The Blockchain Propagation Process

Aapeli Vuorinen

May 31, 2019

In this talk

- What is Bitcoin?
- What are blockchains and how do they work?
- What is the blockchain propagation process?
- Why does the propagation delay matter?

Bitcoin

- An electronic payment system
- Created by “Satoshi Nakamoto” in 2008, implemented in 2009 in the Bitcoin reference client
- Introduced the concept of a blockchain

Blockchain

- A database and a protocol by which a decentralised, global community can reach consensus on a ledger of events without a central authority or clearing house
- Reaching consensus: coming to agreement on the state of the ledger
- Relies heavily on cryptography
- Once the ledger has been built, it is immutable

Addresses: public key cryptography

- Key pair: $(\text{Private key}, \text{Public key})$.
- Two operations:

$$\text{Sign}(\text{Data}, \text{Private key}) \mapsto \text{Signature}$$
$$\text{Verify}(\text{Signature}, \text{Data}, \text{Public key}) \in \{\text{Valid}, \text{Invalid}\}$$

- Impossible to forge signatures
- Bitcoin address: a public key
- Normally use elliptic curve cryptography: discrete logarithm problem on a group constructed on elliptic curves over finite fields

Transactions

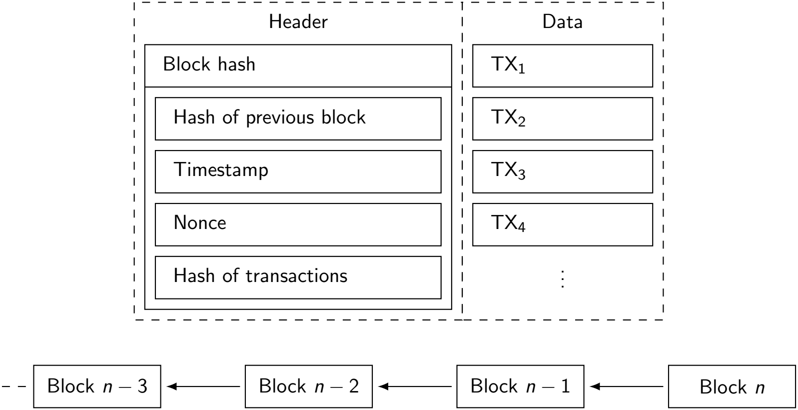
$$\text{Sign}(\text{Data}, \text{Private key}) \mapsto \text{Signature}$$
$$\text{Verify}(\text{Signature}, \text{Data}, \text{Public key}) \in \{\text{Valid}, \text{Invalid}\}$$

- A transaction: sign “Transfer x Bitcoins to account y” with your own private key
- Anyone can verify this!

Double spending

- I simultaneously send Alice and Bob one Bitcoin
- Do I have enough Bitcoins for both transactions?
- If not, then which transaction is valid, i.e. occurred first?
- Need a database: blockchain

Blocks and the blockchain



- A block is a bundle of transactions
- A block always refers to the last block forming a blockchain
- The miner who finds the block gets a reward
- Who gets to create the next block?

Cryptographic hashes

Cryptographic hash functions map any binary string into a number and:

1. input is completely uncorrelated to output, any change in input completely scrambles output;
2. computationally infeasible to find an input that matches any given hash; and
3. computationally infeasible to find two inputs with the same hash.

```
hash("illustration.")=5613492
hash("illustration!")=1668603
hash("illustration?")=6393172
```

- Output should be distributed uniformly over possible outputs!

Proof-of-work

- `hash` $\in \{0, \ldots, n-1\}$
- t = Threshold, controls difficulty
- Example $n = 10000$, $t = 10$:

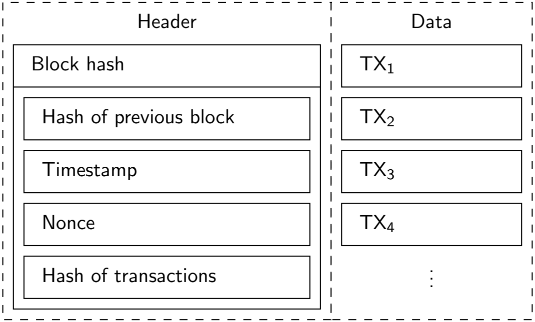
`hash("Message:`Hello audience. Nonce: 1")=1628 ✗
`hash("Message:`Hello audience. Nonce: 2")=8445 ✗
`hash("Message:`Hello audience. Nonce: 3")=4184 ✗
...
`hash("Message:`Hello audience. Nonce: 1084")=9 ✔

