Accelerating Bitcoin's Transaction Processing
Fast Money Grows on Trees, Not Chains

Yonatan Sompolinsky[*]   Aviv Zohar[†]

Abstract

Bitcoin is a potentially disruptive new crypto-currency based on a decentralized open-source protocol which is gradually gaining popularity. Perhaps the most important question that will affect Bitcoin's success, is whether or not it will be able to scale to support the high volume of transactions required from a global currency system.

We investigate the restrictions on the rate of transaction processing in Bitcoin as a function of both the bandwidth available to nodes and the network delay, both of which lower the efficiency of Bitcoin's transaction processing.

The security analysis done by Bitcoin's creator Satoshi Nakamoto [12] assumes that block propagation delays are negligible compared to the time between blocks—an assumption that does not hold when the protocol is required to process transactions at high rates. We improve upon the original analysis and remove this assumption. Using our results, we are able to give bounds on the number of transactions per second the protocol can handle securely. Building on previously published measurements by Decker and Wattenhofer [5], we show these bounds are currently more restrictive by an order of magnitude than the bandwidth needed to stream all transactions. We additionally show how currently planned improvements to the protocol, namely the use of transaction hashes in blocks (instead of complete transaction records), will dramatically alleviate these restrictions.

Finally, we present an easily implementable modification to the way Bitcoin constructs its main data structure, the blockchain, that immensely improves security from attackers, especially when the network operates at high rates. This improvement allows for further increases in the number of transactions processed per second. We show that with our proposed modification, significant speedups can be gained in confirmation time of transactions as well. The block generation rate can be securely increased to more than one block per second – a 600 fold speedup compared to today's rate, while still allowing the network to processes many transactions per second.

1 Introduction

Bitcoin, a potentially disruptive protocol for distributed digital currency, has been slowly gaining traction. Since its initial launch in 2009 by its mysterious developer Satoshi Nakamoto, use of the crypto-currency has been slowly increasing and its value has gone up considerably. As of November 2014, it is valued at over $1000 per bitcoin. Just as any other currency, Bitcoin's value heavily depends on the size of its underlying economy. Several obstacles still lie on the path to wide-spread adoption. These include Bitcoin's unknown regulatory status, and the

[*]School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel, yonatan.sompolinsky@mail.huji.ac.il.

[†]School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel, and Microsoft Research, Israel. avivz@cs.huji.ac.il

relative lack of basic infrastructure that will make it accessible to the masses (this too has been slowly changing as Bitcoin becomes more mainstream). From the computer science perspective however, the main challenge that must be faced is related to Bitcoin's ability to scale to higher transaction rates, and to its ability to quickly process individual transactions. This paper aims to address both of these issues and the connections between them and Bitcoin's security.

As of November 2013, Bitcoin's network processes roughly 60 thousand transactions per day, a number which has been growing, but still amounts to approximately 0.7 transactions per second (TPS) – much lower than Visa's reported scale of approximately 150 million transactions per day (just under 2000 TPS). The relatively low number of transactions is mainly due to Bitcoin's small user-base. Once adoption of the currency increases, the system will need to scale to process transactions at a greater rate than before.

The core idea of the bitcoin protocol is to replace the centralized control of money transmission ordinarily taken up by large organizations such as banks, credit card companies, and other money transmitters, by a large peer-to-peer network. A centrally controlled monetary system is open to intervention: Accounts can be frozen, money can be seized, and fees are high as the channels used for money transfer are controlled by a handful of entities that face little competition. Bitcoin's alternative, is to use the nodes in its large P2P network to verify each other's work and thus ensure that no single entity is able to misbehave. The design of Bitcoin therefore replicates all information at all nodes in the network, allowing each node to become a fully operational part of the money transmission system. This widely replicated data needs to be constantly updated and thus transactions, which are essentially updates to this large database, must be propagated to all nodes in the network. At first glance, this seems to be highly unscalable: every single transaction preformed anywhere in the world is eventually sent to each one of the many nodes in the Bitcoin network. A quick back-of-the-envelope calculation [1] shows that things are still manageable: If bitcoin is to grow to the scale of 2,000 transactions per second worldwide (a rate resembling that of Visa), an internet connection with a bandwidth of approximately 1MB per second would suffice to receive all transactions. This is achieved thanks to the small size of an average transaction which is approximately 0.5KB.[1] In addition to this, all transaction records from Bitcoin's creation are currently saved at every node. Further improvements to the data structures underlying Bitcoin have been proposed so that transactions in the distant past can be safely erased thus reducing storage requirements to a manageable range [4].

The analysis we present in this work shows that there are *additional* limitations on the transaction rates that can be processed by the protocol. These come from the interaction between the delayed propagation of blocks in the network, and bitcoin's security guarantees. We estimate that these restrictions impose a rate of transaction processing that is significantly lower than the limit imposed by the bandwidth required to stream all transactions.

The core of the Bitcoin protocol relies primarily on a data structure called "the blockchain" which essentially encodes a ledger of all transactions from Bitcoin's creation up to the present and is replicated at each node. Each block within the blockchain contains a set of transactions that can be consistently performed based on the state of the ledger represented by the preceding sub-chain. The chain thus forms a serialized record of all accepted transactions. The protocol is configured so that blocks are added to this chain approximately once every 10 minutes. This is accomplished by requiring the node that created the block to solve a computationally difficult problem (essentially generating proof-of-effort [6]). The problem is solved by randomly trying

[1]Arguably, one would need several times this bandwidth to receive and send protocol related messages to *several* neighbors and thus to fully participate in the network.

different inputs, i.e., by brute force (the process of block creation is also called "mining"). Blocks are quickly propagated throughout the entire network, allowing other nodes to build on top of the latest addition to the chain and to include additional transactions. Each block is currently restricted to a size of 1MB. The block size-limit and the block creation rate combined imply a limit on the average number of transactions that can be added to the blockchain per time unit (the current limit is around 3.3 transactions per second).

The effect of delays. Attempting to increase either the block creation rate of the block size may increase the throughput, but both options also adversely affect the protocol to some extent, a fact that has been noted by Decker and Wattenhofer as well [5]. Since the process of block creation is effectively random, it is possible for two blocks to be created simultaneously in the network by two different nodes, each one as a possible addition to the same sub-chain. These two blocks can be consistent with the history, but are mutually conflicting. The Bitcoin protocol ensures that eventually only one of the generated blocks will be accepted by the network. Discarding a block amounts to wasting effort, and this waste is only avoided if blocks are propagated quickly enough through the network so that additional blocks are built on top of them. Any increase in the block size implies that blocks take longer to propagate through the network, and thus many wasted blocks will be built in parallel. In a similar manner, increasing the rate of block creation implies blocks are created more often, and frequently before previous blocks have propagated through the network.

Waiting for transaction confirmation. The main attack that decentralized currency systems must counteract is known as the double-spending attack – in which the attacker attempts to use money in order to pay for a product, and then after some time causes the transaction to be reversed, allowing him to reuse these funds in another transaction. According to Satoshi Nakamoto's original analysis, it is not enough for a transaction to be included in a block for it to be considered irreversible, instead it is only considered secure (with sufficiently high probability) once several blocks have been added to the chain on top of it (the addition of each such block is considered an additional "confirmation"). Therefore, in order to be sufficiently assured that money has been irreversibly transferred, a receiver of funds must await for several confirmations, together taking tens of minutes in expectation. Waiting for a transaction to be included in even a single block takes 10 minutes in expectation and businesses that cannot afford to keep customers waiting this long are forced to accept transactions before any confirmations are generated. This leaves such merchants vulnerable to simple and effective double-spending attacks (see e.g., the Finney Attack [8]). It would therefore be quite useful to shorten the waiting time needed for transaction approval, or even the time needed for a single confirmation.

Indeed, several alternative currencies derived from the Bitcoin protocol have experimented with different, seemingly arbitrary, rates. Examples include Litecoin with a 2.5 minute block creation target, and even Fastcoin that uses 12 seconds. However, as we already noted, this leads to blocks that are wasted due to simultaneous creation, which weakens the security guarantees of the protocol. Satoshi Nakamoto's original analysis [12] heavily depends on the assumption that the time it takes to propagate a block through the bitcoin network is much shorter than the expected time between consecutive blocks. This assumption no longer holds true if we increase the rate of block creation.

Our Contributions:

- We provide and analyze a model of bitcoin's behavior that accounts for delays in the network. We analyze the security of the Bitcoin protocol even when delay is non-negligible.

- We provide estimates and bounds on the limits to which Bitcoin can grow in terms of the rate of transactions processed per second (TPS).

- We analyze and find parameters that optimize the waiting times for transactions while maintaining security guarantees.

Finally, in what is perhaps our main contribution in this paper (presented in Section 8), we propose a protocol modification, namely The Greedy Heaviest-Observed Sub-Tree chain selection rule, that makes use of blocks that are off the main chain to obtain a more secure and scalable system.

With our modification, we demonstrate that high transaction rates no longer compromise bitcoin's security. As a result, we are able to show that a substantial increase in Bitcoin's block generation rate (to around one block per second – a 600 fold speedup compared to today) can be combined with a high transactions throughput, which brings Bitcoin closer to other large scale payment processors in terms of speed of transaction confirmation, and transaction volume. Unlike previous attempts by alternative currencies (e.g., Fastcoin), we are able to increase the rate without suffering from increased susceptibility to 50% attacks.

1.1 Related Work

The original security analysis done by Satoshi [12] has been improved in a whitepaper published by Meni Rosenfeld [16].[2] Several papers have looked at incentive issues related to the operation of the protocol examining issues related to transaction propagation [3], selfish mining [7], and the distribution of rewards within mining-pools [15]. Other work on Bitcoin has looked at its privacy aspects [13, 2], including analysis of its transaction graph [14] which allows to deanonymize some of its users. The Zerocoin protocol has been offered as a way to improve annonimity [11].

Our work deals, among other issues, with enabling fast confirmations for transactions in the network. A paper by Karame *et. al.* discusses similar issues, that relate to possible attacks on nodes that accept zero-confirmation transactions [9]. They suggest several countermeasures that may help avoid such attacks. Their work does not deal with an attack by an adversary with a significant hash-rate, which can compute alternative chains on its own.

A paper closely related to ours is one that was recently published by Decker and Wattenhofer, in which they present a measurement study of message propagation times in the Bitcoin network. They associate delays with the creation of forks in the block tree, and with an increased vulnerability to the 50% attack [5]. As far as we are aware, no other work addresses the issue of Bitcoin's scalability, or its security in a network with delayed block propagation.

2 The Model

2.1 The Block Creation Process

As mentioned above, a block B is completed after a difficult proof-of-work is generated, by essentially random trials. At every trial a different input is fed into a hash function and succeeds with some probability denoted *prob*. A machine that is able to preform many hash computations per second thus performs many trials and generates the proof-of-work with higher probability per time unit. Past attempts at generation do not reveal information that is useful

[2]Many of the papers containing the central ideas on bitcoin and its improvements have only been published as whitepapers, and have not appeared in academic conferences. They have been extensively reviewed and discussed by the Bitcoin community.

for future trials, and so the trials are essentially independent. In practice, machines perform many hashes per second and thus the number of successes per time unit in a given machine is very well approximated by a Poisson process.

The Bitcoin protocol automatically adjusts *prob* in order to react to changes in the total hash-rate of the network, denoted by *hash*, so as to maintain the value of $hash \cdot prob$ constant. Following Satoshi's notation, this constant parameter is denoted by λ, and is to be thought of as the rate of block creation in the network. The current value of this constant, $\frac{1}{600}$ blocks per second, was chosen by Satoshi at Bitcoin's inception, and is hardcoded in the protocol.

