Lecture 7: Liveness of Bitcoin

Principles of Blockchains, University of Illinois,
Professor: Pramod Viswanath
Scribe: Xuechao Wang

February 16, 2021

Abstract

In the previous lecture we have considered *safety* of a block: the event that once a block is deemed to be confirmed (because it is buried deep enough in the longest chain), then the probability it will be deconfirmed is small. While safety is a very important security property of a blockchain protocol, an equally important is *liveness*: this is the event that (honest) transactions get included into blocks, and further that the blocks feature in the longest chain. Liveness ensures that all transactions make their way into the ledger and safety ensures that eventually the transactions stay permanently in the ledger (with high probability). Together liveness and safety ensure the *security* of the blockchain protocol. The focus of this lecture is the liveness of the longest chain protocol. We will see that the longest chain protocol is live exactly under the conditions that the protocol is safe (derived in the previous lecture). We quantify the "level" of liveness and make connections to how fairly the incentives are distributed among the honest and adversarial miners. As part of this study, we identify worst-case adversarial strategies that minimize liveness and fairness (in incentive distribution). Finally we study a simple, but powerful, modification to the longest chain protocol that yields optimal liveness and fairness properties.

Chain Quality

We begin with two properties of a longest chain in the blocktree that will help us quantify the level of liveness of the protocol.

Definition 1 (Chain Growth)**.** *For a blockchain protocol, we define the chain growth of a longest chain $\mathcal{C}$ as the average growth rate of $\mathcal{C}$ (number of blocks per unit time), denoted as* $\mathrm{CG}(\mathcal{C})$.

Definition 2 (Chain Quality)**.** *For a blockchain protocol, we define the chain quality of a longest chain $\mathcal{C}$ as the fraction of honest blocks in $\mathcal{C}$, denoted as* $\mathrm{CQ}(\mathcal{C})$.

Positive chain growth ensures that the chain keeps growing and new blocks are continuously added to the longest chain. Due to the random nature of the mining operation, an honest miner will have a non-zero chance to succeed in mining a block. Thus CG is positive as long as the honest mining power $(1-\beta) > 0$. However, this is not enough to guarantee liveness: this is because a block mined by an adversary may fail to include honest transactions (or even be empty) and the entire longest chain may be made up of such adversarial blocks. However, positive chain quality ensures that a positive fraction of blocks mined by honest nodes enter the longest chain. Thus positive chain growth and positive chain quality together combine to ensure that any honest transaction will eventually be added to a block on the longest chain and to the ledger, which gives us liveness.

Now that liveness is guaranteed, one can study a more fine grained property of *fast time scale liveness*: having enough blocks mined by honest nodes on the longest chain (i.e., high CQ) ensures

that transactions enter the blockchain at a fast clip – thus CQ helps quantify the level of liveness. Furthermore, CQ quantifies *fairness*: since block rewards are provided to blocks in the longest chain, having blocks mined by honest nodes in the longest chain ensures that honest miners are rewarded. Next we will derive lower-bounds on chain growth and chain quality as a function of adversarial fraction of hash power β, total mining rate λ and maximum network delay Δ.

We first derive the lower bound on the chain growth. As we have seen in Lecture 6, under the limit of a very large number of honest miners (each with infinitesimally small mining power), the growth rate of a pure honest chain is $\frac{(1-\beta)\lambda}{1+(1-\beta)\lambda\Delta}$ when the network delays for all messages are exactly Δ. One can see that the adversary really cannot do anything better than this. Delivering an honest block earlier than Δ time or publishing any adversarial block will just increase the chain growth. Therefore, we have the bound:

$$\text{CG} \geq \frac{(1-\beta)\lambda}{1+(1-\beta)\lambda\Delta}. \tag{1}$$

Next we derive a lower-bound on the chain quality. If the chain growth is CG and the adversary can mine blocks at a rate of at most $\beta\lambda$, by the definition of the chain quality, we have a lower bound:

$$\text{CQ} \geq \frac{\text{CG} - \beta\lambda}{\text{CG}}, \tag{2}$$

where the equality is achieved when every adversarial block enters the longest chain. Plugging (1) into (2), we have

$$\text{CQ} \geq \frac{\frac{(1-\beta)\lambda}{1+(1-\beta)\lambda\Delta} - \beta\lambda}{\frac{(1-\beta)\lambda}{1+(1-\beta)\lambda\Delta}} = \frac{1-2\beta-\beta(1-\beta)\lambda\Delta}{1-\beta}. \tag{3}$$

One can check that $\text{CQ} > 0$ if $\frac{(1-\beta)}{1+(1-\beta)\lambda\Delta} > \beta$, which is exactly the same threshold as the safety guarantee as we have seen in Lecture 6. Therefore, we can conclude the following security guarantee on the longest chain protocol.