- Number of hashes before success is geometrically distributed
- Expected hashes before finding one below threshold is n/t
- Probabilistic bounds on number of hashes before threshold is reached

The mining process

- Bitcoin uses this proof-of-work to limit the number of blocks mined
- Randomly selects next miner to build the next step in the ledger
- The threshold is deterministically adjusted every 2016 blocks to control the block arrival rate
- Bitcoin blocks arrive approximately every 10 minutes
- Miners are incentivised by a mining reward

Hashes in blocks and transactions



- Block is identified by its hash: changing a header field or transaction completely scrambles the block hash
- Means contents of blocks are immutable once they are in the chain

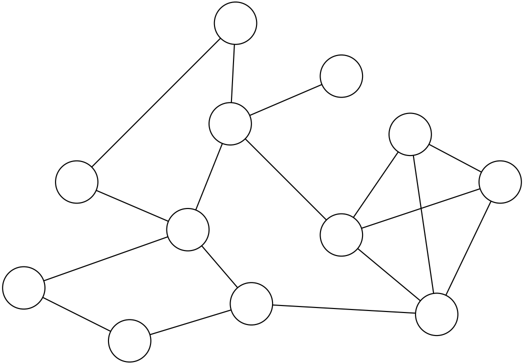
Recap

1. Create transaction object and sign it with the private key corresponding to the Bitcoin address you're sending from
2. Send it to the network and hope miners pick it up
3. Proof-of-work scheme randomly picks next miner
4. The miner hopefully added your transaction to that block (ostensibly if you put a high enough "tip")
5. The miner sends out the block to everyone on the network
6. This new block refers to the last block in the chain
7. The blockchain is advanced by one step

Recap

1. Create transaction object and sign it with the private key corresponding to the Bitcoin address you're sending from
2. Send it to the network and hope miners pick it up
3. Proof-of-work scheme randomly picks next miner
4. The miner hopefully added your transaction to that block (ostensibly if you put a high enough "tip")
5. The miner sends out the block to everyone on the network
6. This new block refers to the last block in the chain
7. The blockchain is advanced by one step

The Bitcoin network

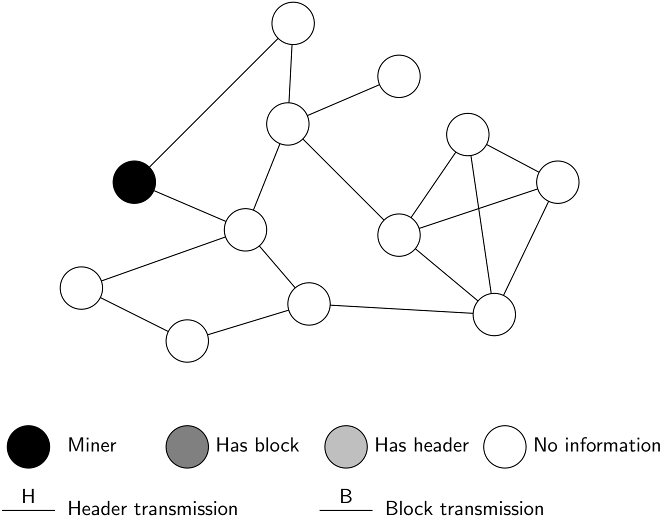


- Servers called “nodes” running a Bitcoin client
- Each node maintains a complete copy of the blockchain
- Each node is connected to random “peers”
- Bitcoin currently has about 8000-12000 nodes

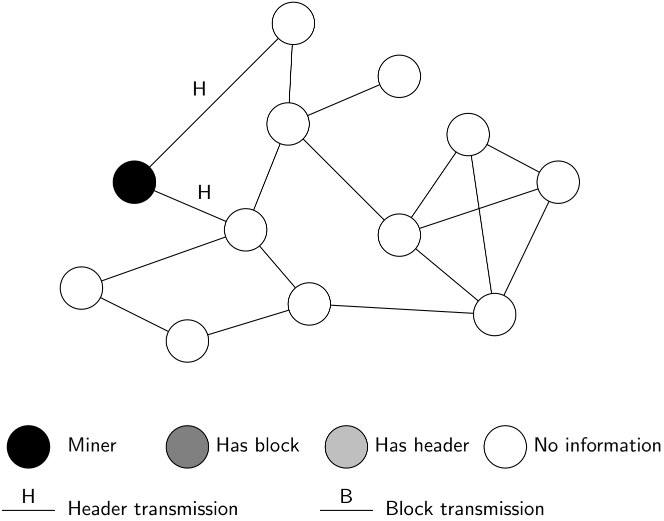
The block propagation process

- A miner finds a new block
- They send out the header to each of their peers
- Each of those peers check the header and forward it to their peers, then ask for the full block
- The peers check the block by verifying transactions before propagation it further
- The network reaches consensus when the block has propagated to every node on the network
- Headers move faster than blocks