2.2 The Network

We model the Bitcoin network as a directed graph $G = (V, E)$. Each node v has some fraction $p_v \geq 0$ of the computational power of the entire network: $\sum_{v \in V} p_v = 1$. Each individual node v in the network thus generates blocks in a Poisson process, with a rate of $p_v \cdot \lambda$, and the entire network combined, generates blocks at a Poisson process with rate λ. Whenever a block is generated by a node, it is immediately sent out to all its neighbors in the network, which continue to propagate the block to their neighbors, until it is eventually known by all nodes.[3] We assume that each edge $e \in E$ has a delay d_e associated with it, which is simply the time it takes for a message to arrive at the other end of the edge.

2.3 The Block Tree

Each block contains in its header the hash of a previous block (typically the most recent one that is known) which we designate as its parent. The hash serves as an identifier of the parent (it is a 256-bit cryptographic hash with negligible probability for collisions). The blocks essentially form a time-developing tree structure that is rooted at the genesis block – the first block created at the moment of Bitcoin's creation; we denote the structure of this tree at time t by $tree(t)$. The depth of block B in the block tree will be denoted, naturally, $depth(B)$. For every block B created by the network, we denote by $time(B)$ its (absolute) creation time, by $owner(B)$, the node which created it. We denote by $parent(B)$ the block upon which B was built, and by $subtree(B)$ the subtree rooted at B.

The structure of the block tree is affected by the blocks that nodes choose to extend. A creator of a new block can specify in its header the hash of any previously known block. The choice of $parent(B)$ therefore depends on the policy of $u = owner(B)$. Formally, this policy is a mapping s_u which maps a block tree $T = (V_T, E_T)$ to a block $B \in V_T$ that is to be the parent of the next block of u. Every node u may posses a different view of the tree (it may not have heard of all created blocks) and thus applies s_u to its currently known tree which we denote $tree_u(t)$; note that $tree_u(t)$ is a subtree of $tree(t)$ with the same root–the genesis block.

The bitcoin protocol currently requires nodes to build new blocks at the end of the longest chain that is known to them. Accordingly, we denote by $longest(t)$ the deepest leaf in $tree(t)$, and by $longest_u(t)$ the deepest leaf in $tree_u(t)$. Unless explicitly stated otherwise, we assume nodes follow this rule. That is, $s_u(tree_u(t)) = longest_u(t)$.

2.4 The Main Chain

The term "main chain", which intuitively amounts to the valid transaction history, corresponds to the path from the genesis block to $longest(t)$. We are particularly interested in the growth

[3]Blocks are only propagated if they have been verified and accepted into the blockchain.

of the main chain. Formally, the time it takes the main chain to advance from length $n-1$ to n is a random variable that we denote as τ_n. We denote $\tau = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \tau_n$, and $\beta = \frac{1}{E[\tau]}$. β is the rate of block addition to the main chain, while λ is the rate of block addition to the block tree.[4]

Another parameter embedded in the protocol is the maximal block size, denoted by b. We assume throughout the paper that there is high demand for transaction processing and that blocks are always full to the limit.

The "efficiency of the network" in including blocks in the main chain, as is represented by the relation between b, λ, and β, is our prime concern in this paper.

Finally, we define the primary measure of Bitcoin's scalability as the number of transactions per second (TPS) the system adds to the history, in expectation.

The TPS is the rate of growth of the main chain, multiplied by the size of blocks, and divided by the average size of a transaction. Thus,

$$TPS(\lambda, b) := \beta(\lambda, b) \cdot b \cdot K$$

where K is the average number of transactions per KB.

3 Susceptibility to Double Spend Attacks

As described above, the protocol includes two parameters, namely, the block creation rate λ and the block size b. A naive attempt to increase the transaction throughput is to simply increase both parameters. As we later show, both changes can result in a less secure protocol, and longer waiting times for transaction authorization. We first briefly explain the double-spend attack and how it is dealt with, and the 50% attack which is its more severe form (the reader is referred to Satoshi Nakamoto's original paper [12] for a more detailed exposition). We go on to formulate the exact requirement which resilience against the 50% attack imposes.

3.1 Fork Resolution and Double-Spending

Since the blockchain, which represents the state of all "accounts", is kept locally at each node, it is imperative that any update will be propagated to the entire network. Nodes which receive a transaction verify its validity (based on cryptographic signatures it contains and on the current status of accounts as represented by their local copy of the blockchain) and send it to all their neighbors. If nodes have identical copies of the blockchain, they will accept and reject transactions in a mutually consistent manner. The danger which is in fact common to many distributed system is that nodes will possess different world-views which may lead to a given transaction being accepted by some of the nodes while a conflicting transaction is accepted by others (a conflicting transaction is essentially one that moves the same bitcoins to a different address).[5]

Transactions are bundled into blocks, which are then propagated through the network (each nodes sends new blocks to its neighbors). Conflicting blocks cannot be accepted together – only one should form the basis for the up-to-date world view of the network. Bitcoin uses two main rules to reduce and eliminate such inconsistencies:

[4]See Theorem 54, Chapter 2 in [17] for the compatibility of these two interpretations of β.

[5]This is in fact a variant of the concensus problem [10] in which nodes in a distributed system must come to agreement about the state of the world.

1. Block creation is made difficult by the requirement to present a proof-of-work for each generated block.
2. Alternative chain branches are accepted by a node if they make up a longer blockchain (or more precisely, if they represent a harder combined proof of work).

The first rule ensures us that blocks are rarely created in the network. Therefore, under conditions of relatively fast block propagation, a block that is created is usually sent to all nodes in the network and is accepted. This greatly reduces conflict. Still, it is possible for two blocks to be created at the same time by far-away nodes in the network. In this case, nodes that adopted one of the versions continue to build their respective main chains, until finally one succeeds in adding a new block on top of the conflicted block. In this case, the tie is broken, as the new block forms a longer chain and is accepted along with it history by all nodes. Ties may in reality last longer, but the race conditions that form ensure us that they will be broken quite quickly, in expectation.

The replacement of the current world-view with an alternative one has far reaching consequences: some transactions may be removed from the current ledger. This fact can be used by an attacker to reverse transactions. The attacker may pay some merchant and then secretly create a blockchain that is longer than that of the network that does not include his payment. By releasing this chain he can trigger a switch that effectively erases the transaction, or redirects the payment elsewhere. This is a difficult undertaking, since the honest nodes usually have a great deal of computational power, and the attacker must get very lucky if he is to replace long chains. The longer the chain, the more difficult it becomes to generate the proof-of-work required to replace it. Satoshi's original security analysis defines a policy for receivers of payments: a transaction is only considered sufficiently irreversible after it was included in a block and some n additional blocks were built on top of it. With this policy, Satoshi shows that the probability of a successful attack can be made arbitrarily low. As a receiver of funds waits for more blocks (larger n), this probability goes down exponentially.

However, if an attacker has more computational power than the rest of the network combined (i.e., it holds at least 50% of the computing power), it is always able to generate blocks faster than the rest of the network and thus to reverse transactions at will (given enough time). This stronger form of attack is known as the 50% attack.

In fact, the assumption that at least 50% of the computational power is required for such an attack to succeed with high probability is inaccurate. If we assume the attacker is centralized and does not suffer from delays, he can beat a network that does suffer from delays using fewer resources. We formulate the exact conditions for safety from this attack, and amend Satoshi's analysis below. We return to the analysis of the weaker double spend attack in Sections 6 and 7.

3.2 The 50 Percent Attack

We begin, as Satoshi has, with the assumption that the attacker has a fraction q of the computational power. Denote by λ_a, λ_h the block creation rate of the attacker and the honest nodes respectively, and by $\lambda = \lambda_a + \lambda_h$ their joint rate. Our assumption is:

$$\lambda_a = q\lambda \quad ; \quad \lambda_h = (1-q)\lambda$$

Note however, that there is no way for honest nodes to differentiate between themselves and the attacker until the attack has actually begun. The attacker may or may not participate in block creation prior to the attack. We therefore denote by λ_{rep} the observed rate of block creation in the system (before the attack).

Proposition 3.1. *If the network's observed block rate is λ_{rep}, for a given block size, we have $\beta = \beta(\lambda)$. Suppose that $\beta(\lambda_{rep}) \geq \frac{q}{1-q}\lambda_{rep}$, then the network is secure against a q-percent attack. Furthermore, an attacker is most effective if he does not participate in block mining before the attack.*

Proof. If a fraction f of the attacker's blocks were included in λ_{rep} prior to the attack, then $\lambda_{rep} = \lambda_h + f \cdot \lambda_a$.

$$\Rightarrow \lambda_h = \lambda_{rep} - f \cdot \lambda_a = \lambda_{rep} - f\frac{q}{1-q}\lambda_h$$

$$\Rightarrow \lambda_h = \frac{\lambda_{rep}}{1 + f\frac{q}{1-q}}$$

We therefore have,

$$\beta(\lambda_h) = \beta\left(\frac{\lambda_{rep}}{1 + f\frac{q}{1-q}}\right) \geq^1 \beta(\lambda_{rep})$$

$$\geq^2 \frac{q}{1-q}\lambda_{rep} \geq^3 \frac{q}{1-q}\lambda_h =^4 \lambda_a,$$

In the above, inequality 1 follows from β's monotonicity, 2 follows from the proposition's assumption, 3 from the fact that λ_{rep} includes the honest network's rate, and 4 from the initial assumption on the attacker's hash-rate.

The attacker's chain thus grows faster than the longest chain in the honest network's tree.

The lower f is, the tighter the first inequality, and the smaller the gap between the rate of network and that of the attacker — making the attack easier to carry out. Therefore, an attacker is most effective when $f = 0$. □

4 Growth of the Longest Chain

Following the above discussion, we wish to analyze how quickly the longest-chain grows, that is, to characterize β's behavior for different block sizes and block creation rates. This is a difficult undertaking for a general network since β highly depends on the structure of the network graph, on the specific delays, and the distribution of hashing power, which are in fact unknown and difficult to measure. We focus on a simple but effective approach to the problem, that will provide us with a lower bound and an upper bound for the estimation of β, which we will be able to use for estimates on the order of magnitude of the TPS without knowledge of the exact network topology.

4.1 Two Nodes and a Link

We begin by analytically deriving the growth rate of the chain in the simplest possible non-trivial case: a graph with only two nodes. This seemingly simple set up turns out to be quite involved, and also yields a useful upper bound for other graphs.

Theorem 4.1. *Let our network consist of only two nodes, u and v, with a block generation rate of $p\lambda$ and $q\lambda$ respectively, and fix the block size b. The delay between u and v is some $d = d(b)$. Then:*

$$\beta(\lambda) = \frac{(p_u\lambda)^2 e^{p_u\lambda 2d} - (p_v\lambda)^2 e^{p_v\lambda 2d}}{p_u\lambda e^{p_u\lambda 2d} - p_v\lambda e^{p_v\lambda 2d}}$$

In the above setting, v and u create blocks separately, and whenever one completes a block it sends the message with its new block through the link, to arrive at its counterpart d seconds later; in these d seconds the node still continues with the attempt to build new blocks and lengthen its own version of the main chain. Thus messages about blocks of the same depth (which were created by u and v roughly at the same time) may simultaneously be traveling in opposite directions on the link.

Proof of Theorem 4.1. In order to count the number of blocks that fail to enter the main chain, we notice that such an event occurs precisely when two blocks of the same height have been created.

Consider a block U of node u. We say that the *window* of U is created d time units before U's creation, and is gone d time units after it. Notice, that a block U is built upon any of v's blocks that was created before U's window was created, and also that block U arrives at node v exactly at the end of U's window.

We say that U is "threatened" at a given time, if U's window has been created, and the chain at v is of length $depth(U) - 1$ (this time interval is contained in U's window). During this period, the next block created by v will be of the same depth as U and one of the blocks is wasted. We define *open* as the time that elapsed from U's window's creation to the moment at which it becomes threatened, and define *close* as the time that elapsed from its window's creation until it ceases to be threatened.

