Proof of Work, “mining,” and P2Pool

An overview of Bitcoin’s consensus model

Foreword: Distributed Consensus is strictly speaking impossible

Impossibility of Distributed Consensus with One Faulty Process

5. *Conclusion*

We have shown that a natural and important problem of fault-tolerant cooperative computing cannot be solved in a totally asynchronous model of computation. These results do not show that such problems cannot be "solved" in practice; rather, they point up the need for more refined models of distributed computing that better reflect realistic assumptions about processor and communication timings, and for less stringent requirements on the solution to such problems. (For example, termination might be required only with probability 1.) Subsequent to the original announcement of these results [12], progress has been made along both of these lines [1–4, 9, 10, 20, 25].

Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. J. ACM 32, 2 (April 1985), 374-382. https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf

Building Blocks

Prerequirements to understanding mining

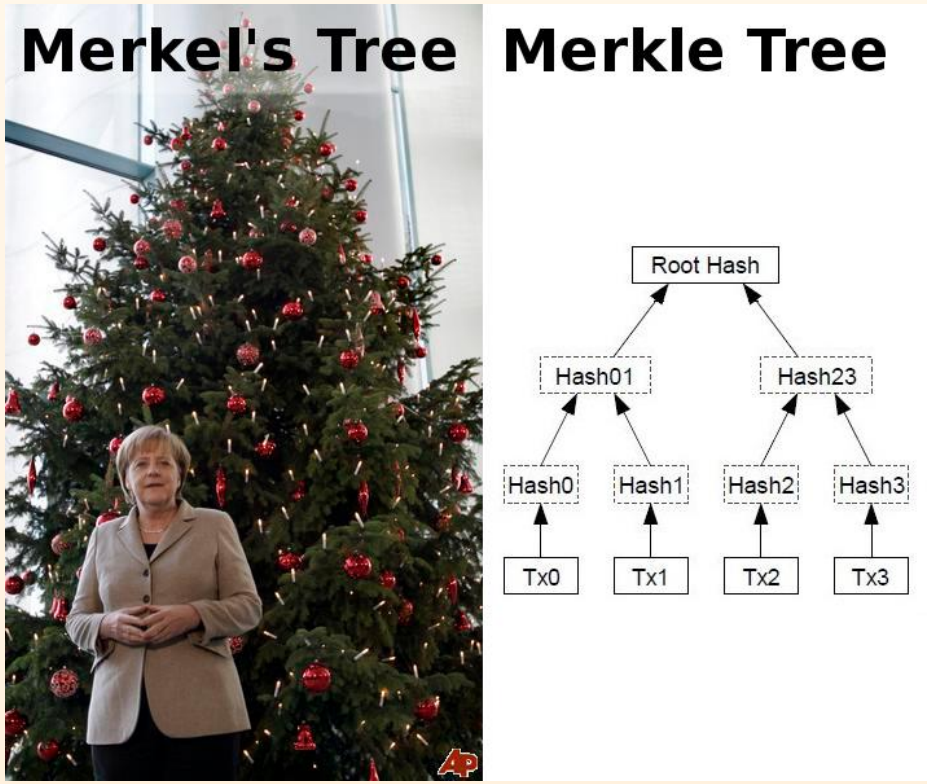
Hash functions

$f : \text{  } \rightarrow \text{  }$

(not this)