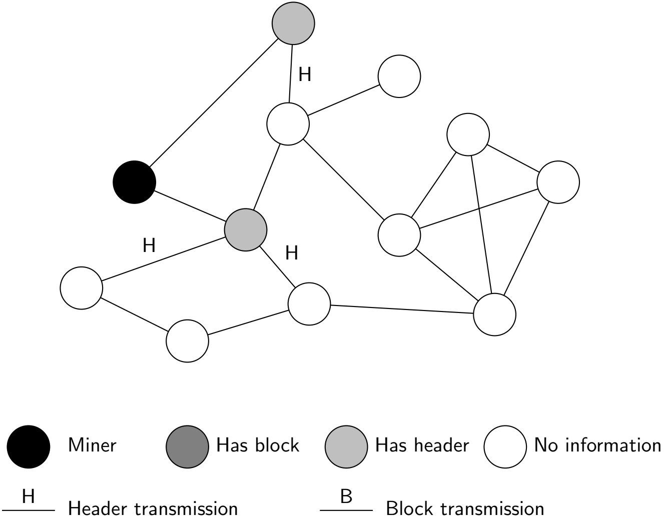
Block propagation



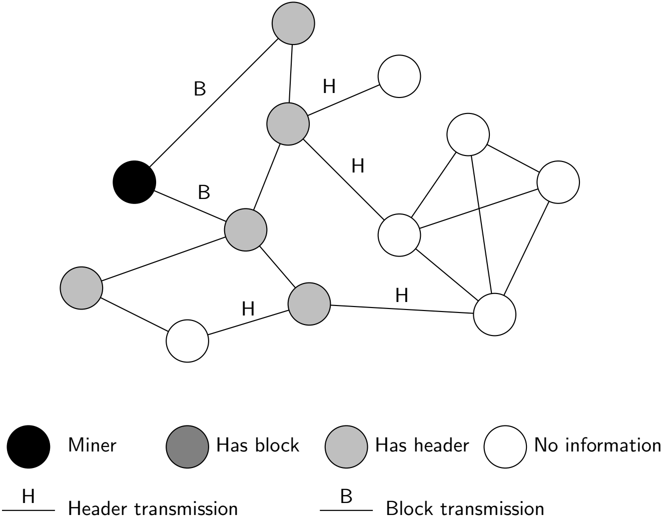
Block propagation



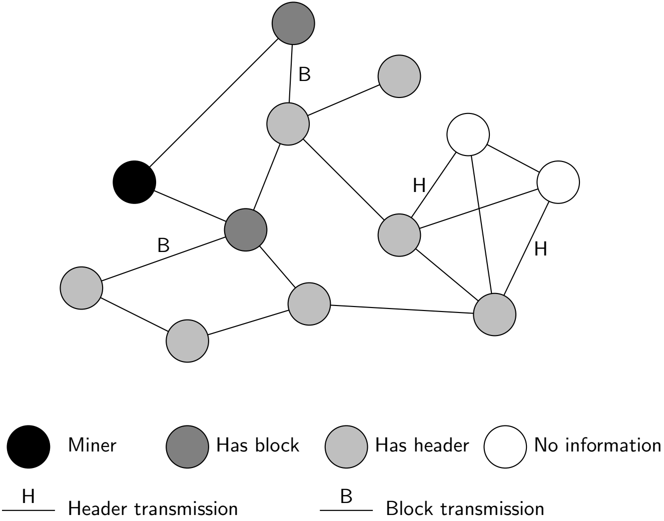
Block propagation



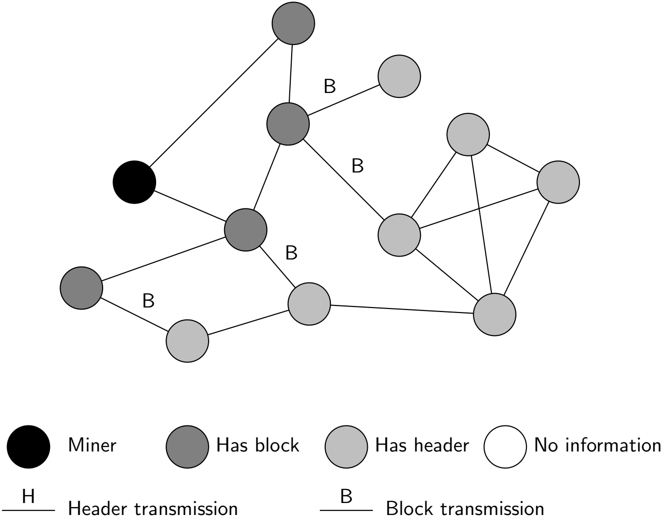
Block propagation



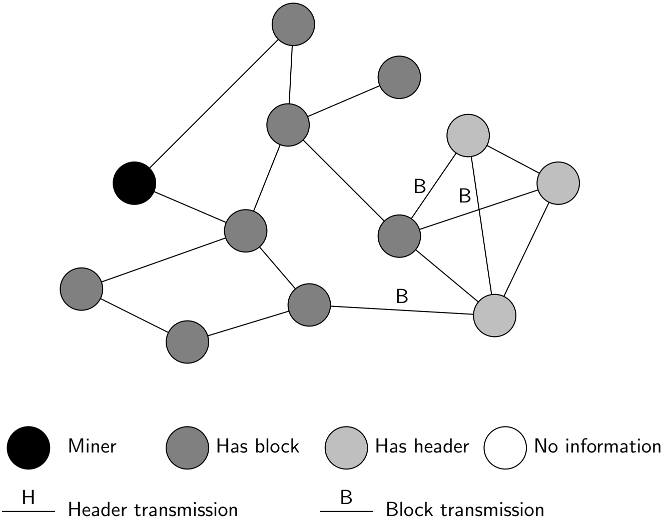
Block propagation



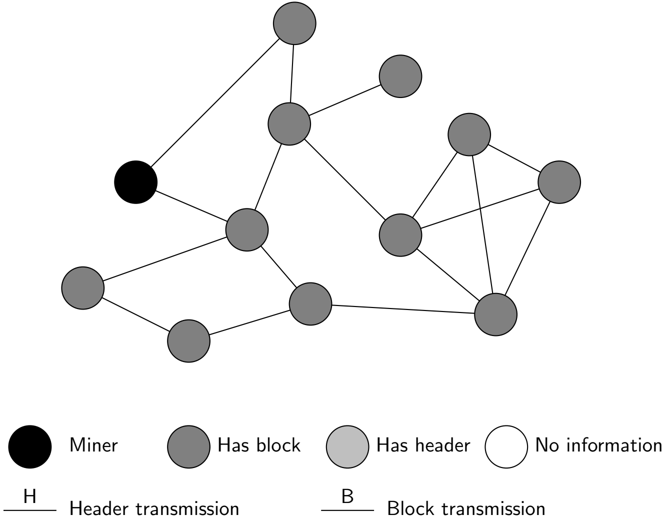
Block propagation



Block propagation