Notice that the closure of U can occur in two ways: either 2d time has passed from the U-window creation, and v received a message containing U, or v generated a competing block of the same height before that. Therefore, the difference between the moment U is opened and the moment it is closed is between 0 and 2d. In addition, notice that two blocks of u cannot be simultaneously threatened (v's chain cannot be shorter by 1 from both their depths at the same time).

Assuming block U_n's window was created at a time that we shall denote as time 0, $open(U_n)$ and $close(U_n)$ are random variables taking values in $[0, 2d]$, for whom we have $close(U_n) \geq open(U_n)$. The distribution of $open(U_n)$ is composed of a continuous part on the region $(0, 2d]$, and a discrete part on the atomic event $\{open(U_n) = 0\}$. We denote the former by $\alpha_n(x)$, for $x \in (0, 2d]$, and the latter by $\alpha_{n,0}$. Similarly, $close(U_n)$'s probability distribution has a continuous part which we denote $\omega_n(x)$ on $[0, 2d)$, and a discrete part $\omega_{n,2d}$ for the atomic event $\{close(U_n) = 2d\}$.[6]

We denote by f_u and f_v the pdf's of the exponential random variables with rates $p_u\lambda$ and $p_v\lambda$, respectively. We claim that the following relations hold:

$$\begin{aligned}\alpha_n(x) \quad &= \quad \int_x^{2d} \omega_{n-1}(y) \cdot f_u(y - x)dy + \\ &\quad\ \omega_{n-1,2d} \cdot f_u(2d - x), \quad 0 < x \leq 2d\end{aligned} \tag{1}$$

$$\begin{aligned}\omega_n(x) \quad &= \quad \int_0^{x} \alpha_n(z) \cdot f_v(x - z)dy + \\ &\quad\ \alpha_{n,0} \cdot f_v(x), \quad 0 \leq x < 2d\end{aligned} \tag{2}$$

[6]We avoided defining the pdf's α_n and ω_n on the entire closed segment $[0, 2d]$, although it can be done by continuity; if defined so, one needs to be careful to distinguish between $\alpha_n(0)$ and $\alpha_{n,0}$ (respectively between $\omega_{n,2d}$, and $\omega_n(2d)$) which are different in essence.

Indeed, starting with Equation 1, U_n opens x seconds after the window creation if and only if for some y, U_{n-1} closed y seconds after its window creation (with probability $\omega_{n-1}(y)$ for $y < 2d$ and $\omega_{n-1,2d}$ for $y = 2d$), and the gap between their respective creations was $y-x$ seconds ($f_u(y-x)$). This calculation is relevant only to $x > 0$, as only under the assumption that U_n's window creation preceded U_{n-1}'s closure the period between U_{n-1}'s opening and closing (y) contains that between U_n's window creation and opening (x).

Regarding Equation 2, U_n closes x seconds after its window creation if and only if for some z, z seconds passed between U_n's window creation and its opening (with probability $\alpha_n(z)$ for $z > 0$ and $\alpha_{n,0}$ for $z = 0$), and $x-z$ seconds between the later and its closing ($f_v(x-z)$). That the gap between the opening and the closing of U_n is controlled by f_v is true only in the region $x < 2d$.

The processes *open*(U_n) and *close*(U_n) are Markovian, and we now write equations 1 and 2 applied to their limiting distributions, $\alpha(x), \alpha_0$ and $\omega(x), \omega_{2d}$:

$$\begin{aligned}\alpha(x) &= \int_x^{2d} \omega(y)\cdot f_u(y-x)dy + \\ &\quad \omega_{2d}\cdot f_u(2d-x), \quad 0 < x \leq 2d \end{aligned} \tag{3}$$

$$\begin{aligned}\omega(x) &= \int_0^{x} \alpha(z)\cdot f_v(x-z)dy + \\ &\quad \alpha_0\cdot f_v(x), \quad 0 \leq x < 2d \end{aligned} \tag{4}$$

These equations resolve to a differential equation system:

$$\left(\begin{array}{c}\alpha\\ \omega\end{array}\right)' = \begin{pmatrix} p_u\lambda & -p_u\lambda \\ p_v\lambda & -p_v\lambda\end{pmatrix}\cdot\left(\begin{array}{c}\alpha\\ \omega\end{array}\right)$$

whose solution is:

$$\left(\begin{array}{c}\alpha(x)\\ \omega(x)\end{array}\right) = \frac{A}{S}\left(\begin{array}{c}p_u\lambda(e^{S\cdot x}-1)\\ p_v\lambda(e^{S\cdot x}-1)\end{array}\right) + \left(\begin{array}{c}\alpha(0)\\ \omega(0)\end{array}\right) \tag{5}$$
$$\text{for } A = \alpha(0)-\omega(0) \quad ; \quad S = p_u\lambda - p_v\lambda.$$

Lemma 4.2. *Equation 5 implies*

$$\omega_{2d} = \frac{p_u\lambda - p_v\lambda}{p_u\lambda - p_v\lambda e^{-(p_u\lambda - p_v\lambda)2d}}.$$

(See the appendix for a proof of the lemma.)

By the definition of ω_{2d}, it is precisely the fraction of u's blocks that have no conflicting blocks created by v. The blocks which contribute to the growth of the main chain are can thus be counted by considering all of v's blocks as valid, and adding to those all of u's unconflicted blocks. Altogether, we obtain

$$\beta(\lambda) = p_v\lambda + \omega_{2d}\cdot p_u\lambda =$$
$$p_v\lambda + \frac{p_u\lambda - p_v\lambda}{p_u\lambda - p_v\lambda e^{-(p_u\lambda - p_v\lambda)2d}}p_u\lambda =$$
$$\frac{(p_u\lambda)^2 e^{p_u\lambda 2d} - (p_v\lambda)^2 e^{p_v\lambda 2d}}{p_u\lambda e^{p_u\lambda 2d} - p_v\lambda e^{p_v\lambda 2d}}.$$

This concludes the proof of Theorem 4.1 □

Notice that the above analysis assumes $p_u \neq p_v$ (many expressions were divided by $p_u\lambda - p_v\lambda$). As the functions involved in the proof are all bounded and continuous, we can take the limit of the expression above and obtain one for the case $p_u = p_v = 0.5$. The resulting expression for β is then (by application of L'Hôpital's rule):

$$\beta = \frac{\lambda(1 + \frac{d\lambda}{2})}{1 + d\lambda}. \tag{6}$$

It should be mentioned that $\frac{\beta}{\lambda}$ here depends only on the parameter $d\lambda$ and not on λ or d independently. The reason for this lies in the fact that increasing d and decreasing λ by the same factor is equivalent to re-scaling the unit of time. This re-scaling does not affect the fraction of un-wasted blocks $\frac{\beta}{\lambda}$.

This model can also be viewed as approximating a larger network in which nodes are partitioned to two large clusters. The delay between the two clusters is large, but internal delay is low. This is an optimistic assumption on the network structure: nodes in the same cluster receive blocks instantly.

4.2 A Lower Bound

We now seek a useful lower bound on the growth rate of the main chain. We assume that the delay diameter of the network is bounded from above by D. We present the following bound:

Lemma 4.3. *Let G=(V,E) be a network graph with delays diameter D. Then the rate at which the longest chain grows is at least $\beta(\lambda, b) = \frac{\lambda}{1+\lambda\cdot D}$*

Proof. We follow a sequence of block creation events for blocks $U_0, U_1, U_2, \ldots$ such that each block U_{i+1} is the first block to be created after D time has passed from the creation of the previous block U_i (i.e., there has been sufficient time to send U_i to all nodes in the network. That is, U_{i+1} is the first block such that $time(U_{i+1}) - D > time(U_i)$. Let us now make the following claim.

Claim 4.4. *Let $U_0, U_1, U_2, \ldots$ be a series of blocks that were created at least D time units apart. Then for all $n \in \mathbb{N}$ $Depth(U_n) - Depth(U_0) \geq n$*

The claim can be proven by induction. It is trivially true for $n = 0$. Now we assume that the claim is true for $n = k$, and show it is true for $n = k + 1$. by $time(U_k)$ we have $Depth(U_k) - Depth(U_0) \geq n$. Then, consider the time at which block U_{k+1} is created. The node that created it has done so after hearing about block U_k, it therefore has a chain that is at least of length k (by the induction assumption and because Chains can only grow or be replaced by longer chains). Therefore U_{k+1} is built at depth that is at least 1 more than U_k.

Now that we have established the claim, we can turn to calculating the lower-bound of β. Denote by $X_i = time(U_i) - time(U_{i-1})$ the random variable of time between block creations. Notice that X_i's are i.i.d. random variables (because the time interval they denote is exactly D time units for the block to spread plus an exponentially distributed waiting time for the next block's creation somewhere in the network). Also note that $\beta \geq E[\frac{1}{n}\sum_{i=1}^{n} X_i]^{-1}$, as the chain grows by at least n during the time $\sum_{i=1}^{n} X_i$. We therefore have $\beta \geq \frac{1}{E[X_1]}$. Additionally, we know that $E[X_1] = D + E[Y]$, where Y is a random variable with an exponential distribution with parameter λ. $E[Y] = \frac{1}{\lambda}$.

$$\beta \geq \frac{1}{D + \frac{1}{\lambda}} = \frac{\lambda}{1 + \lambda \cdot D}$$

□

Lemma 4.3 can be shown to be tight in the sense that for any larger growth rate β' than the bound it provides, it is possible to construct a network graph that grows slower than β'. This is achieved via the clique with n nodes where the delay on all edges is exactly D, and each node has $1/n$'th of the computational power, for sufficiently large n. This lower bound can thus be thought of as approximating a pessimistic assumption on the network, for a given diameter: that of a highly distributed network (with roughly equal division of hash power) in which nodes are equally distanced from each other.

4.3 Delay and the Size of Blocks

As we have already seen, the delay in the network is a highly significant factor that impacts the rate of growth of the main chain, and thus affects both the waiting time for transactions and the transaction rate. We are therefore very interested in estimates of the delay in the network. A measurement study which was recently presented by Decker and Wattenhofer addresses this very issue [5]. They have set up a node on the Bitcoin network that connected to as many accessible nodes as possible. Since each such nodes announces new blocks to its neighbors, it is possible to record these events and estimate the time it takes blocks to propagate.

Their findings include the existence of a strong correlation between the size of the block being sent and the time it took to send the block. In particular, there is a minimal delay experienced by even the smallest blocks, and an additional delay that each KB adds linearly. This is very simply explained by the combination of the constant propagation delay for blocks over a link, and the transmission delay which is due to bandwidth restrictions and block verification times which is roughly linear in the block size. Both of these types of delays accumulate as blocks are propagated across several hops resulting in long delays to reach the entire network.

Figure 1, which is based on raw data that Decker and Wattenhoffer have generously shared with us, depicts this linear effect quite clearly.

The interesting point is that the linear dependence on the block size, which is characteristic of a single link, also holds in aggregate for the entire network. Following this observation, we adopt a linear model of the delay:

$$D_{50\%}(b) = D_{prop} + D_{bw} \cdot b \tag{7}$$

The time it takes to get to 50% of the network's nodes is quite accurately described by the best fit of such a linear relation to the data. The fit parameters are: $D_{prop} = 1.80$ seconds, and $D_{bw} = 0.066$ seconds per KB. Notice that D_{prop} is a measure of aggregate propagation delay, and D_{bw} is an aggregate measure in units of seconds per KB.

