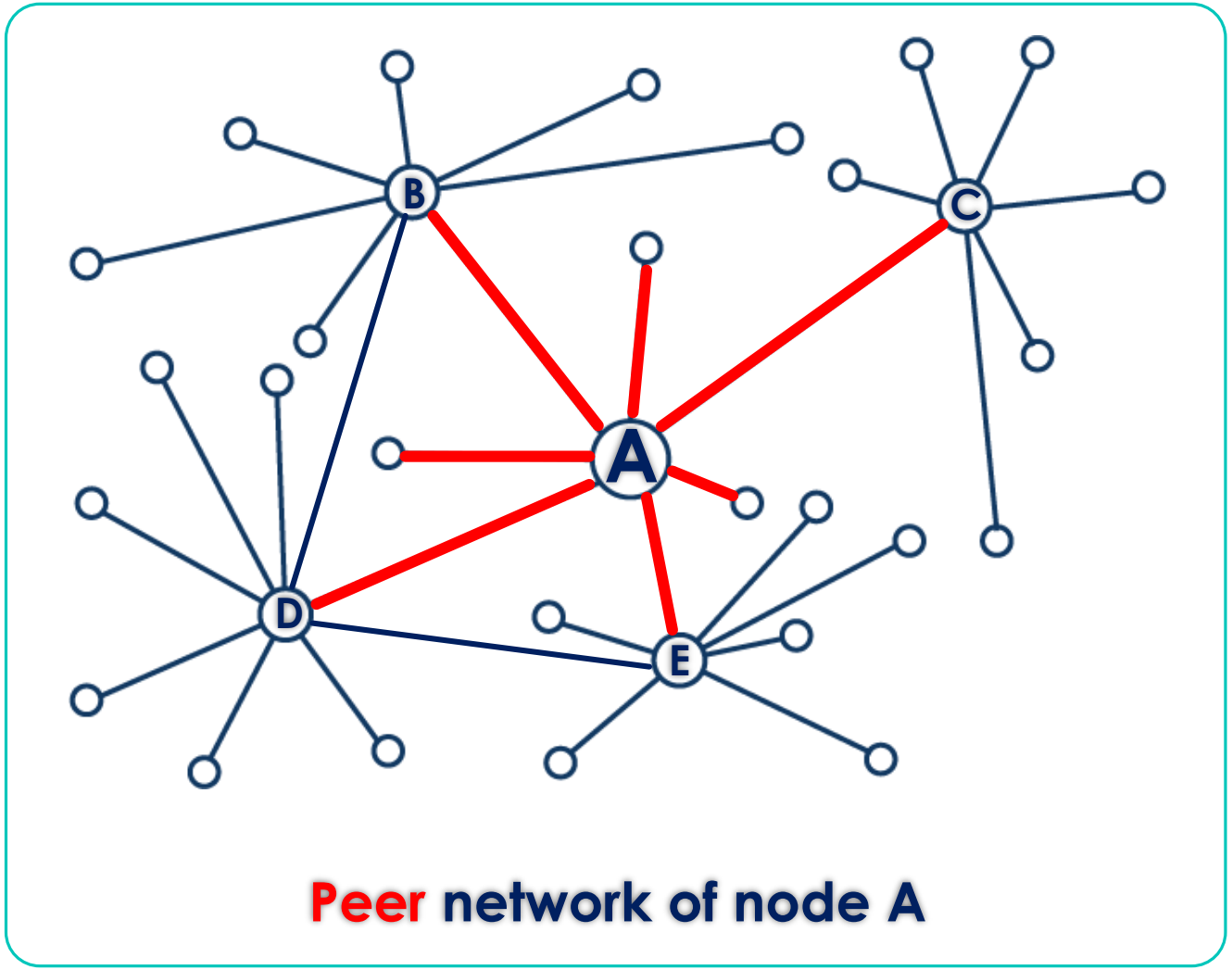
Presenter: Dae Kwang Lee

Adviser: Matthew Anderson

What is Bitcoin?