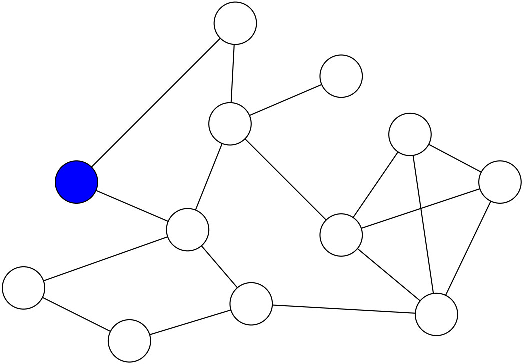
Block propagation



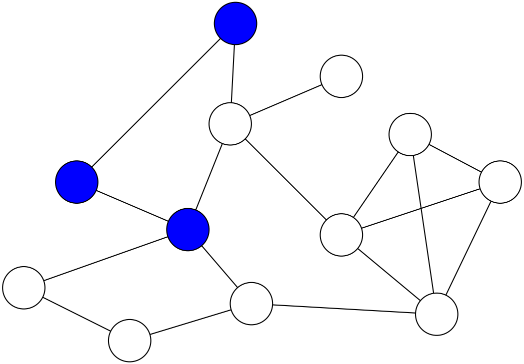
Block propagation delay

- Processing: time taken to verify transactions against current ledger
- Transmission: time taken for block to travel across wire
- Longer delay leads to longer time until consensus
- Slower transaction processing time
- **Increases rate of “forks”**