We make use of this linear relation to extrapolate the delay of block propagation for different block sizes, and specifically use the parameters of the best fit to estimate the time it takes to reach 50% of the nodes. We also assume that reaching 50% of the nodes is equivalent to reaching 50% of the total hash power in the network.[7]

5 Optimizations

In the previous section we explored two specific network models inducing different behaviors of β as a function of λ and b. Equipped with these models, we now proceed to the task of analyzing and bounding the TPS.

[7]The usual caveats apply: the measurements represent the state of the network at a given time, and changes have likely occurred. Additionally, there is no exact measurement of the block creation time; it is only estimated by the first message about the block, which in fact implies delays are longer than shown above.

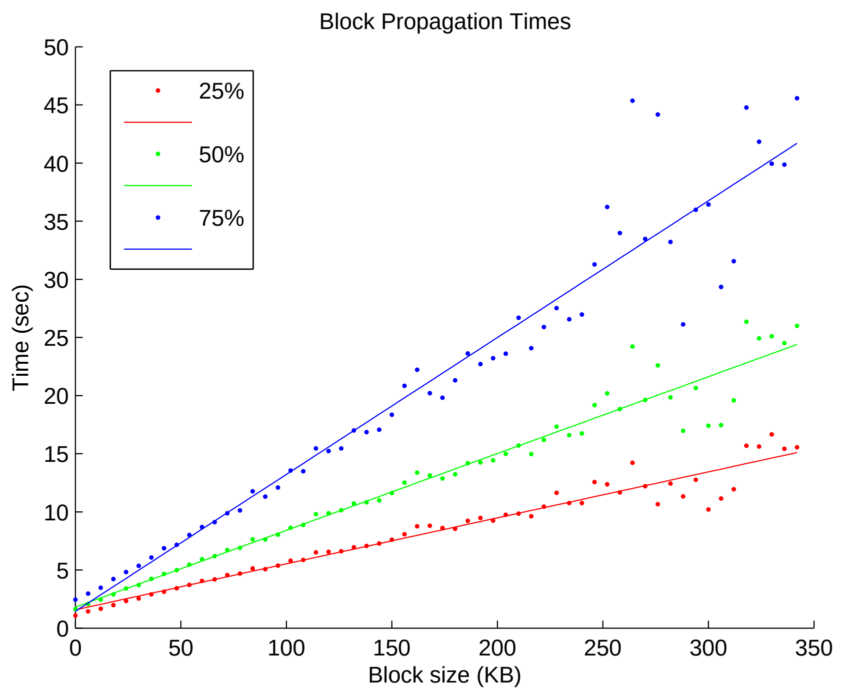

Figure 1: The relation between the block size and the time it took to reach 25% (red), 50% (green), and 75% (blue) of monitored nodes, based on data provided by Decker and Wattenhofer [5].

5.1 An Optimistic Estimate of the Achievable TPS

We begin by utilizing our result for the 2-node network, along with the assumption on the linearity of delay. We assume here that the network is well approximated by two clusters that have very low delays internally.

We consider the case $p_u = p_v = 0.5$, with the corresponding expression for the growth rate taken from equation 6:

$$\beta(\lambda, b) = \frac{\lambda(1 + \frac{(d_{prop} + d_{bw} \cdot b)\lambda}{2})}{1 + (d_{prop} + d_{bw} \cdot b)\lambda}.$$

Recall the security constraint in Proposition 3.1, which is required for resilience to a 50% attack. When $\frac{1}{2} < q$ the attacker has more hash-rate than that of the two nodes combined and we cannot be resilient against his attacks. On the other hand, when $q < \frac{1}{3}$, the attacker has less hash-rate than each node separately, and thus every node alone is enough to resist the 50% attack, regardless of the values of λ and b (as a node suffers no internal delay). We are thus interested in the range $q \in (\frac{1}{3}, \frac{1}{2})$, in which the constraint on β is non-trivial:

$$\frac{\lambda(1 + \frac{d(b) \cdot \lambda}{2})}{1 + d(b) \cdot \lambda} \geq \frac{q}{1-q}\lambda \iff d(b) \cdot \lambda \leq \frac{2 - 4q}{3q - 1}.$$

It is easy to see that this single constraint is satisfied as an equality when maximizing the

TPS, which induces the following relation between λ and b:

$$\lambda(b) = \frac{2-4q}{3q-1}\frac{1}{d(b)} = \frac{2-4q}{3q-1}\frac{1}{d_{prop} + d_{bw} \cdot b}.$$

We differentiate the TPS to find the optimal b:

$$\frac{\partial TPS}{\partial b} = \frac{\partial}{\partial b}\left(b \cdot \beta(\lambda, b) \cdot K\right) = \frac{\partial}{\partial b}(b\frac{q}{1-q}\lambda(b)K) =$$
$$\frac{q}{1-q}\frac{2-4q}{3q-1}\frac{d_{prop}}{(d_{prop} + d_{bw} \cdot b)^2}K > 0.$$

Thus, a limitless increase of b, and a corresponding adjustment of $\lambda(b)$, yields the optimal TPS, whose value is then upper bounded by:

$$\lim_{b\to\infty}\left(b\frac{q}{1-q} \cdot \frac{2-4q}{3q-1} \cdot \frac{1}{d_{prop} + d_{bw} \cdot b}K\right) =$$
$$\frac{q}{1-q} \cdot \frac{2-4q}{3q-1} \cdot \frac{K}{d_{bw}}.$$

This bound can be applied as an optimistic assumption on the network structure in the following way: We have seen that under the conditions measured by Decker and Wattenhofer, the time it takes a block to reach 50% of the nodes is about $d_{50\%}(b) = 1.80 + 0.066 \cdot b$ seconds. To give an upper bound, we assume 50% of the network hears about a block immediately at its creation, while the rest 50% hear about it after $d_{50\%}(b)$ seconds. This optimistic assumption on the network's structure admits to a treatment in our two node model with $p_u = p_v = 0.5$. We can thus apply the above upper bound with $d_{bw} = 0.066$.

e.g., for $K = 2$ (transactions per KB) and $q = 0.4$ we obtain $TPS < 40.4$ transactions per second. Notice that this value of 40.4 TPS is two orders of magnitude lower than the restriction imposed by a bandwidth of 1MB per second, even though we have taken a very optimistic assumption on the structure. The bound above also allows us to consider the gains we will get once transaction hashes are encoded into the block instead of the entire transaction records (a change which is currently planned by Bitcoin's core developers). In this case, transaction hashes, which are only 32 bytes each, make up blocks that are smaller by a factor of 16. The bound on TPS for the corresponding value of $K = 32$ is also 16 times higher, and an entire order of magnitude is gained: $TPS < 646.4$.[8]

5.2 A Pessimistic Estimate of the Achievable TPS

A pessimistic estimate on the achievable TPS for a network with a given delay diameter $D(b)$ can easily be obtained by applying Lemma 4.3. However, the delay diameter of the Bitcoin network is unknown, and, moreover, by its strict definition, it can be affected by extremely negligible distant nodes. It is thus wiser to analyze the "truncated" network, that is, to look at a certain percentage of the network and apply Lemma 4.3's lower bound on the growth rate of this faction of the network. The rest of the network can of course only increase the growth rate.

[8]here we assume that nodes propagate blocks even if transaction hashes represent unknown transactions, which later arrive. This is similar to the pipelining idea that Decker and Wattenhofer discuss in [5].

We thus look only at a fraction of x of the network, and turn to the corresponding security constraint from Proposition 3.1:

$$\beta(\lambda, b) = \frac{x\lambda}{1 + D(b) \cdot x\lambda} \geq \frac{q}{1-q}\lambda$$

$$\iff \lambda \leq \frac{\frac{1}{q} - (1 + \frac{1}{x})}{D_{prop} + D_{bw} \cdot b},$$

where the delay constants D_{prop} and D_{bw} depend on the fraction of the network we wish to arrive at, namely, x.

This inequality becomes an equality when maximizing the TPS, and thus

$$\lambda(b) = \frac{\frac{1}{q} - (1 + \frac{1}{x})}{D_x(b)} = \frac{\frac{1}{q} - (1 + \frac{1}{x})}{D_{prop} + D_{bw} \cdot b}.$$

The achievable TPS is then bounded from below by

$$TPS(\lambda, b) = b \cdot \beta(\lambda, b) \cdot K \geq$$

$$b\frac{q}{1-q}\lambda(b)K = b\frac{q}{1-q}\frac{\frac{1}{q} - (1 + \frac{1}{x})}{D_{prop} + D_{bw} \cdot b}K.$$

Here, again, limitlessly increasing b maximizes the TPS, and thus *the optimal* TPS's value is lower bounded by

$$\frac{q}{1-q}\frac{\frac{1}{q} - (1 + \frac{1}{x})}{D_{bw}}K.$$

For instance, when $x = 0.5$ we have $D_{bw} = 0.066$ seconds, and taking $K = 2$ and $q = 0.25$ in order to be safe from a 25% attack, we obtain $TPS > 10.1$ transactions per second.

Again, once transactions are more compactly encoded in blocks (in the form of hashes setting $K = 32$), this estimate grows by a factor of 16 to $TPS = 161.6$

Notice that while this bound is not too low, it is obtained in exchange for a greater security risk than the upper bound we showed perviously (we guarantee resilience only to a 25% attack in this case). This unsatisfying large gap between the two network models comes from the large difference between the pessimistic and optimistic assumptions. This difference is not accidental: There is indeed a wide possible variety of underlying network structures which yield different performance levels. Due to the difficulty in learning the exact structure, setting the parameters correctly is indeed a difficult undertaking. We provide better guarantees of security in our improved protocol in Section 8.

6 Security in Networks with Delay

We have so far considered only the effect that delayed block propagation has on the 50% attack. The implications also naturally extend to attackers with a smaller fraction of the computational resources; Even an attacker with a modest hash-rate can still succeed in a double-spending attack if it is lucky enough to generate many blocks in a quick burst. Satoshi, in his original paper, asserts the security of Bitcoin by demonstrating the very low probability of events that lead to such successful attacks. We stress, however, that Satoshi's assumption was that block propagation times are negligible compared to the time it takes to generate them. Removing this assumption complicates matters, as the production of blocks in a general network with delays is no longer a Poisson process. In particular, the intervals between successive lengthening events of the longest chain may not be independent.

Our main result in this section shows that (under some reasonable assumptions) networks in which the main chain grows at a rate of β, are at least as secure as a network with no delays that produces blocks at the same rate β. This provides a practical approach to security: measure the growth rate of the main chain empirically (a process which is already occurring in the Bitcoin protocol in order to adjust the difficulty level of the proof of work), and then simply apply the standard security analysis (e.g., from [16]) with this measured rate instead of the block creation rate λ.

A key piece in the security analysis, is bounding the probability that an attacker will be able to generate more blocks than those added to the network's longest chain, given that the attacker starts at a disadvantage of X_0 blocks. In Satoshi's analysis, if the network's block generation rate is λ_h, and the attacker's is γ, this probability is *exactly* $\left(\frac{\gamma}{\lambda_h}\right)^{X_0+1}$. The theorem below shows us that we need only substitute β for λ_h in this expression, and what we get is an upper bound on the probability of successful attacks. In other words, the network with delays, is just as secure (if not more secure) than a network with no delays and rate β. This result is somewhat surprising, given the potentially complex behavior of the main chain's growth.

Theorem 6.1. *Consider a network G with delays. Let $1/\tilde{\beta}$ be an upper-bound on the expected waiting time for the next lengthening of the main chain, given* any history. *Let $\gamma \leq \tilde{\beta}$ be the block creation rate of the attacker (according to a Poisson process), and suppose the network's longest chain is longer than the attacker's by X_0 blocks. Then the probability that the attacker will succeed in extending his chain to be longer than the network's is at most $\left(\frac{\gamma}{\tilde{\beta}}\right)^{X_0+1}$.*

The bound in Theorem 6.1 is later combined with a distribution over such possible gaps to give the full security analysis (see e.g., [16], or alternatively, our treatment in Section 7).

