# Bitcoin with Just-In-Time Block Propagation

Department of Mathematical Sciences, Stellenbosch University, 7600 Stellenbosch, South Africa

Honours Project 2021

Bitcoin miners receive copies of all transactions as they are generated. Each miner gathers several validated transactions into a candidate block. Miners compete to append their candidate blocks to their blockchain. The process of appending a block to the blockchain requires the miners to solve a computationally difficult cryptographic problem.

If a miner computes a valid solution then the newly mined block is appended to the blockchain at the miner. The block is copied to every miner in the Bitcoin network and the block is appended to the blockchains at these miners.

## Blockchain splits

Each miner maintains its own blockchain. The blockchains are locally updated by the miners in such a way that the blockchains at each miner are either identical, or if they differ then the differences will soon be resolved and the blockchains will become identical.

Mined blocks are subject to communication delays as they are broadcast through the Bitcoin network. If two or more miners compute a valid solution more or less simultaneously, then the blockchain splits. The split will be resolved when the next block is mined.

Split avoidance schemes already exist in Bitcoin to rapidly communicate the contents of a mined block among the miners. Instead of broadcasting blocks of transactions, *thin* blocks are broadcast. Thin blocks contain hashes of transactions rather than transactions. The idea is that most of the miners have already received most of the transactions in the currently mined block. Sending them again in a broadcast block is unnecessary.

## The just-in-time protocol

There is a simple way to avoid blockchain splits, namely to communicate the transaction contents of a block *before* the block is mined. When a block is mined, the miner broadcasts only the block header. The header is small and will rapidly be communicated to all the miners. When the header arrives, it will most likely find that the corresponding body, which was broadcast at the end of the *previous* mining interval (which occurred on average 10 minutes ago), will have already arrived at the destination miner.

Let $\mathbf{B_i} = (H_i, B_i)$ denote block $\mathbf{B}_i$ at depth $i > 0$ in the blockchain with header $H_i$ and body $B_i$. The header $H_i$ contains metadata including a timestamp, the nonce, a hash pointer to $\mathbf{B}_{i-1}$, the Merkle root, and the public key of the miner. The latter is an addition to the standard header.

**Initialization.** An empty body $B_1$ is stored at every miner.

**Just-in-time protocol.** Consider block mining interval $i > 0$ Assume that miner $N$ computes a valid solution for block $\mathbf{B}_i = (H_i, B_i)$. Miner $N$

- appends block $\mathbf{B}_i$ to its blockchain,
- broadcasts the header $H_i$ to the other miners,
- forms the body $B_{i+1}$ from transactions that arrived at miner $N$ during the current mining interval,
- broadcasts the body $B_{i+1}$ to the other miners.

When the body $B_{i+1}$ arrives at miner $L$

- if the predecessor block $\mathbf{B}_i$ is missing, then $B_{i+1}$ is stored in the orphan pool at miner $L$ and $\mathbf{B}_i$ is fetched (duplicate blocks are deleted).
- miner $L$ inserts its coinbase into the body $B_{i+1}$,
- miner $L$ forms the header $H_{i+1}$ and starts mining $\mathbf{B}_{i+1}$.

When the header $H_i$ broadcast by miner $N$ arrives at miner $M$

- if the body $B_i$ is missing, then $B_i$ is fetched (duplicate bodies are deleted),
- the coinbase in the body $B_i$ is updated so that the block reward and fees are recorded as having being credited to the miner $N$ of the block $\mathbf{B}_i$,
- the Merkle root stored in the header $H_i$ must be the same as the Merkle root computed over the updated body $B_i$,
- the block $\mathbf{B}_i$ is validated and appended to the blockchain at miner $M$.

With reference to the rules above, during the first mining interval the empty block $\mathbf{B}_1$ is mined and the header $H_1$ is copied to the other miners. The body $B_2$ is formed containing transactions that arrived during the first mining interval. The body $B_2$ is copied to all the miners.

During the second mining interval the header $H_1$ arrives at all the miners. These miners already have a copy of the empty body $B_1$. At each miner the empty block $\mathbf{B}_1$ is formed and appended to the blockchain. The block $\mathbf{B}_2$ is mined and the header $H_2$ is copied to the other miners. The body $B_3$ is formed containing transactions that arrived during the second mining interval. The body $B_3$ is copied to all the miners, and so on for the following mining intervals

In general, consider block $\mathbf{B}_i$ which was mined by miner $N$ during the $i^{\text{th}}$ interval where $i > 0$. When block $\mathbf{B}_i$ was mined, the header $H_i$ was broadcast. The header $H_i$ is short and will quickly propagate to all the miners. When the header arrives at miner $L$ it will most likely find that the body $B_i$, which was broadcast at the end of interval $i - 1$, has already arrived at miner $L$.

## The project

We will extend our blockchain simulator to model the just-in-time protocol. We will investigate the relationship between the block mining interval and the size of the block body. If the body is too large, the body may not arrive before the corresponding header arrives. We will investigate the percentage of the time that the blockchains are in consensus.

## References

[1] K Ayinala, B-Y Choi and S Song. Accelerating Block Propagation in PoW Blockchain Networks with Pipelining and Chunking (PiChu), 2020 IEEE International Conference on Blockchain.

Funded by EU Erasmus: Living Laboratory in Climate Change (Grant Reference Number 618209).