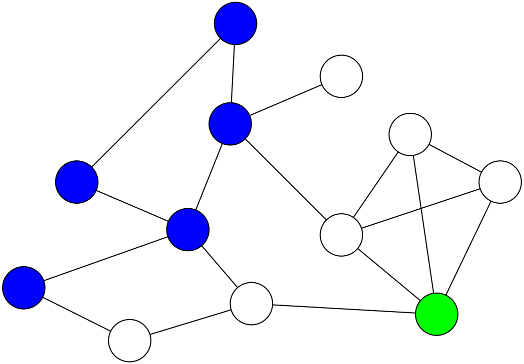
Forks



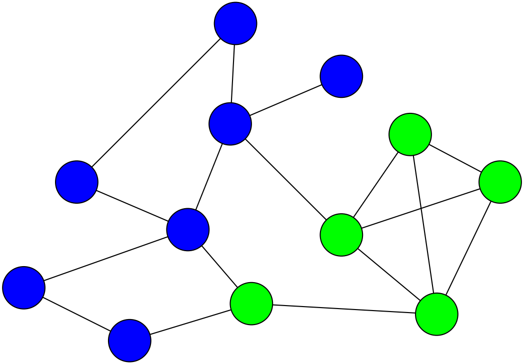
Forks



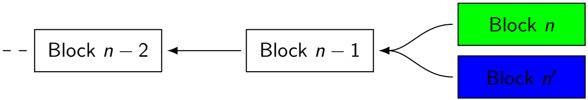
Forks



Forks

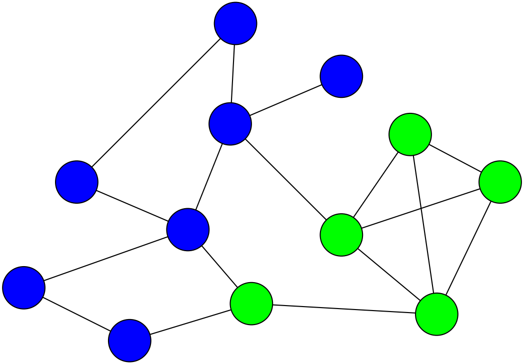


Forks

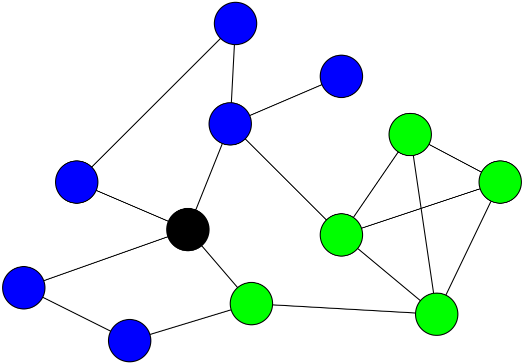


- Sometimes another miner will find a block before the block has propagated through a network
- This causes a fork in the blockchain
- **Rate of forks is influenced by the propagation delay and the block arrival rate**

