Lecture 17: Bootstrapping Blockchains

Principles of Blockchains, University of Illinois,
Professor: Pramod Viswanath
Scribe: Suryanarayana Sankagiri

March 25, 2021

Abstract

In the lectures so far, we have studied the security and performance of blockchains in a *steady state*, i.e., when the participation is steady and engaged. In this lecture, we study the *transient* phase of blockchains: how to securely and efficiently *bootstrap* blockchains via careful incentives and design choices. We discuss several practical approaches to bootstrapping PoW and PoS blockchains. We also discuss popular methods to start new blockchains piggybacking off existing blockchains, especially Ethereum.

Introduction

Today, we take the security of well-established blockchains like Bitcoin and Ethereum for granted. Partly, this assurance is based off of the past track record of the blockchains; there has not been any fatal attack on them so far. Additionally, we have provable guarantees, based on mathematical models, that show that if an adversary wants to attack the system, they must control close to 50% of the mining power in the system. In Bitcoin, currently, the total mining power in the system is enough to compute more than 10^{20} hashes per second. To compare, a good GPU today could compute around 10^8 hashes per second. Thus, it seems *economically infeasible* for an adversary to own that much computing power. Put differently, Bitcoin's security comes from the extensive computing power already invested into the system.

In its initial days, Bitcoin did not have such a strong level of security, simply because there were a much smaller number of miners in the system. In those days, the Bitcoin hash rate was close to 10^6 hashes per second (the mining difficulty has changed by around fourteen orders of magnitude to keep the mining rate constant). Matching this hash rate would not have been hard for an adversary (even if they were limited to the computers of 2009). Even today, a new blockchain would presumably start off with a much smaller total computation power than what Bitcoin and Ethereum have today. How, then, do we preserve security of these blockchains? In this lecture, we will explore some methods to 'bootstrap' blockchains, first for Proof-of-Work systems and then for Proof-of-Stake systems. Finally, we will look at methods of piggybacking of existing blockchains.

Bootstrapping PoW Blockchains

We discuss a few different principles to bootstrap Proof of Work blockchains. These principles are complementary, and multiple of them can be applied simultaneously to the same system.

Incentives via block rewards

The first security measure we discuss is an incentive-based one, which argues that miners are likely to be rational; they would not attack the system if they would hurt themselves economically by doing

so. In an incentive-compatible blockchain system, it'd be extremely unlikely that a set of miners controlling a majority of the mining power would collude together to attack the system, since doing so would hurt them economically. It does not provide any security guarantees if indeed, for some reason, a 51% adversary does attack the system.

We know that, in steady-state operation, mining rewards in PoW blockchains provide an incentive to miners to behave honestly. In fact, these rewards can incentivize miners to act honestly in the initial phases of the blockchain too. One way to do so is to give early miners an extra incentive in terms of greater mining rewards, as was done in Bitcoin. Here, the mining reward began with 50 bitcoins per mined block. It is reduced by half every few years (210,000 blocks), and is currently at 6.25 bitcoins per block. Thus, early miners had an extra incentive to remain honest.

In terms of its dollar value at that time, the initial block rewards were not worth much. However, it was believed that the price of Bitcoin would increase once it gained greater acceptance; indeed, this belief has been validated. A miner with significant mining power would stand to gain a lot in the future if it helped maintain Bitcoin's security, and thereby, its reputation, by following the protocol. If, instead, it performed a double spend attack, it would get a one-time gain from the attack. As a penalty, it would lose the value of all its mining rewards, as the attack would drive the price of Bitcoin down to zero very fast. Thus, correct incentives helped maintain Bitcoin's security in the initial years.

Checkpointing

A second method of bootstrapping PoW blockchains is to employ a *checkpointing mechanism*. Broadly speaking, this method employs a trusted party (or a group of parties) that checkpoints blocks at regular intervals, and new blocks must be mined below the latest checkpoint to be considered valid. Users can use the checkpoints as a basis of confirmation: i.e., they confirm blocks up to the last checkpoint. The checkpointing mechanism is essentially a BFT (permissioned) consensus protocol, executed by a special set of checkpointers.

In the previous lecture, we saw that checkpointing (via a finality gadget) can help secure the longest-chain protocol under periods of asynchrony. Similarly, the checkpointing mechanism can also act as a guard-rail against an adversary with more than 50% mining power, as long as the checkpointing mechanism acts correctly. The important point to note here is that the checkpointers are not chosen via mining. Rather, they are a set of entities (corporations, foundations, etc.) with some credibility who are appointed just for the sake of securing a PoW blockchain in its initial stages. Thus, it is fair to assume that a majority (or super-majority) of them will be honest.

