Hathor Network[1]

A Novel, Scalable, and Decentralized Architecture for a General Distributed Ledger

Hathor is a transactional consensus platform comprised of an entirely novel architecture, based on concepts from both directed acyclic graph (DAG) and blockchain technologies combined. **We solve the problems of scalability and decentralization maintenance among distributed ledger networks** by including a chain of mined blocks *inside* a DAG of transactions. The blockchain ensures security when the number of transactions per second is small, whereas the DAG prevails when the number increases significantly.

A highly scalable distributed ledger where transactions can be processed for multiple purposes with no fees will be created through its own network. At the same time, mining rewards in the form of newly generated tokens will enforce a fair distribution of economic and computational resources through decentralization and 99.99% uptime without any form of central coordination. **Hathor is the direct result of 7+ years of academic research from its founding members,** throughout which they have been stress-testing its assumptions and alpha versions.

We see scalability as a significant challenge in cryptocurrencies. IOTA made great strides towards scalability, but its solution does not seem to work when the number of transactions per second is small. Hathor's architecture lies between the ones from Bitcoin and IOTA and presents a solution to scaling, centralization and spam issues.

Thanks to the academic research conducted by Dr. Marcelo Brogliato and published as part of his Ph.D. thesis, we have a detailed analysis of Hathor's architecture through simulations. **The primary result is that it works correctly under *any* number of transactions per second.** The higher the number of transactions per second, the faster new transactions are confirmed.

As in IOTA, new transactions confirm previous ones, forming a DAG. Each transaction has its own proof-of-work which is solved by the issuer before propagating the transactions in the network. As in the case of Bitcoin, miners find new blocks which form a blockchain inside the DAG. Blocks collect newly-generated tokens and confirm all the transactions in the DAG. Each transaction has an accumulated weight which expresses the necessary effort to break the transaction, similar to the number of confirmations in Bitcoin.

Hathor Foundation

In the coming months, we will open the Hathor Foundation as a way to raise funds to be used for research, partnerships, regulation, education, and development related to Hathor and its emerging ecosystem. Although there are a few details yet to be discussed, we plan to do a pre-sale of Hathor's genesis tokens, which would be created when the network is launched. **Our first round goal is to raise $5MM to open the foundation,** sign vesting contracts with developers and advisors and launch the network. Investors would also have votes in the foundation's board.

[1] `https://hathor.network/` — contact@hathor.network — Written on June 20, 2018

Hathor: An alternative towards a scalable cryptocurrency

Marcelo Salhab Brogliato

June 15, 2018

1 Introduction

We are watching History being made.
Or History being repeated.

— David Collum, Cornell University, 2013

The primary problem for creating digital money is how to prevent double spending. As the money is digital, and copies can be made *ad nauseam*, what can prevent counterfeiting? What would prevent users from sending copies of the same money to two (or more) people? That is precisely the problem solved by Bitcoin and its underlying Blockchain technology. The current solution behind fiat money is having a single issuer, a central bank — then trusting the financial institutions and regulators.

The concept of transferring money using cryptography as an underlying technology was shortly presented in 1983 by Chaum [13] and was deepened in a theoretical paper in 1985 [14]. However, it was only in 1988 that Chaum et al. [15] created the term *electronic cash* and also proposed a basic and practical scheme which yielded untraceability yet allowed to trace double spendings.

According to Barber et al. [4], despite the 30-year literature on e-cash, most of the proposed schemes requires a central authority which controls the currency issuance and prevents double spending [10, 11, 13, 36]. Some papers even propose solutions in a similar trajectory to Bitcoin, such as hash chain [47] and preventing double spending using peer-to-peer networks [27, 37]. The no central point of trust and predictable money supply together with a clever solution to the double-spending problem is what separates Bitcoin from the previous e-cash philosophies.

Bitcoin (BTC) is a digital currency, also known as digital money, internet money, and cryptocurrency. It is the first currency based on cryptography techniques which are distributed, decentralized, and with no central bank.

Bitcoin is a computer network in which nodes act like clerks performing clearing. A transaction clearing consists of ensuring that the transaction is settled according to the rules. In order to do that, every node stores a copy of Bitcoin's ledger, which records both all transactions and users' balance. When new transactions are added to the ledger, the balances are updated. It is said that Bitcoin is distributed because its ledger is public and is stored in thousands of computers. Even though the ledger is public, balances are anonymous, and no one knows who owns which funds[1]. If an attacker tries to change anything, the remaining of the network is able to detect it and ignore the change.

[1]There are some techniques which may de-anonymize transactions in specific situations, even when users are using Tor network. For further information, see Biryukov et al. [5], Jawaheri et al. [29], ShenTu and Yu [42].

Bitcoin is considered decentralized because there is no authority (or government) who decides its future. Every decision must be accepted by its community, and no one can enforce their will. Every change proposal must be submitted to the community who will discuss the matter and come to a verdict. If the majority of Bitcoin's community agrees on a decision, they just have to update their clearing process accordingly, and the changes are applied.

The security of Bitcoin relies on digital signature technology and network agreement. While digital signature ensures ownership, i.e., the funds may only be spent by their owners, and nobody else; the network agreement both prevents double spending and ensures that all processed transactions have sufficient funds. In short, every transaction must spend only unspent funds, must have enough funds available, and must be signed by its owners, authorizing its execution. Only when all these requirements are met, the funds are transferred.

Bitcoin provides interesting incentives to all players (users and miners). On the one hand, users may have incentives to use Bitcoin because (i) the fees are small and do not depend on the amount being transferred — but only in the size (in bytes) of the transaction —; (ii) the transfers will be confirmed in a well-known period; (iii) it is not possible to revert an already confirmed transfer, not even with a judicial order; and (iv) and the currency issuance rate is well-known and preset in Bitcoin's rules, which makes Bitcoin's supply predictable and trustworthy, different from fiat currencies which depends on decisions of their central banks — i.e., it would be virtually impossible to face a hyper inflation in Bitcoin due to currency issuance. On the other hand, miners have incentive to mine Bitcoin because new Bitcoins are found every ten minutes, and they may also collect the fees of unconfirmed transactions. It is important to highlight that anyone can become a miner, and there is no entry cost besides the mining equipment. These incentives have kept the Bitcoin network up and running since 2009 with barely any interruptions (99.99% uptime). For further information about incentives, see Catalini and Gans [12], Ma et al. [33].

Since 2009, Bitcoin has been growing and becoming more and more used all around the world. It started as an experiment based on a seminal work by Nakamoto [34] and expanded to the most important and successful cryptocurrency with a highly volatile $192 billion market capitalization, as of this writing [17]. There are hundreds of companies investing in different uses of the technology, from exchanges to debit cards, and billions of dollars being invested in the new markets based on Bitcoin's technology.

Despite Bitcoin's huge success, there are still many challenges to be overcome. We will focus on the following challenges: scaling, spamming, and centralization. One important challenge that we will skip is to reduce the size of the ledger (or blockchain), which today is around 125GB and is growing at a rate of 4.5GB per month [8].

The network must scale to support hundreds of transactions per second, while its capacity is around only eight transactions per second. Thus, the more Bitcoin becomes popular, the more saturated the network is. Network saturation has many side effects and may affect the players' incentive to keep the network running. The transaction fees have to be increased to compete for faster confirmation. The pool of unconfirmed transactions grows indefinitely, which may cause some transactions to be discarded due to low memory space available, as the previously predictable confirmation time of transactions becomes unpredictable.

The scaling problem is not precisely an issue of Bitcoin, but an issue of the Blockchain technology. Hence, all other Blockchain-based cryptocurrencies have the same limitations, such as Litecoin, Bitcoin Cash, and Ethereum. One may argue that increasing the maximum block size is a feasible solution to scaling, but I would say that it is just a temporary solution which buys some time until the next network saturation.

Bitcoin seems to have the most decentralized network among the cryptocurrencies, even so, there are few miners and mining pools which together control over 50% of the network's computing (hash)power (for details, see Gencer et al. [23]). Hence, they have an oversized

influence when it comes to changes in the Bitcoin protocol's behavior. They may also cooperate in an attack, voiding transactions which seemed confirmed. The more decentralized, the more trustworthy Bitcoin is. This centralization problem is seen as an important challenge.

Generating new transactions in Bitcoin has a tiny computational cost because one only has to generate the transaction itself, digitally sign it, and propagate it in the Bitcoin network. On the one hand, it means that any device is capable of generating new transactions, but, on the other hand, it makes Bitcoin susceptible to spam attacks. One may generate hundreds of thousands of new valid transactions, overloading the unconfirmed transactions pool and saturating the network. This spam problem has happened several times and affects Bitcoin's trustworthy. Parker [39] reports a possible spam attack lasting at least contiguous 18 months.

The number of ideas and publications focusing on improving Bitcoin's design and overcoming those challenges is increasing every day. Many of these proposals are organized into BIPs (Bitcoin Improvement Proposals) which are discussed and implemented by the community; while others come in the form of whitepapers and alternative software forks (which would include the need of a protocol upgrade). Other proposals are published in blogs and forums, describing new cryptocurrencies. Bitcoin's community hardly ever publishes their ideas in academic journals, preferring instead, of BIPs, white papers, and web discussions.

After the launch of Bitcoin, more than 1,000 other cryptocurrencies have been created [18]. In general, they are Bitcoin-like, which means they use similar technologies, including the blockchain. Some cryptocurrencies differs a lot from Bitcoin, like the ones which use the Directed Acyclic Graph (DAG) model [21, 31, 32, 40, 44, 46]. We are especially interested in one of them: Iota.

Iota uses a DAG model, called tangle, which has a different design than Bitcoin's blockchain. It has neither mining nor confirmation blocks and transaction fees. Each transaction has its own proof-of-work[2] and is used to confirm other transactions, forming a directed acyclic graph of transactions. Thus, a transaction is said to be confirmed when there is enough proof-of-work from the transactions confirming it directly or indirectly. There is no other way to confirm transactions but generating new transactions.

In Iota, as transactions confirm transactions, the network benefits from a high volume of new transactions. Therefore, theoretically, it scales to any large number of transactions per second. The scaling problem of tangle is exactly the opposite of Bitcoin's: it must have at least a given number of transaction per seconds; otherwise, the transactions are not confirmed, and the cryptocurrency does not work. While Iota's network has not reached this minimum number of transactions per second, it uses a central coordinator which works as a trustworthy node [45].

Every transaction confirmed by the central coordinator is assumed to be valid and cannot be reverted. The remaining of the network can verify a confirmation through the central coordinator's digital signature. The coordinator will not be necessary anymore when the number of transactions per second reaches a minimum value, but Iota's developers cannot say precisely what is this minimum value. This just elucidates that the tangle does not seem to work properly under a low volume of transactions (and fluctuations in the number of transactions per second may severely affect Iota's trustworthiness).

The present work intends to propose and analyze a new architecture, named Hathor, which lies between Bitcoin and Iota and may be a viable solution to both scaling, centralization, and spam problems.

[2]The mechanism that assures the immutability is the proof-of-work, which makes it computationally infeasible to tamper with transactions. It will be explained later in details.

2 Hathor's architecture

This work introduces Hathor's architecture, which lies between Bitcoin's and Iota's and may be a solution to scaling, centralization, and spam issues.

Like Iota, new transactions confirm previous ones, forming a Directed Acyclic Graph (DAG). For this, each transaction has its own proof-of-work which is solved by the issuer before propagating the transactions in the network. Like Bitcoin, miners find new "blocks" every 10 minutes in which they collect the fees and newly generated tokens. Each transaction has an "accumulated weight" which express the required effort to break the transaction, similar to Bitcoin's number of confirmations.

In Hathor, there are two difficulty levels: (i) one for new transactions which are just moving tokens around, and (ii) another one for "blocks" which are generating new tokens and collecting fees. The first may be adjusted to prevent spammers, which would spend too many resources to generate a great number of new transactions, whereas the latter is adjusted every 2,016 blocks to keep the pace of blocks on every 2 minutes.

Both miners and users will be working on proof-of-work, decentralizing even more the network's hash rate. Even though the users' difficulty is less than the miners', the hash rate will increase with every new user. The more transactions arrive, the higher the total hash rate. This may have good consequences in governance, which we will further discuss.

There is a trade-off about the *difficulty* of new transactions. The higher it is, the harder it is to generate new transactions, preventing spammers but also making it harder for IoT devices generate new transactions. This difficulty may even be increased when a spam attack is in course and reduced when it is gone. If the difficulty is too high, IoT devices may sign their transactions and send them to another devices which have a greater hash rate and will solve their proof-of-work faster.

It also seems interesting to have this difficulty depending on new transaction's size (in bytes) and amount being moved. The idea here is to require more work when high amounts are at stake. It would not affect IoT devices, which are expected to usually move smaller amounts. Regarding the transaction's size, requiring more work for larger transactions may make sense because they may prevent abuses, such as a denial-of-service attack using enormous transactions which would consume a lot of node's bandwidth and disk space.

Another important security matter is that each transaction has to confirm all its inputs, i.e., there must be a confirmation path between all the transactions of the inputs and the transaction which are spending them. It is always possible since there is at least one confirmation path between any transaction and a tip. This ensures that, when a conflict is resolved, only the sub-DAG with root at the invalidated transaction will be affected. The remaining parts of the DAG remains the same.

The transactions are classified into three groups: (i) confirmed transactions, (ii) in-progress transactions, and (iii) unconfirmed transactions (tips). The confirmed transactions are the ones which have already been settled, i.e., their accumulated weights have reached a minimum level. The unconfirmed transactions (tips) are the brand new transactions which have not been confirmed even once yet, i.e., their accumulated weights are zero. The in-progress transactions are in the middle. They have already been confirmed a few times, but not enough to reach the minimum level required to be a confirmed transaction. For simplicity, the pending transactions encompass both in-progress and unconfirmed transactions.

Another transaction classification concerns its validation by the network. A transaction is said to be network validated if there are confirming paths from all tips to the transactions, i.e., the whole network has validated that the transaction is valid. It is important to notice that a transaction may be network validated but still pending, or even be confirmed but not network validated.

A *block* is just a regular transaction with no inputs which confirms a previous block and at least two in-progress transactions or tips. There may be any number of outputs provided that they comply with the number of newly generated tokens. Each block collects all fees from all transactions confirmed by it which have not been confirmed by another block before. The blocks are ordered according to their timestamp. If two blocks have the same timestamp, the block hash is used as a tiebreaker.

In the low load scenario, there is a small number of new transactions coming into the network, which means they give a minor contribution to confirmations. In this case, the confirmation is held mostly by blocks. On the other hand, in the high load scenarios, there is a large number of new transactions giving a major contribution to confirmations. In this case, the blocks strengthen the confirmations, but many of them will have already been confirmed before the next blocks are found. The higher the number of new transactions, the faster the transactions are confirmed. The blocks assure a "maximum confirmation time".

The incentive scheme which keeps the network running is the same as Bitcoin's. Miners go towards fees and newly generate coins, whereas users just want to exchange their tokens. When there is no new transaction to be confirmed, the miners keep the network up and running while they find new blocks.

2.1 Transaction confirmation

Hathor uses similar concepts of weight and accumulated weight as Iota's. The weight depends only in the transaction itself, whereas the accumulated weight depends on its confirmations.

The weight of a transaction is calculated as $w = \log_2(k)$ where k is the average number of hashes required to solve its proof-of-work.

The accumulated weight is the average number of hashes required to solve the proof-of-work of the transaction itself plus all the transactions which confirm it. Let A be a transaction, its accumulated weight w_A is calculated as $\log_2(2^{w_A} + \sum_{A \rightsquigarrow P} 2^{w_P})$.

In Bitcoin, it is well-known that one should wait at least "six confirmations" before accepting a transaction. This Bitcoin's criteria is based on some results presented in Satoshi's seminal work [34] and derived here in more detail in Chapter 10. Adopting six confirmations is the same as demanding from attackers a minimal effort of six times the network's hash rate to successfully double spend those tokens. Let H be the total hash rate of the network. Then, as $\mathbf{E}(Y_6) = 60$ minutes, it will be necessary to calculate, on average, $\mathbf{E}(Y_6) \cdot H = 60 \cdot 60 \cdot H$ hashes to solve the proof-of-work of 6 blocks.

Therefore, in order to have the same level of security as Bitcoin, a transaction is said to be confirmed when its accumulated weight is greater than or equal to $\log_2(\mathbf{E}(Y_6) \cdot H) = \log_2(6 \cdot 128 \cdot H) = 7 + \log_2(6) + \log_2(H)$, where H is calculated as the total hash rate of the miners plus the total hash rate of new transactions.

2.2 Time between blocks

The hash function used for the proof-of-work (PoW) is the same as Bitcoin: *SHA-256* applied twice. Thus, most of the math analysis we have already done before is just the same.