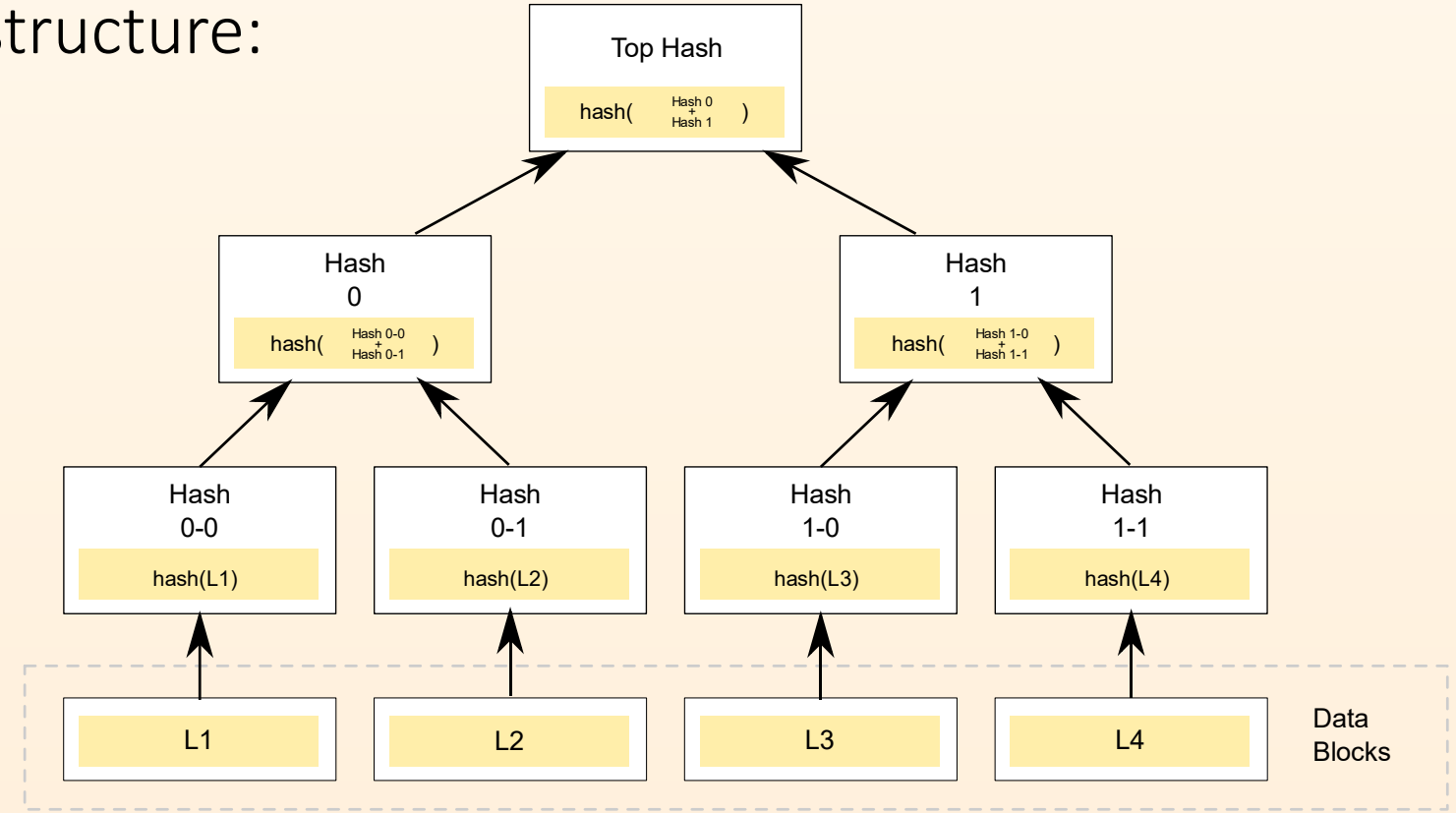
- Hash functions map bit strings of arbitrary length to fixed length bit strings.
  - $\mathrm{hash}(\mathrm{preimage}) = \mathrm{digest}$
- *Cryptographic* hash functions are:-
  - Preimage resistant: Given the digest it's infeasible to compute the preimage;-
  - Collision resistant: It's infeasible to find two preimages which map to the same digest.
  - Their outputs are uniformly distributed and uncorrelated to their inputs.