Let us examine the security of such a checkpointing based protocol under the presence of an adversary with majority mining power. The safety of the protocol is easy to argue; it follows from the safety of the checkpointing mechanism. However, liveness is harder to achieve. A simple implementation of a checkpointing system could lead to poor liveness guarantees (e.g., see this work). Figure 1 shows the variation of the chain quality, a metric of liveness, with epoch length and adversarial mining power.

Liveness is important to have in order to incentivize honest nodes to stay in the system. Indeed, if honest blocks do not get included in the ledger, honest miners will not be able to accrue any reward, and will therefore leave the system. In earlier checkpoint-based protocols, we were willing to forego liveness when the network was asynchronous. This is because periods of asynchrony are typically small and infrequent. Here, ensuring liveness is important because an adversary may retain a mining majority until a large number of users join the system, which could take time.

There are two major principles to have a liveness-guaranteeing checkpointing mechanism (under $> 50\%$ adversary). The first is that the checkpointers should generate a fresh random string with every new checkpoint, and this random string should be included in all descendants of the checkpoint block. Such a mechanism ensures that all blocks below a checkpoint block must be mined *after* the block has been marked as a checkpoint. The adversary cannot take undue advantage of its mining

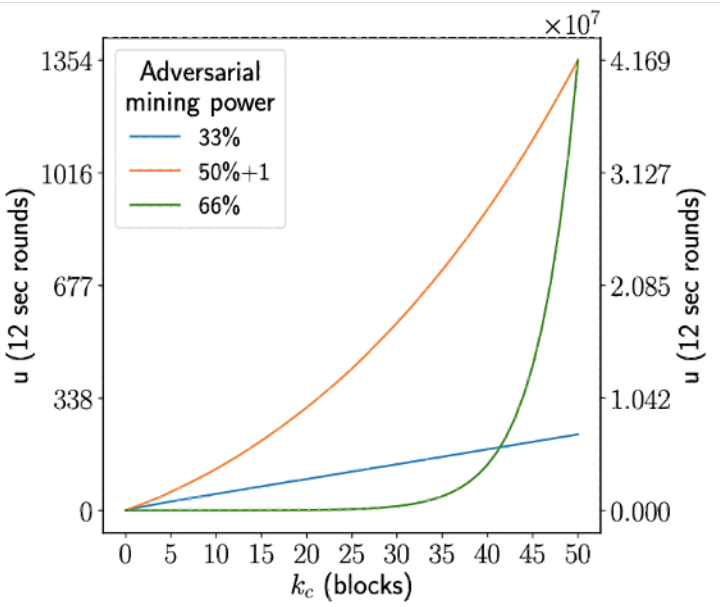

Figure 1: Chain Quality degrades sharply with an increase in the adversarial mining power, and with an increase in the epoch length (period between checkpoints)

power by mining many blocks beforehand; the counter of blocks is set to zero with every checkpoint block.

The second principle is for checkpoints to introduce reference links to blocks that are not on the main chain, which are likely to be honest blocks. The main idea is that honest blocks are produced in proportion to the honest mining power, but they may not be on the main chain due to the adversary ignoring them. These reference links point to honest blocks and bring them into the ledger, leading to a chain quality that is equal to the honest mining power. The idea of using reference links to improve chain quality was also used in FruitChains. An entire system, with both these principles, is illustrated in Figure 2.

Historically, in Bitcoin, Satoshi Nakamoto used to issue checkpoints at regular intervals. This practice was discontinued in 2014, presumably once they realized that Bitcoin has sufficient mining power to be secure on its own. In this case, it was a single trusted honest party that was issuing checkpoints. A more robust method would be to have a permissioned set of users doing so.

Hardfork

In the world of blockchains, a *hard fork* (or hardfork) is a point in the blockchain where the protocol undergoes a major change, and these changes are incompatible with the previous version. Blocks produced as per the new protocol will be deemed invalid as per the old protocol and vice-versa. A hard fork could be used to introduce new features into the blockchain, in order to improve its security or performance. Hard forks could also be used to correct security bugs in the existing codes. This happened with Ethereum in the case of the DAO vulnerability.