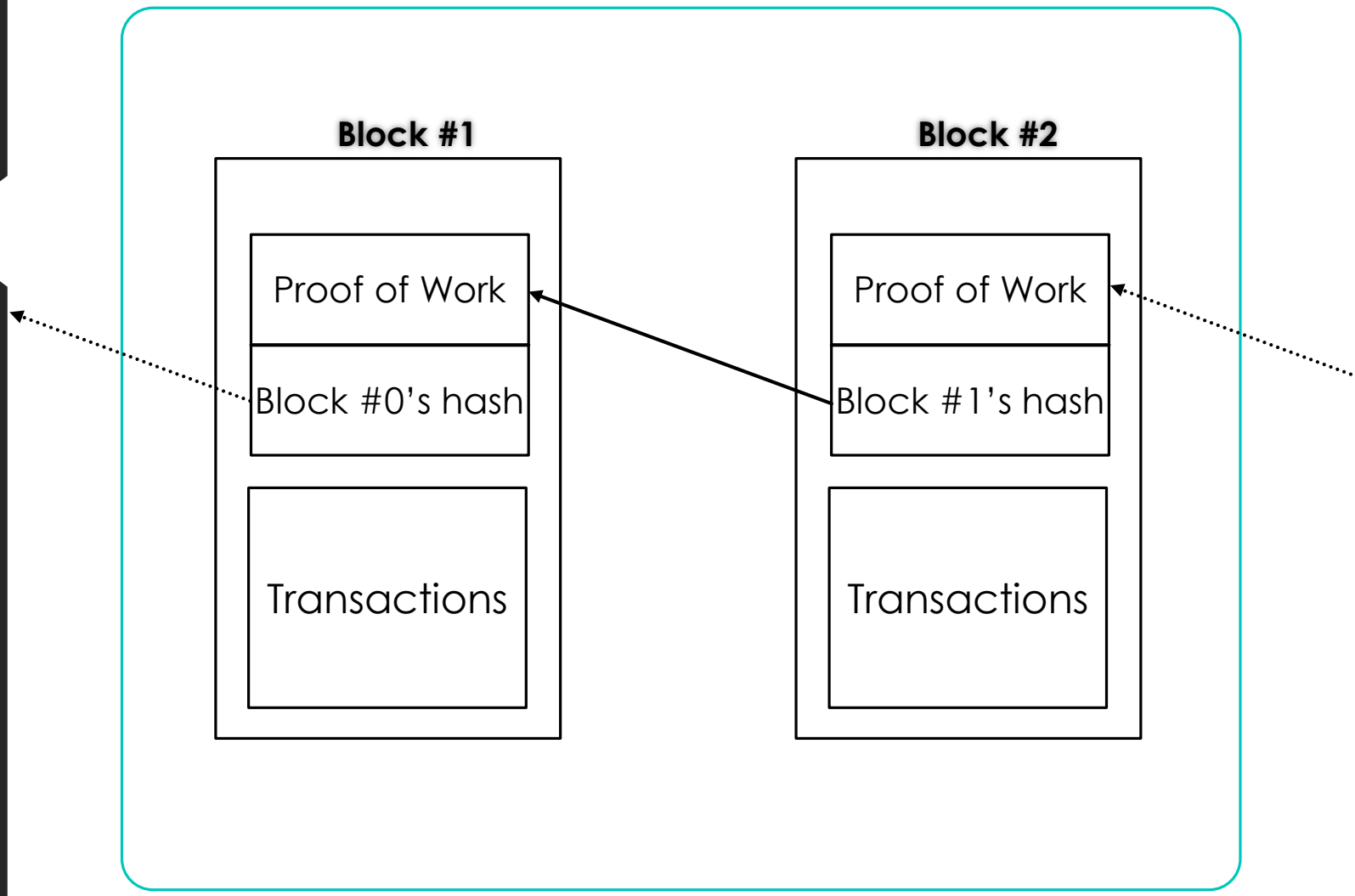
Decentralized peer-to-peer electronic payment system

- Each node stores a copy of the public transaction history
- Transactions verified by nodes
- Nodes send new transactions to their peers



Transaction

- **Input:** previous output hash
- **Output:** instructions for sending bitcoins