Merkle trees



(not to be confused with Merkel's tree)

- Problem: How do we compute a hash of list of items?
- Naïve approach: Concatenate items, compute hash.
  - Not flexible: Need all items to verify the inclusion of one.
- Instead, compute the hashes of each item and combine them in a tree-like structure:



Basic data structures used in Bitcoin

(sorry, there's no common misconception about this one)

Transactions

- Version
- Markers (segwit)
- List of inputs
  - Previous output
  - Input script
  - Sequence number
- List of outputs
  - Value
  - Output script
- List of witnesses (segwit)
- Lock time

- The first transaction in a block is a special one:
  - Called the ‘generation’ transaction.
  - Has only one input, the ‘coinbase’.
  - Previous output = 0
  - Input script was originally ignored, but now has certain restrictions.

Block headers

- Version
- Previous block hash
- Transaction Merkle tree root
- Timestamp
- Difficulty
- Nonce

Blocks

- Block header
- List of transactions

Proof of Work

How to approximate consensus with expensive computations

Requirements for PoW

- Easy to verify solutions
- Hard to find solutions
- Difficulty of finding solutions can be precisely quantified
- **Probability** of finding solutions proportional only to computational power
- Provably inseparable from the block it secures
- Depending only on the current state of the blockchain

Would a useful application like protein folding be a good option?

Hashcash based PoW

- Invented by Adam Back in 1997 — www.hashcash.org
- A valid proof satisfies $\mathrm{hash}(\mathrm{preimage}) < \mathrm{target}$
- We can set $\mathrm{preimage} = \mathrm{data} \parallel \mathrm{nonce}$ to commit data to the digest while still being able to find a nonce satisfying the validity condition.

- Bitcoin uses SHA-256 applied twice as its block header hashing algorithm, referred to as HASH256 in the source code.
- The *valid* chain with the highest *cumulative work* is chosen as the correct.

Attacks against PoW

1. Attacker tries to create an alternative chain with higher PoW:
   - Has 100% success rate if the attacker controls more than 50% of the network's hash power (known as the 51% attack), but with a less powerful attacker the success rate might still be high enough to be financially sensible.
   - An attacker does not need to own mining hardware: hash power can be controlled by a mining pool (GHash.io) or rented (Nicehash, www.crypto51.app).
2. "Selfish Mining": An attacker creates a short private alternative chain. She can take advantage of others experiencing propagation delays and carefully choose whether to mine on top of her own chain (and hence not experience propagation delays and increase her apparent hashrate), or switch to the public chain. Can be profitable with just a third of the network's hashrate. https://bitcoinmagazine.com/articles/selfish-mining-a-25-attack-against-the-bitcoin-network-1383578440/

How long should I wait?

- The probability an attacker succeeds is strictly positive. (Consistent with the impossibility of Distributed Consensus)
- However, it decreases exponentially with the number of confirmations.
- A common suggestion is to wait for 6 confirmations (expected 1 hour). Is that enough? https://www.desmos.com/calculator/fxcydh6vgw

Analysis of hashrate-based double-spending, Meni Rosenfeld, https://arxiv.org/pdf/1402.2009.pdf

Block times questions

Assume a hashrate and difficulty corresponding to 1 block per 10 minutes. If I uniformly randomly pick a point in time, what is the expected time between the previous block and the next block?

- 7 minutes
- 10 minutes
- 15 minutes
- 20 minutes

20 minutes

Assuming the Bitcoin hashrate is perfectly constant, and all blocks have exact timestamps (corresponding to the time they were mined). Which of the options below is closest to the expected time retargetting periods will take?

- 2 weeks minus 10 minutes
- 2 weeks
- 2 weeks plus 10 minutes
- 2 weeks plus 20 minutes

2 weeks, 20 minutes, 1.19 seconds;
i.e. 2016/2014 × 2 weeks

Credit to Russell O'Connor and Pieter Wuille.