To understand hard forks better, consider the following example. Suppose the miners of Bitcoin decide that they would like to increase the mining rate of Bitcoin by a factor of ten, in order to increase its throughput. To increase the mining rate, they would have to lower the difficulty of the blocks. In essence, the block difficulty adjustment formula must be changed from the current version. If all nodes agree to undertake this change simultaneously, then the protocol will make a

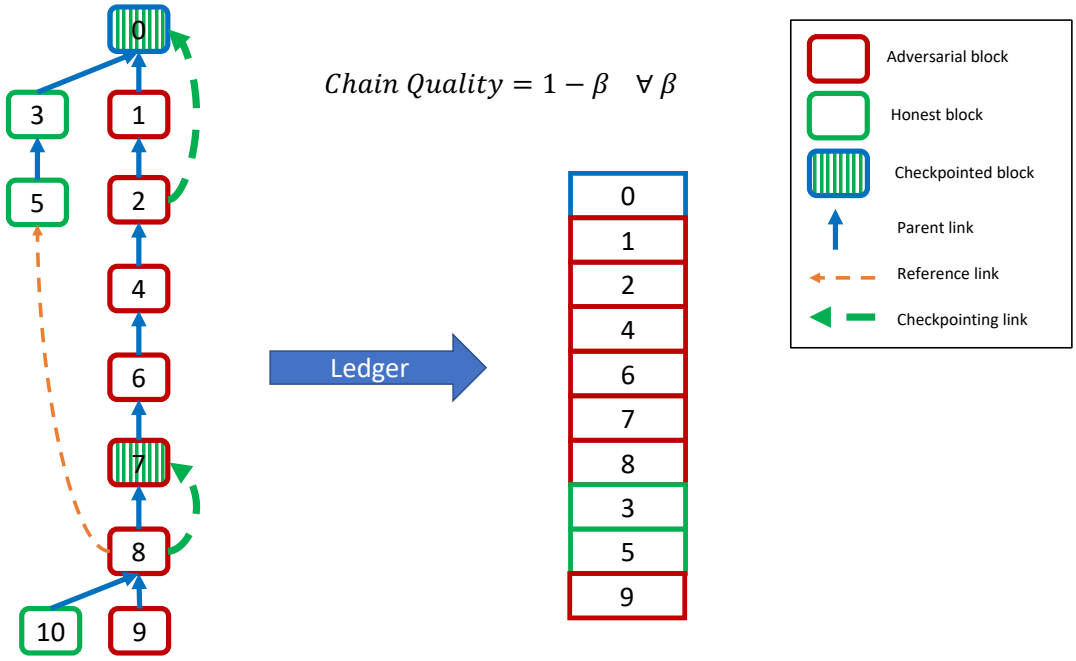

Figure 2: The Advocate system, which provides optimal chain quality even in the presence of an adversarial majority. Such a system provides a useful checkpointing mechanism to bootstrap PoW blockchains.

smooth transition to the new version. However, this is not possible in a decentralized system. Some nodes will be unaware of the new changes for a while and therefore they will ignore blocks produced by the new rule, treating them as invalid. Because of this, there will be a major fork in the system; this phenomenon gives rise to the term hard fork.

In some cases, all nodes eventually agree to the software update. In this case, the fork with the older format of blocks gets stalled, while the fork with the newer format continues to grow. However, there are instances when some nodes prefer the updates, while others do not. In this case, two forks continue forever. In essence, the fork with the updated protocol is a new cryptocurrency/blockchain. The (hard) fork marks the beginning of the new system. Some nodes migrate to the new system, while others remain in the old one. Note that this is not a security vulnerability; once a node clearly decides which version of the software it wants to follow, the blockchain continues to guarantee security. Many cryptocurrencies have been launched in this way, the most popular being Bitcoin Cash, which is a hard fork from Bitcoin. The history of hard forks from Bitcoin can be found here.

A cryptocurrency that is launched as a hard fork from an established blockchain retains the distribution of coins (or more generally, the state of the system) as recorded in the last block before the fork. If the new system is appealing, it very quickly gathers a large number of miners (i.e., a large computing power) since miners in the old system can seamlessly transit to the new system. This gives the system a strong security backing, as it starts in a fairly decentralized fashion. In addition, the new system inherits most of the security features from the existing blockchain; only the new features must be verified.

Soft Forks

Soft forks, in contrast to hard forks, are software updates that are backward compatible. In such an update, even those who have not updated their software will treat new blocks as valid. It is

possible that those following the new update treat the old blocks as invalid. In this case, as long as a majority of miners upgrade their software, the blockchain continues to function smoothly; blocks produced by the old software will quickly get stale. Compared to a hard fork, a soft fork allows users to slowly transit to the new system. However, for the soft fork to come into effect, a majority of the miners should be running the updated software. Just like hard forks, soft forks are also used to include better features to an existing system.