Let X be the number of trials to solve the PoW and T be the time between blocks. We already know that X follows a geometric distribution with $p = \frac{A}{2^{256}}$, where A is inversely proportional to the difficulty, i.e., the smaller the A, the higher the difficulty. We also know that T follows an exponential distribution with $\lambda = \frac{1}{\eta}$, where η is the average time between blocks. As proved before, $\eta p H = 1$, thus let's define $w = \log_2(\mathbf{E}(X)) = \log_2(1/p) = \log_2(\eta H) = \log_2(\eta) + \log_2(H)$.

For the blocks, $\eta = 128$, thus, $\mathbf{E}(T) = \eta = 128$ seconds, and $\mathbf{V}(T) = \eta^2 = 16,384$, where T is the random variable of the time between two consecutive blocks. The symmetrical confidence interval of T with $\alpha = 10\%$ is $[6.56, 383.45]$, i.e., 90% of the cases the distance between blocks will be between 6 seconds and just under 7 minutes.

Let H be the hash rate of the miners, then, the weight of the blocks is calculated by

$$w_{\text{blocks}} = \log_2(128) + \log_2(H) = 7 + \log_2(H)$$

This weight will be updated every 675 blocks (which should happen every 24 hours) to take into consideration the change of H. First, H will be estimated using the fact that $\mathbf{E}(Y_{2016}) = \frac{2016}{\lambda} = 2016\eta = \frac{2016}{pH} = \frac{2016 \cdot 2^w}{H}$. Thus, $H = \frac{2016 \cdot 2^w}{\Delta t} \Rightarrow \log_2(H) = w + \log_2(2016) - \log_2(\Delta t)$, where Δt is the time between the latest weight update and now. Finally,

$$w_{\text{new}} = 7 + w_{\text{old}} + \log_2(2016) - \log_2(\Delta t)$$

Notice that, if the hash rate has not changed, then $\Delta t = 128 \cdot 2016$ and $w_{\text{new}} = w_{\text{old}}$.

2.3 Weight of the transactions

There is a trade-off which must be considered in the weight of the transactions: the higher the weight, the better to prevent spam, but the worse to microtransactions. So, trying to fulfill both necessities, the weight will be a function of the transaction's size (in bytes) and total amount:

$$w_{\text{tx}} = \log_2(\text{size}) + \log_2(\text{amount}) + 0.5$$

Although the transaction size depends on the implementation, a typical transaction with 2 inputs and 2 outputs would have approximately 188 bytes. So, for instance, transfering 50,000 tokens would require a weight of 23, which means an average of 2^{23} trials to solve the proof-of-work.

2.4 Issuance rate

Hathor issues tokens every block, thus it is similar to Bitcoin. Even so, an important decision is whether it will issue a limited number of tokens or not. Bitcoin chose to issue a limited number of tokens. On the other hand, Ethereum and others chose to issue an ilimited number of tokens.

Both ways, the number of issued tokens is predictable. This feature itself brings confidence to the community, who will not face monetary intervention (unless they agree to) — this comes in contrast with fiat money in which the central authority may change the monetary policy to adjust to political demands.

Bitcoin issuance rate started in 50 tokens per block and decreases over time. Every 210,000 blocks—4 years, on average—it halves the number of tokens issued per block. As Bitcoin smallest fraction is 10^{-8}, after the 33th reduction it will stop issuing new tokens since $2^{33} > 50 \cdot 10^{-8}$. In total, the number of issued tokens will never exceed 21 million. For further information, see [7].

2.5 Transaction fees

In Bitcoin, the value of the fee is calculated as the difference between the transaction's outputs and the inputs. For instance, a transaction with inputs summing 8,000 and outputs summing 7,000 is paying 1,000 of fees.

In Hathor, each transaction may pay a fee to the next block which confirms it. But, even if the fee is zero, the miners are forced to confirm at least two pending transactions. These two pending transactions also confirm other transactions which may have no fees. In summary, transactions with no fees cannot be left behind and will always be confirmed by both blocks and other transactions.

One may arguee that it allows the whole network to never pay fees and they are right. In the beginning, miners' incentive is driven by new tokens instead of fees. In the long term, it may be necessary to require a minimum fee to keep the miners working, depending on the chosen issuance policy.

2.6 Transaction validation

A transaction will be considered valid when it complies with the following rules: (i) it spends only unspent outputs; (ii) the sum of the inputs is greater than or equal to the sum of the outputs; (iii) the number of inputs is at most 256; (iv) the number of outputs is at most 256; (v) it confirms at least two pending transactions; (vi) it solves the proof-of-work with the correct weight.

The transaction has a timestamp field which is used to record when the transaction was generated. This timestamp field must be in UTC time to prevent timezone issues. It must also be within at most 5 minutes from the current time, otherwise the transaction will be discarded.

The digital signature is used to ensure that only the owners may spend their tokens. It will be calculated signing the transaction's input and output only. This allows the transaction to be signed in one device and to be sent to another device that will choose which transaction will be confirmed and will solve the proof-of-work.

Services of solving proof-of-work may also be offered by companies. They give their customers a wallet address and they send the payment inside of the transaction itself. This allows IoT devices to save energy, delegating the task of solving the proof-of-work.

In case of transaction conflict, in which two transactions try to spend the same tokens, the one with higher accumulated weight is chosen and the other is invalidated. Although it is not a possible policy in Iota because of the submarine attack, Hathor does not have the same problem. In Hathor, like in Bitcoin, the submarine attack is only possible if the attacker has a hash rate higher than the whole network, including the miners. In other words, when analysing the double spending attack, Hathor is as safe as Bitcoin.

2.7 Orphan blocks

Different from Bitcoin, there is no orphan blocks in Hathor. Unless a block confirms either directly or indirectly an invalid transaction, every block is valid. There is no need to leave a block behind since its proof-of-work is increasing transactions' accumulated weight.

If two blocks are trying to collect fees from the same transactions, the one with higher timestamp will be discarded.

2.8 Governance

In general, cryptocurrencies are decentralized, which means there is no central authority who decides its future, i.e., no one can enforce their will. Every decision must be accepted by its community, which means the community must agree. But what happens if they do not agree? When a consensus is not reached, the rules remain the same and the cryptocurrency may stall. The lack of a central authority may generate long debates, split the community, slow down

strategic decision-making, and, ultimately, come to a "civil war"—it is precisely what happened between Bitcoin Core (BTC) and Bitcoin Cash (BCH) in 2017 [16, 43].

Governance is an important part of a cryptocurrency because it must evolve, which means its community must agree into changing the rules. Governance is an agreement of how the community will proceed to change the rules.

Despite the large literature available about governance, what separates Blockchain-based cryptocurrencies from them is the decentralization (against the hierarchical model). The number of papers about governance in decentralized cryptocurrencies has been growing, but it still lacks a solution. Hacker [25] resorts to the theory of complex systems and proposes a governance framework for decentralized cryptocurrencies, which is, in summary, a centralized coordination entity. Hsieh et al. [28] has analyzed the effects of governance in returns using panel data on several cryptocurrencies. They present a deeper discussion about the parts of a governance mechanism and concludes that

> "... on the one hand, investors value cryptocurrencies' core value proposition, rooted in decentralization; but on the other hand, are suspicious of decentralized governance at higher levels in the organization because they could slow down strategic decision-making (e.g., regarding the introduction of new innovations) or create information asymmetries between investors and technologists."

I believe that the solution to a good governance will come from financial incentives to all players to find a common ground. In Bitcoin, users have less bargaining power than miners because they do not contribute with work, whereas, in Hathor, both miners and users are working together. So, when it comes to changing the rules, the bargaining power is more distributed than in Bitcoin. The distribution depends on the ratio of the miners' hashpower and the users' hashpower. The higher the ratio, the closer to Bitcoin's governance. The lower the ratio, the higher the bargaining power of the users.

2.9 Expected number of tips

It is intuitive that the number of tips depends on the number of transactions per second. The higher the number of transactions per second, the higher the number of tips. We can model the number of tips at time t as a stochastic process, which means it changes over time following some probability distribution. Stochastic processes have two important states: transient and steady. A process is in transient state when its properties are changing over time, whereas it is in steady state when its properties has already converged.

Let's assume that the number of transactions per second is constant and does not change over time. Thus, at the beginning, there will be no tips, and the number will increase until it converges to its stable quantity. The process is in transient state until it reaches stability. Then, it is in the steady state. Notice that the number of tips also change in steady state, but its average does not. It just floats around the average.

If the number of transactions per second increases, the process returns to the transient state and moves towards the new steady state. In practice, the number of transactions per second is always changing, so is the steady state.

Popov and Labs [40] has modeled the stochatic process of the total number of tips. It proves that the average number of tips, in the steady state, is:

$$\frac{\lambda_{\text{TX}} \cdot 2^{w_{\text{TX}}}}{\log(2) H}$$

A simple idea to get to this equation is through flow analysis: In steady state, we may say that the number of new tips must equal the number of tips being confirmed in a given time

window. Thus, in order to estimate the number of tips in the steady state, let's consider the process in which the rate of inward tips is λ_{TX}, whereas the rate of outward tips is ρ.

Let K be the number of tips at a given time. Thus, after Δt seconds, $\Delta\text{tips} = \lambda_{\text{TX}}\Delta t - \rho K \Delta t$. In the steady state, $\Delta\text{tips} = 0$, hence $K = \frac{\lambda_{\text{TX}}}{\rho}$.

The rate of outward tips is proportional to the rate in which devices solve the proof-of-work of transactions. Let H be the devices' hash rate, then, $\rho = \alpha \cdot H \cdot 2^{-w_{\text{TX}}}$, where α is the coefficient of proportionality.

Finally,

$$K = \frac{\lambda_{\text{TX}}}{\alpha}\frac{2^{w_{\text{TX}}}}{H}$$

If new transactions would always confirm two tips, then α would equal 2. But, it may confirm one or even zero tips because concurrent new transactions may confirm their tips first. Thus, $0 < \alpha \leq 2$. According to the equation found in Popov and Labs [40], $\alpha = \log(2) = 0.693147$.

3 Methodology

The methodology we have used is computer simulation. Through the simulation of many scenarios of Hathor, we will understand how the network behaves in complex scenarios, including when the load suddenly increases, and when the network is under attack.

The simulator has been developed using an event-based design which is capable of running hours of simulation in just a few minutes. It creates agents who decide to make a transaction, then they select which transactions will be confirmed, next they spend some time working in the proof-of-work, and, finally, they propagate the transaction to the network. The other agents receive the transaction and may accept or deny it. The agents may use different parameters among themselves.

When a new transaction emerges, it chooses two tips to confirm before solving the proof-of-work. When it finishes solving the proof-of-work, the transaction is propagated and becomes a tip. So, two new transactions may choose the same tips to confirm. If there are t tips, a new transaction will randomly choose 2 out of these t tips, even if they have already been chosen by other new transactions — in fact, they do not know which have been chosen because these new transactions have not been propagated yet.

When a new transaction is added to the Hathor's network, it uses a depth-first search [19] to update the aggregated weight of the directly and indirectly confirmed transactions. The depth-first search is interrupted when it reaches a transaction which the accumulated weight is larger than a given threshold. This interruption significantly increases the overall performance of the simulation. If the accumulated weight of the whole DAG is an important metric, the whole DAG may be updated in specific times to get a measurement (instead of every new transaction).

Simulator's random variables are all sampled from their distributions. The time between two transactions is sampled from an exponential distribution with λ_{TX}. The number of attempts to find a solution of the proof-of-work is sampled from a geometric distribution with $p = 2^w$, where w can be either w_{TX} or w_{block}. The amount of time spent to solve the proof-of-work is calculated dividing the number of attempts by the hash rate of the device (which could be either a miner or an user). The time between blocks is just the amount of time spent to solve the proof-of-work with w_{block}.

Transactions (and blocks) do not have inputs, outputs, and scripts. They have only pointers to other transactions, which form the DAG. They also store their weight, accumulated weight, timestamp, and some statistics used for reports.

New miners or users may be added or removed any time during a simulation. This allows the simulation of many different scenarios, such as increasing the number of miners, a sudden increase in the number of transactions per second, a sudden decrease in the number of miners, and so forth.

It is also possible to create metrics, which sample a statistic every Δt seconds. There are two metrics available: (i) TipsMetric, which stores the number of tips at a given simulation time, and (ii) UtterlyAcceptanceMetric, which finds the new utterly accepted transactions and store how long it took.

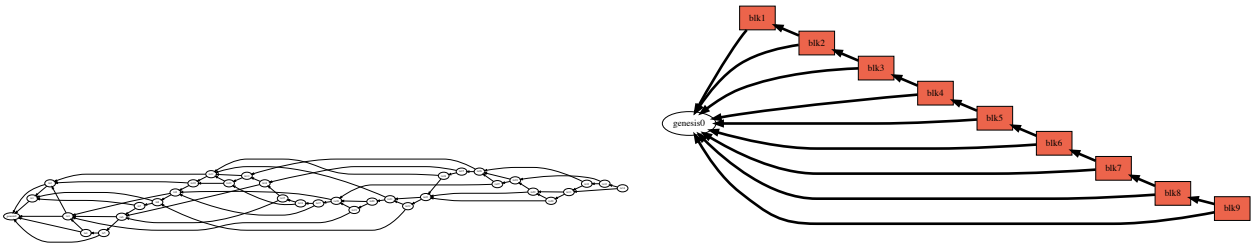
To find the network validated transactions, I run a breadth-first search (bfs) for each tip. Transactions visited in all searches are being confirmed by all tips and, by definition, are network validated transactions.

4 Analysis of Hathor

Hathor's architecture lies between Iota and Bitcoin's architectures. It is similar to Bitcoin's architecture when the number of transactions per second is low, while it is similar to Iota's architecture when the number of transactions per second is high. There is no "switching" between Bitcoin and Iota. It just behaves like one or another according to the network. In order to check this statement, I have performed some simulations.

First, two simulations in the extreme cases: (i) no transactions, (ii) no miners. The first should have precisely the same behavior as Bitcoin's blocks with one block after the other forming a long chain. The latter should have a similar behavior as Iota's, forming a Direct Acyclic Graph (DAG) with new transactions confirming previous ones. We can see in Figure 1 that both cases seem to be correct.

Next, I have run a more realistic simulation, with both miners and transactions. As we can see in Figure 2, Hathor's structure is a mix of Iota and Bitcoin's structure. The transactions are forming a DAG while, in parallel, the blocks are forming a chain inside the DAG.



(a) No miners

(b) No transactions

Figure 1: Visualization of a Hathor's graph in two particular cases: (a) no miners, (b) no transactions. It shows that when there are no miners, Hathor is similar to Iota (same structure, but different parameters), and when there is no transactions, it is similar to Bitcoin.

5 Confirmation time

Another evidence that Hathor lies between Iota and Bitcoin is found when comparing the time to confirm a transaction. In this context, a transaction is said to be confirmed when it has reached an accumulated weight similar to six times the hash rate of the whole network (miners and new transactions). This criteria is equivalent to the well-known "6 confirmations" of Bitcoin, which is adopted by almost the whole ecosystem.