The proof of this theorem, which we defer to the appendix, makes use of tools from martingale theory, and is quite involved. A central key in it is the following lemma which (intuitively) is used to show that the waiting time in networks is less "bursty" than a Poisson process with the same rate.

Lemma 6.2. *Let ς be a random variable with increasing hazard rate function. Then, $\forall k \in \mathbb{N}, \quad E[\varsigma^k] \leq k!E^k[\varsigma]$.*

Note that the almost inverse inequality, $E[\varsigma^k] \geq E^k[\varsigma]$, stems from Jensen's inequality.

Although the condition in the Theorem 6.1 is realistic for most networks, it still is not the most general case. We finish this section by suggesting that these results are valid to all networks with any configuration:

Conjecture 6.3. *If the longest chain grows at a rate of $\tilde{\beta}$ and the attacker's rate is some $\gamma < \tilde{\beta}$, the probability that an attacker will catch up is upper bounded by $\left(\frac{\gamma}{\tilde{\beta}}\right)^{X_0+1}$.*

7 Waiting Time Optimization

Given that there is always some probability that an attack by a miner working off the main chain will succeed, a transaction receiver needs to decide when to consider a transaction *sufficiently* irreversible and release the good or service. His policy can be described as a function $n(t, r, q, \beta)$, where r is the error he allows, $q \leq \frac{1}{2}$ is the upper bound on the fraction of the hash-rate controlled by the attacker, β is the rate of growth of the main chain, and t the time that elapsed since the transaction was broadcast to the network. If the transaction receiver observes n blocks (confirmations) atop his transaction by time t, he approves it only if $n \geq n(t, r, q, \beta)$, and otherwise waits for n to increase. [9]

An acceptance policy ought to take into account the ratio $\frac{q\lambda}{\beta}$ which is the parameter determining how good the attacker is relative to the network. Notice that for any $k \geq 0$, $\beta(k\lambda) \leq k\beta(\lambda)$, since every increase in the block creation rate is only partially translated to an increase in the growth rate, due to the waste in conflicting blocks, (an equality holds for a network with no delays). Therefore, the parameter $\frac{q\lambda}{\beta}$ increases as a function of λ, making the attack easier.

Therefore there exists a simple trade-off in the waiting time for transaction approval between high rates and low ones: a high rate makes an attack easier, and so a larger number of blocks needs to pile on top of a transaction to gain sufficient irreversibility. On the other hand, these very blocks arrive faster. We seek an optimal λ, that minimizes the expected waiting time for the inequality $n \geq n(t, r, q, \beta)$ to be satisfied, where $\beta = \beta(\lambda, b)$.

We now present the client-policy. If t seconds have passed since the transaction was received, the probability that the attacker has completed k blocks is $\zeta_k := e^{-q\lambda t} \frac{(q\lambda t)^k}{k!}$. If the honest network completed n blocks within this period, the attack surely succeeds for $n < k$, and for $n \geq k$ succeeds with probability upper bounded by $\left(\frac{q\lambda}{\beta}\right)^{n-k+1}$ (Theorem 6.1). This justifies the following acceptance policy:

$$n(t, r, q, \beta) := \min_n \left\{ \sum_{k=0}^{n} \zeta_k \cdot \left(\frac{q\lambda}{\beta}\right)^{n-k+1} + \sum_{k=n+1}^{\infty} \zeta_k \leq r \right\}$$

The behavior of β as a function of λ, b is subject to the network structure and to the specific delays on each link. We take the lower bound model described in Subsection 4.2, $\beta = \frac{x\lambda}{1+D\cdot x\lambda}$. To illustrate the possible speed up, we assume every block arrives at 50% of the nodes within 4 seconds (which roughly correlates to the propagation time of small blocks containing only transaction hashes, at around 32KB). To give a more concrete number, we numerically compute the optimal block creation rate when defending from an attacker with 10% of the hash-rate, and allowing for a 1% success probability of an attack. The optimal block rate in this case is: $\lambda_{opt} \approx 0.29$ blocks per second, corresponding to an expected waiting time of ≈ 56 seconds.

These results demonstrate a significant speed up in authorization compared to the current block creation rate (of 10 minutes per block). It is important to note, however, that these gains highly depend on the specific profile of the attacker. In fact, in some network configurations that match the assumptions above, an attacker with just over 24% of the hash-rate can successfully execute a so-called 50% attack, i.e., to replace the main chain at will. This should be taken into account when evaluating the security of alternative coins (such as Fastcoin with its rate of 12 seconds per block). In the next section, we present a protocol modification that completely removes this flaw.

[9]Previous work, such as [12, 16] considered simpler policies that accepted transactions based on the number of confirmations alone and did not take time into account. We make use of this added information, which also gives some minor improvements in waiting times.

8 The Greedy Heaviest-Observed Sub-Tree (GHOST)

In this section we present our main contribution to the protocol: a new policy for the selection of the main chain in the block tree. The advantage of this suggested change to the protocol, is that it maintains the threshold of hash power needed to successfully pull off a 50% attack at 50%, even if the network suffers from extreme delays and the attacker does not. This allows the protocol designer to set high block creation rates and large block sizes without the fear of approaching the 50%-attack's cliff edge, which in turn implies that high confirmation rates and a high transaction throughput can both be maintained.

The basic observation behind the protocol modification that we suggest, is that blocks that are off the main chain can still contribute to a chain's irreversibility. Consider for example a block B, and two blocks that were created on top of it C_1 and C_2, i.e., $parent(C_1) = parent(C_2) = B$. The Bitcoin protocol, at its current form, will eventually adopt only one of the sub-chains rooted at C_1 and C_2, and will discard the other. Note however, that both blocks were created by nodes that have accepted block B and its entire history as correct. The heaviest sub-tree protocol we suggest makes use of this fact, and adds additional weight to block B, helping to ensure that it will be part of the main chain.

Recall our definition from Section 2.3; node u chooses the *parent* of its blocks according to a policy $s_u(T)$, which maps a tree T to a block in T, which essentially represents the main chain u has accepted. Formally, our new protocol is a new *parent* selection policy. This new policy redefines the main chain, which is what should be regarded as the valid branch of transaction history.

For a block B in a block tree T, let $subtree(B)$ be the subtree rooted at B, and let $Children_T(B)$ be the set of blocks directly referencing B as their parent. Finally, denote by $GHOST(T)$ the parent selection policy we propose, defined as the output of the following algorithm.

Algorithm 1. *Greedy Heaviest-Observed Sub-Tree (GHOST).*
Input: Block tree T.

```
1. set $B \leftarrow$ Genesis Block
2. if $Children_T(B) = \emptyset$ then return(B) and exit.
3. else update $B \leftarrow \operatorname*{argmax}_{C \in Children_T(B)} \{\#subtree_T(C)\}$[10]
4. goto line 2
```

The algorithm follows a path from the root of the tree (the genesis block) and chooses at each fork the block leading to the heaviest subtree. For instance, if the blocks C_1 and C_2 mentioned earlier have children D_1 and D_2 respectively, then $\#subtree_T(B) = 5$. Consequently, an alternative chain of length 4 rooted at $parent(B)$, will not override block B, despite the fact that it is longer than any of the branches extending B. This makes forks in the block tree above B of no consequence to its secureness — every addition of a block to $subtree(B)$ makes it harder to omit B from the main chain, even if it does not extend $subtree(B)$'s height.

Since the $GHOST$ protocol exploits the work invested in conflicting blocks to enhance their ancestor's security, it is useful to know how quickly a given block is considered in the main chain of all nodes in the network. We discuss this, and other basic properties of the mechanism below.

[10]We are in fact interested in the sub-tree with the hardest combined proof-of-work, but for the sake of conciseness, we write the size of the subtree instead.

8.1 Basic Properties of GHOST

Given that *GHOST* is a new chain selection rule, it is imperative to first show that all nodes eventually adopt the same history when following it. For every block B define by ψ_B the earliest moment at which it was either abandoned by all nodes, or adopted by them all. We call the adoption of a block by all nodes the *collapse* of the fork to its subtree. In addition, denote $n_B := \#subtree_T(B)$ for $T = tree(\psi_B)$; n_B is the size of $subtree(B)$ at time ψ_B.

Proposition 8.1 (The Convergence of History)**.** $Pr(\psi_B < \infty) = 1$. *In other words, every block is eventually either fully abandoned or fully adopted.*

To prove the proposition, we make use of the following claim.

Claim 8.2. $E[\psi_B] < \infty$

Proof. Let D be the delay diameter of the network. Assume that at time $t > time(B)$, block B is neither adopted by all nodes, nor abandoned by all of them. Denote by $\mathcal{E}_t$ the event in which the next block creation in the system occurs between times $t + D$ and $t + 2D$, and then no other block is produced until time $t + 3D$. We argue that once such an event occurs, block B is either adopted or abandoned by all nodes. Between time t and $t + D$, all nodes learn of all existing blocks (as no new ones are manufactured), each pair of leaves of the block tree that have nodes actively trying to extend them must have equal weight subtrees rooted at some common ancestor. A single block is then created which breaks these ties, and another D time units allow this block to propagate to all nodes, which causes them to switch to a single shared history. Notice that $Pr(\mathcal{E}_t)$ is uniformly (in t) lower bounded by a positive number. Hence the expected waiting time for the first event is finite, implying $E[\psi_B] < \infty$. (See "Awaiting the almost inevitable" in [18], Chapter 10.11). □

Proof of Proposition 8.1. If $\psi_B = \infty$ then there are always new blocks added to $subtree(B)$ (as well as to conflicting subtrees), and so the size of $subtree(B)$ before the collapse (which never actually occurs, in this case) is infinite. This must occur with probability 0, as by Claim 8.2 we know that $E[n_B]$ is finite. □

We now show that the *GHOST* chain selection rule is resilient to 50% attacks, even at high rates (in contrast to the longest-chain rule which, as we have shown, deteriorates in such cases).

Proposition 8.3 (Resilience from 50% attacks)**.** *If* $\lambda_h > \lambda_a$*, then for any* $r > 0$*, there exists* $\tau \in \mathbb{R}$*, such that, if* $T = tree(time(B) + \tau)$*,* $GHOST(T) \in subtree_T(B)$*, then the probability that* B *will ever be out of the main chain after* $(time(B) + \tau)$*, is smaller than* r*.*

Contrast the statement above with the one in Proposition 3.1, where the attacker's rate needs only to exceed $\beta(\lambda, b)$ which may be significantly smaller than λ_h.

Proof of Proposition 8.3. The event in which B is eventually discarded from the main chain is contained in the event that a collapse has yet to occur (i.e., $\psi_B \geq time(B) + \tau$). Relying again on the finiteness of $E[\psi_B]$ (Proposition 8.2), it follows that $\exists \tau \in \mathbb{R}$, after which we have either abandoned B, or all nodes adopted it. In the former case, the proposition holds trivially. In the latter case, blocks are now built in B's subtree at the rate of λ_h, which is higher than λ_a. Thus, if we wait long enough, the honest subtree above B will be larger than the one constructed by the attacker, with sufficiently high probability. □

8.2 Growth of The Main Chain

Next, we seek a lower bound for the rate of growth of the main chain in networks with delay diameter D, using the *GHOST* chain selection rule. As the protocol no longer selects the longest chain, it can be expected that the rate of growth will be somewhat slower than in the longest-chain rule. Indeed, the lower bound is somewhat smaller than that of the longest-chain rule:

Proposition 8.4. *Let D be the delay diameter of a network which generates blocks at the rate of λ. When all nodes follow the GHOST rule, the main chain grows at a rate β lower bounded by $\beta \geq \frac{\lambda}{1+2D\lambda}$.*

(Compare with Lemma 4.3)
The Lemma follows as an immediate consequence of the following claim:

Claim 8.5. *Let B be a block in tree T, then regardless of history, the expected waiting time for the creation of the last child of B is upper bounded by: $2D + \frac{1}{\lambda}$.*

Proof. Let C be the first block created after D seconds have passed from B's creation. Denote by τ the time from B's creation until C has been created and yet another D seconds elapsed. We argue that $E[\tau] \leq 2D + 1/\lambda$. This is easy to see: It takes $1/\lambda$ seconds in expectation to create block C, an event which can only occur after D seconds have passed from B's creation. Then, we deterministically wait another D seconds to propagate C to the entire network.