Theorem 1. *The longest chain protocol is safe and live, exactly under the following condition:* $\frac{(1-\beta)}{1+(1-\beta)\lambda\Delta} > \beta$.

In Lecture 6, we have seen that the private attack achieves the security threshold. Now a natural question would be: is there a specific adversarial strategy that achieves the minimum chain quality? A "selfish mining" attack described in this paper, answers this question and is discussed next. Before that, one can see that to achieve the minimum chain quality, the attack has to guarantee two properties:

- every adversarial block enters the longest chain; so every adversarial block is useful.
- every adversarial block displaces an honest block in the longest chain; so every adversarial block is doubly useful.

Selfish Mining

Consider the following adversarial strategy (see Figure 1):

- The adversary always mines on the block at the tip of the longest chain, whether the chain is private or public. Upon successful mining, the adversary maintains the block in private to release it at an appropriate time (discussed next).
- When an honest miner publishes a block the adversary will release a previously mined block at the same level (if it has one). We assume that the adversary can break ties in its favor, so honest miners will mine on the adversarial block.

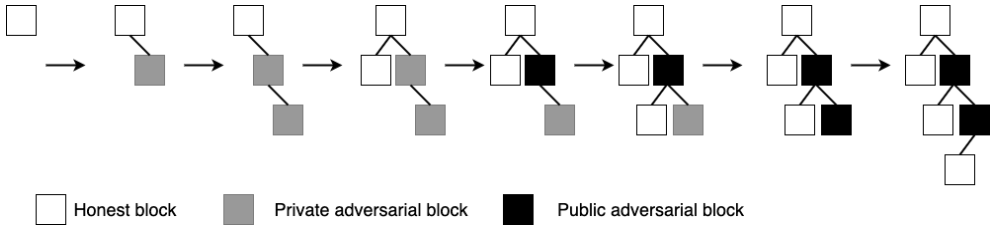

Figure 1: Selfish mining example.

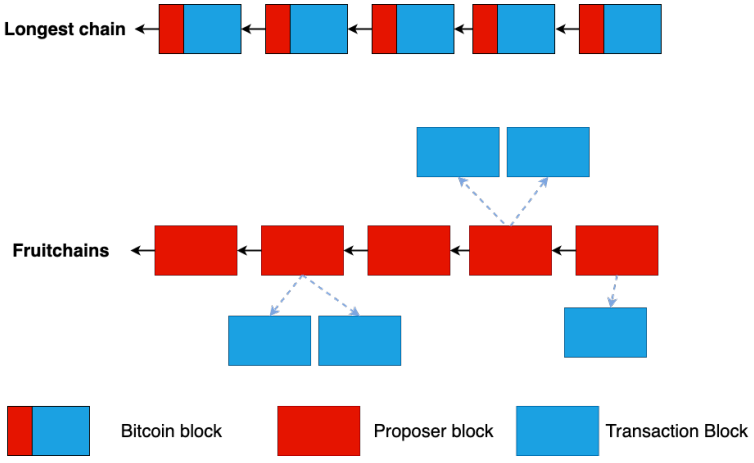

Figure 2: Comparison between blocks in the longest chain protocol and the transaction and proposer blocks in Fruitchains.

Optimal Chain Quality

If the adversary decided to follow protocol, then the chain quality of the longest chain protocol would be CQ $= 1 - \beta$; indeed one cannot expect any CQ better than this. A natural question would be: can we improve the design of the longest chain protocol to achieve this optimal chain quality under arbitrary adversarial strategy? Especially interesting would be if this design were only a slight modification of the longest chain protocol. Fruitchains is one such modification that achieves optimal CQ, and at fast time scales.