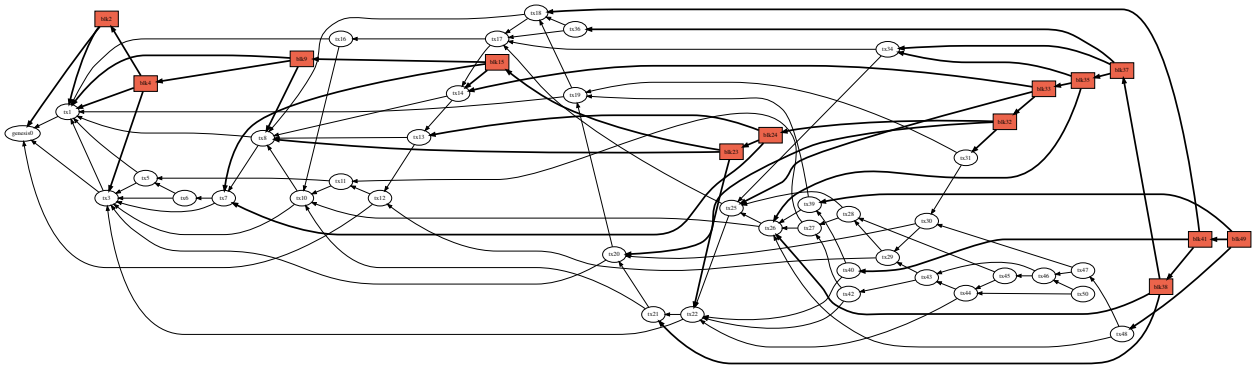
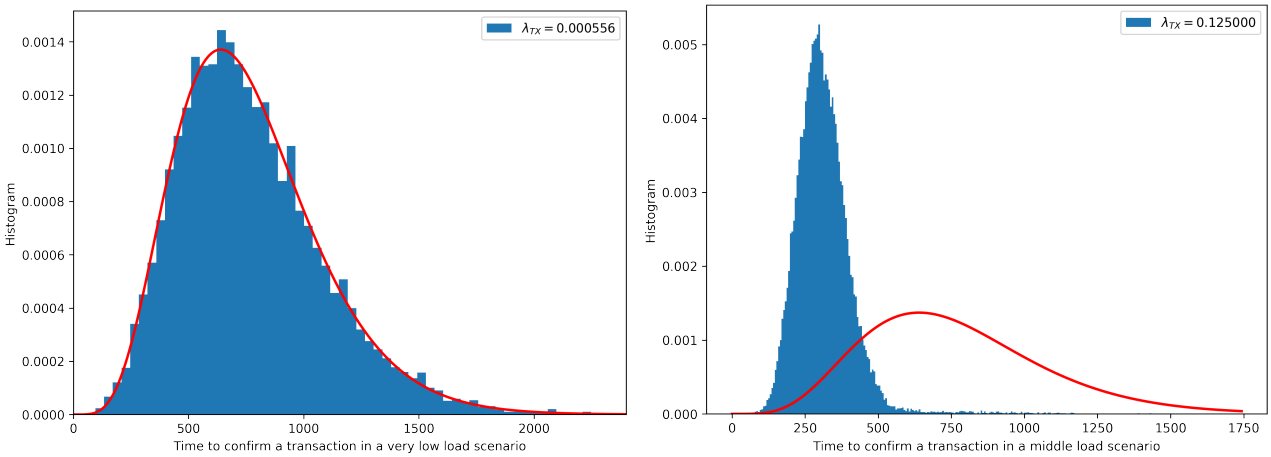


Figure 2: Visualization of a Hathor's graph with transactions and blocks. Red boxes are blocks, and white circles are simple transactions. The arrows show the confirmations.

Thus, I have run a simulation in which miners are majority and there are few transactions. In Figure 3, we may see a good fit between the confirmation time of a transaction and the theoretical distribution of the time to find six blocks in Bitcoin (which is Y_6 and follows an Erlang distribution). The blocks create "maximum confirmation time", since they are found with a precise pace, i.e., when there is not enough new transactions coming, the confirmation is done by the blocks. But, when the load is increased, Hathor's confirmation time is reduced and diverges from Bitcoin's time distribution. The reasoning is that confirmations coming from other transactions start to play an important role and accelerates the speed of confirmations, i.e., there is no need to wait for the next blocks, because the transactions are confirming themselves. This is what allows Hathor to scale and support higher volumes, indeed.



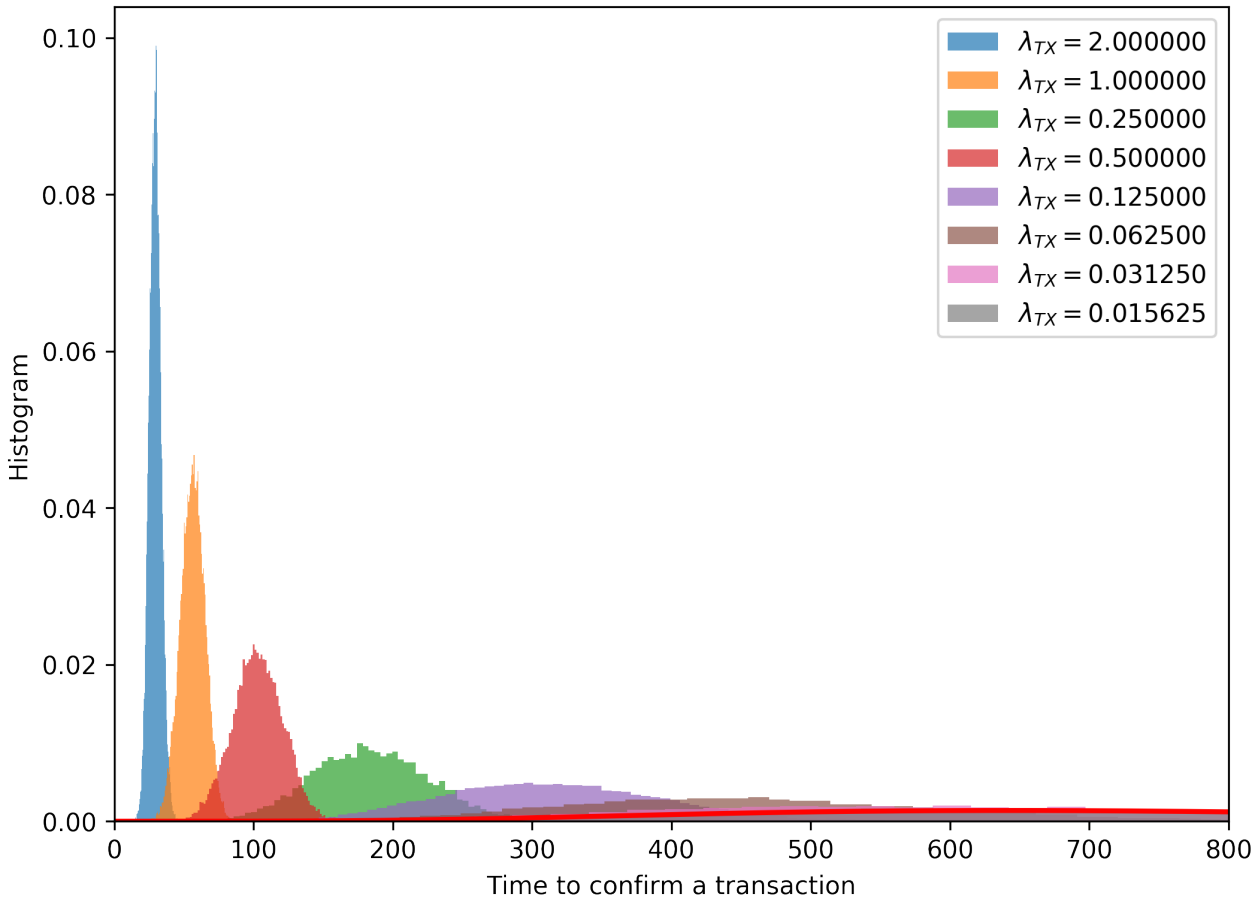
(a) Low load

(b) Mid load

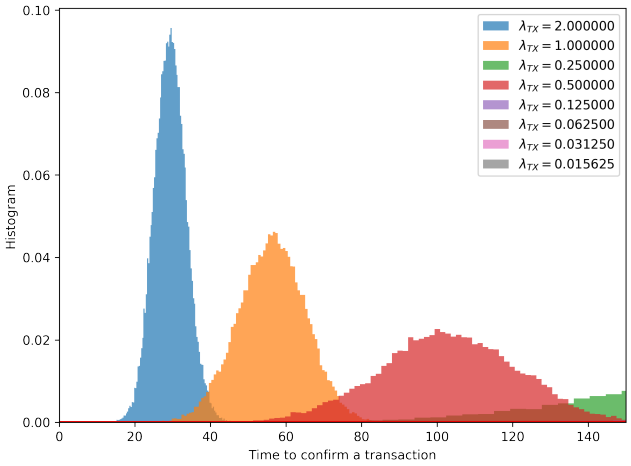
Figure 3: Confirmation time in two scenarios: (a) low load, (b) mid load. The red curve is the distribuion of the time to find six blocks in Bitcoin (which follows an Erlang distribution). As we can notice, in the low load scenario, Hathor's confirmation time behaves just like Bitcoin's. When the load is increased, it starts to diverge from Bitcoin's distribution.

We may see Hathor's confirmation time moving from Bitcoin's to Iota's in Figure 4. Notice that the confirmation timer is getting smaller as the number of transactions per second increases. Figure 4c is a zoom-in in the right side, and we can see again the good fit between Hathor Bitcoin's confirmation time under low load.

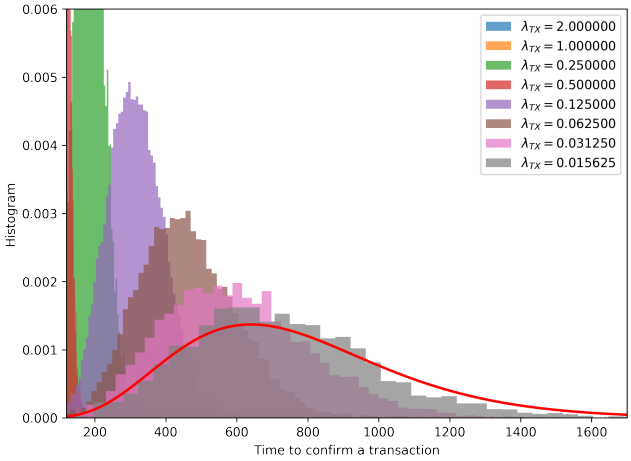
But, what would happen if, instead of changing the number of transactions per second, we change the relative hash power between miners and transactions? In previous simulations, the



(a) The whole picture



(b) Zoom in the left part of the above chart

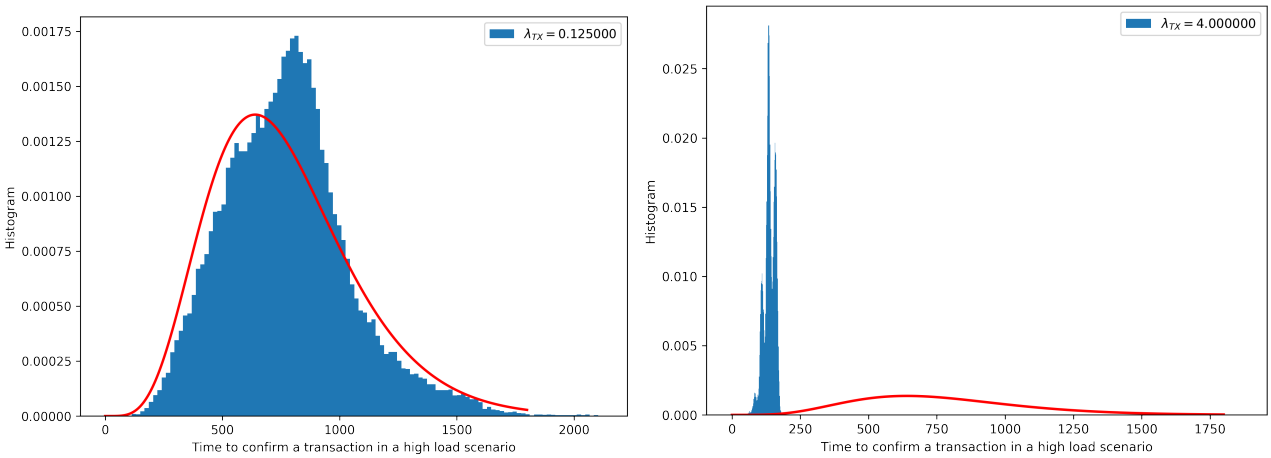


(c) Zoom in the right part of the above chart

Figure 4: Confirmation time in many scenarios, moving from a low load ($\lambda_{\mathrm{TX}} = 0.015625$) to a high load ($\lambda_{\mathrm{TX}} = 2$).

miners had a hash rate in the same magnitude as the transactions. In Figure 5, we can see the same simulation as in Figure 3b, but with the miners' hash rate ten times the transactions'. Besides the difference in the shape of the distribution, we can see that it is moving back towards Bitcoin's confirmation time distribution. It also makes sense because increasing the miners' hash rate increases the required minimum accumulated weighted for confirmed transactions. Therefore, more transactions are necessary to give more accumulated weight. As the number of transactions per second was not changed, most of the work of confirmations were done by

blocks (and not by transactions). To confirm this idea, I kept the miners' hash rate ten times the transactions' and increased 16 times the number of transactions per second. As we can see in Figure 5b, when the number of transactions per second is increased, its role in the accumulated weight also increases and it goes farther from Bitcoin's distribution.

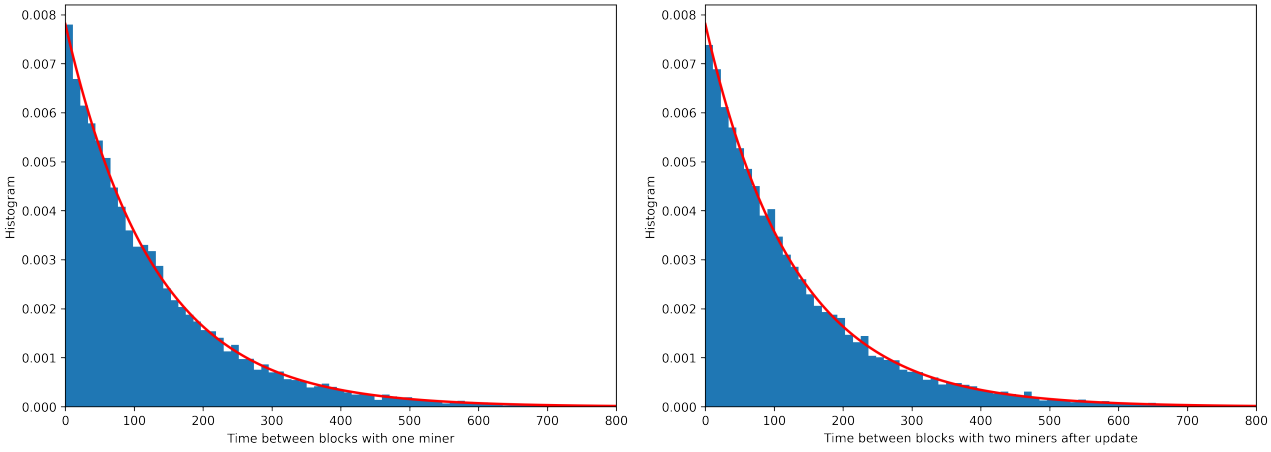


(a) Same load as in Figure 3b

(b) High load, 16 times of (a)

Figure 5: Confirmation time with miners' hash rate ten times the transactions'.

Finally, even with both blocks and transactions, Hathor's blocks are similar to Bitcoin's blocks, and they share the same math. To confirm that, see Figure 6, where the red curve is Bitcoin's distribution of time between blocks and the blue histogram is Hathor's time between blocks. I also made several tests adding and removing miners to check the difficulty adjustment and it worked properly.



(a) 1 miner

(b) 2 miners after difficulty adjustment

Figure 6: Histogram of time between blocks. The red curve the Bitcoin's theoretical distribution of time between blocks. As we can notice, the fit is very good.

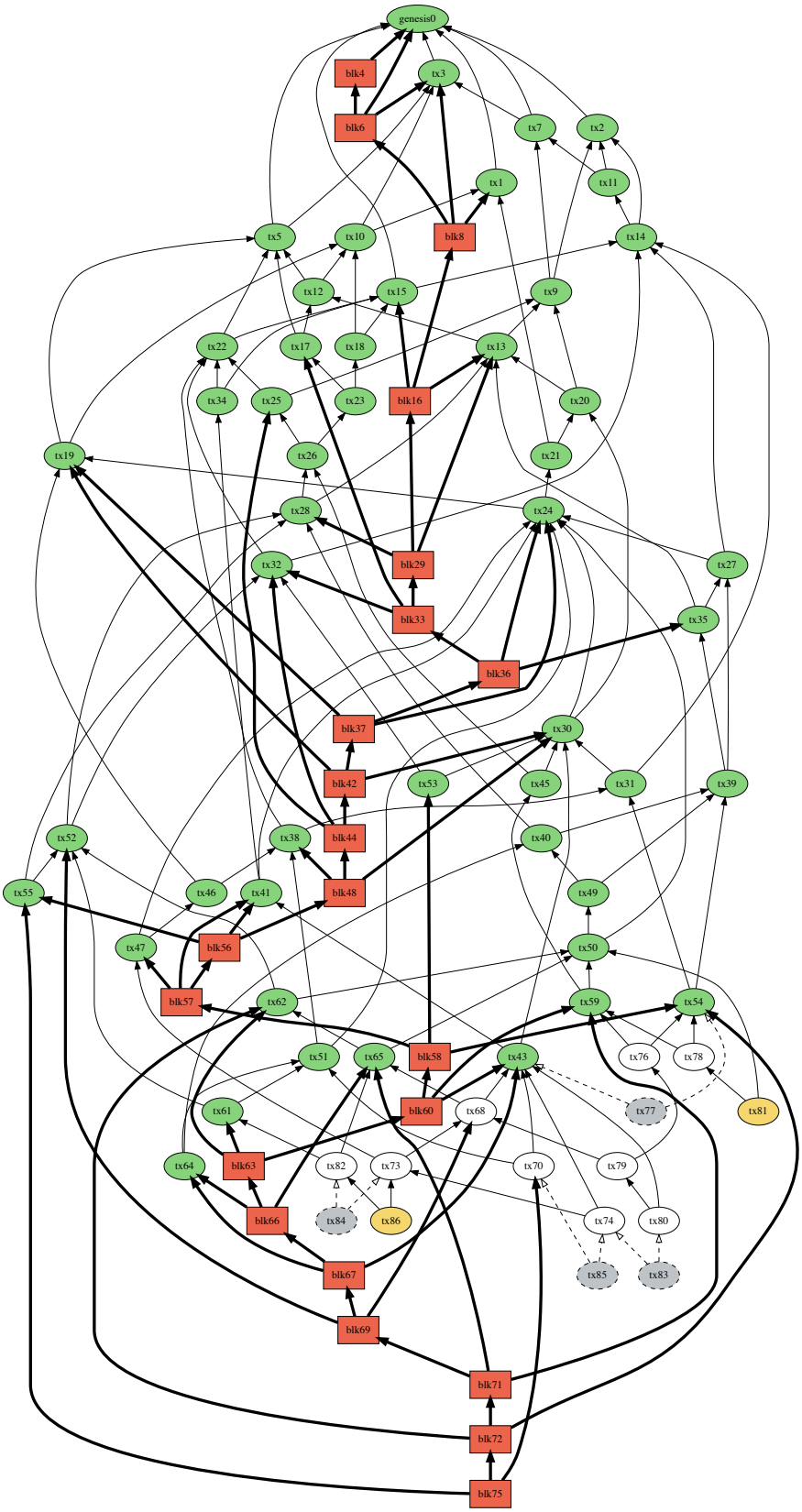


Figure 7: Visualization of a Hathor's graph with transactions and blocks. Red boxes are blocks; green circles are confirmed transactions; white circles are in-progress transactions; yellow circles are unconfirmed transactions (tips); and grey circles are transactions solving the proof-of-work which have not been propagated yet. The arrows show the confirmation chain. Block's arrows are in bold.