We claim that after τ seconds from B's creation, B will have no more children. Let us examine the two possible cases:
Case I: C is a descendant of B. Once C has been propagated to all nodes, no node considers B a leaf, and the *GHOST* chain selection rule only extends leaves (in the subtree known to the extending node).
Case II: C is not a descendant of B. Because B was propagated to all nodes before C was created, the node that extended C was well aware of B, but did not extend it. It therefore had a strictly heavier sub-tree than B is part of after the creation of C. D seconds later, block C is known to all other nodes, along with its entire supporting subtree. In this case, B will not be extended directly either – nodes have switched away from B if no other children extend it, or have switched to its descendants if it does have children. □

The result above can be used to bound the throughput of transactions for a network following the *GHOST* rule.[11] Recall that $TPS = b\cdot\beta\cdot K$. Using the inequality written in Proposition 8.4, $TPS \geq b\frac{1}{\frac{1}{\lambda}+2D}K$. By Proposition 3.1, the 50% attack imposes no constraint on the choice of b or λ, leaving the bounds on the TPS decided only by the physical limits on nodes' bandwidth.

Following the same method as in Subsection 5.2, we assume 50% of the network hears about every block of size b within $1.80 + 0.066 \cdot b$ seconds (and truncate the rest of the network). Accordingly, the TPS is at least

$$TPS \geq b\frac{1}{\frac{2}{\lambda} + 2(1.80 + 0.066\cdot b)}K = \frac{32}{\frac{\frac{2}{\lambda}+3.60}{b} + 0.132}, \tag{8}$$

[11] When carefully examining the proof of Proposition 8.4, it can be shown that it is also applicable to truncated networks, just as our lower bound for the longest-chain rule.

where we took $K = 32$ transactions per KB (which amounts to embedding only transaction hashes in blocks). This means that for any fixed rate λ, choosing the block size b to be large enough we obtain at least ≈ 242.4 transactions per second.

Practically, individual nodes' limited bandwidth imposes an additional constraint on these parameters. If every node's bandwidth is at least 0.427 MB per second, then the network is able to maintain a rate of $\lambda = 1$ blocks per second and $b = 320$ KB per block, with 10000 transaction hashes per block, achieving a TPS of more than 214.0.

This illustrates the ability of *GHOST* to attain both very high block creation rates and a high TPS.

8.3 The Rate of Collapse

In Subsection 8.1 we have discussed the collpase time ψ_B for any block B, and its implications to the growth and convergence of the main chain in *GHOST*. Long living forks imply longer waiting times until the entire network contributes confirmations to a block, and further implies long waiting times for transaction authorization. It can prove useful to further investigate how fast the collapse at B occurs. We do this for a simple model including only two forks, each with equal contributing hash power. Even this seemingly simple case proves to be non-trivial.

Theorem 8.6. *Consider a network with two nodes: u, v with equal hash-rates $\lambda/2$ which are connected by a single link with delay d. Assume that the network follows the GHOST selection rule, and let U and V be conflicting blocks in the network's block tree (one belonging to u, and the other to v). Then: $E[n_B] \leq \frac{(d\lambda)^2}{8} + \frac{d\lambda}{2}$.*

See the appendix for a proof of this theorem. The theorem gives an upper bound for the special configuration of two nodes; we conjecture, however, that it is the worst case, and that in general setups collapses occur even faster.

8.4 Implementation Details

Below, we outline some additional details about the use and implementation of the *GHOST* chain selection rule.

Waiting for authorization. While *GHOST* is generally more secure against attackers, it is important that clients be able to discern when transactions have been made sufficiently irreversible. We suggest the following strategy for clients:
First, listen to the network to determine when forks below the transaction have collapsed. From this point on, the network adds confirmations at a rate of λ_h. Then, apply the policy outlined in Section 7 for $n(t, r, q, \lambda_h)$, where t includes the time it takes for the chain to collapse, and n is the number of blocks in the sub-tree above the transaction, rather than the length of the chain. [12]

Retargeting (difficulty adjustment). Given potentially complex relations between the growth rate of the main chain and the rate of created blocks, and the fact that *GHOST* depends more on the total rate of block creation, we suggest a change in the way difficulty adjustments to the proof-of-work are done. Instead of targeting a certain rate of growth for the longest chain, i.e., β (which is Bitcoin's current strategy), we suggest that the total rate of block creation be kept constant (λ). As our protocol requires knowledge of off chain blocks by all nodes, we propose that information about off chain blocks be embedded inside each block

[12]Notice also that the value of λ_h is substituted for β. Herein lies the improvement.

(blocks can simply hold hashes of other blocks they consider off-chain). This can be used to measure and re-target the difficulty level so as to keep the total block creation rate constant.

Fees and minted coins. While *GHOST* does make use of off-chain blocks to secure the protocol, we believe it is best to allocate fees and mined coins only to the creators of blocks that are on the main chain, similarly to how the longest chain rule works today. The rate of minting can be adjusted independently from the block creation rate (but in a very similar way) by adjusting the amount of minted coins per block given the measured number of blocks in the recent past (e.g., in a 2 week window).

9 Conclusions

This paper has focused primarily on the effect network delays have on Bitcoin's transaction processing. We have shown that limitations that are due to delay in the network can in fact be more restrictive than bandwidth requirements. We have additionally analyzed the security of the protocol when delay is non-negligible compared to the block creation rate. Our results underscore the importance of the health of the network to Bitcoin's security and scalability. Finally, we presented our suggestion for the modification of the protocol, which helps secure Bitcoin even when working at high transaction rates.

Many additional research questions must be addressed in light of our results:

Additional Attacks. While we have analyzed the security of Bitcoin against an attack by a miner that attempts to mine an alternative chain secretely, other forms of attack on the network must be better understood. In particular, an attacker who can degrade the honest network performance, and can lower the rate of block propagation. Another form of attack can be one in which the attacker tries to use its own computational power to keep network forks from collapsing. This can be done by shifting computational power to generate blocks for the chain that is currently behind. A network that is split in this way is also somewhat less secure.

Congestion Collapse and Delays. Our network model included only a basic treatment of network delays. In particular, our assumption that the delay of a link is linearly dependent on the block size is only an ideal assumption that holds under low network load conditions. Under heavier load, links become congested and delays can increase non-linearly.

Dynamic Adjustment. The block creation rate and the block size can be ideally set for given network conditions. As the network changes and grows, different conditions may be ideal for different parameter settings. The task of changing parameters can thus be seen as a congestion control problem – optimizing the throughput of transactions to current network conditions.

Bitcoin's Decentralization. Bitcoin's decentralized nature is supported by lack of barriers to enter the mining business. Even a small firm (or indeed a single individual) can be profitable from the very beginning without needing to hold a large scale mining operation. Admittedly, the latest mining hardware is required, but any quantity would do. The risk to decentralization arises whenever there are increasing returns to scale, i.e., when large miners manage to profit over small ones. The delay in the network creates such a situation, by enabling large miners to get more than their fair share of the blocks, e.g., as delays go to infinity, all blocks in the network are produced by the node with the greatest hash-rate. The minimal requirements on bandwidth also create barriers – large miners can more easily pay for high bandwidth which gets them better network connectivity and helps them spread their blocks faster than their competitors are able to.

Simplified Protocol Verification. Our calculations above have shown, among other things, that it is possible to greatly speed up bitcoin block creation, and still remain resilient to attackers possessing up to 50% of the hash-rate. The benefits of these speedups would be fast confirmation times for individual transactions. Such changes, however, pose potential problems to light nodes that do not participate in mining, but rather download block headers alone, and request proofs of the existence of certain transactions in the blockchain from a full node (such proofs are simply the branch of the merkle tree leading to these transactions). These nodes will face a larger number of block headers to download. It is therefore of great interest to create light nodes that can, for example, verify the blockchain probabilistically, without needing to download all headers.

10 Acknowledgements

The authors would like to thank Christian Decker and Roger Wattenhofer for generously making their measurement data available. We thank Prof. Haim Sompolinsky for his contribution to an earlier version of Lemma 6.2, but primarily for his continuous support of the 1st author. Finally, we thank Gavin Andresen for his comments, and Meni Rosenfeld for useful discussions.

This research was supported in part by the Israel Science Foundation (grants no.616/13, and 1773/13)

References

[1] Bitcoin wiki, scalibility. https://en.bitcoin.it/wiki/Scalability.

[2] E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. *IACR Cryptology ePrint Archive*, 2012:596, 2012.

[3] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On bitcoin and red balloons. In *The 13th ACM Conference on Electronic Commerce*, pages 56–73. ACM, 2012.

[4] J. Bruce. Purely p2p crypto-currency with finite mini-blockchain (white paper). https://bitcointalk.org/index.php?topic=195275.0.

[5] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing (P2P), Trento, Italy*, September 2013.

[6] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology (CRYPTO 92)*, pages 139–147. Springer, 1993.

[7] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *arXiv:1311.0243*, 2013.

[8] H. Finney. The Finney attack (the Bitcoin Talk forum). https://bitcointalk.org/index.php?topic=3441.msg48384.

[9] G. O. Karame, E. Androulaki, and S. Capkun. Double-spending fast payments in bitcoin. In *The 2012 ACM conference on Computer and communications security*, pages 906–917. ACM, 2012.

[10] N. A. Lynch. *Distributed algorithms.* Morgan Kaufmann, 1996.

[11] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE Symposium on Security and Privacy*, 2013.

[12] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[13] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Security and Privacy in Social Networks*, pages 197–223. Springer, 2013.

[14] D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. *IACR Cryptology ePrint Archive*, 2012:584, 2012.

[15] M. Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[16] M. Rosenfeld. Analysis of hashrate-based double spending. https://bitcoil.co.il.Doublespend.pdf, 2012,.

[17] R. Serfozo. *Basics of applied stochastic processes.* Springer, 2009.

[18] D. Williams. *Probability with martingales.* Cambridge university press, 1991.