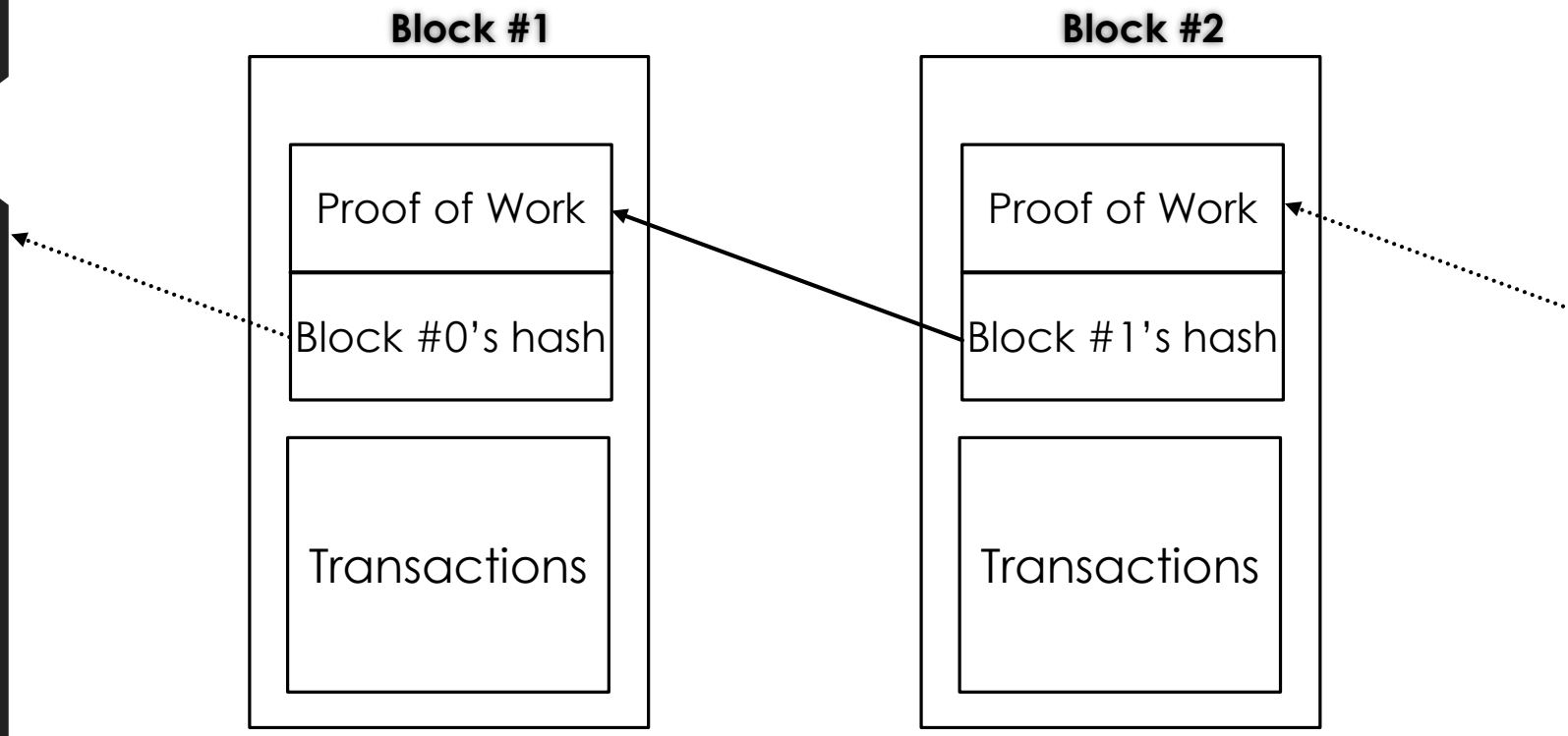


Blockchain

- **Miner**
  - Collects transactions & generates a block by solving computational puzzle*
  - Sends block to peers
  - Mines a 1MB block / 10min
  - Incentivized with bitcoin and **RESIDUALS**