6 Visualizing the network

Visualizing the network is not simple because the number of transactions and blocks is high, thus, arranging them and their edges is a non-trivial task. Therefore, most of the visualization are just part of the DAG which shows a window of time.

To a better visualization of a Hathor's network, I run a simulation classifying the transactions in either confirmed, in-progress, or unconfirmed (tips). I also showed the transactions solving the proof-of-work which had not been propagated yet. See Figure 7 and notice the chain of blocks inside the DAG. Confirmed transactions are in green circles, while in-progress transactions are in white circles and tips are in yellow circles. The blocks are in red boxes form a chain inside the DAG. Finally, the new transactions which are solving the proof-of-work and had not been propagated are in grey dashed circles.

As new transactions have to chose two previous transactions to confirm, and just after they start to work in the proof-of-work, two new transactions eventually may chose the same tip because they do not know each other yet. So, if new transactions are coming in a low pace, the width of the swarm is small because the number of new transactions simultaneously solving the proof-of-work is also small. But, when new transactions are coming in a high volume, the width of the swarm increases because new transactions are choosing the same tip over and over.

In summary, the width of the swarm depends on the number of transactions per second. The greater the number of new transactions per second, the larger the width of the swarm.

To visualize the change in the width of the swarm, I run a simulation with a constant number of transactions per second, which was increased only in a specific window of time. The result can be seen in Figure 8, where the number of transactions per second suddenly increases, and the width increases until it reaches a stable value. Then, the number of transactions per second decreases, and the width returns to the previous value.

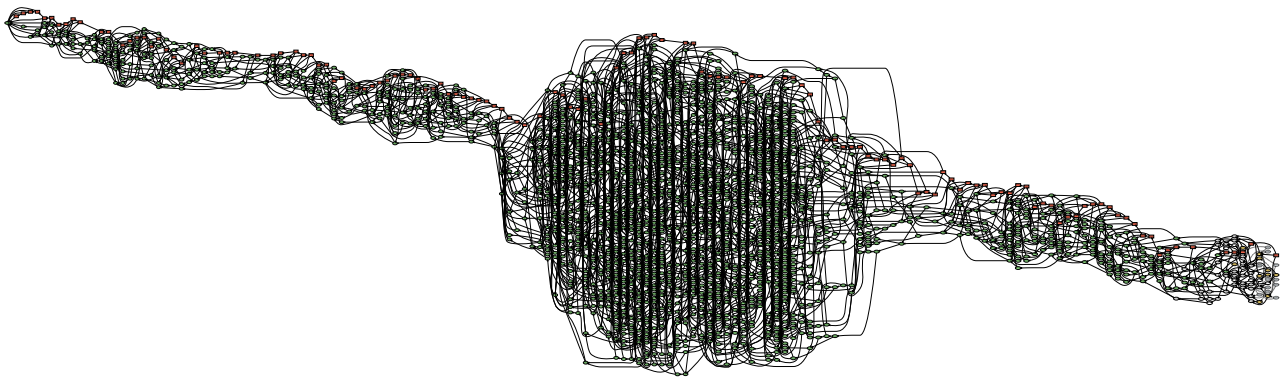


Figure 8: DAG visualization when the loading is changed over time.

7 Number of tips

The number of tips at a given time also depends on the number of new transactions per second. The relationship between them was empirically measured in several simulations with different λ_{TX}.

Analyzing Figure 9, it is easy to notice that both the average and the standard deviation of the number of tips increase when λ_{TX} increases. According to Popov and Labs [40], the average has the following equation:

$$\frac{\lambda_{\text{TX}}}{\log(2)} \frac{2^{w_{\text{TX}}}}{H}$$

In our simulation, $H = 100,000$ and $w_{\text{TX}} = 17$, so, the average number of tips should be equal to $1.89 \cdot \lambda_{\text{TX}}$. This model works best for low values of λ_{TX} and diverges a little for higher values. For instance, if $\lambda_{\text{TX}} = 32$ tx/s, this equation predicts 60.48 tips on average yet we can check in the empirical result that the average is around 55 tips.

When $\lambda_{\text{TX}} \to 0$, the number of tips goes towards one. The explanation is that there will be only one new transaction solving the proof-of-work per turn. So, new transactions will always confirm two tips, reducing the number of tips by one—each transaction confirms two tips and create one new one, hence the balance is -1. As it is impossible to have less than one tip, it converges to one.

When there is only one tip, new transactions must chose this only tip and another in-progress transaction. It should only happen in very low load scenarios, like during the launch of the network itself.

8 Network validated transactions

When all tips are confirming a transaction directly or indirectly, it is said that the transaction is network validated. It means that the whole network has checked the transactions and agrees that it is valid.

When a transaction is network validated, all new transactions and blocks will confirm that transaction. Thus, its aggregated weight will increase as fast as possible.

Let λ_{TX} be the number of new transactions per second. If λ_{TX} is constant, it means that, on average, there will be $\lambda_{\text{TX}}\Delta t$ new transactions after Δt seconds (because the number of new transactions after Δt seconds follows a Poisson distribution). All these transactions will confirm the network validated transactions. Hence, the number of transactions confirming a network validated transactions grows linearly. This result was also predicted by Popov and Labs [40, p.14].

Suppose a transaction has just become network validated. Let acc_0 be its accumulated weight when it became network validated, η be the average time between blocks, w_{TX} be the average transaction weight, and w_{BLK} be the average block weight. Then,

$$\begin{aligned}
\text{acc}(\Delta t) &= \log_2 \left(2^{\text{acc}_0} + \lambda_{\text{TX}}\Delta t \cdot 2^{w_{\text{TX}}} + \left\lfloor \frac{\Delta t}{\eta} \right\rfloor 2^{w_{\text{BLK}}} \right) \\
&= \text{acc}_0 + \log_2 \left(1 + \lambda_{\text{TX}}\Delta t \cdot 2^{w_{\text{TX}} - \text{acc}_0} + \left\lfloor \frac{\Delta t}{\eta} \right\rfloor 2^{w_{\text{BLK}} - \text{acc}_0} \right) \\
&\simeq \text{acc}_0 + \log_2 \left(1 + \lambda_{\text{TX}}\Delta t \cdot 2^{w_{\text{TX}} - \text{acc}_0} + \frac{\Delta t}{\eta} 2^{w_{\text{BLK}} - \text{acc}_0} \right) \\
&= \text{acc}_0 + \log_2 \left(1 + \lambda_{\text{TX}}\Delta t \cdot 2^{w_{\text{TX}} - \text{acc}_0} + \Delta t \cdot \eta^{-1} 2^{w_{\text{BLK}} - \text{acc}_0} \right) \\
&= \text{acc}_0 + \log_2 \left[1 + \Delta t \cdot 2^{-\text{acc}_0} \cdot (\lambda_{\text{TX}} 2^{w_{\text{TX}}} + \eta^{-1} 2^{w_{\text{BLK}}}) \right]
\end{aligned}$$

Therefore, after being network validated, the accumulated weight of a transaction grows logarithmically.

In Figure 10, we can see how long it takes for a transaction to be network validated in different scenarios. The time is quite the same for λ_{TX} less than one, but it changes for higher values.

It is interesting to notice that there is a trade-off when λ_{TX} increases. On the one hand, new transactions grow the DAG and accelerate the network validation, whereas, on the other hand, both the number of tips and the width of the swarm increases, dispersing this acceleration. The results show that, in fact, the time to be network validated increases with λ_{TX}.

Anyway, for up to 32 tx/s (and $\eta = 128$), it is reasonable to state that most transactions will be network validated after 35 seconds.

9 Conclusion

Bitcoin's underlying technology, blockchain, has been called by many as a major invention, even comparable to the invention of the internet. But it is unlikely that Bitcoin and blockchain have achieved the final or most optimal design for a secure and scalable electronic transaction system. In this work, I proposed and analysed a new architecture named Hathor, which seems a scalable alternative to Bitcoin.

Today, Bitcoin network can barely handle 8 transactions per second without increasing the unconfirmed transaction list to hundreds of thousands — several transactions take days to be confirmed. In order to increase Bitcoin's capacity, its community has first proposed and implemented segregated witness, which improved scalability yet was not enough. Finally, they proposed the lightning network, which is in development and should be available in the next months. I believe these proposals relieve the network—a temporary solution—, but do not solve the scalability problem.

Hathor's architecture allows a great number of transactions per second, since new transactions confirm previous ones (and there is no such thing as "maximum block size"). The more transactions are coming, the faster previous transactions will be confirmed. It is the opposite of Bitcoin because the network benefits from high volume scenarios. As I have shown, Hathor seems to solve the scalability problem present in Blockchain-based cryptocurrencies.

As the transactions also have a proof-of-work, it becomes harder to perform a spam attack. The attacker would spend a considerable amount of computational resources to solve the proof-of-work of every transaction, and the amount of work depends on the transaction's weight parameter. Future work may explore automatic adjustments in transaction's weight to improve spam prevention. For instance, the network can detect a higher number of new transactions coming and increase the transaction's weight for a while. Or else, the transaction's weight may be a function of the time between an output being spent and its spending transaction, so, transferring the same tokens over and over in a small window of time would require more work. Anyway, the transaction's weight seems to tackle the spam issue. The new challenge is to set a proper transaction's weight which would prevent spam without impairing IoT devices.

The last, but not least, challenge is the hashpower centralization. Although Bitcoin seems to have the most decentralized network among cryptocurrencies, there are few miners and mining pools which *together* control over 50% of the network's computing (hash)power [23]. Hence, they have an oversized influence when it comes to changes in the Bitcoin protocol's behavior. Hathor's architecture splits the hashpower among miners and users. Even if miners have more individual hashpower than users, because they would have rigs with appropriate cooling and energy supply, I believe their aggregate hashpower will not surpass users' aggregate hashpower when millions of devices are generating transactions. Future IoT devices may even come with an application-specific integrated circuit (asic) designed to solve Hathor's proof-of-work without spending too much battery. Future work may check common IoT processors' hashpower, which would allows us to estimate how many devices would be necessary to surpass miners' hashpower.

Even though I have proposed to update block's weight every 24 hours (or 675 blocks),

this was an arbitrary number. Future work may explore whether it would be feasible to continuously update block's weight, or what would be the optimal number of blocks between each update. I believe that the challenge of a continuous update approach would be preventing outdated nodes to discard valid blocks when two or more blocks were being propagated through Hathor's network. Maybe a solution would be allowing a range of block's weight instead of a single value, but future work would have to check whether this can be exploited by attackers.

I also presented a mathematical analysis of Blockchain, going though mining, hashpower change, orphan blocks, and double-spending attacks. Most of the presented results may directly be applied to Hathor's blocks, since their foundations are the same. As Popov and Labs [40] has already analyzed Tangle, I have just applied their results with a few extensions.

Future work may also further analyze other possibilities of attack in Hathor's network, such as malicious device not using random tip selection. Another major challenge affecting all cryptocurrencies is disk space use. How would one wipe out part of the blocks and transactions without putting security in risk? At first, all blocks and transactions are required to check whether anyone (including computer viruses) has tampered with transactions which had already been validated and stored in disk.

References

[1] Babaioff, M., Dobzinski, S., Oren, S., and Zohar, A. (2012). On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73. ACM.

[2] Bahack, L. (2013). Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*.

[3] Bailey, N. T. (1954). On queueing processes with bulk service. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 80–87.

[4] Barber, S., Boyen, X., Shi, E., and Uzun, E. (2012). Bitter to better—how to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*, pages 399–414. Springer.

[5] Biryukov, A., Khovratovich, D., and Pustogarov, I. (2014). Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM.

[6] BitcoinStats (2013-2017). Data propagation. `http://bitcoinstats.com/network/propagation/`.

[7] BitcoinWiki (2017). Controlled supply. `https://en.bitcoin.it/wiki/Controlled_supply`.

[8] Blockchain.info. Bitcoin blockchain size. `https://blockchain.info/charts/blocks-size`. Last accessed on July 14, 2017.

[9] Bonneau, J., Felten, E. W., Goldfeder, S., Kroll, J. A., and Narayanan, A. (2016). Why buy when you can rent? bribery attacks on bitcoin consensus.

[10] Camenisch, J., Hohenberger, S., and Lysyanskaya, A. (2005). Compact e-cash. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 302–321. Springer.

[11] Canard, S. and Gouget, A. (2007). Divisible e-cash systems can be truly anonymous. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 482–497. Springer.

[12] Catalini, C. and Gans, J. S. (2016). Some simple economics of the blockchain. Technical report, National Bureau of Economic Research.

[13] Chaum, D. (1983). Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer.

[14] Chaum, D. (1985). Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044.

[15] Chaum, D., Fiat, A., and Naor, M. (1988). Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography*, pages 319–327. Springer.

[16] Chen, L. Y. and Nakamura, Y. (2017). Bitcoin is having a civil war right as it enters a critical month. https://www.bloomberg.com/news/articles/2017-07-10/bitcoin-risks-splintering-as-civil-war-enters-critical-month.

[17] CoinMarketCap. Bitcoin market capitalizations. http://coinmarketcap.com/currencies/bitcoin/. Last accessed on July 14, 2017.

[18] CoinMarketCap. Cryptocurrency market capitalizations. https://coinmarketcap.com/. Last accessed on July 14, 2017.

[19] Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

[20] Decker, C. and Wattenhofer, R. (2013). Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE.

[21] Discussion (2014). Dag, a generalized blockchain. https://nxtforum.org/proof-of-stake-algorithm/dag-a-generalized-blockchain/ (registration at nxtforum.org required).

[22] Dobbertin, H., Bosselaers, A., and Preneel, B. (1996). Ripemd-160: A strengthened version of ripemd. In *Fast Software Encryption*, pages 71–82. Springer.

[23] Gencer, A. E., Basu, S., Eyal, I., van Renesse, R., and Sirer, E. G. (2018). Decentralization in bitcoin and ethereum networks. *arXiv preprint arXiv:1801.03998*.

[24] Gilbert, H. and Handschuh, H. (2003). Security analysis of sha-256 and sisters. In *International workshop on selected areas in cryptography*, pages 175–193. Springer.

[25] Hacker, P. (2017). Corporate governance for complex cryptocurrencies? a framework for stability and decision making in blockchain-based monetary systems.

[26] Heilman, E., Kendler, A., Zohar, A., and Goldberg, S. (2015). Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium*, pages 129–144.

[27] Hoepman, J.-H. (2007). Distributed double spending prevention. In *International Workshop on Security Protocols*, pages 152–165. Springer.

[28] Hsieh, Y.-Y., Vergne, J.-P., and Wang, S. (2017). The internal and external governance of blockchain-based organizations: Evidence from cryptocurrencies.

[29] Jawaheri, H. A., Sabah, M. A., Boshmaf, Y., and Erbad, A. (2018). When a small leak sinks a great ship: Deanonymizing tor hidden service users through bitcoin transactions analysis. *arXiv preprint arXiv:1801.07501*.

[30] Karame, G., Androulaki, E., and Capkun, S. (2012). Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012(248).