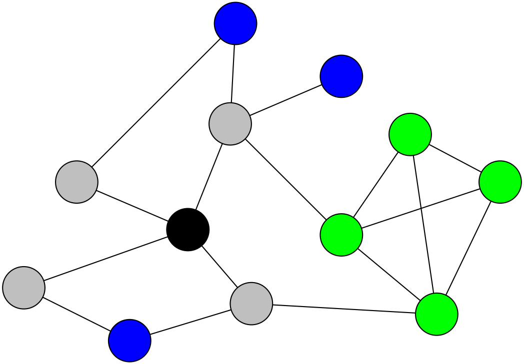
Resolution of forks



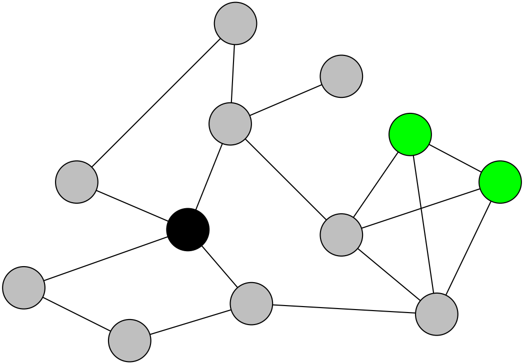
Resolution of forks



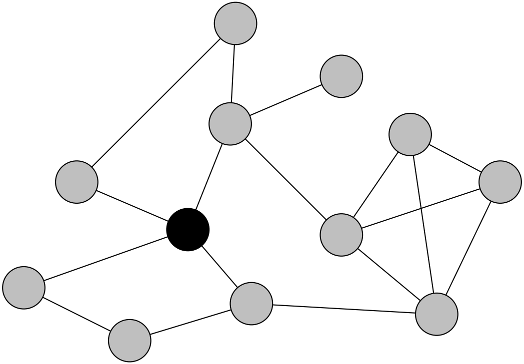
Resolution of forks



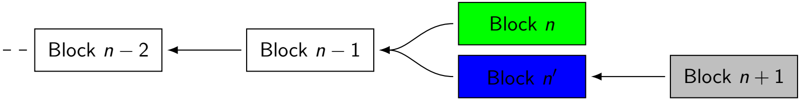
Resolution of forks



Resolution of forks



Resolution of forks



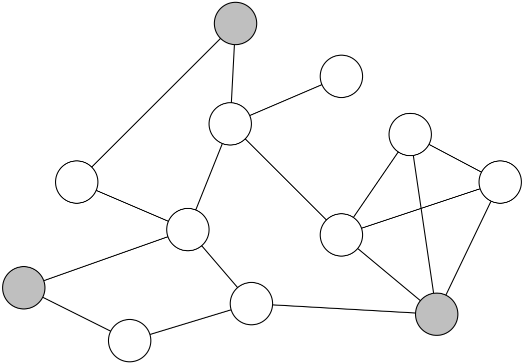
- Eventually a miner on one of the two forks finds a new block
- The whole network switches to that block and the fork is resolved
- Nodes always switch to longest chain

Selfish-mine strategy

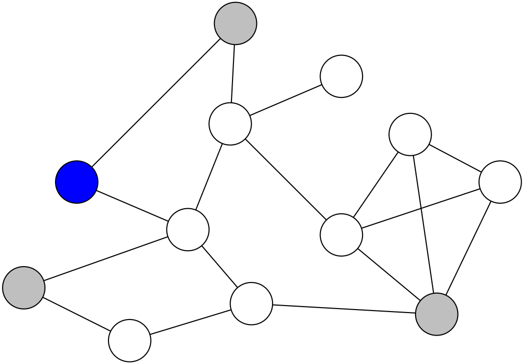
J. Göbel, H.P. Keeler, A.E. Krzesinski, and P.G. Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104:23 – 41, 2016.

- Allows adversarial miners to inflate their share of mining rewards when there is a propagation delay