In Bitcoin, new transaction types (or new scripts) have been introduced as soft forks. A prominent example of this is the SegWit update, which stands for *segregated witness*. Recall that in Bitcoin, blocks have a maximum size of $1MB$. This places a limit on the number of transactions that can be included in a block. SegWit is a proposal that reduces the size of transactions in Bitcoin, which allows one to include more transactions per block while still obeying the $1MB$ limit. The basic idea of SegWit is to keep the signatures on transactions outside the main block. As such, all transactions in Bitcoin must be signed. However, the signature can take up to 65% of the space in a transaction. Separating the signature reduces the transaction size separately. The signatures are still present; they are just moved to an additional part of the block that does not count as part of the $1MB$ limit. The phrase "segregated witness" reflects this design principle (the term witness refers to the signatures here).

Proof-of-Burn

Proof-of-burn is a method by which users can migrate from an established blockchain system like Bitcoin to a new one. To burn coins, nodes send Bitcoins to a verifiably unspendable address (say, an address with a fixed public key for which it is virtually impossible to come up with a corresponding secret key). In exchange, they receive a reward in the native currency of the new blockchain. Users may also receive mining rights proportional to the amount of coins they have burned. In effect, this creates a new system where the mining power is proportional to the money one has burned. Compared to Proof-of-Work, this method is less resource intensive, and equally secure. It piggybacks off an established blockchain. You can read more about this idea here.

Bootstrapping PoS Blockchains

In any Proof of Stake system, block proposers are chosen based on the stake that nodes have recorded in the ledger. Thus, for a PoS system to get started, some amount of stake (coins) must be distributed amongst the initial participants. Moreover, this initial stake distribution *must be recorded in the genesis block* (unlike a PoW blockchain where the genesis block can have no record of coins and ownership, as in Bitcoin). Whatever be the initial distribution, a Proof-of-Stake system could suffer from a rich-get-richer phenomenon. Those with higher stake are more likely to mine blocks, and thereby accrue block/transaction rewards. This would then increase the chance of them mining future blocks.

If left unchecked, the compounding of wealth could lead to extensive centralization. One way to reduce the compounding of wealth effect is to start off with a large amount of initial stake distributed fairly uniformly across nodes. Here, the initial stake is large relative to the block/transaction rewards. A second way to reduce this effect is to re-design the mining rewards with time: if rewards increase with time, the stake distribution after some time is likely to be more uniform than the case where rewards remain constant (or decrease).

There are different ways to gather an initial set of stakeholders. These strategies could be used by any cryptocurrency, even a PoW based one; however, they are *essential* for a PoS based blockchain to get off the ground. One method could be a proof-of-burn strategy, where one must pay in some form to gain coins of a new system. A different method could be to just distribute a small amount of coins for free to as many different users as possible. Many companies offer rewards on their service upon

joining or to existing users for giving referrals. Others simply give out freebies. This marketing ploy, when adopted by cryptocurrencies, is called an *airdrop*. For example, one could gain some coins upon tweeting about the new currency. Apart from gaining an initial base of stakeholders, the currency also gains more visibility/popularity.

A popular term used during the launch of a new cryptocurrency is that of an ICO, which stands for initial coin offering. This is modeled (and named) after the concept of an IPO (initial public offering). Here, new tokens are offered for a fixed price in some fiat currency (dollars). ICOs are often used as an investment opportunity by users, but they carry the same risks as a company IPO; arguably, they are riskier.

Bootstrapping via Layer 2 Solutions

Today, many new tokens are launched on the Ethereum Virtual Machine Platform. This platform is extremely flexible, and supports a wide range of functionalities. Such tokens take the Ethereum ledger's security (based on Nakamoto consensus) for granted, and are built for very specific applications. Each new token is essentially a new cryptocurrency, which can be used for its own specific purpose (e.g., it can be used only on Walmart). By piggy-backing off of Ethereum, it derives many of the basic safety features for free.

A vast majority of tokens on Ethereum are "ERC-20" tokens. ERC-20 is a standard for tokens, which sets some basic rules, such as how tokens can be transferred, how new tokens can be generated, etc. Having a common set of rules ensures these tokens are compatible and can be exchanged with each other easily. Moreover, they are also compatible with the backbone Ethereum system (e.g., all gas fees in ERC-20 tokens are denominated in ETH, the native token of Ethereum).

Once we have a token on the Ethereum system, one can employ creative ways to get people to adopt the token, via smart contracts. A bonding curve contract is a special contract that adjusts the price of a token (in terms of ETH or Dollars) based on its demand. As the popularity of a contract soars, its price increases. Thus, early participants get better deals. Similarly, ICOs can also be automated via smart contracts.