***Proof Of Work**

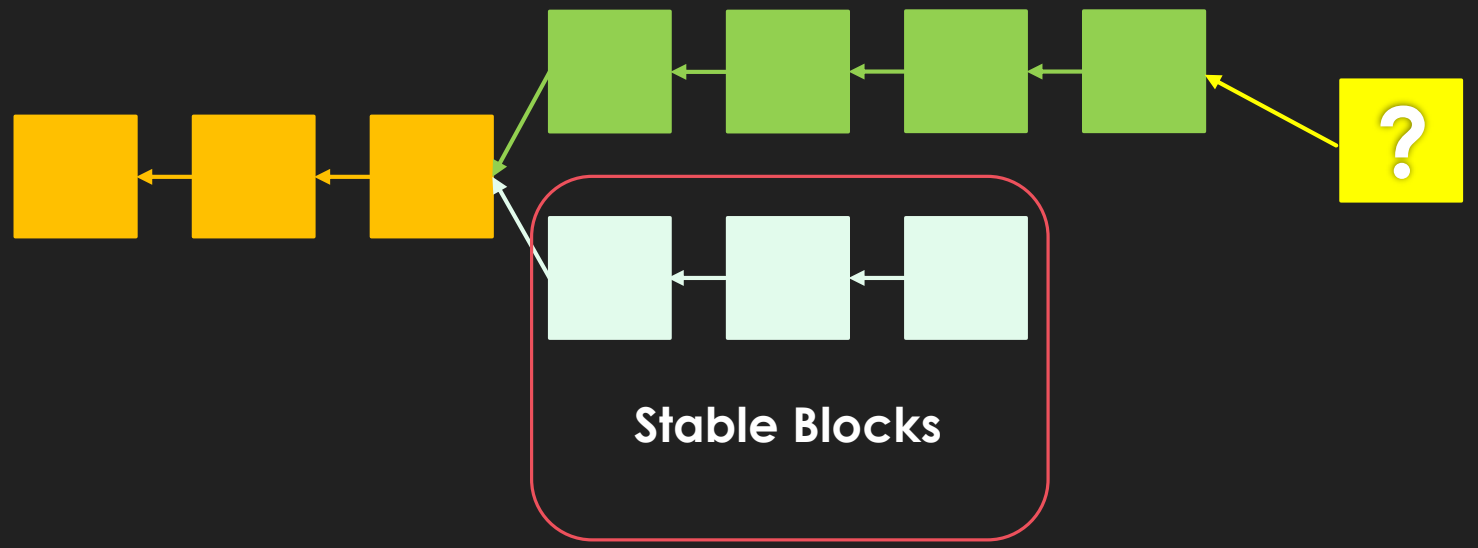
- Hash (previous block's hash + transactions + nonce) <= target



Blockchain Forking and Primitive Solution

Forking happens when..

- Duplicate blocks are generated
- Each node has different history
- Transactions are not validated!
- May develop into **selfish-mining** to revert transactions and double-spend



NAKAMOTO CONSENSUS

- Nodes agree on this policy
- Resolves by adding block on the longest chain

Is there a faster policy that generates less stale blocks?