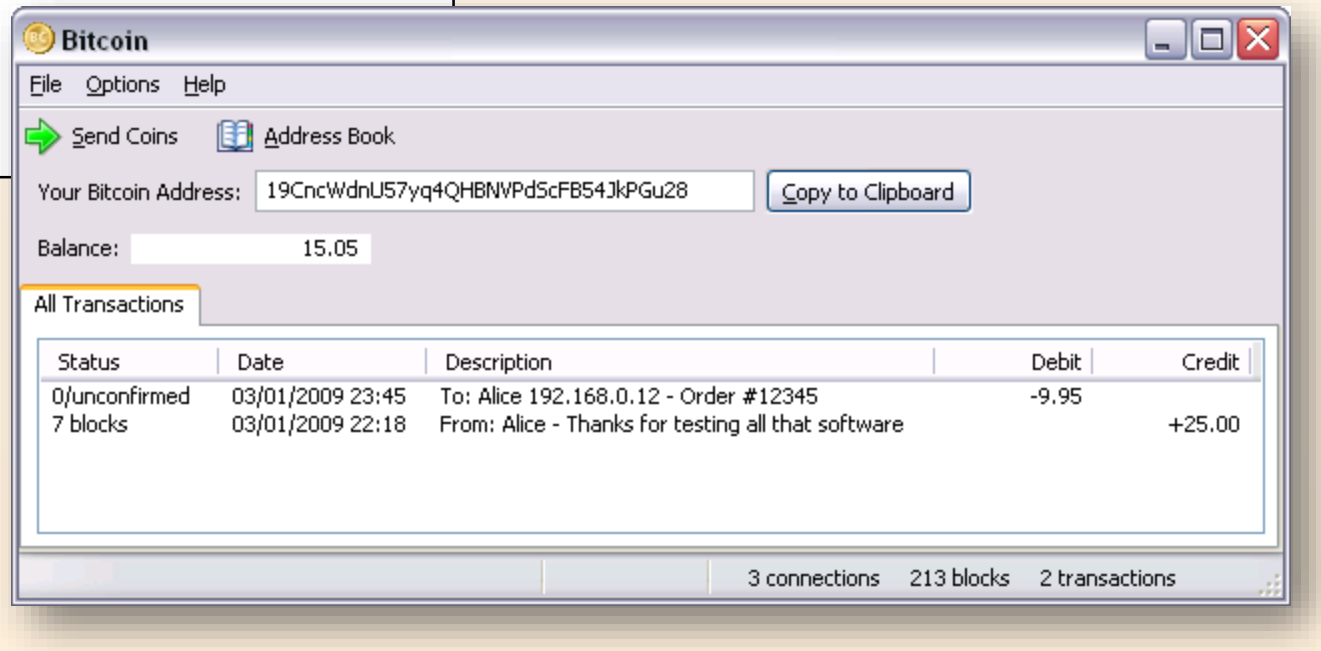
Evolution of Mining

A decade of finding preimages

Humble beginnings

You can get coins by getting someone to send you some, or **turn on Options->Generate Coins to run a node and generate blocks**. I made the proof-of-work difficulty ridiculously easy to start with, so for a little while in the beginning a typical PC will be able to generate coins in just a few hours. It'll get a lot harder when competition makes the automatic adjustment drive up the difficulty. Generated coins must wait 120 blocks to mature before they can be spent.

www.bitcoin.org, 2009



The Satoshi client gets modularity

- The Satoshi client got a JSON-RPC API over HTTP. One of the commands available was **getwork**, which allowed external mining software to interact with the client.
- Getwork only sends a block header and the target the server is willing to accept. Hence the miner can only modify the nonce and needs to poll the server constantly for new work (when a new block arrives).

The first mining pool

- With the advent of GPU miners it became more difficult for people to mine solo.
- In 2010, user Slush created the first mining pool using getwork at mining.bitcoin.cz.
- In December 2010 they mined their first two blocks with a hashrate of about 1.4 GH/s.

Vanilla getwork becomes too little

- The requirement to constantly poll the server for new work and the lack of flexibility in what can be modified by a miner made the development of extensions necessary.
- Long-polling allowed the server to respond to a request only when new work becomes available.
- Time rolling allowed miners to update the block timestamp.

Stratum Mining Protocol

- With the advent of ASIC miners even getwork with extensions became insufficient.
- Stratum sends miners not only the block header but also the generation transaction (and some of the internal nodes of the Merkle tree). Hence miners can modify also a portion of the generation transaction known as the extranonce.
- Stratum also allows the server itself to send notifications with new work to miners.
- Uses the plain, line based JSON-RPC protocol.
- Also invented by Slush.