A Proof of Lemma 4.2

Lemma 4.2:
The following equation

$$\begin{pmatrix} \alpha(x) \\ \omega(x) \end{pmatrix} = \frac{A}{S}\begin{pmatrix} p_u\lambda(e^{S\cdot x}-1) \\ p_v\lambda(e^{S\cdot x}-1) \end{pmatrix} + \begin{pmatrix} \alpha(0) \\ \omega(0) \end{pmatrix} \tag{9}$$
$$\text{for } A=\alpha(0)-\omega(0) \quad ; \quad S=p_u\lambda-p_v\lambda,$$

implies:

$$\omega_{2d} = \frac{p_u\lambda - p_v\lambda}{p_u\lambda - p_v\lambda e^{-(p_u\lambda-p_v\lambda)2d}}.$$

Proof of Lemma 4.2. By equation 1, $\alpha(2d)=\omega_{2d}p_u\lambda$, and by 2, $\omega(0)=\alpha_0 p_v\lambda$. Therefore,

$$\omega(x) = \hat{A}p_v\lambda(e^{(p_u\lambda-p_v\lambda)x}-1)+\alpha_0 p_v\lambda, \text{ for } \hat{A} := \frac{A}{p_u\lambda-p_v\lambda} \Rightarrow$$
$$\omega'(x) = p_v\lambda A e^{(p_u\lambda-p_v\lambda)x}.$$

By 4.1, $\omega'(x)=p_v\lambda(\alpha(x)-\omega(x))$, and therefore,

$$\alpha(x)-\omega(x) = Ae^{(p_u\lambda-p_v\lambda)x} \Rightarrow \alpha'(x)=p_u\lambda(\alpha(x)-\omega(x)) \Rightarrow$$
$$\alpha(x) = \int_0^x p_u\lambda(\alpha(t)-\omega(t))dt+\alpha(0) = p_u\lambda\int_0^x Ae^{(p_u\lambda-p_v\lambda)t}dt+\alpha(0) =$$
$$\frac{p_u\lambda\cdot A}{p_u\lambda-p_v\lambda}(e^{(p_u\lambda-p_v\lambda)x}-1)+\alpha(0).$$

Using

$$\alpha(0)=\omega(0)+A=\alpha_0 p_v\lambda+A$$

we obtain

$$\alpha(x) = \frac{p_u\lambda \cdot A}{p_u\lambda - p_v\lambda}(e^{(p_u\lambda - p_v\lambda)x} - 1) + \alpha_0 p_v\lambda + A,$$

and, in particular,

$$\alpha(2d) = \frac{p_u\lambda \cdot A}{p_u\lambda - p_v\lambda}(E - 1) + \alpha_0 p_v\lambda + A.$$

Denote: $\hat{A} := \frac{A}{p_u\lambda - p_v\lambda}$, $E := e^{(p_u\lambda - p_v\lambda)2d}$. We have,

$$\hat{A} = \frac{\alpha(2d) - \alpha_0 p_v\lambda}{p_u\lambda E - p_v\lambda} = \frac{\omega_{2d} p_u\lambda - \alpha_0 p_v\lambda}{p_u\lambda E - p_v\lambda}.$$

We have thus obtained explicit expressions for $\alpha(x)$ and $\omega(x)$ subject to the parameters α_0 and ω_{2d}:

$$\alpha(x) = \hat{A}(p_u\lambda e^{(p_u\lambda - p_v\lambda)x} - p_v\lambda) + \alpha_0 p_v\lambda$$
$$\omega(x) = \hat{A}(p_v\lambda e^{(p_u\lambda - p_v\lambda)x} - p_v\lambda) + \alpha_0 p_v\lambda$$

By the definition of α we know that α's integral over the range $(0, 2d]$ should be $1 - \alpha_0$:

$$1 - \alpha_0 = \int_0^{2d} \left(\hat{A}(p_u\lambda e^{(p_u\lambda - p_v\lambda)t} - p_v\lambda) + \alpha_0 p_v\lambda\right) dt =$$
$$\hat{A}\left(\frac{p_u\lambda(E-1)}{p_u\lambda - p_v\lambda} - 2d \cdot p_v\lambda\right) + 2d \cdot \alpha_0 p_v\lambda =$$
$$\hat{A}\left(\hat{E} - 2d \cdot p_v\lambda\right) + 2d \cdot \alpha_0 p_v\lambda,$$

for $\hat{E} := \frac{p_u\lambda(E-1)}{p_u\lambda - p_v\lambda}$. Therefore,

$$\alpha_0 = 1 - \frac{\omega_{2d} p_u\lambda - \alpha_0 p_v\lambda}{p_u\lambda E - p_v\lambda}\left(\hat{E} - 2d \cdot p_v\lambda\right) - 2d \cdot \alpha_0 p_v\lambda$$

Similarly, the integral of ω over $[0, 2d)$ should be $1 - \omega_{2d}$, and combining this with the relation $\alpha(x) - \omega(x) = Ae^{(p_u\lambda - p_v\lambda)x}$ we obtain:

$$1 - \alpha_0 - (1 - \omega_{2d}) = \int_0^{2d} Ae^{(p_u\lambda - p_v\lambda)t} dt = \hat{A}(E - 1) \Rightarrow$$
$$\omega_{2d} - \alpha_0 = \hat{A}(E-1) = \frac{\omega_{2d} p_u\lambda - \alpha_0 p_v\lambda}{p_u\lambda E - p_v\lambda}(E - 1) \Rightarrow$$
$$\frac{\omega_{2d}}{\alpha_0} - 1 = \frac{\frac{\omega_{2d}}{\alpha_0} p_u\lambda - p_v\lambda}{p_u\lambda E - p_v\lambda}(E-1) \Rightarrow$$
$$\left(1 - \frac{p_u\lambda(E-1)}{p_u\lambda E - p_v\lambda}\right)\frac{\omega_{2d}}{\alpha_0} = 1 - \frac{p_v\lambda}{p_u\lambda E - p_v\lambda}(E-1) \Rightarrow$$
$$\frac{\omega_{2d}}{\alpha_0} = \frac{1 - \frac{p_v\lambda}{p_u\lambda E - p_v\lambda}(E-1)}{1 - \frac{p_u\lambda(E-1)}{p_u\lambda E - p_v\lambda}} = E$$

Therefore,

$$\omega_{2d} = E \cdot \alpha_0 = E\left(1 - \frac{\omega_{2d} p_u \lambda - \frac{\omega_{2d}}{E} p_v \lambda}{p_u \lambda E - p_v \lambda}\left(\hat{E} - 2d \cdot p_v \lambda\right) - 2d \cdot \frac{\omega_{2d}}{E} p_v \lambda\right) =$$

$$E\left(1 - \frac{\omega_{2d}}{E}\left(\hat{E} - 2d \cdot p_v \lambda\right) - 2d \cdot \frac{\omega_{2d}}{E} p_v \lambda\right) = E - \omega_{2d}\hat{E} \Rightarrow$$

$$\omega_{2d} = \frac{E}{1 + \hat{E}_u} = \frac{e^{(p_u\lambda - p_v\lambda)2d}}{1 + \frac{p_u\lambda(e^{(p_u\lambda - p_v\lambda)2d} - 1)}{p_u\lambda - p_v\lambda}} =$$

$$\frac{e^{(p_u\lambda - p_v\lambda)2d}}{p_u\lambda e^{(p_u\lambda - p_v\lambda)2d} - p_v\lambda} \cdot (p_u\lambda - p_v\lambda) =$$

$$\frac{p_u\lambda - p_v\lambda}{p_u\lambda - p_v\lambda e^{-(p_u\lambda - p_v\lambda)2d}}$$

□

B Proof of Lemma 6.2

Lemma 6.2:
Let ς be a nonnegative random variable with increasing hazard rate function. Then, $\forall k \in \mathbb{N}$

$$E[\varsigma^k] \leq k! E^k[\varsigma].$$

Proof of Lemma 6.2. By induction on k. The base case $k = 0$ is trivial. For $k + 1$ we have:

$$E[\varsigma^k] = \int_0^\infty \varsigma^{k+1} f(\varsigma) = \int_0^\infty \varsigma^{k+1}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma =$$

$$[\varsigma^{k+1} \cdot -e^{-\Lambda(\varsigma)}]_0^\infty$$

$$+ \int_0^\infty (k+1)\varsigma^k e^{-\Lambda(\varsigma)}d\varsigma = (k+1)\int_0^\infty \frac{\varsigma^k}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma$$

On the other hand,

$$E[\varsigma] = \int_0^\infty \varsigma f(\varsigma) = \int_0^\infty \varsigma\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma =$$

$$[\varsigma^\cdot - e^{-\Lambda(\varsigma)}]_0^\infty + \int_0^\infty e^{-\Lambda(\varsigma)}d\varsigma = \int_0^\infty e^{-\Lambda(\varsigma)}d\varsigma,$$

and thus,

$$(k+1)!E^{k+1}[\varsigma] = (k+1)k!E^k[\varsigma]E[\varsigma] =$$

$$(k+1)k!E^k[\varsigma]\int_0^\infty e^{-\Lambda(\varsigma)}d\varsigma$$

$$= (k+1)\int_0^\infty \frac{k!E^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma$$

It is thus sufficient to prove that,

$$(k+1)\int_0^\infty \frac{\varsigma^k}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma \leq (k+1)\int_0^\infty \frac{k!E^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma,$$

or, equivalently, that

$$\int_0^\infty \frac{\varsigma^k - k!E^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma \leq 0.$$

Using the induction hypothesis we obtain:

$$\int_0^\infty \frac{\varsigma^k - k!E^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma =$$

$$\int_0^{k!E^k[\varsigma]} \frac{\varsigma^k - k!E^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma$$

$$+\int_{k!E^k[\varsigma]}^\infty \frac{\varsigma^k - k!E^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma$$

$$\leq^1 \frac{1}{\lambda(k!E^k[\varsigma])}\int_0^{k!E^k[\varsigma]} (\varsigma^k - k!E^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma +$$

$$\frac{1}{\lambda(k!E^k[\varsigma])}\int_{k!E^k[\varsigma]}^\infty (\varsigma^k - k!E^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma =$$

$$= \frac{1}{\lambda(k!E^k[\varsigma])}\int_0^\infty (\varsigma^k - k!E^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma \leq^2 0,$$

where we used λ's monotonicity in 1 and the induction hypothesis in 2. □

Lemma B.1. *Let ς be as in Lemma 6.2, let f be its pdf, and denote $\beta := E[\varsigma]^{-1}$. Then for any constant $0 < \gamma < \beta$ the function $H_{\gamma,\beta}$ obtains a positive root a_0 smaller than $\frac{\gamma}{\beta}$, where*

$$H_{\gamma,\beta}(a) := \int_0^\infty f(\varsigma)e^{\gamma(\frac{1}{a}-1)\varsigma}d\varsigma - \frac{1}{a}.$$

Proof of Lemma B.1. We have $H_{\gamma,\beta}(0) = \infty$. $H_{\gamma,\beta}$ is continuous in a and thus, by The Intermediate Value Theorem, it suffices to show that $H_{\gamma,\beta}(\frac{\gamma}{\beta}) \leq 0$.
Put:

$$H(\gamma) := H_{\gamma,\beta}(\frac{\gamma}{\beta})$$

We need to show that $H(\gamma) \leq 0$, and we do so by showing that its Taylor series elements (around β) are all (!) nonpositive. That is, we show that

$$H^{(k)}(\beta)\frac{(\gamma-\beta)^k}{k!} \leq 0,$$

and this will imply,

$$H(\gamma) = \Sigma_{k=0}^\infty H^{(k)}(\beta)\frac{(\gamma-\beta)^k}{k!} \leq 0.$$

Indeed,

$$H^{(k)}(\beta) = \frac{d^k}{d\gamma^k}\left\{\int_0^\infty f(t)e^{\gamma(\frac{1}{\frac{\gamma}{\beta}}-1)t}dt - \frac{1}{\frac{\gamma}{\beta}}\right\}_{\gamma=\beta} =$$

$$\frac{d^k}{d\gamma^k}\left\{\int_0^\infty f(t)e^{(\beta-\gamma)t}dt - \frac{\beta}{\gamma}\right\}_{\gamma=\beta} =$$

$$\left\{\int_0^\infty (-t)^k f(t) e^{\gamma(\frac{1}{a}-1)t} dt + \frac{k!\beta}{(-\gamma)^{k+1}}\right\}_{\gamma=\beta} =$$
$$\int_0^\infty (-t)^k f(t) dt + k!\beta^{-k}(-1)^{k+1}.$$

As the Taylor elements of $H(\gamma)$ are of alternating signs (recall $\gamma < \beta$), it suffices to show the inequalities $H^{(k)}(\beta) \leq 0$ and $H^{(k)}(\beta) \geq 0$ for even and odd k's respectively. This, in turn, generates to showing that for all k:

$$\int_0^\infty t^k f(t) dt \leq k!\beta^{-k},$$

which was proved in Lemma 6.2. □

C Proof of Theorem 6.1

Theorem 6.1:
Consider a network G with delays. Let $1/\tilde{\beta}$ be an upper-bound on the expected waiting time for the next lengthening of the main chain, given *any history.* Let $\gamma \leq \tilde{\beta}$ be the block creation rate of the attacker (according to a Poisson process), and suppose the network's longest chain is longer than the attacker's by X_0 blocks. Then the probability that the attacker will succeed in extending his chain to be longer than the network's is at most $\left(\frac{\gamma}{\tilde{\beta}}\right)^{X_0+1}$.