Nakamoto Consensus

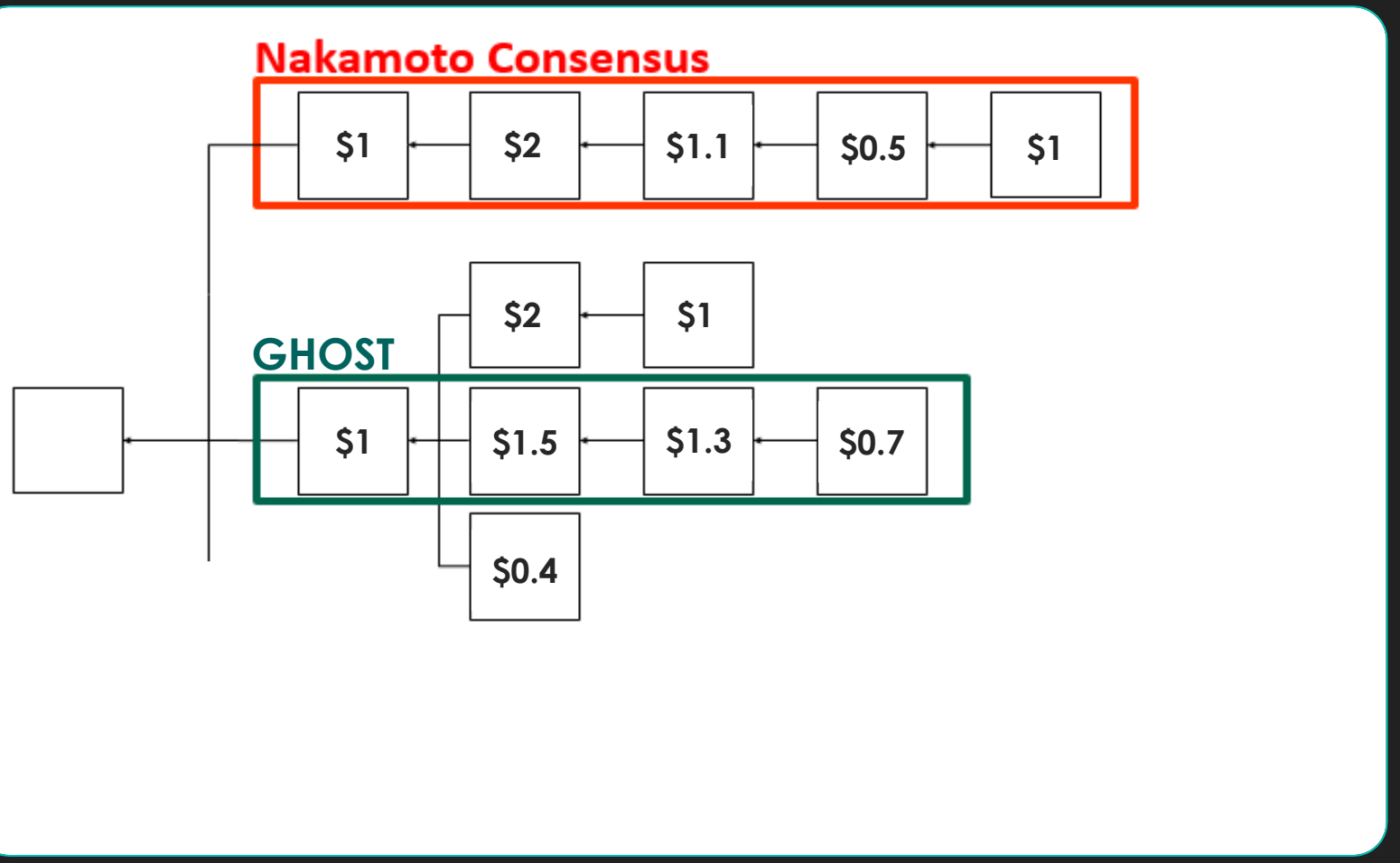
- Adds block onto the longest chain

Greedy Heaviest Observed SubTree (GHOST) 2013

- Adds block on the heaviest subtree at each fork
- Faster but generates more stale blocks