An alternative to Stratum

- At the same time Stratum was developed the Satoshi client got a new mining related RPC command.
- **GetBlockTemplate** sends a full potential block to clients and allows a lot of flexibility in what can be modified.
- Its advantage is that it potentially allows miners to choose which transactions they want to mine, making pooled mining more decentralized.
- However, its verbosity made it impractical for mining hardware which only performs hashing.
- Nonetheless it is commonly used by pool servers to communicate with the Bitcoin client.

Multipools/ hashrate rentals

- Multipools direct their hashpower towards the most profitable coin at any given time. It requires no configuration from hashrate providers, who just expect the pool operator to manage their computational power.
- Hashrate providers can also direct their miners at a service which will sell the hashpower to anyone. In that case miners don’t need to care about which coin they will be mining as they are paid according to the demand for hashpower.
- In both cases hashrate providers can specify in which currency they are paid, possibly not a cryptocurrency at all.

The future

- Mining hardware has come close to the limits of current technology so we will likely not see the same huge increases in hashrate as before (for now).
- This might lead to more affordable hardware as the first mover advantage of manufacturers diminishes.
- BetterHash is a protocol which similarly to GBT intends to give miners more options in terms of modifying blocks. However it is a binary protocol so it’s more efficient than GBT.

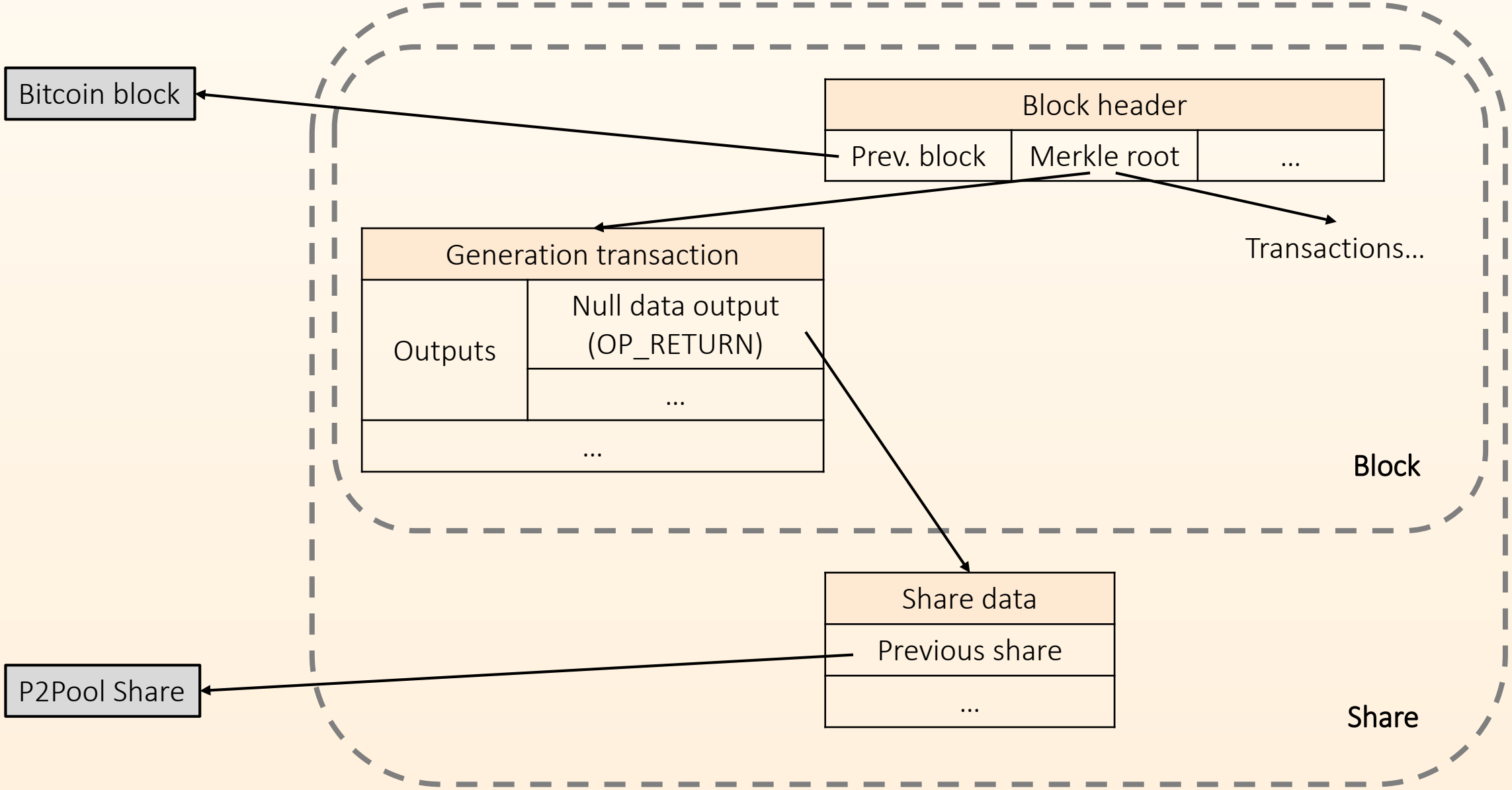
Issues with mining in Bitcoin (and other cryptocurrencies)