[31] Lerner, S. D. (2015). Dagcoin: a cryptocurrency without blocks. Available at https://bitslog.wordpress.com/2015/09/11/dagcoin/.

[32] Lewenberg, Y., Sompolinsky, Y., and Zohar, A. (2015). Inclusive block chain protocols. Available at http://www.cs.huji.ac.il/~avivz/pubs/15/inclusivebtc.pdf.

[33] Ma, J., Gans, J. S., and Tourky, R. (2018). Market structure in bitcoin mining. Technical report, National Bureau of Economic Research.

[34] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Available at https://bitcoin.org/bitcoin.pdf.

[35] Neudecker, T., Andelfinger, P., and Hartenstein, H. (2015). A simulation model for analysis of attacks on the bitcoin peer-to-peer network. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 1327–1332. IEEE.

[36] Okamoto, T. (1995). An efficient divisible electronic cash scheme. In *Annual International Cryptology Conference*, pages 438–451. Springer.

[37] Osipkov, I., Vasserman, E. Y., Hopper, N., and Kim, Y. (2007). Combating double-spending using cooperative p2p systems. In *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*, pages 41–41. IEEE.

[38] Ozisik, A. P., Bissias, G., and Levine, B. N. Estimation of miner hash rates and consensus on blockchains. Technical report, Tech. rep., PDF available from arxiv. org and https://www. cs. umass. edu/ brian/status-reports. pdf (June 2017).

[39] Parker, L. (2017). Bitcoin 'spam attack' stressed network for at least 18 months, claims software developer. https://bravenewcoin.com/news/bitcoin-spam-attack-stressed-network-for-at-least-18-months-claims-software-developer/.

[40] Popov, S. and Labs, J. (2016). The tangle. Available at https://iota.org/IOTA_Whitepaper.pdf.

[41] Shah, D. et al. (2009). Gossip algorithms. *Foundations and Trends® in Networking*, 3(1):1–125.

[42] ShenTu, Q. and Yu, J. (2015). Research on anonymization and de-anonymization in the bitcoin system. *arXiv preprint arXiv:1510.07782*.

[43] Shin, L. (2017). Bitcoin cash skyrockets, bitcoin price drops as civil war continues. https://www.forbes.com/sites/laurashin/2017/11/12/bitcoin-cash-skyrockets-bitcoin-price-drops-as-civil-war-continues/#67ffeed635b5.

[44] Sompolinsky, Y. and Zohar, A. (2013). Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. Available at https://eprint.iacr.org/2013/881.pdf.

[45] Sønstebø, D. The transparency compendium. https://blog.iota.org/the-transparency-compendium-26aa5bb8e260. Last accessed on July 20, 2017.

[46] Vorick, D. (2015). Getting rid of blocks. Available at slides.com/davidvorick/braids.

[47] Zongkai, Y., Weimin, L., and Yunmeng, T. (2004). A new fair micropayment system based on hash chain. In *e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004 IEEE International Conference on*, pages 139–145. IEEE.

10 Appendix: Analysis of Bitcoin

The primary objective of this chapter is to increase the understanding of Bitcoin through mathematical tools.

10.1 Hash function

Hash functions has been widely studied in computer science. In short, a hash function $h : \{0,1\}^\infty \to \{0,1\}^n$ has the following properties:

1. $x = y \Rightarrow h(x) = h(y)$
2. $h(x) \sim \mathcal{U}(0, 2^n - 1)$, where $\mathcal{U}$ is the uniform distribution, i.e., $\forall a \in [0, 2^n - 1], \mathbf{P}(h(x) = a) = \frac{1}{2^n}$

In other words, when two inputs are the same, they have the same output. But, when the inputs are different, their outputs are uniformly distributed. Clearly, the hash functions are surjective but not injective. They are not injective because the image of h has only 2^n elements and the domain has infinite elements. When $x \neq y$ and $h(x) = h(y)$, we say that x and y are a collision. A hash function is considered to be safe when it is unknown how to quickly find a collision of a given hash, i.e., one has to check all possible values until the correct one is found (known as the brute-force attack).

Bitcoin uses two hash functions: HASH-160 and HASH-256. The first has $n = 160$ and consists of the composition of *SHA-256* and *RIPEMD-160*. The latter has $n = 256$ and applies *SHA-256* twice. The first is used in transactions' scripts and the latter in the mining algorithm. For both hash functions, it is infeasible to run a brute-force attack because it would demand, on average, either 2^{160} or 2^{256} trials, and those would take a tremendous amount of time even for the fastest known processors.

For further information about hash functions, see Dobbertin et al. [22], Gilbert and Handschuh [24].

10.2 Mining one block

Let $\mathbb{B}$ be the set of Bitcoin blocks and $h : \mathbb{B} \to \{0,1\}^{256}$ be the Bitcoin *HASH-256* function. The mining process consists of finding $x \in \mathbb{B}$ such as $h(x) < A$, where A is a given threshold. The smaller the A, the harder to find a new block. In fact, $\mathbf{P}(h(x) < A) = \frac{A}{2^{256}}$.

Hence, in order to find a new block, one must try different inputs $(x_1, x_2, \ldots, x_k)$ until they find a solution, i.e., all attempts will fail ($h(x_i) \geq A$ for $i < k$) but the last ($h(x_k) < A$). The probability of finding a solution exactly in the k^{th} attempt follows a geometric distribution. Let X be the number of attempts until a success, then $\mathbf{P}(X = k) = (1-p)^{k-1}p$, where $p = \frac{A}{2^{256}}$. Also, we have $\mathbf{P}(X \leq k) = 1 - (1-p)^k$. The average number of attempts is $\mathbf{E}(X) = 1/p$ and the variance is $\mathbf{V}(X) = \frac{1-p}{p^2}$.

In the Bitcoin protocol, the given number A is adjusted so that the network would find a new a block every 10 minutes, on average. Suppose that the Bitcoin network is able to calculate H hashes per second — H is the total hash rate of the network. The time required to find a solution would be $T = X/H$, and $\mathbf{E}(T) = \mathbf{E}(X)/H$ would be the average number of seconds to find a new block. So, the rule of finding a new block every 10 minutes ($\eta = 600$ seconds) — on average — leads to the following equation: $\mathbf{E}(T) = \eta = 600$. So, $\mathbf{E}(T) = \mathbf{E}(X)/H = \frac{1}{pH} = \eta = 600 \Rightarrow p = \frac{1}{\eta H}$. Finally, $\mathbf{E}(X) = \eta H$, $\mathbf{E}(T) = \eta$, $\mathbf{V}(X) = (\eta H)^2 - \eta H$, and $\mathbf{V}(T) = \eta^2 - \eta/H$.

The cumulative distribution function (CDF) of T is $\mathbf{P}(T \leq t) = \mathbf{P}(X/H \leq t) = \mathbf{P}(X \leq tH) = 1 - (1-p)^{tH} = 1 - \left(1 - \frac{1}{\eta H}\right)^{tH}$. But, as the Bitcoin network hash rate is really large, we may approximate the CDF of T by $\lim_{H\to\infty} \mathbf{P}(T \leq t) = 1 - e^{-\frac{t}{\eta}}$, which is equal to the CDF of the exponential distribution with parameter $\lambda = \frac{1}{\eta}$.

Theorem 1. *When $H \to +\infty$, the time between blocks follows an exponential distribution with parameter $\lambda = \frac{1}{\eta}$, i.e., $\lim_{H\to+\infty} \mathbf{P}(T \leq t) = 1 - e^{-\frac{t}{\eta}}$.*

Proof.

$$\begin{aligned}
\mathbf{P}(T \leq t) &= 1 - (1-p)^{tH} \\
&= 1 - \left(1 - \frac{1}{\eta H}\right)^{tH}
\end{aligned}$$

Replacing $u = \eta H$,

$$\begin{aligned}
\lim_{H\to+\infty} \mathbf{P}(T \leq t) &= \lim_{u\to+\infty} 1 - \left(1 - \frac{1}{u}\right)^{\frac{tu}{\eta}} \\
&= \lim_{u\to+\infty} 1 - \left[\left(1 - \frac{1}{u}\right)^{u}\right]^{\frac{t}{\eta}} \\
&= 1 - (1/e)^{\frac{t}{\eta}} \\
&= 1 - e^{-\frac{t}{\eta}}
\end{aligned}$$

□

Now, we would like to understand from which value of H it is reasonable to assume that T follows an exponential distribution.

Theorem 2. $x > M \Rightarrow |(1 + 1/x)^x - e| < e/M$.

Proof. Let's use the classical inequality $\frac{x}{1+x} < \log(1+x) < x$ for $x > -1$. So, $\frac{1/x}{1+1/x} < \log(1 + x) < 1/x$. Simplifying, $\frac{1/x}{1+1/x} = 1/(1 + x)$. Thus, $1/(1 + x) < \log(1 + 1/x) < 1/x \Rightarrow x/(1 + x) < x \log(1 + 1/x) < 1$.

As $\log(1 + \frac{1}{M}) > 0$ and $1 < 1 + \log(1 + \frac{1}{M})$.

$x > M \Rightarrow 1/x < 1/M \Rightarrow 1 + 1/x < 1 + 1/M \Rightarrow 1/(1 + 1/x) > 1/(1 + 1/M) \Rightarrow x/(1 + x) > M/(1 + M)$.

Again, $\log(1 + x) < x \Rightarrow \log(1 - 1/M) < -1/M \Rightarrow 1 + \log(1 - 1/M) < (M - 1)/M < M/(1 + M)$, since $(x - 1)/x < x/(x + 1)$.

Hence, $1 + \log(1 - 1/M) < M/(1 + M) < x/(1 + x) < x \log(1 + 1/x)$, and $x \log(1 + 1/x) < 1 < 1 + \log(1 + \frac{1}{M})$.

Finally,

$$
\begin{aligned}
1+\log(1-1/M) &< x\log(1+1/x) < 1+\log(1+\frac{1}{M}) \\
e^{1+\log(1-1/M)} &< e^{x\log(1+1/x)} < e^{1+\log(1+\frac{1}{M})} \\
e\cdot e^{\log(1-1/M)} &< e^{\log((1+1/x)^x)} < e\cdot e^{\log(1+\frac{1}{M})} \\
e(1-1/M) &< (1+1/x)^x < e(1+\frac{1}{M}) \\
e-e/M &< (1+1/x)^x < e+e/M \\
-e/M &< (1+1/x)^x - e < e/M
\end{aligned}
$$

Therefore, $|(1+1/x)^x - e| < e/M$. □

We may consider H big enough to say that T follows an exponential distribution when $e/H < \epsilon$, where ϵ is the maximum approximation error. When $\epsilon = 10^{-6} \Rightarrow H > e \cdot 10^6$. So, when $H > 2.6$Mh/s, our approximation is good enough.

The symmetrical confidence interval with level α would be $[t_0, t_1]$, where $\mathbf{P}(t_0 < T < t_1) = 1-\alpha$, $\mathbf{P}(T < t_0) = \alpha/2$, and $\mathbf{P}(T > t_1) = \alpha/2$. These conditions give the following equations: $1-e^{-t_0/\eta} = \alpha/2$, and $e^{-t_1/\eta} = \alpha/2$. Solving these equations, we have $t_0 = -\eta\ln(1-\alpha/2)$, and $t_1 = -\eta\ln(\alpha/2)$.

For instance, if $\alpha = 10\%$, then $t_0 = 30.77$ and $t_1 = 1797.44$ (or $[0.51, 30.76]$ in minutes). Thus, 90% of the time the intervals between blocks are between 30 seconds and 30 minutes, with average of 10 minutes.

The fact that the time between blocks follows an exponential distribution with $\lambda = 1/\eta = pH$ may be used to estimate the total network's hash rate (or a miner's hash rate). For further information, see [38].

10.3 Mining several blocks

Let $T_1, T_2, T_3, \ldots, T_n$ be the time to find the first block (T_1), then the time to find the second block (T_2), and so on. Let's analyze the distribution of $Y_n = \sum_{i=1}^{n} T_i$ which is the total time to find the next n blocks. As Y_n is the sum of random variables which follow an exponential distribution with same $\lambda = \frac{1}{\eta}$, then $Y_n \sim \text{Erlang}(n, \frac{1}{\eta})$. Thus, the CDF of Y would be $\mathbf{P}(Y_n < t) = 1 - \sum_{k=0}^{n-1} \frac{1}{k!} e^{-\lambda t}(\lambda t)^k$.

Many exchanges require at least six confirmations in order to accept a deposit in Bitcoin. So, for $n = 6$, $\mathbf{P}(Y_6 < 1 \text{ hour}) = \mathbf{P}(Y_6 < 3600) = 0.5543$, i.e., only 55% of the deposits will be accepted in one hour. The symmetrical confidence interval with $\alpha = 10\%$ is $[27, 105]$ in minutes. Thus, 90% of the times, it will take between 27 minutes and 1 hour and 45 minutes to have your deposit accepted — assuming that your transaction will be confirmed in the very next block. The pdf of Y_6 is shown in Figure 11, in which the 10% symmetrical confidence interval is shown in the white area. The average total time of six confirmations is $\mathbf{E}(Y_6) = 6 \cdot 600 = 3600 = 60$ minutes.

10.4 Mining for a miner

Let's analyze the probability of finding a new block for a miner who has α percent of the network's total hash rate. Let $T_\alpha = \frac{X}{\alpha H}$ be the time required for the miner to find a new block. As $T_\alpha = \left(\frac{1}{\alpha}\right) T$, when $H \to +\infty$, T_α also follows an exponential with parameter $\lambda_\alpha = \frac{\alpha}{\eta}$. Hence, we confirm the intuition that the miner with α percent of the network's total hash power will find α percent of the blocks.

Theorem 3. *When the miner with α percent of the network's total hash rate is part of the mining network,* $\mathbf{P}(\textit{next block is from } T_\alpha) = \alpha$.

Proof.

$$\begin{aligned}\mathbf{P}(\text{next block is from } T_\alpha) &= P\left(T_\alpha = \min\{T_\alpha, T_{1-\alpha}\}\right)\\ &= \frac{\lambda_\alpha}{\lambda_\alpha + \lambda_{1-\alpha}}\\ &= \frac{\alpha/\eta}{\alpha/\eta + (1-\alpha)/\eta}\\ &= \frac{\alpha}{\alpha + 1 - \alpha}\\ &= \alpha.\end{aligned}$$

□

Theorem 4. *When one miner with α percent of the network's total hash rate multiplies their hash rate by m, the probability of this miner find the next block is multiplied by $\frac{m}{m\alpha+1-\alpha}$.*

Proof. When miners increase their hash rate, they also increase the network's total hash rate. Let H be the network's hash rate before the increase. Thus, the network's total hash rate after the increase is $H + (m-1)\alpha H = (1 - \alpha + m\alpha)H$. So,

$$\begin{aligned}\mathbf{P}(\text{next block is from } T_{m\alpha}) &= P\left(T_{m\alpha} = \min\{T_{m\alpha}, T_{1-\alpha}\}\right)\\ &= \frac{\lambda_{m\alpha}}{\lambda_{m\alpha} + \lambda_{1-\alpha}}\\ &= \frac{m\alpha/\eta}{m\alpha/\eta + (1-\alpha)/\eta}\\ &= \frac{m\alpha}{m\alpha + 1 - \alpha}\\ &= \alpha\left(\frac{m}{m\alpha + 1 - \alpha}\right).\end{aligned}$$

□

Corollary. *If one miner has a really tiny percent of the network's total hash rate, then multiplying their hash rate by m approximately multiplies their probability of finding the next block by m.*

Proof.

$$\lim_{\alpha\to 0} \mathbf{P}(\text{next block is from } T_{m\alpha}) = \lim_{\alpha\to 0} \frac{m}{m\alpha + 1 - \alpha} = m.$$

□

That way, it is not exactly correct to say that when one doubles their hash rate, their probability will double as well. It is only true for small miners.

10.5 Orphan blocks

An orphan block would be created if a new block is found during the propagation time of a new block. Let α be the percentage of the total hash rate of the node which is outdated, and Δt the propagation time in seconds. Thus, $\mathbf{P}(\text{new orphan}) = \mathbf{P}(T < \Delta t) = 1 - e^{-\frac{\alpha\Delta t}{\eta}}$.

Bitcoin peer-to-peer network is a gossip network, where miners are semi-randomly connected to each other, and each miner sends all information it receives to all its peers. According to Decker and Wattenhofer [20], the average time for a new block propagate over the network is 12.6 seconds, while the 95% percentile is 40 seconds, which indicates a long-tail distribution. BitcoinStats [6] has measured the propagation time between 2013 and 2017. During 2017, the worst daily 90% percentile was 21 seconds. Notice that both results may not be contradictory because the Bitcoin network is continuously evolving.