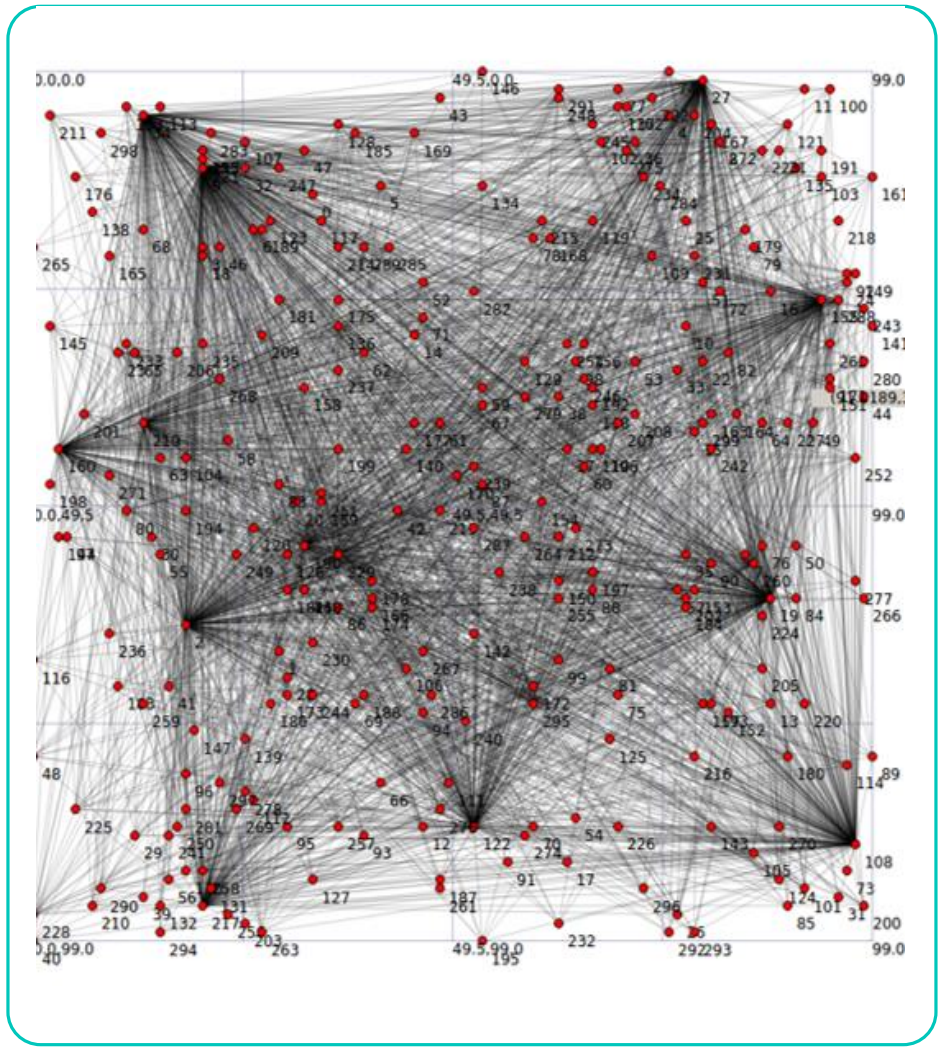
Highest Residual Selection Policy (HIRES)

- Adds block on the most expensive subtree at each fork



Bitcoin Simulator

- **Arthur Gervais**
- Collects block size data from Blockchain.Info from May 2015 to November 2015
- 16 threads of mining activities
- Can adjust block parameters
- Returns each node's copy of the ledger



Methods

I. Collected txFee data to build probability distribution
II. Used txFee probability distribution during mining
III. Made miners to pick the highest residual for each mining activity
IV. Optimized HR policy to go five level down