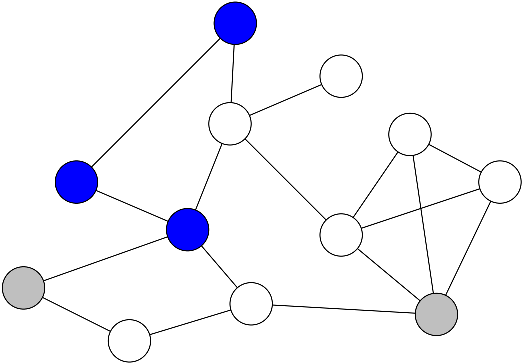
Selfish-mine strategy



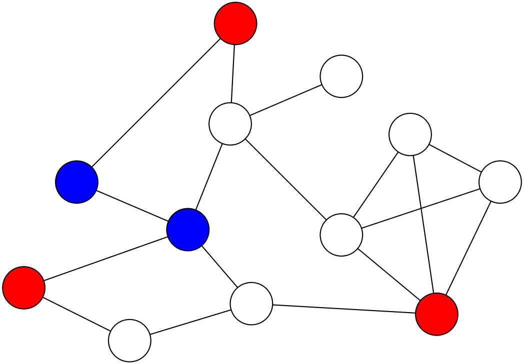
Selfish-mine strategy



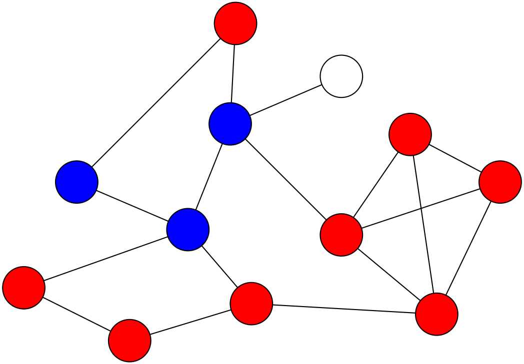
Selfish-mine strategy



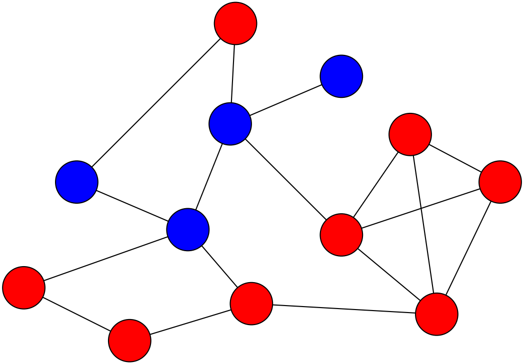
Selfish-mine strategy



Selfish-mine strategy

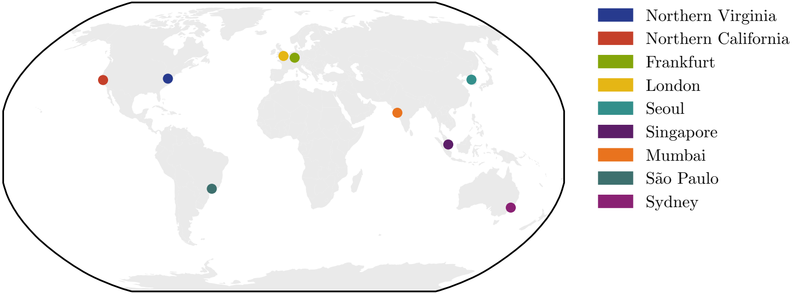


Selfish-mine strategy



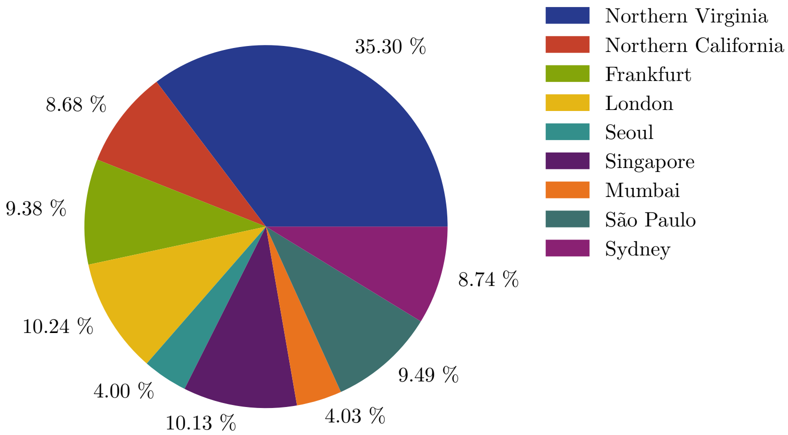
My project

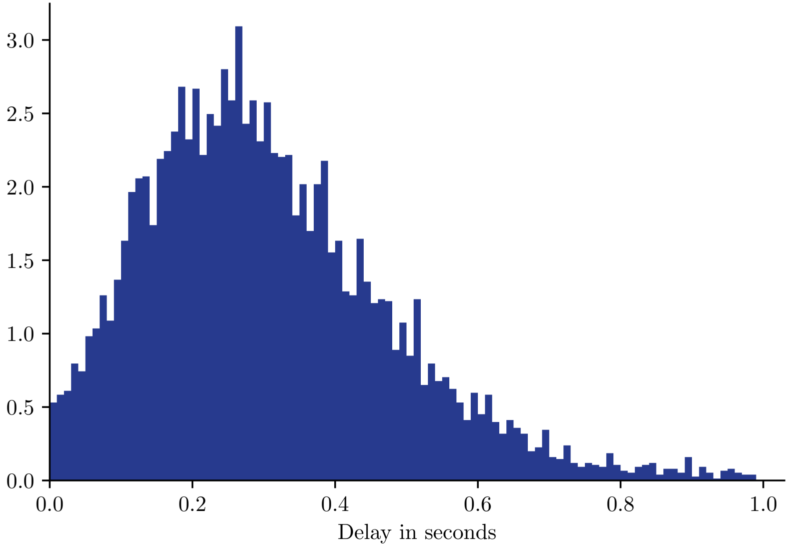
- Wrote code to observe Bitcoin network
- Set up a global observational experiment
- Observed parts of the propagation process of 14810 blocks
- Total 137 gigabytes data, 20.6 million messages



block	time	node_id	node_location	peer_ip	peer_port
000...3df98	1542887007.19418	4	London	62.152.58.16	9421
000...a7919	1543748261.60442	9	Sydney	52.198.169.28	8333
000...c9e85	1544975472.21318	1	Northern Virginia	2a01:4f8:191:4174::2	8333
000...19fca	1547242755.84717	1	Northern Virginia	47.88.192.215	8333
000...dd0e6	1547487249.97776	4	London	108.56.233.194	8333
000...3b14d	1547900706.97837	6	Singapore	117.52.98.78	8333
000...eeae7	1549261393.32542	6	Singapore	81.209.69.107	8333
000...713e3	1549319396.6256	1	Northern Virginia	144.76.5.41	8433
000...f4059	1550359795.0164	1	Northern Virginia	54.64.245.84	8333
000...f5404	1551099161.4188	1	Northern Virginia	185.186.209.210	8333