Proof of Theorem 6.1. Let τ_n be the waiting time for the nth lengthening of the main chain. Let $f_{\tau_n|\tau_{n-1},...,\tau_1}$ be the conditional pdf of τ_n given $\tau_{n-1},...,\tau_1$. Denote $\beta := E[\tau_n \mid \tau_{n-1},...,\tau_1]^{-1}$ for some *given* history (that is, for some realization of the τ_i's up to $n-1$).

The random variable τ_n given a history is nonnegative with increasing hazard rate. Indeed, when a node creates a new block it broadcasts it to the network, and as more and more nodes learn about it, more hash-rate is contributed to the effort of creating the next one and thereby lengthening the main chain. If meanwhile a conflicting block was created elsewhere, still more hash-rate is working on lengthening the main chain, just on a different version of it.

By our assumption, $\beta \geq \tilde{\beta}$, and thus $\forall k \in \mathbb{N}, \beta^{-k} \leq \tilde{\beta}^{-k}$. As a corollary of Lemma 6.2 we get that $E[\tau_n^k \mid \tau_{n-1},...,\tau_1] \leq k!\tilde{\beta}^{-k}$. Embedding this fact to the end of Lemma B.1's proof, we are provided with the inequality $H_{\gamma,\tilde{\beta}}(\frac{\gamma}{\tilde{\beta}}) \leq 0$.

The attacker's chain is built according to a Poisson process in the worst case, whose rate we denoted by γ. Let N_2 be the event-count (random variable) of this process, namely, $N_2(t) := max\{n \mid \sum_{j=1}^n \tau_n \leq t\}$. Define, $X_n := n - N_2(\sum_{j=1}^n \tau_n)$, and $Y_n := (\frac{\gamma}{\tilde{\beta}})^{X_n}$.

The process $X = (X_n)$ represents the gap between the lengths of the attacker's chain and the (worst-case) main chain, in favor of the later, as the nth lengthening of the later occurred.

We claim that $Y = (Y_n)$ is a super-martingale, namely that for any history, $E[Y_{n+1} \mid Y_n,...,Y_0] \leq Y_n$. Indeed, while the value of X_{n+1} depends naturally on $\tau_{n+1},...,\tau_1$, the increment $X_{n+1} - X_n$ given a history $\tau_n,...,\tau_1$ is controlled by the random variable τ_n given this history, with the pdf $f_{\tau_n|\tau_{n-1},...,\tau_1}$ which we abbreviate f. We have:

$$E\left[Y_{n+1} \mid Y_n, ..., Y_0\right] = E\left[(\frac{\gamma}{\tilde{\beta}})^{X_{n+1}} \mid (\frac{\gamma}{\tilde{\beta}})^{X_n}, ..., (\frac{\gamma}{\tilde{\beta}})^{X_0}\right] =^1$$

$$\sum_{k=0}^{\infty} \int_0^{\infty} f(\tau_{n+1}) \frac{e^{-\gamma\tau_{n+1}}(\gamma\tau_{n+1})^k}{k!} (\frac{\gamma}{\tilde{\beta}})^{X_n+1-k} d\tau_{n+1} =^2$$

$$(\frac{\gamma}{\tilde{\beta}})^{X_n+1} \cdot \sum_{k=0}^{\infty} \int_0^{\infty} f(\tau_{n+1}) \frac{e^{-\gamma\tau_{n+1}}(\gamma\tau_{n+1})^k}{k!} (\frac{\gamma}{\tilde{\beta}})^{-k} d\tau_{n+1} =$$

$$(\frac{\gamma}{\tilde{\beta}})^{X_n+1} \cdot \int_0^{\infty} f(\tau_{n+1}) e^{-\gamma\tau_{n+1}} \sum_{k=0}^{\infty} \frac{(\frac{\gamma\tau_{n+1}}{\frac{\gamma}{\tilde{\beta}}})^k}{k!} d\tau_{n+1} =$$

$$(\frac{\gamma}{\tilde{\beta}})^{X_n+1} \cdot \int_0^{\infty} f(\tau_{n+1}) e^{-\gamma\tau_{n+1}} e^{\frac{1}{\frac{\gamma}{\tilde{\beta}}}\gamma\tau_{n+1}} d\tau_{n+1} =$$

$$(\frac{\gamma}{\tilde{\beta}})^{X_n} \cdot \frac{\gamma}{\tilde{\beta}} \cdot \int_0^{\infty} f(\tau_{n+1}) e^{\gamma(\frac{1}{\frac{\gamma}{\tilde{\beta}}}-1)\tau_{n+1}} d\tau_{n+1} \leq^3$$

$$(\frac{\gamma}{\tilde{\beta}})^{X_n} = Y_n.$$

Equality 1 is due to the attacker's chain advancing during the waiting time τ_{n+1} according to a Poisson process with rate $\tau_{n+1} \cdot \gamma$. In 2 we made explicit the fact that $(\frac{\gamma}{\tilde{\beta}})^{X_n}$ is a constant in the σ-algebra corresponding to the natural filtration (usually denoted by $\sigma(X_n, ..., X_1)$). Finally, by the proof of Lemma B.1, $H_{\gamma,\tilde{\beta}}(\frac{\gamma}{\tilde{\beta}}) \leq 0$, thus inequality 3.

Let $x_1 < X_0 < x_2$ be some fixed constants, let the stopping time π be defined by $\pi := \min\{n \mid X_n \leq x_1 \vee X_n \geq x_2\}$, and finally, define the event $E_{x_1,x_2} := \{X_\pi = x_2\}$ (i.e., "X reached x_2 before it reached x_1"). By Doob's Optional Stopping Theorem (See [18], p. 100-101) applied to the super martingale Y, we have

$$(\frac{\gamma}{\tilde{\beta}})^{X_0} = Y_0 \geq E[Y_\pi] =$$

$$Pr(E_{x_1,x_2}) \cdot (\frac{\gamma}{\tilde{\beta}})^{x_2} + Pr(E^c_{x_1,x_2}) \cdot (\frac{\gamma}{\tilde{\beta}})^{x_1} \Rightarrow$$

$$(\frac{\gamma}{\tilde{\beta}})^{X_0} - (\frac{\gamma}{\tilde{\beta}})^{x_1} \geq Pr(E_{x_1,x_2}) \cdot \left((\frac{\gamma}{\tilde{\beta}})^{x_2} - (\frac{\gamma}{\tilde{\beta}})^{x_1}\right) \Rightarrow$$

$$Pr(E_{x_1,x_2}) \geq \frac{(\frac{\gamma}{\tilde{\beta}})^{X_0} - (\frac{\gamma}{\tilde{\beta}})^{x_1}}{(\frac{\gamma}{\tilde{\beta}})^{x_2} - (\frac{\gamma}{\tilde{\beta}})^{x_1}}.$$

Taking $x_1 = -1$ and $x_2 \to \infty$ we obtain a lower bound on the probability that the gap between the chains will never reach minus 1: $1 - (\frac{\gamma}{\tilde{\beta}})^{X_0+1}$. The success probability of an attack is thus upper bounded by $(\frac{\gamma}{\tilde{\beta}})^{X_0+1}$. □

Note that an almost identical method shows that if the random variables τ_n are i.i.d then there exists an $a_0 \leq \frac{\gamma}{\tilde{\beta}}$ such that $Y := a_0^X$ is a martingale.

D Proof of Theorem 8.6

Theorem 8.6:
Consider a network with two nodes: u, v with equal hash-rates $\lambda/2$ which are connected by a

single link with delay d. Assume that the network follows the *GHOST* selection rule, and let U and V be conflicting blocks in the network's block tree (one belonging to u, and the other to v). Then:

$$E[n_B] \leq \frac{(d\lambda)^2}{8} + \frac{d\lambda}{2}.$$

Proof of Theorem 8.6. We define a state x_n representing the time gap between the of creation the n'th block by each of the nodes, in favor of u.

It is clear that whenever $|x_n| > d$, a collapse has occurred, as this means a message from u about a new block has arrived at v without the later creating a corresponding block in time, or vice versa.

In order to count n_U, we recursively express the expected number of *additional* blocks in *subtree*(U), given the current state x_n. We denote this by $h(x_n)$.

Given that the time gap x_{n+1} is positive, its value depends on the next block creation of v, and thus follows an exponential distribution with rate $\lambda/2$; the same argument applies to the case $x_{n+1} < 0$. If $|x_{n+1}| < d$, the expected addition to *subtree*(U) (conditioned on the current state) is simply $1 + h(x_{n+1})$, otherwise, it is exactly 0. We express $h()$ as a sum of two functions $f(), g()$. One for the case in which the time gap increases in favor of u (f), and one for the case in which it decreases (g). By symmetry, the probability for these events is $\frac{1}{2}$. This justifies the following equations for f, g and h:

$$f(x) := \frac{1}{2}\int_x^d \mu e^{-\mu(t-x)}(h(t)+1)dt = e^{\mu x}\int_x^d \mu e^{-\mu t}\frac{h(t)+1}{2}dt$$

$$g(x) := \frac{1}{2}\int_{-d}^x \mu e^{-\mu(x-u)}(h(u)+1)du = e^{-\mu x}\int_{-d}^x \mu e^{\mu u}\frac{h(t)+1}{2}du$$

$$h(x) = f(x) + g(x).$$

Then, $\frac{df}{dx} = \mu e^{\mu x}\int_x^d \mu e^{-\mu t}\frac{h(t)+1}{2}dt + e^{\mu x}\cdot -1 \cdot \mu e^{-\mu x}\frac{h(x)+1}{2} =$

$$\mu f(x) - \mu\frac{h(x)+1}{2} = \mu f(x) - \mu\frac{f(x)+g(x)+1}{2} = \frac{\mu}{2}(f(x) - g(x) - 1)$$

And similarly, $\frac{dg}{dx} = \frac{\mu}{2}(f(x) - g(x) + 1)$

Differentiating f and g we obtain the following linear non homogeneous differential system:

$$\begin{pmatrix} f \\ g \end{pmatrix}' = \begin{pmatrix} \frac{\mu}{2} & -\frac{\mu}{2} \\ \frac{\mu}{2} & -\frac{\mu}{2} \end{pmatrix} \cdot \begin{pmatrix} f \\ g \end{pmatrix} + \begin{pmatrix} -\frac{\mu}{2} \\ \frac{\mu}{2} \end{pmatrix}$$

with the following boundary conditions:

$$f(d) = 0, g(-d) = 0.$$

Solving this system yields:

$$\begin{aligned} f(x) &= \frac{1}{4}\left((d\mu)^2 - (x\mu)^2 + 2d\mu - 2x\mu\right) \\ g(x) &= \frac{1}{4}\left((d\mu)^2 - (x\mu)^2 + 2d\mu + 2x\mu\right) \\ h(x) &= \frac{1}{2}\left((d\mu)^2 - (x\mu)^2 + 2d\mu\right) \end{aligned}$$

As the state at which the competition begins is $x = 0$, by symmetry, we get that the expected number of blocks until a collapse is $h(0) = \frac{(d\mu)^2}{2} + d\mu$ blocks. $\square$