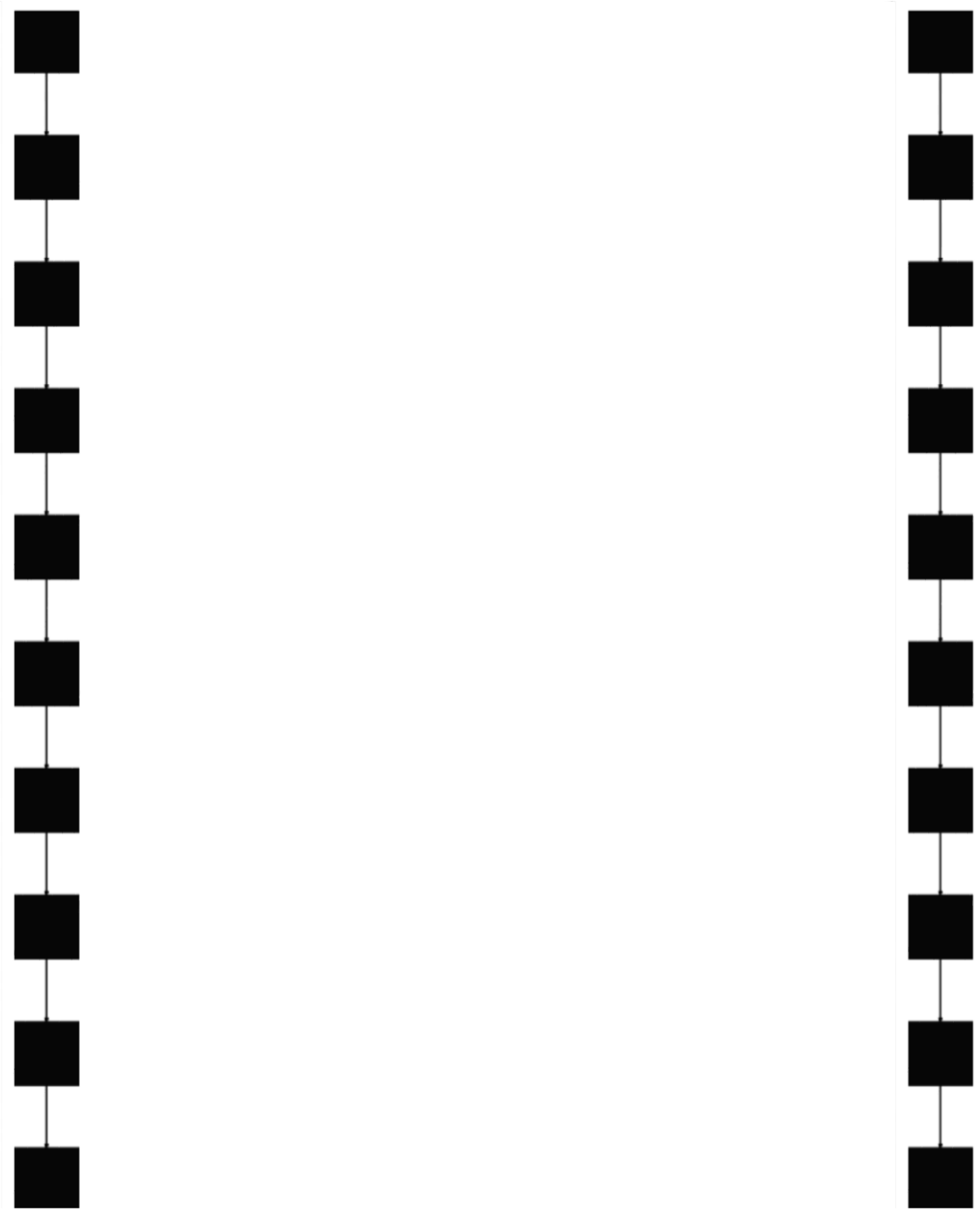
Experiments:

I. Typical parameters
II. Extreme parameters
III. Extreme parameters with selfish mining

Experiment I

Left: HIRES

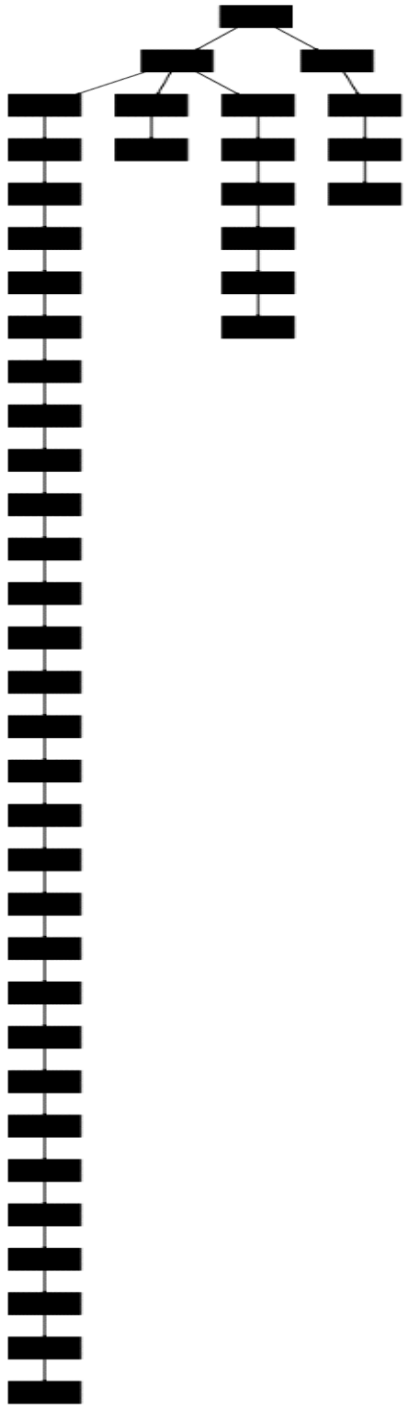
Right: NAKAMOTO

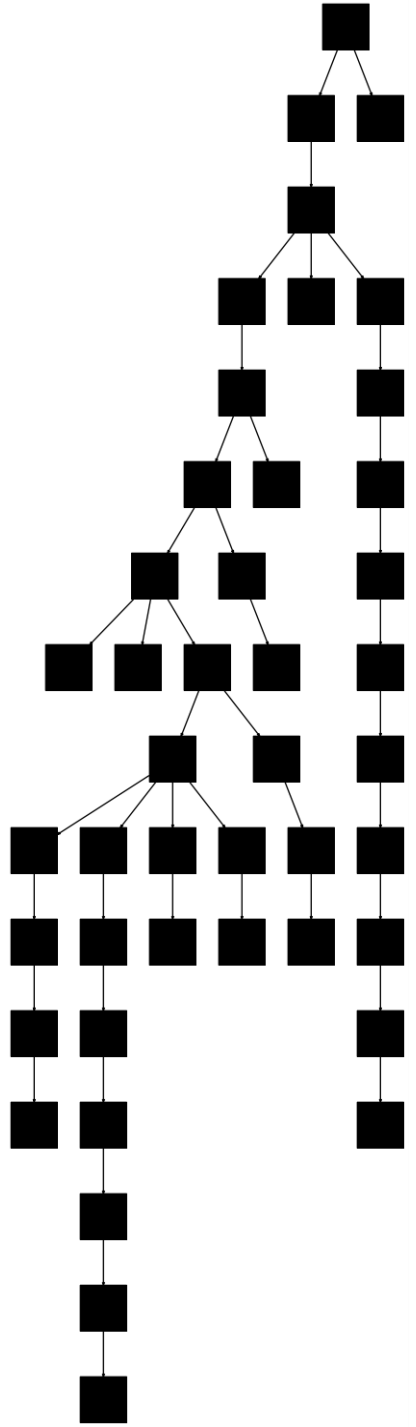


Experiment II

Left: HIRES

Right: NAKAMOTO





Experiment III

Left: HIRES

Right: NAKAMOTO

Result

Parameters	Generating 100 1MB blocks within 6s and distributing to 500 nodes	Generating 100 1MB blocks within 6s while selfish miner > 50%
Nakamoto	▪ # Total blocks: 42.120 ▪ # Stale blocks: **28.094** (66.7%) ▪ Mean Block Propagation Time: **87.073**	▪ # Total blocks: 102.13 ▪ # Stale blocks: **46.10** (45.1%) ▪ Honest Mining Income = 55.46 ▪ Attacker Income = 54.92 (-0.009%)
HIRES	▪ # Total blocks: 31.60 ▪ # Stale blocks: **12.1** (38.3%) ▪ Mean Block Propagation Time: **128.02**	▪ # Total blocks: 99.36 ▪ # Stale blocks: **38.012** (38.2%) ▪ Honest Mining Income = 59.30 ▪ Attacker Income = 55.206 (-6.9%)

Conclusion:

1. **HIRES is slower and generates less stale blocks**
2. **HIRES incentivizes attackers less than honest miners**
3. **HIRES contradicts my hypothesis based on the GHOST**

- Experiment I
  - Both policies generate 0 stale blocks
- Experiment II
  - HIRES:
    - Less stale blocks
    - Less blocks in total (timeout expired)
    - Greater block propagation time
- Experiment III
  - HIRES:
    - Less stale blocks
    - Attacker loses more money

Future Work

Optimize the new policy to propagate more blocks

I. Fast propagation
II. Micro payment
III. Makes fewer stale blocks

THANK YOU