- Not sufficiently decentralized. While pool owners themselves don't generally own a lot of mining hardware, it isn't certain that miners would abandon a malicious mining pool (see GHash.io).
- Most "miners" aren't actually miners. Mining originally referred to attesting that a block chain is valid, but with most hashpower providers not doing any verification and not participating in transaction selection, the term "hasher" started being used.
- The energy usage problem: wildly exaggerated by mainstream media. Most mining uses renewable energy sources (mainly hydroelectric power) for economic reasons.

P2Pool

A decentralized mining protocol

- P2Pool (peer-to-peer mining pool; since 2011) creates a blockchain (called the sharechain) parallel to Bitcoin's blockchain.
- Shares in the sharechain are potentially valid Bitcoin blocks, but their hash is higher than Bitcoin's target.
- Shares reference the Bitcoin block they extend, but also the previous share. Hence P2Pool nodes keep track of the work done by each participant.
- When a share matches the Bitcoin difficulty, it is published to the Bitcoin network and the participants are paid.
- P2Pool nodes build only upon shares which conform to P2Pool's rules. Particularly, they ensure that the generation transaction pays each miner fairly according to the sharechain.


Bitcoin block
Block header
Prev. block
Merkle root
…
Transactions…
Generation transaction
Outputs
Null data output
(OP_RETURN)
…
…
Block
Share data
Previous share
…
P2Pool Share
Share

- P2pool runs a local Stratum server for mining hardware to connect to, and has a web interface with statistics.

- Some users may choose to not run their own P2Pool node and connect to a trusted public node.
- This has become a popular way to use P2Pool, and while not perfect, it is an improvement since setting up a new public P2Pool node is much simpler than an independent mining pool.

Successes

- It is a good option for smaller cryptocurrencies since modifying it to support a new coin is relatively simple.

- P2Pool was one of the first pools where SegWit was implemented.
- This has made it quite popular with altcoins which had adopted SegWit. Vertcoin, for example, set up two P2Pool networks to allow smaller miners to remain competitive.