For instance, if a node has 10% of the total hash rate and it takes 30 seconds to receive the update, then $\mathbf{P}(\text{new orphan}) = 1 - e^{-\frac{0.1 \cdot 30}{600}} = 0.004987$, which is almost 0.5%. I would say that a node with 10% of the total hash rate would be well connected and it would take less time to receive the update, so, the probability would be even smaller than 0.5%.

Another important factor is that, as Bitcoin is open-source, miners are free to change the gossip algorithm, which leads to the network incentives. See Babaioff et al. [1] for an analysis of the incentives to miners forward new blocks and transactions in the network.

For further information about gossip algorithms, see Shah et al. [41].

10.6 Analysis of network's hash rate change

The difficulty, given by the number A, is adjusted every 2016 blocks. As, $\mathbf{P}(13 \text{ days} < Y_{2016} < 15 \text{ days}) = \mathbf{P}(13 \cdot 24 \cdot 3600 < Y_{2016} < 14 \cdot 24 \cdot 3600) = 0.9986$, it is expected that the total time to find 2016 blocks will be between 13 and 15 days, assuming that the network's hash rate remains constant. If it takes less than the expected time, it means that the network's total hash rate has increased. While if it takes more than the expected time, it means that the network's total hash rate has decreased. So, let's analyze what happens when the network's hash rate changes significantly.

Let $H \cdot u(t)$ be the network's total hash rate over time. So, the number of hashes calculated in t seconds is $H \int_0^t u(t)dt$. Hence, $\mathbf{P}(T \leq t) = \mathbf{P}(X \leq H \int_0^t u(t)dt)$. When $H \to +\infty$, $\mathbf{P}(T \leq t) = 1 - e^{-\frac{1}{\eta} \int_0^t u(t)dt}$, and the pdf of T is $\frac{u(t)}{\eta} \cdot e^{-\frac{1}{\eta} \int_0^t u(t)dt}$.

10.6.1 Hash rate suddenly changing

Let's say that the network's total hash rate has suddenly multiplied by α. So, $u(t) = \alpha$, $\int_0^t u(t)dt = \alpha t$, and T also follows an exponential distribution, but with $\lambda = \frac{\alpha}{\eta}$. Thus, $Y_n^\alpha = \sum_{i=1}^n T_i^\alpha \sim \text{Erlang}(n, \frac{\alpha}{\eta})$. Thus, $\mathbf{E}[Y_n^\alpha] = \frac{\mathbf{E}[Y_n]}{\alpha}$, i.e., the average total time required to find n blocks will be divided by α, while $\mathbf{V}[Y_n^\alpha] = \frac{\mathbf{V}[Y_n]}{\alpha^2}$ and the variance will be divided by α^2. Hence, on one hand, when the network's hash rate increases ($\alpha > 1$), the 2016 blocks will be found earlier. On the other hand, when the network's hash rate decreases ($\alpha < 1$), the 2016 blocks will be found later.

For example, if the network's total hash rate suddenly doubles ($\alpha = 2$), then $\mathbf{P}(6.5 \text{ days} < Y_{2016} < 7.5 \text{ days}) = 0.9986$, and the time required to find 2016 blocks halved. On the other side, if the network's total hash rate suddenly halves ($\alpha = 0.5$), then $\mathbf{P}(27 \text{ days} < Y_{2016} < 29 \text{ days}) = 0.9469$, and the time required to find 2016 blocks doubled. It is an important conclusion, since it shows that even if half of the network stops mining, it will only double the time to the next difficulty adjustment, i.e., the time between blocks will be 20 minutes for, at most, the next 29 days, at which point the adjustment will occur and everything will be back to the normal 10 minutes between blocks.

10.6.2 Hash rate smoothly changing

Let $u(t) = \frac{1+abx}{1+bx}$. It is an useful function because $u(0) = 1$ and $\lim_{t\to\infty} u(t) = a$. The bigger the b, the faster $u(t) \to a$. For example, if $a = 2$, it means H would be smoothly doubling. If $a = 0.5$, it means H would be smoothly halving.

It is easy to integrate $u(t)$ because $\frac{1+abx}{1+bx} = \frac{1-a}{1+bx} + a$, which yields $\int_0^t u(x)dx = at + \frac{1-a}{b}\log(1+bt)$. So,

$$F_T(t) = 1 - (1+bt)^{\frac{\lambda(a-1)}{b}} e^{-\lambda at}.$$

$$f_T(t) = \lambda \left(\frac{1+abt}{1+bt}\right) (1+bt)^{\frac{\lambda(a-1)}{b}} e^{-\lambda at}.$$

Assuming that $n = \frac{\lambda(a-1)}{b}$ is integer, we have:

$$F_T(t) = 1 - (1+bt)^n e^{-\lambda at}$$

Let $\mathcal{L}$ be the Laplace Transform. Thus,

$$\begin{aligned}
\mathcal{L}\{F_T(t)\} &= \mathcal{L}\{1 - (1+bt)^n e^{-\lambda at}\} \\
&= \mathcal{L}\{1\} - \mathcal{L}\{(1+bt)^n e^{-\lambda at}\} && (\mathcal{L} \text{ is a linear operator}) \\
&= \frac{1}{s} - \mathcal{L}\{(1+bt)^n e^{-\lambda at}\} \\
&= \frac{1}{s} - \sum_{k=0}^{n} \binom{n}{k} b^k \mathcal{L}\{t^k e^{-\lambda at}\} \\
&= \frac{1}{s} - \sum_{k=0}^{n} \binom{n}{k} b^k \frac{k!}{(s+\lambda a)^{k+1}}
\end{aligned}$$

Hence, as $\mathcal{L}\{f_T(t)\} = s\mathcal{L}\{F_T(t)\}$,

$$\mathcal{L}\{f_T(t)\} = 1 - \sum_{k=0}^{n} \binom{n}{k} \frac{s b^k k!}{(s+\lambda a)^{k+1}}$$

Then,

$$
\begin{aligned}
\frac{d}{ds}\mathcal{L}\{f_T(t)\} &= -\sum_{k=0}^{n}\binom{n}{k}b^k k!\frac{d}{ds}\frac{s}{(s+\lambda a)^{k+1}} \\
&= -\sum_{k=0}^{n}\binom{n}{k}b^k k!\left[\frac{1}{(s+a\lambda)^{k+1}} - \frac{s(k+1)}{(s+a\lambda)^{k+1}}\right] \\
\frac{d}{ds}\mathcal{L}\{f_T(t)\}|_{s=0} &= -\sum_{k=0}^{n}\binom{n}{k}b^k k!\frac{1}{(\lambda a)^{k+1}} \\
&= -\frac{1}{a\lambda}\sum_{k=0}^{n}\binom{n}{k}k!\left(\frac{b}{\lambda a}\right)^k \\
&= -\frac{1}{a\lambda}\sum_{k=0}^{n}\frac{n!}{(n-k)!}\left(\frac{b}{\lambda a}\right)^k \\
&= -\frac{1}{a\lambda}\left[n!\sum_{k=0}^{n}\frac{1}{(n-k)!}\left(\frac{b}{\lambda a}\right)^k\right] \\
&= -\frac{1}{a\lambda}\left[n!\sum_{k=0}^{n}\frac{1}{k!}\left(\frac{b}{\lambda a}\right)^{n-k}\right] && (k \to n-k) \\
&= -\frac{1}{a\lambda}\left[n!\left(\frac{b}{\lambda a}\right)^n\sum_{k=0}^{n}\frac{1}{k!}\left(\frac{b}{\lambda a}\right)^{-k}\right] \\
&= -\frac{1}{a\lambda}\left[n!\left(\frac{b}{\lambda a}\right)^n\sum_{k=0}^{n}\frac{1}{k!}\left(\frac{\lambda a}{b}\right)^{k}\right]
\end{aligned}
$$

Finally, as $\mathbf{E}[T] = -\mathcal{L}\{f_T(t)\}|_{s=0}$,

$$
\mathbf{E}[T] = \frac{1}{\lambda a}\left[n!\left(\frac{b}{\lambda a}\right)^n\sum_{k=0}^{n}\frac{1}{k!}\left(\frac{\lambda a}{b}\right)^{k}\right], \text{ where } n = \frac{\lambda(a-1)}{b}
$$

Let's check this equation for already known scenarios. When $a = 1$, then $n = 0$ and $\mathbf{E}[T] = 1/\lambda$. When $b \to +\infty$, it reduces to the case in which the hash rate is multiplied by a, which we have already studied. In fact, $b \to +\infty$ yields $n \to 0$, $u(t) \to a$, and $\mathbf{E}[T] = \frac{1}{\lambda a}$.

Theorem 5.

$$
a > 1 \text{ and } x > M \Rightarrow \left|\frac{1+abx}{1+bx} - a\right| < \frac{a-1}{1+bM}
$$

Proof. $x > M \Rightarrow \frac{1}{1+bx} < \frac{1}{1+bM}$. As $1-a<0$, $\frac{1-a}{1+bx} > \frac{1-a}{1+bM}$. Thus, $\frac{1-a}{1+bM} < \frac{1-a}{1+bx} + a - a = \frac{1+abx}{1+bx} - a < 0 < \frac{a-1}{1+bM}$. Hence, $-\frac{a-1}{1+bM} < \frac{1+abx}{1+bx} - a < \frac{a-1}{1+bM}$. □

For instance, if we would like to know the impact of smoothly double the hash rate in the next week, then the parameters would be $\lambda = 1/600$, $a = 2$, $M = 1 \text{ week} = 3600 \cdot 24 \cdot 7 = 604,800$, b can be calculated using $\epsilon = \frac{a-1}{1+bM} < 0.01$, which yields $b > 0.000163690$ and $n < 10.1818$. So, for $n = 10$, then $b = 0.000166666$ and $\epsilon = 0.009823 < 0.01$, as expected. Finally, $\mathbf{E}[T] = 557.65$. In other words, during the next week, the average time between blocks will be 9 minutes and 17 seconds, instead of the normal 10 minutes. If the hash rate had suddenly doubled, the average time between blocks would be 5 minutes.

10.6.3 Piecewise linear model of hash rate change

Let's analyze what would happen if the network's hash rate is growing linearly with angular coefficient a^2, i.e., $u(a,b,t) = a^2t + b$. Thus, $\mathbf{P}(T \leq t) = 1 - e^{-\frac{bt+a^2t^2/2}{\eta}}$.

It is well known that $\mathbf{E}(T) = \int_0^\infty 1 - \mathbf{P}(T \leq t)dt$. Thus, replacing $y = \frac{a^2t+b}{a\sqrt{2\eta}}$, and using the fact that $\int_0^\infty e^{-x^2}dx = \frac{\sqrt{\pi}}{2}\operatorname{erf}(x)$, we have:

$$\begin{aligned}
\mathbf{E}(T)|_{t_1}^{t_2} &= \int_{t_1}^{t_2} \exp\left(-\frac{bt + a^2t^2/2}{\eta}\right) dt \\
&= \frac{\sqrt{2\eta}}{a} \exp\left(\frac{b^2}{2a^2\eta}\right) \int_{y_1}^{y_2} \exp(-y^2)dy \\
&= \frac{\sqrt{2\eta}}{a} \exp\left(\frac{b^2}{2a^2\eta}\right) \frac{\sqrt{\pi}}{2}[\operatorname{erf}(y_1) - \operatorname{erf}(y_2)] \\
&= \frac{\sqrt{2\pi\eta}}{2a} \exp\left(\frac{b^2}{2a^2\eta}\right) [\operatorname{erf}(y_2) - \operatorname{erf}(y_1)] \qquad (1)
\end{aligned}$$

Where $y_1 = \frac{a^2t_1+b}{a\sqrt{2\eta}}$ and $y_2 = \frac{a^2t_2+b}{a\sqrt{2\eta}}$.

Thus, $\mathbf{E}(T) = \mathbf{E}(T)|_0^\infty$. When $t_1 = 0 \Rightarrow y_1 = \frac{b^2}{2\sqrt{2\eta}}$ and $t_2 \to \infty \Rightarrow y_2 \to \infty \Rightarrow \operatorname{erf}(y_2) = 1$, then:

$$\mathbf{E}(T) = \frac{\sqrt{2\pi\eta}}{2a} \exp\left(\frac{b^2}{2a^2\eta}\right) \left[1 - \operatorname{erf}\left(\frac{1}{a\sqrt{2\eta}}\right)\right]$$

10.6.4 Comparison of the models

In order to compare the hash rate change models, namely (i) suddenly changing, (ii) smoothly changing, and (iii) linearly changing, I have applied each of them to the same scenarios. In the first scenario, the hashrate will double in the next week, whereas, in the second scenario, it will halve in the next week.

In both the smoothly change model and the linear change model, I could have calculated each model's average time between blocks during one week. But, it would not give us much information, because the estimated average time between models would be increasing (or decreasing) more and more as the days goes by. And we are really interested in the average time between blocks throughout the days, and not the average of one week.

Thus, I have analyzed a piecewise hash rate change, i.e., I have calculated the average time between blocks for each hour throughout the week. First, I split the whole week into $24 \cdot 7$ intervals, $(t_0, t_1, t_2, \ldots, t_168)$, where $t_i = 3600i$. Then, I calculated the average for each interval (t_k, t_{k+1}). Let H_k^0 and H_k^1 be the initial and final hash rate of the (t_k, t_{k+1}) interval. So, I also ensured the continuity of the hash rate between consecutive intervals, i.e., $H_k^1 = H_{k+1}^0$.

I compared both the smoothly change model and the linear change model with the suddenly changing model. The difference between them is negligible. Let ϵ be the maximum absolute error between the models, than $\epsilon < 0.8$ and $\epsilon/H < 0.2\%$, for all intervals. The maximum absolute error between the linear and the suddenly changing models can be seen in Figure 12.

Therefore, we may conclude that it is reasonable to approximate the average time between blocks using only the suddenly changing model in each interval of one hour.

The average time between blocks throughout the days can be seen in Figure 13. It was calculated using the suddenly changing model with the hash rate changing linearly during the week.

10.7 Attack in the Bitcoin network

There are many possible ways to attack the Bitcoin network [2, 9, 26, 30, 35]. In this section, we are interested in a particular attack: the double spending attack.

In the double spending attack, the attacker's send some funds to the victim, let's say a merchant. They wait for k confirmations of the transaction, and the victim delivers the good or the service to the attacker. Then, the attacker mine enough blocks with a conflicting transaction, double spending the funds which was sent to the victim. If the attacker is successful, the original transaction will be *erased* and the victim will be left with no funds at all. In order to be successful, the attacker must propagate more blocks than the network in the same period, propagating a chain longer than the main chain. Hence, we would like to understand what the odds are that the attacker will be successful. This attack was originally discussed by Nakamoto [34].

In order to maximize their odds, the attacker must start to mine the new blocks as soon as they send the funds to the victim. In this moment, it starts to mine in the head of the blockchain, just like the rest of the network. So, in the beginning, the attacker and the network are in exactly the same point.

Let βH be the hash rate of the attackers, and γH be the network's hash rate without the attackers. Thus, when $H \to +\infty$, we already know that $T_{\text{attackers}}$ and T_{network} follow exponential distributions with parameters $\lambda_{\text{attacker}} = \frac{\beta}{\eta}$ and $\lambda_{\text{network}} = \frac{\gamma}{\eta}$, respectively.

As [34] has done, we will also model the attack using the Gambler's Ruin. In this game, a gambler wins \$1 at each round, with probability p, and loses \$1, with probability $1 - p$. The rounds are independent. The gambler starts with \$$k$ plays continuously until he either accumulates a target amount of \$$m$, or loses all his money. Let $\rho = \frac{1-p}{p}$, then the probability of losing his fortune is:

$$\mathbf{P}(\text{losing his fortune}) = \begin{cases} \frac{\rho^k - \rho^m}{1-\rho^m}, \text{ if } \rho \neq 1, \\ \frac{m-k}{m}, \text{ if } \rho = 1. \end{cases}$$

When $m \to +\infty$,

$$\mathbf{P}(\text{losing his fortune}) = \begin{cases} \rho^k, \text{ if } \rho < 1, \\ 1, \text{ if } \rho \geq 1. \end{cases}$$

The gambler winning \$1 is the same as the network finding a new block, the gambler losing \$1 is the same as the attacker finding a new block. The initial \$$k$ is the same as the number of blocks the attacker is behind the network. Thus, the gambler loses his fortune is the same as the attacker successfully finds k or more blocks than the network, i.e., losing his fortune means that the attack was successful.

In our case, $p = \frac{\lambda_{\text{network}}}{\lambda_{\text{network}} + \lambda_{\text{attacker}}} = \frac{\gamma}{\beta+\gamma}$, thus $\rho = \frac{\beta}{\gamma}$. Hence, $\rho < 1 \Leftrightarrow \beta < \gamma$.