In the longest chain protocol, safety of a block (provided by how many blocks are mined underneath) and the incentives associated with a block (both the transaction rewards and block rewards, cf. Lecture 5) are *coupled*. The key idea of Fruitchains is to *separate* the two properties – incentives and safety – by embodying them in two separate block structures:

- a *transaction block*, which contains all the transactions (or `data`) of the blocks in the longest chain protocol;
- a *proposer block*, which contains the header of the blocks in the longest chain protocol.

It is important to cryptographically *couple* the two types of blocks and this is done via embedding the hash of each transaction block inside one of the proposer blocks (see Figure 2). The two types of blocks are further coupled during the mining process via "2 for 1 mining", also known as *cryptographic sortition*. This is described next.

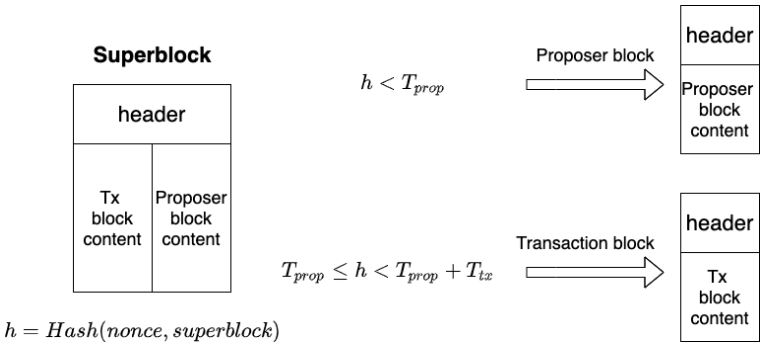

Figure 3: Superblock containing a transaction block and a proposer block is "two-for-one" mined; the resulting cryptographic sortition identifies either the transaction or proposer block as the winner of the mining process.

Cryptographic Sortition

In the mining process, the two types of blocks – transaction and proposer blocks – are mined *together*. A miner first forms a "superblock", essentially the same as a regular block in the longest chain protocol: the superblock contains two parts, a transaction block and a proposer block. However this is where the similarity ends. We say a superblock is successfully mined if

$$Hash(\mathsf{nonce}, \mathsf{superblock}) < T_{\text{tx}} + T_{\text{prop}}. \tag{4}$$

Further, every successfully mined superblock is identified as either a transaction block or a proposer block based on the hash output:

- identify the superblock as a proposer block if the hash output is less than T_{prop};
- identify the superblock as a transaction block if the hash output is in the range $[T_{\text{prop}}, T_{\text{tx}} + T_{\text{prop}}]$.

Due to the random output nature of hash functions, the probability a superblock is identified as a proposer (transaction) block is proportional to T_{prop} (T_{tx}), respectively. This process where two distinct types of blocks are mined jointly and only one of the blocks is identified as the output of the mining process is known as "two-for-one mining" or "cryptographic sortition" (see Figure 3).

Fruitchain Protocol

We can put together the cryptographic sortition process to summarize the Fruitchains protocol. Every miner creates a superblock containing a transaction block (consisting of transactions not in the ledger) and a proposer block (which has a hash pointer to the parent proposer block and hash pointers to *all* the transaction blocks that are not referred by its proposer ancestors). The proposer block mining is conducted using the longest chain rule and transaction blocks have no chaining or constraints. See Figure 2 for illustration. The success of the superblock mining process is decided by Equation (4) and the actual hash output decides whether to interpret the superblock as a transaction block or a proposer block. The ledger is an ordered list of transactions and is generated as follows: order the proposer blocks according to the longest chain, order the transaction blocks referred to by each proposer block (e.g., ordered based on the hash value of the transaction block) and further order the transactions within a transaction block (e.g., lexicographic ordering of the transactions in the Merkle tree representation). We argue the safety and liveness properties of the fruitchain protocol as follows.