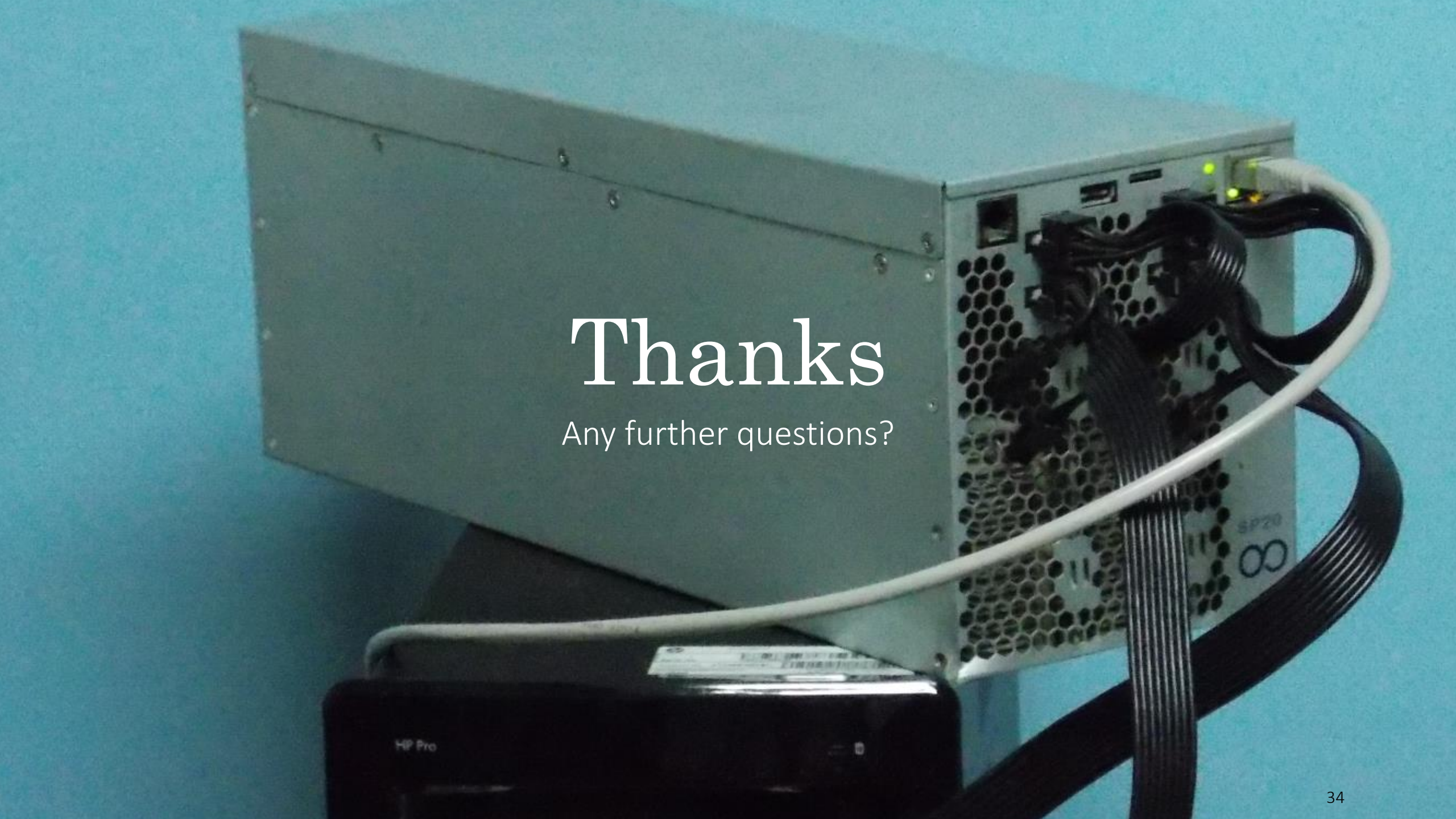
Drawbacks

- Centralized pools are inherently more efficient as they don’t have overhead for P2P communication.
- The requirement for propagating new shares frequently (one every 30 seconds on average) makes it hard to scale.
- Every participant who expects a payout has a separate output in the generation transaction.
- It’s written in Python, further limiting its efficiency.
- Development has stagnated, as its author is no longer involved in the cryptocurrency space, and the code quality is not great.

Future (hopefully) improvements

- Rewrite P2Pool in an efficient language, in a modular way.
- Incorporate advances in propagation techniques and possibly rework the consensus protocol.

- Once it becomes a viable alternative, encourage its use.



Thanks

Any further questions?