Suppose that the attacker is mining with the network. Suddenly, he stops mining with the network and starts attacking, i.e., starts to mine in another chain. In this scenario, since the attacker's hash rate is not mining with the network anymore, $\gamma = 1 - \beta$. Thus, $\beta < \gamma \Rightarrow \beta < 0.5 \Leftrightarrow \rho < 1$. Here comes the conclusion that, if the attacker has 50% or more of the network's hash rate, then his attack will be certainly successful. We got exactly the same equations and conclusions as [34].

But this scenario seems not to be the optimal attack, because the attacker has waited k confirmations before starting the attack. A better approach would be to start attacking just after propagating the transaction. In this case, our previous model is not good, because even if the attacker have found more blocks than the network, he cannot propagate those blocks before the network has found k confirmations. So, we have to model the probabilities before

the network has found the k block. Then, if the attacker has more blocks than the network, he has successfully attacked. Otherwise, we return to the previous model, in which the attacker must still find more blocks.

Theorem 6. *Assuming that the attacker starts the attack just after publishing the transaction, the probability of the attacker has already found exactly s blocks while it waits the network to find k blocks is* $\mathbf{P}(S = s) = \binom{k+s-1}{s}(1-p)^s p^k$.

Proof. The attacker must find exactly s blocks while the network must find exactly k blocks. It is as they would be walking the grid from the point $(0, 0)$ to (s, k), where it is only allowed to go up or right, like in Figure 14. When the attacker finds a block, it would be a movement to the right. When the network finds a block, it would be an upward movement. No matter the order which the blocks are found, all the paths occur with probability $(1-p)^s p^k$.

The walking ends when $(\cdot, k)$ is reached, i.e., when the network finds k blocks, regardless of how many blocks the attacker has found – i.e., it is not allowed to walk above the line $(\cdot, k)$. Thus, the number of paths between $(0, 0)$ and (s, k) moving only upward or to the right, without going into the line $(\cdot, k)$ is exactly the number of paths between $(0, 0)$ and $(s, k-1)$, which is equal to the number of permutations of the sequence $(u, u, \ldots, u, r, r, \ldots, r)$ in which there are s movements to the right (r) and $k-1$ upward movements (u). This number of permutations is $\frac{(k-1+s)!}{s!(k-1)!} = \binom{(k-1)+(s)}{s}$ because there are s repetitions of the element r and $k-1$ repetitions of the element u.

Finally, the probability is $\binom{k+s-1}{s}(1-p)^s p^k$.

□

Assuming that the attacker starts mining just after publishing the victim's transaction, the probability of the attacker will have found more than k blocks while it waits the network to find k blocks is $\mathbf{P}(S \geq k) = \sum_{s=k}^{\infty} \binom{k+s-1}{s}(1-p)^s p^k$.

Theorem 7.

$$\mathbf{P}(S \geq k) = 1 - \sum_{s=0}^{k-1} \binom{k+s-1}{s}(1-p)^s p^k.$$

Proof. Let's use the following identity:

$$\frac{1}{(1-z)^{a+1}} = \sum_{i=0}^{\infty} \binom{i+a}{i} z^i, \text{ for } |z| < 1$$

Thus, replacing $z = 1-p$, $i = s$, and $a = k-1$, we have:

$$\frac{1}{p^k} = \sum_{s=0}^{\infty} \binom{s+k-1}{s}(1-p)^s$$

$$1 = \sum_{s=0}^{\infty} \binom{s+k-1}{s}(1-p)^s p^k.$$

Now, just split $\sum_{s=0}^{\infty} = \sum_{s=0}^{k-1} + \sum_{s=k}^{\infty}$ and it is done. □

Using this last theorem, we moved from an infinity sum to a finity sum.

Theorem 8. *Let* $p = \frac{\gamma}{\beta+\gamma}$.

$$\mathbf{P}(\textit{successful attack}) = \begin{cases} 1 - \sum_{s=0}^{k-1} \binom{k+s-1}{s} \left((1-p)^s p^k - (1-p)^k p^s\right), & p \geq 0.5 \\ 1, & p < 0.5. \end{cases}$$

Proof.

$$\mathbf{P}(\text{successful attack}) = \mathbf{P}(S \geq k) + \sum_{i=0}^{k-1} \mathbf{P}(s = i)\rho^{k-i}$$

□

For $k = 6$, $p = 0.9$, $\mathbf{P}(\text{successful attack}) = 0.0005914121600000266$.
For $k = 6$, $p = 0.7$, $\mathbf{P}(\text{successful attack}) = 0.15644958192000014$.

10.8 Confirmation time and network capacity

Let's say that when a new transaction is propagated it is enqueued in the unconfirmed transaction queue. Then, when a new block is found, some of these transactions in the queue are confirmed. We are interested in some measures of the queue, like the expected time to confirm a transaction and the queue's length.

Let's assume that all transactions have exactly the same size S and pay exactly the same fee. If the Bitcoin block's maximum size is M, there would be room for $s = \lfloor M/S \rfloor$ transactions in each block.

Using the results from Bailey [3], we have found that $\pi_n = \frac{z_s - 1}{z_s^{n+1}}$ is the probability of having n unconfirmed transactions in the pool subjected to $s > m$, where $m = \frac{\lambda_{\text{TX}}}{\lambda_{\text{blocks}}}$ and z_s is the single root of the polynomial $z^s(1 + m(1 - z)) - 1$ with $|z_s| > 1$. In this case, the average size of the unconfirmed transaction pool is $\mathbf{E}(\pi) = \frac{1}{z_s - 1}$.

When $s > m$, the probabilities π_n form a simple geometric series with common ratio smaller than one, which means the probabilities are exponentially decreasing. Since $\pi_n \to 0$ when $n \to \infty$, we may interpret it as a stable system, i.e., the unconfirmed transactions pool size is finite.

When $s \leq m$, the system is unstable, which means the unconfirmed transactions pool size keeps growing towards infinity. In this case, the system is not capable of processing the demand for a long period of time.

Using the fact that $m = \frac{\lambda_{\text{TX}}}{\lambda_{\text{blocks}}}$ and $\lambda_{\text{blocks}} = 1/\eta$, the stability condition $s > m$ is reached when $\lambda_{\text{TX}} < s/\eta$.

In the Bitcoin network, the average number of transactions per block is $s = 2,250$, so, the system is stable when $\lambda_{\text{TX}} < 2,250/600 = 3.75$ tx/s. Therefore, 3.75 is the maximum number of new transactions per second that the Bitcoin network may handle. When $\lambda_{\text{TX}} > 3.75$ tx/s, the unconfirmed transaction pool starts to grow indefinitely.

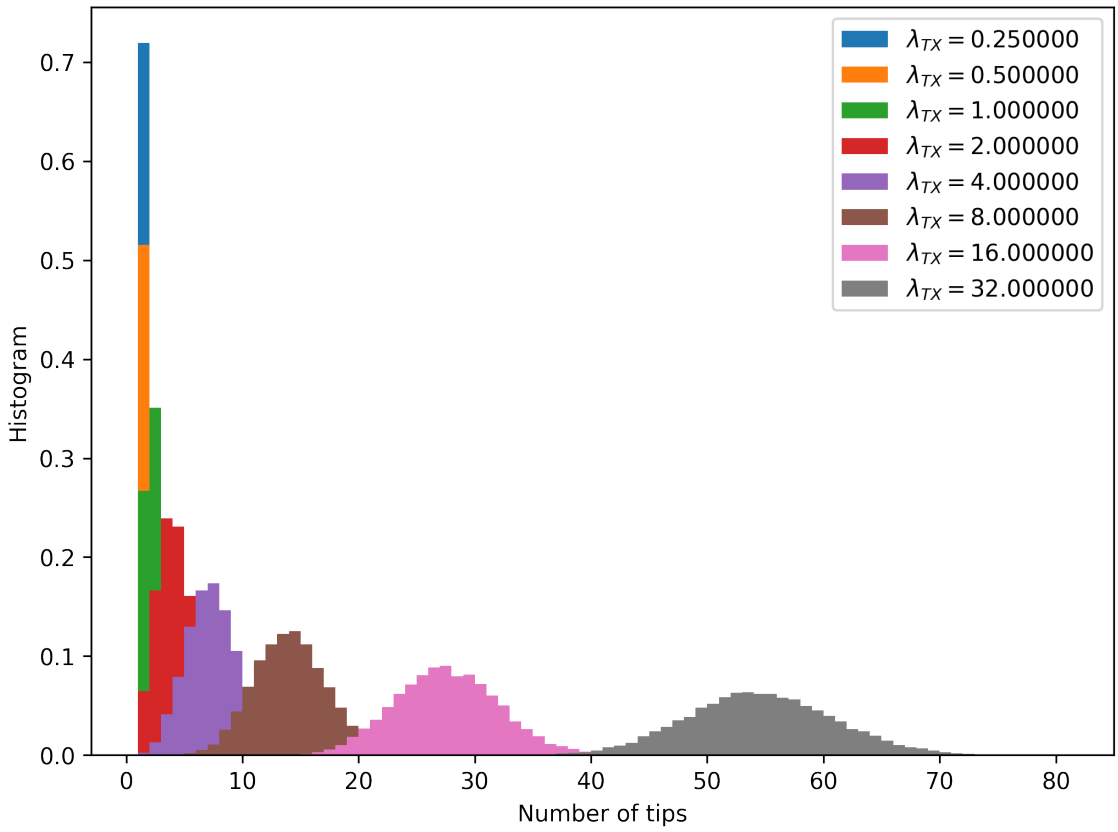
When the system is stable, the average waiting time of a transaction to be confirmed is $\mathbf{E}(w) = \frac{1}{\lambda_{\text{TX}}(z_s - 1)}$.

$m \ll s$ yields $z_s \to 1 + 1/m$. Thus, the average number of unconfirmed transactions $\mathbf{E}(\pi) \to m$ and the average waiting time $\mathbf{E}(w) \to \frac{1}{\lambda_{\text{blocks}}} = \eta = 600$ seconds. In the Bitcoin network, $m \ll s$ is reached when $\lambda_{\text{TX}} \ll 3.75$ tx/s. In other words, when the number of new transactions per second is way smaller than 3.75 tx/s, the average waiting time of a transaction to be confirmed is 600 seconds, which means, on average, all transactions will be confirmed in the next block.

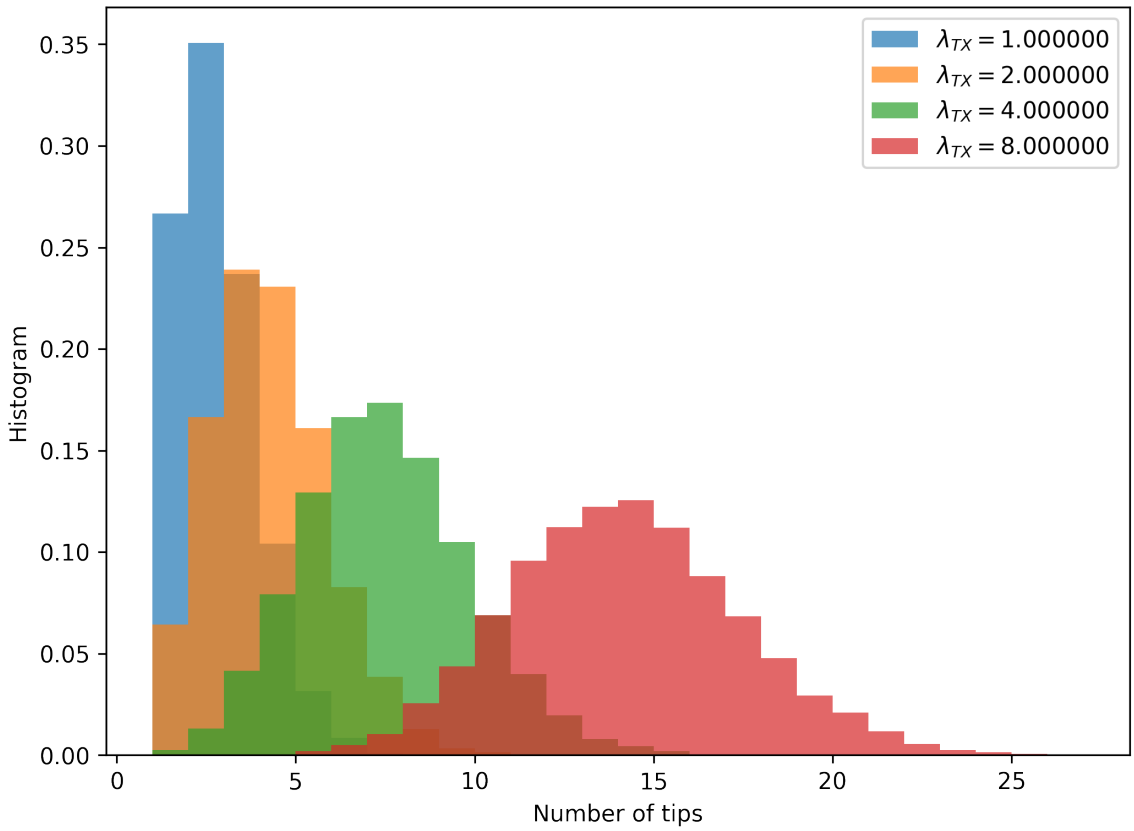
But, $\lambda_{\text{TX}} \to s/\eta$ yields $z_s \to 1$. Hence, $\mathbf{E}(\pi) \to +\infty$, which means the system is going towards instability.

Therefore, we conclude that the Bitcoin network capacity is $\lambda_{\text{blocks}} = s/\eta = s/600$ transactions per second, where s is the average number of transactions per block.

For instance, in order to be a stable system and process 15 transactions per second, each block would have to confirm, on average, 9,000 transactions. Bitcoin's network is really far from this point.