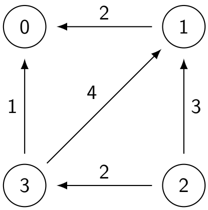
block	size	weight	height	time	nonce	bits	nTx	pool
000...048dd	1191087	3993369	550706	1542627049	1206282193	172a4e2f	1900	Slush
000...373a9	154481	534173	553369	1544510774	3681510874	1731d97c	305	Unknown
000...dd7e9	649274	2359871	555601	1545843940	4123099930	17371ef4	648	Unknown
000...84206	1283944	3993127	559626	1548178688	2688550637	172fd633	3133	AntPool
000...735ea	1262291	3993185	559645	1548188137	2421018921	172fd633	2524	BTC.com
000...10bc7	559802	1993133	562220	1549670573	2587934320	17306835	476	AntPool
000...574e1	697494	2340000	562268	1549698426	1796028615	17306835	1700	Poolin
000...16419	1288903	3993037	562793	1550026180	1918332176	172e6f88	2873	BTC.top
000...f3348	1134403	3993103	564931	1551302350	2869178554	172e5b50	2040	BTC.com
000...01cf8	1317204	3992916	565279	1551493180	1664069908	172e5b50	3273	Slush


35.30 %
8.68 %
9.38 %
10.24 %
4.00 %
10.13 %
4.03 %
9.49 %
8.74 %
Northern Virginia
Northern California
Frankfurt
London
Seoul
Singapore
Mumbai
São Paulo
Sydney


3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Delay in seconds

Phase-type distributions

- Markov chain $\{X_t\}_{t\geq 0}$ on finite state space $\mathcal{S} = \{0, \ldots, p\}$
- Phase-type generator $\boldsymbol{T}$, initial distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_p)$
- $\tau := \inf\{t \geq 0 \,|\, X_t = 0\}$ is phase-type, $\tau \sim \mathrm{PH}(\boldsymbol{\pi}, \boldsymbol{T})$



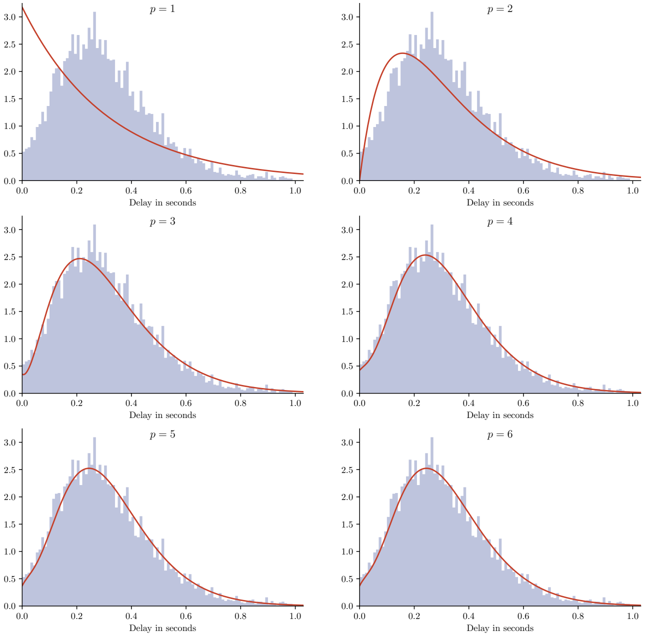
$$\boldsymbol{T} = \begin{pmatrix} -2 & 0 & 0 \\ 3 & -5 & 2 \\ 4 & 0 & -5 \end{pmatrix}, \qquad \boldsymbol{\pi} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Why phase-type distributions?

- Phase-type distributions are dense in set of all non-negative distributions (under weak convergence)
- So given any non-negative distribution, we can approximate it to arbitrary precision using phase-type distributions

EM-algorithm

- State of the art technique for fitting phase-type distributions
- Formulates the problem as an incomplete data problem
- Finding transition probabilities and initial distribution is very easy if we observed the whole Markov chain
- Compute MLEs using the Expectation-Maximisation algorithm
- Improved Julia code by Patrick Laub for performing fitting phase-types to data with this algorithm
- Use parallelisation to speed up fitting time for extremely large datasets



$p = 1$
$p = 2$
$p = 3$
$p = 4$
$p = 5$
$p = 6$
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0
Delay in seconds

Recap

- Wanted to study the Bitcoin propagation process
- Created a data collection tool to observe a most of the Bitcoin network
- Collected a large amount of data on arrival times of blocks
- Improved EM-algorithm for fitting phase-type distributions
- Fitted a phase-type distribution to some arrival difference data

Website

- All the data is open online!
- It's cleaned and processed and just over 2 GB.
- Also included: thesis, code for fitting and data collection, this presentation
- `https://bitcoin.aapelivuorinen.com/`