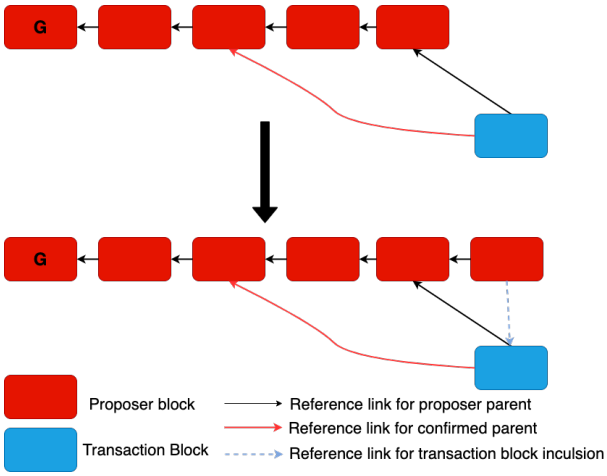

Figure 4: The Fruitchains protocol.

Safety. Let λ_{tx} and λ_{prop} be the mining rate of the transaction blocks and proposer blocks respectively; these mining rates are directly proportional to the difficulty targets T_{tx} and T_{prop}, respectively. The safety of this protocol follows directly from the safety of the longest chain protocol: as long as λ_{prop} is small enough, i.e.,

$$\frac{(1-\beta)}{1+(1-\beta)\lambda_{\text{prop}}\Delta} > \beta,$$

the k-deep proposer block in the longest chain will be permanent with high probability (for large enough k). This stabilizes the ledger guaranteeing safety.

Liveness. The main claim is that the optimal CQ of $1-\beta$ is achieved regardless of the adversarial strategy on transaction and proposer block mining. We first examine the selfish mining attack: even if an adversary tries to "erase" some proposer blocks mined by an honest party (which contains some honest transaction blocks) by selfish mining, by the liveness (positive CQ and CG) of the the longest chain protocol, eventually an honest party will mine a new proposer block including those transaction blocks that were displaced and with large enough k, this proposer block will be safely included in the ledger. We know that the mining of the honest and adversarial transaction blocks follows Poisson process with rate $(1-\beta)\lambda_{\text{tx}}$ and $\beta\lambda_{\text{tx}}$ respectively. And by the liveness of the the longest chain protocol, every honest transaction blocks will be eventually included into the proposer chain. Therefore, by considering an average on the entire longest proposer chain, we have

$$\text{CQ} \geq \frac{(1-\beta)\lambda_{\text{tx}}}{(1-\beta)\lambda_{\text{tx}}+\beta\lambda_{\text{tx}}} = 1-\beta.$$

Note that here we define chain quality as the fraction of *honest transaction blocks* instead of honest proposer blocks, and further determining mining rewards by transaction block yields fair rewards. It is interesting to note that the safety and optimal CQ properties of the Fruitchains protocol do not constrain the mining rate λ_{tx} of transaction blocks (determined by the difficulty target T_{tx}). However, having a low difficulty target T_{tx} will lead to a large rate of production of transaction blocks which will need to be distributed over the underlying network. In the next lecture we will see that the constraint on the transaction block mining rate imposed by a careful modeling of the network capabilities (capacity and delay).

Finetuning the Fruitchains protocol. The Fruitchains protocol described above guarantees long term optimal chain quality. However, in the context of a short timescale, the protocol is vulnerable to a private block attack: an attacker could keep successfully mined transaction blocks private, and suddenly release a large number of them at the same time, thereby creating a very high fraction of

adversarial transaction blocks in some small segment of the proposer chain. This issue is resolved by requiring that a transaction block should "hang" from a proposer block which is not too far from the proposer block which includes it. Each the transaction block has two parent blocks, a confirmed parent and a proposer parent: the confirmed parent is a recently stabilized/confirmed proposer block (i.e., k-deep block) that the transaction is hanging from; the proposer parent should be the tip of the longest proposer chain. Note that a proposer block also has a confirmed parent because the transaction block mining and proposer block mining are conjoined by the two-for-one mining process, although the proposer block generation and interpretation does not care about this field. See Figure 4 for an illustration. We say that a transaction B_{tx} is recent with respect to a proposer chain $\mathcal{C}$ if the confirmed parent of B_{tx} is a block that is at most R deep in $\mathcal{C}$, where R is a *recency* parameter. Our goal is achieved by requiring proposer blocks only include recent transaction blocks. By setting the recency parameter R reasonably large, any transaction block mined by an honest player will be included sufficiently deeply in the proposer chain. Therefore, the optimal CQ property continues to hold true. Moreover, since only recently mined transaction blocks can be included, the optimal CQ is achieved even for short segments of the longest chain.