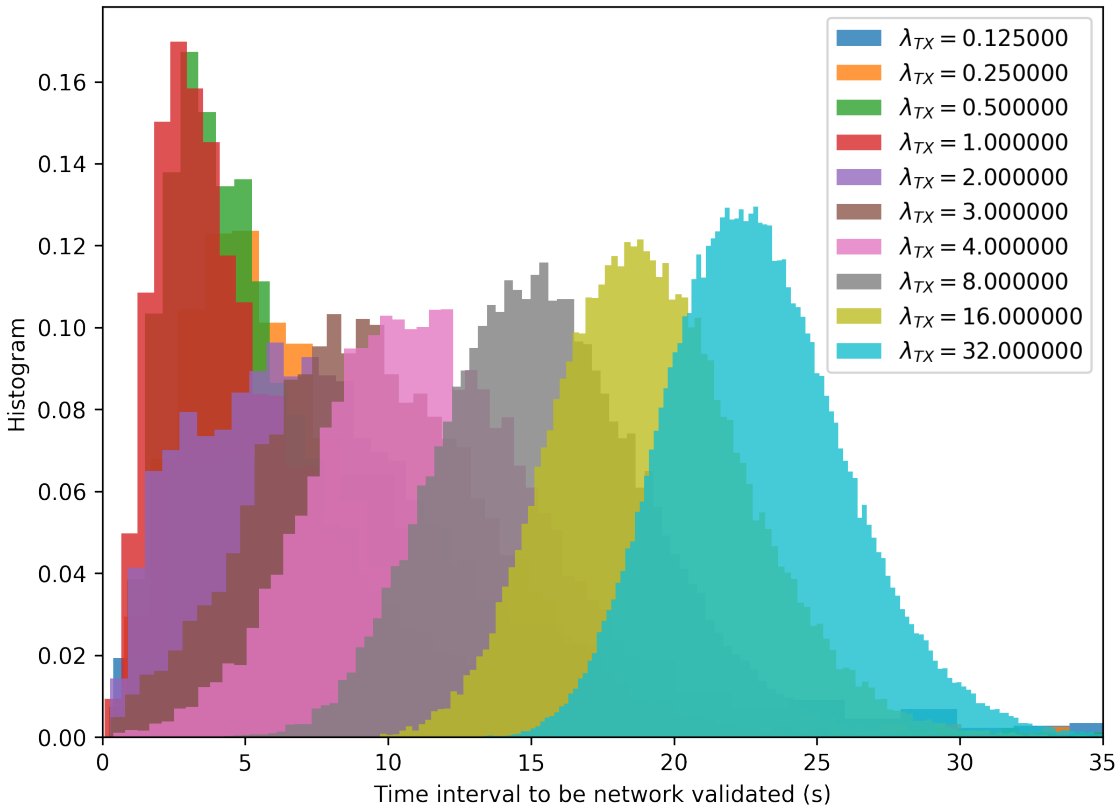


(a) All load scenarios

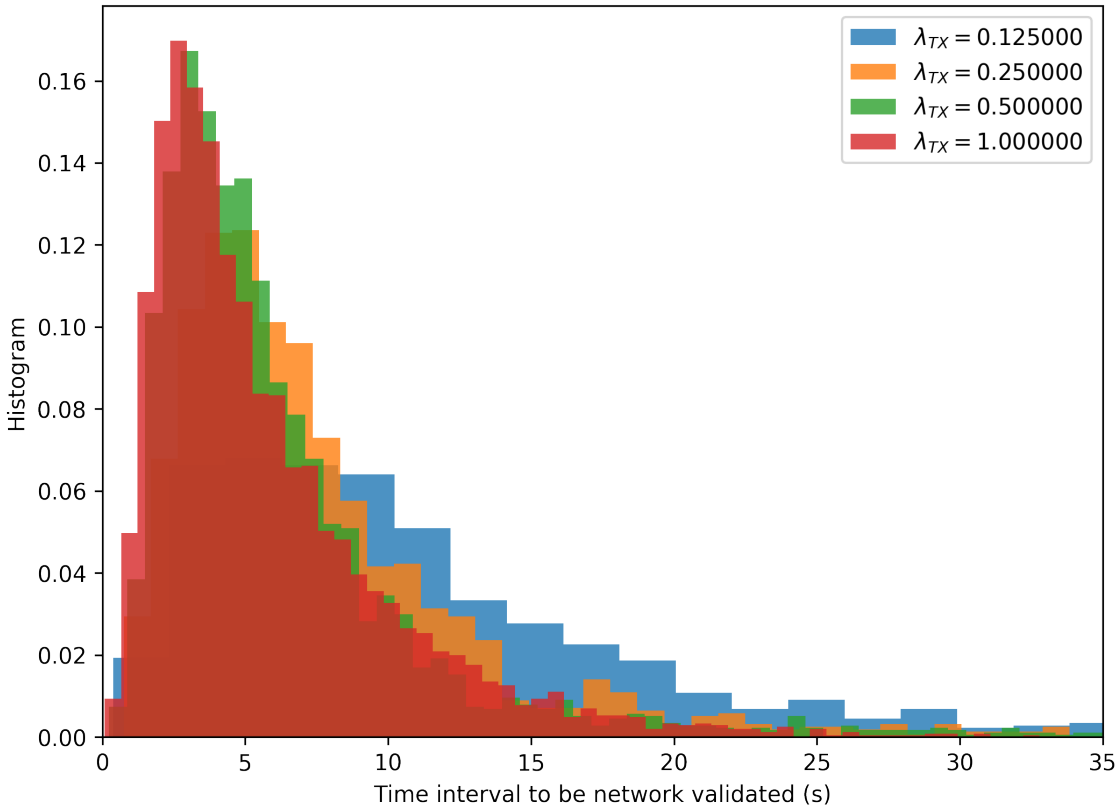


(b) Zoom in the left side of (a)

Figure 9: Histogram of the number of tips for different load scenarios. As expected, the number of tips increases with λ_{TX}.



(a) All load scenarios



(b) Only low load scenarios

Figure 10: Histogram of the time it takes for a transaction to be network validated. A transaction is said to be network validated when all tips are confirming it directly or indirectly.

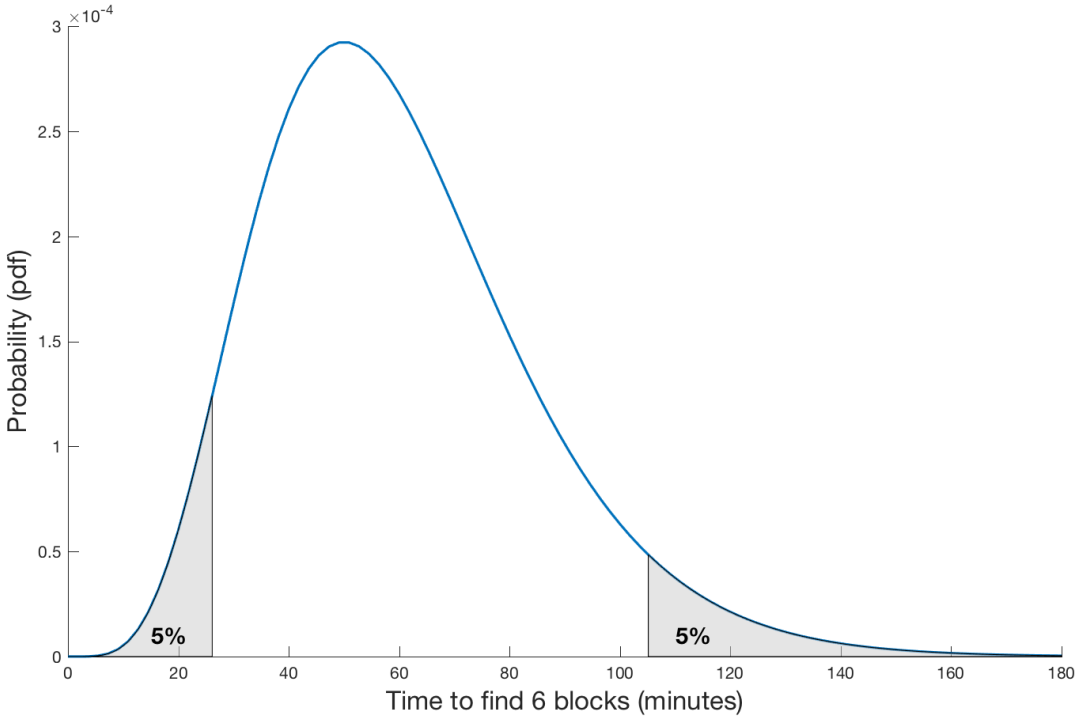


Figure 11: Probability density function of Y_6, i.e., probability of finding 6 blocks after time t. The shaded areas shows the lower 5% and upper 5% of the pdf.

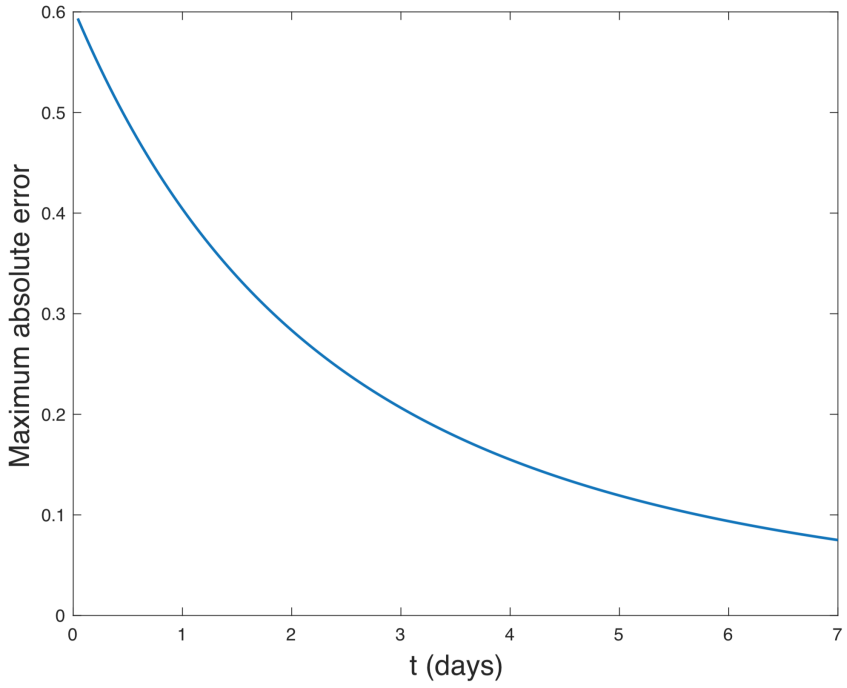
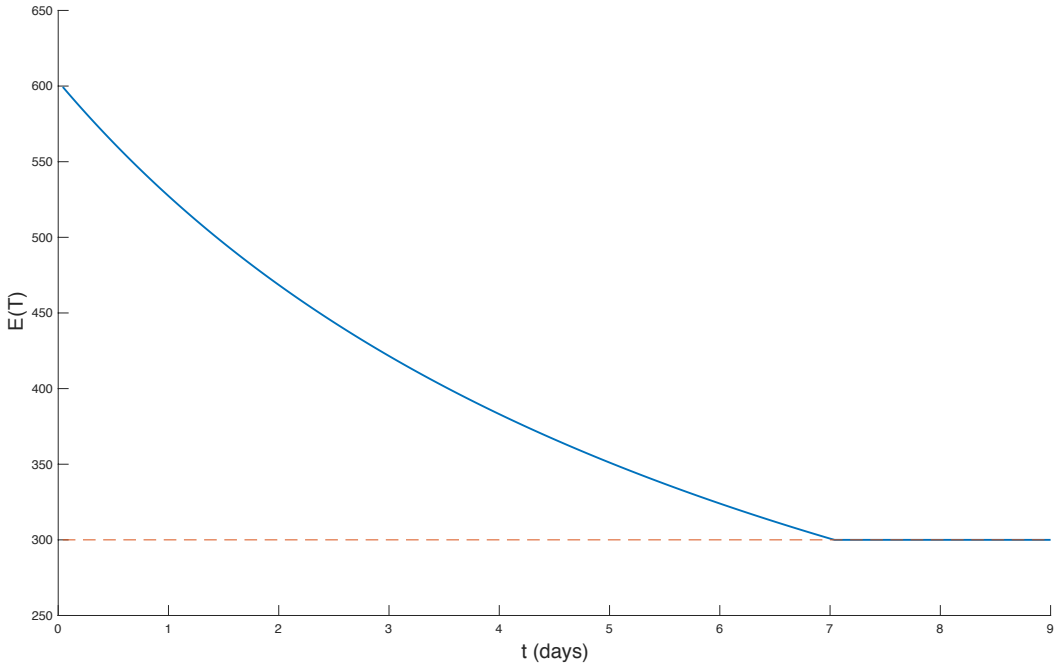
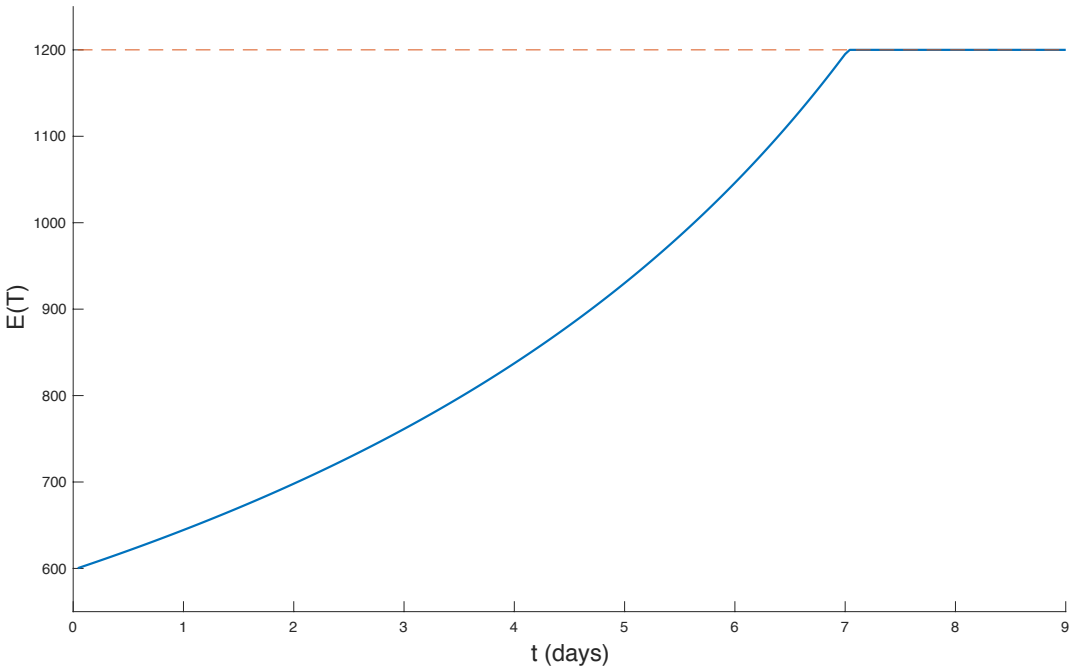


Figure 12: Maximum absolute error between the linear and the suddently change models.



(a) Doubling the hash rate



(b) Halving the hash rate

Figure 13: The average time between blocks when the hash rate changes over time.

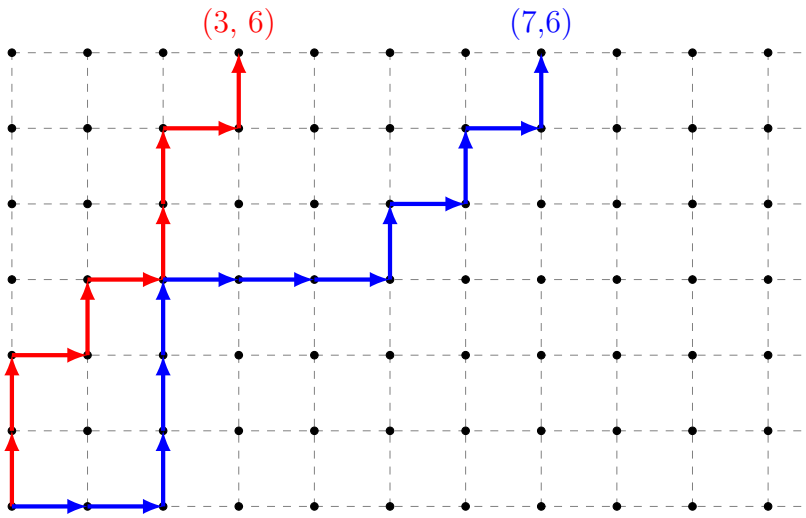


Figure 14: Both the attacker and the network are mining. Each step up is a new block found by the network with probability p. Each step right is a new block found by the attacker with probability $1-p$. It ends when the network finds k blocks — in this example, $k=6$. The red path has probability $p^6(1-p)^3$, while the blue path has probability $p^6(1-p)^7$. Notice that the blue path is a successfull attack, because the attacker has found more blocks than the network. In the red path, the attacker still have to catch up 3 blocks to have a successful attack, which happens with probability ρ^3, if $p < 0.5$.

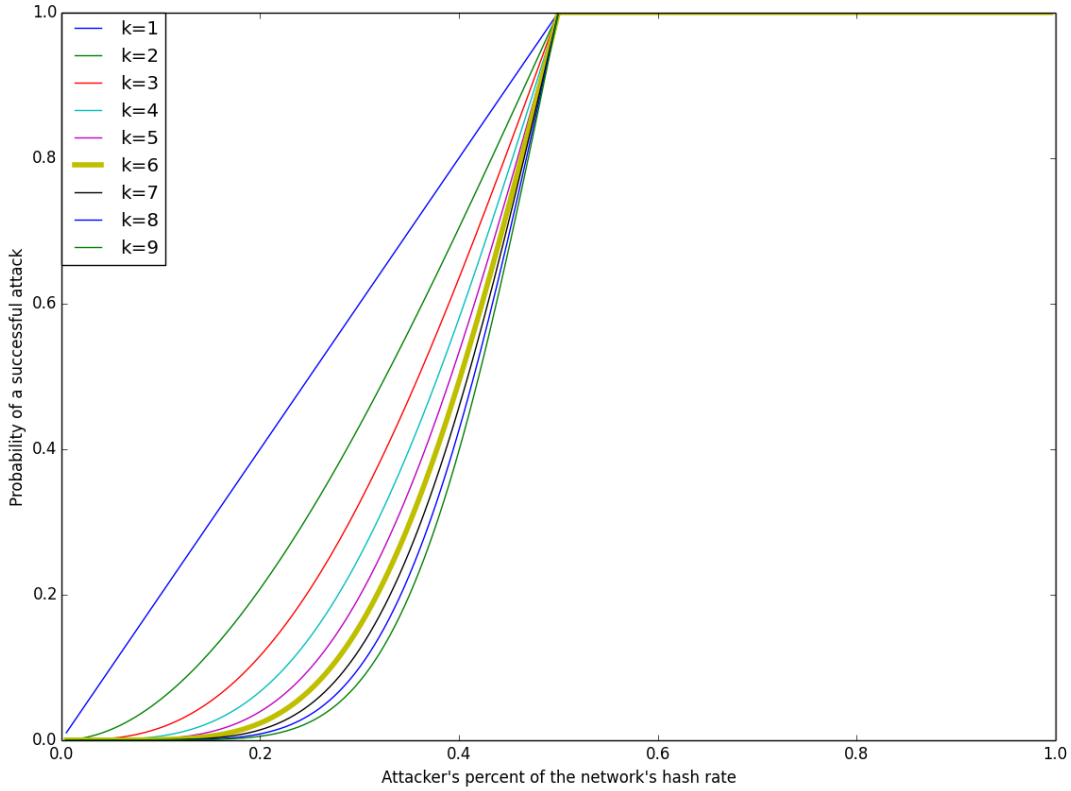


Figure 15: Probability of a successful attack according to the network's hash rate of the attacker (β).